ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 35

Settore Concorsuale: 09/G1 – Automatica
Settore Scientifico Disciplinare: ING-INF/04 – Automatica

A ROBOTIC PLATFORM FOR PRECISION AGRICULTURE AND APPLICATIONS

Presentata da: Dario Mengoli

Coordinatore Dottorato

Prof. Michele Monaci

Supervisore

Prof. Lorenzo Marconi

Esame finale anno 2023

Abstract

Agricultural techniques have been improved over the centuries to match with the growing demand of an increase in global population. As the current world number of people is surpassing 8 billion individuals, with an exponential growth, and planet resources are reducing, farming applications are facing new challenges to satisfy global needs. The recent technology advancements in terms of robotic platforms can be exploited to help farmers increase their production with an increment in efficiency that may also lead to cost reductions.

Many research groups from different universities are investigating new farming techniques by building new robotic platform to automate several farming tasks in the field. As the orchard management is one of the most challenging applications because of its tree structure and the required interaction with the environment, it was targeted also by the University of Bologna research group to provide a customized solution addressing new concept for agricultural vehicles.

The result of this research has blossomed into a new lightweight tracked vehicle capable of performing autonomous navigation both in the open-filed scenario and while travelling inside orchards for what has been called in-row navigation. The mechanical design concept, together with customized software implementation has been detailed to highlight the strengths of the platform and some further improvements envisioned to improve the overall performances.

Static stability testing has proved that the vehicle can withstand steep slopes scenarios with more than 40° of inclination. Some improvements have also been investigated to refine the estimation of the slippage that occurs during turning maneuvers and that is typical of skid-steering tracked vehicles.

The software architecture has been implemented using the Robot Operating System (ROS) framework, so to exploit community available packages related to common and basic functions, such as sensor interfaces, while allowing dedicated custom implementation of the navigation algorithm developed.

Real-world testing inside the university's experimental orchards located in Cadriano (Bologna, Italy) have proven the robustness and stability of the solution with more than 800 hours of fieldwork.

The vehicle has also enabled a wide range of autonomous tasks such as spraying, mowing, and on-the-field data collection capabilities. The latter can be exploited to automatically estimate relevant orchard properties such as fruit counting and sizing, canopy properties estimation, and autonomous fruit harvesting with post-harvesting estimations.

Sintesi

Le tecniche agricole sono state migliorate nel corso dei secoli per soddisfare la crescente domanda di aumento della popolazione mondiale. Poiché l'attuale numero di persone sta superando gli 8 miliardi con una crescita esponenziale, e le risorse del pianeta si stanno contemporaneamente riducendo, le applicazioni agricole stanno affrontando nuove sfide per soddisfare le esigenze globali. I recenti progressi tecnologici in termini di piattaforme robotiche possono essere sfruttati per aiutare gli agricoltori ad aumentare la loro produzione con un incremento dell'efficienza che può anche portare a una riduzione dei costi.

Molti gruppi di ricerca di diverse università stanno studiando nuove tecniche e costruendo nuove piattaforme robotiche per automatizzare diverse attività agricole sul campo. Poiché la gestione del frutteto è una delle applicazioni più impegnative, a causa della sua struttura arborea e della necessità di interagire con l'ambiente, questa è stata presa in considerazione anche dal gruppo di ricerca dell'Università di Bologna per fornire una soluzione personalizzata che sviluppi un nuovo concetto di veicolo agricolo.

Il risultato di questa ricerca si è concretizzato in un nuovo veicolo cingolato leggero, in grado di effettuare una navigazione autonoma sia nello scenario di pieno campo che all'interno dei frutteti per quella che è stata definita navigazione interfilare. Il concetto di progettazione meccanica, insieme all'implementazione del software sviluppato, sono stati dettagliati per evidenziare i punti di forza della piattaforma ed alcuni ulteriori miglioramenti previsti per incrementarne le prestazioni complessive.

I test di stabilità statica hanno dimostrato che il veicolo è in grado di resistere a scenari di pendii ripidi con oltre 40° di inclinazione. Sono stati inoltre studiati alcuni miglioramenti per affinare la stima dello slittamento che si verifica durante le manovre di svolta, tipico dei veicoli cingolati.

L'architettura software è stata implementata utilizzando il framework Robot Operating System (ROS), in modo da sfruttare i pacchetti resi disponibili dalla comunità e relativi alle funzioni di base, come ad esempio le interfacce dei sensori, e consentendo al contempo un'implementazione personalizzata degli algoritmi di navigazione sviluppati.

I test in condizioni reali all'interno dei frutteti sperimentali dell'università, situati a Cadriano (Bologna, Italia), hanno dimostrato la robustezza e la stabilità della soluzione con oltre 800 ore di lavoro sul campo.

Il veicolo ha inoltre permesso di abilitare e svolgere un'ampia gamma di attività agricole in maniera autonoma, come l'irrorazione, la trinciatura e la raccolta di dati sul campo. Questi ultimi possono essere sfruttati per stimare automaticamente le proprietà più rilevanti del frutteto, come il conteggio e la misurazione dei frutti, la stima delle proprietà della chioma e la raccolta autonoma dei frutti con alcune stime post-raccolta.

Table of Contents

Chapter 1. Introduction

In this chapter, the ideas behind Precision Agriculture and the utilization of robotics to facilitate PA applications are presented. Also highlighted are the rationale for this effort and the current status of robotic applications in agriculture.

As global earth population reached 8 billion people with an increasing trend [1], the need for food production is one of the main challenges of the future years. As more than 70 percent of global freshwater withdrawals and half of the world's habitable land is used for agriculture [2], it is clear that an optimization of the food production chain is needed to ensure the survival of humanity. During the centuries, farmers have always tried to increase agricultural techniques, by exploiting technology advancements, in order to increase harvest and overcome weather or seasonal limitations.

1.1. A brief history of the agricultural evolution

1.1.1. Ancient agriculture

The birth of agriculture dates around 10500 B.C. with the first cultivation of wild seeds such as barley, wheat, and lentils. Around 9800 B.C. the first domestic forms of such seeds born by successive selections from the wild forms are cultivated. Early cultivation occurs not only in the Fertile Crescent but also independently and at different times at other sites in both Eurasia and the Americas. Ancient Egypt saw the emergence of the plow and hoe, tools that facilitated agricultural productivity and encouraged the settling of populations. With the consequent development of animal husbandry, mainly cattle, which is also used as motive power, agriculture progresses and enables the sustenance of more people. In the Middle Ages new crops such as rice, citrus, spinach, and cotton arrived in Europe from the East. After 1500 A.D. from America come potato, corn, tomato, peppers, and beans.

1.1.2. Technology and industrial improvements

In ancient times, biennial field rotation was used to increase production, whereby one half of the field was sown while the other half was left fallow, i.e., fallow, then the following year the fields were reversed. With the advent of cattle raising, it was possible to use manure as fertilizer so as to increase the yield of the land. In addition, cattle could be used as motive power for plowing and moving commodities, greatly alleviating farm labor. In order to be able to exploit even difficultly arable land, the cultivation of vines, citrus and other tree species took hold.

With the advent of the industrial revolution in the 19th century, new mechanical technologies and new chemical fertilization techniques were introduced in agriculture, which on the one hand led to an increase in productivity and a simultaneous shift of labor to industry. Numerous reclamations of state-owned and swampy lands were initiated, which made it possible to support the large increase in urban population at the time.

1.1.3. Modern agriculture

The need for ever higher production at ever lower cost has led to the emergence of intensive agriculture. Such practices lead to the increasing use of fertilizers, pesticides, genetic engineering, and mechanization. Agricultural mechanization, the result of the development of engineering in the 1900s, has mainly led to a reduction in the use of labor, especially seasonal and low-skilled labor, but has only partly contributed to reducing production costs.

Unfortunately, the development of intensive agriculture is accompanied by a strong environmental impact and a depletion of biodiversity, thus organic farming was born, but it

remains a niche sector with high product costs. Another problem lies in the fact that the development of agricultural mechanization has not progressed uniformly in all agricultural sectors, for example, open field agriculture has enjoyed mechanization the most while tree crops and viticulture have lagged behind.

Figure 1 - Arable crops mechanization

1.1.4. The advent of precision agriculture

One answer to these problems, along with the need to provide sustenance for an ever-increasing world population, is most likely to come from precision agriculture. The concept of precision agriculture originated in the U.S. in the early 1980s, developing, with new computer techniques and technologies, the virtuous practice of small farmers who daily monitored both the soil in their field and the state of plant growth and then decided what to do.

This practice basically uses three technologies: mechanization to provide increasingly precise and high-performance operational tools, monitoring to know the condition of the soil and plants in real time, and artificial intelligence decision-making processes supported by large data bases to determine how and when to make the necessary interventions. Precision monitoring of field conditions and the use of decision-making processes derived from increasingly accurate mathematical models enable targeted use of treatments and irrigation, reducing consumption of pesticides, insecticides, and water, with clear benefits for the environment. The possibility of using self-driving vehicles for agricultural activities, designed to operate best in the various environmental types (open field, orchard or vineyard rows, hilly terrain, etc.) allow a significant reduction in the use of seasonal and low-skilled labor, thus significantly reducing production costs.

Continuous monitoring of the chemical and physical conditions of the soil, possibly divided into larger or smaller plots depending on the variability of the same, temporally correlated don the state and development of plants allows for the realization of very important historical sequences of big data to be used to support decision-making processes.

Figure 2 - Field monitoring and Variable Rate Application are key aspects of Precision Agriculture

We can therefore conclude that precision agriculture leads to a reduction in the use of pesticides and fertilizers, and a better use of water, with related benefits for the environment, in addition, automation leads to the reduction of labor that combined with the increase in productivity that all this technology produces lead to certain economic benefits.

For all these reasons, we can hope that precision agriculture is a very encouraging prospect for the future evolution of world agriculture that can then better meet the needs of the entire population.

1.2. The role of robotic platforms in Precision Agriculture

For the primary sector, precision agriculture and autonomous vehicles, as well as safety measures for agricultural machine operators, are subjects of increasing interest. Agriculture is indeed one of the sectors that features the deadliest accidents every year [3]–[5], and tractor overturning is by far the most common cause of injury or death for operators [3]–[6].

Autonomous or Unmanned Ground Vehicles (AGVs or UGVs) can be a powerful tool in minimizing the rampant issue of fatal injuries. Their ability to run autonomously and decrease reliance on operators in the field can provide a much-needed solution, not just for better safety, but also as a practical response to workforce shortage both in terms of quantity and skill. Furthermore, their usage can increase the timeliness and accuracy of farming tasks execution with a high level of repeatability, and they can even surpass traditional methods due to existing regulation concerning human health, such as limitations on daily vibration exposure of the operators. Lastly, UGVs can be beneficial in extreme conditions or hazardous terrains, which could lead to tractor overturns when using traditional machinery.

The origins of PA can be traced back to the 1980s, but early prototypes of self-navigating vehicles were developed as far back as 30 years before. The ultimate goal of PA is to allocate resources and time accurately in order to get the best results while minimizing inputs - known as "doing the right thing, in the right place and in the right time" [7]. It allows for more efficient use of resources, either by reducing inputs without compromising on quality or obtaining a better output from existing resources [8].

The economic and environmental consequences must also be considered along with the technical aspects, since avoiding overlap during field operations (such as spraying and spreading)

can significantly reduce the amount of inputs and increase the working capacity, resulting in a more sustainable environment and economy.

PA's deep interest in autonomous driving systems stems from the need to relieve the operator of two main aspects of driving:

- The physical one, in which the driver uses their movements and physical effort to turn the steering wheel and operate levers and pedals to travel along the expected path.
- The mental one, as the operator uses their intelligence to control the implements and needs continuous attention to complete the specific operation with all the maneuvers required by the task.

Indeed, agricultural machine drivers typically have two core duties: operating the vehicle and managing the equipment to maintain an adequate level of productivity. Despite the stability and repetitiveness of the tasks over the last decades, the design and components of both tractors and related tools have seen radical changes over time and properly operate all the systems in a field environment has become increasingly complicated, especially for elderly operators. This has favored the current research topics and solutions for PA, such as development of UGVs powered by artificial intelligence algorithms.

1.3. Research Activity on Autonomous and Robotic Ground Vehicles for Agricultural Tasks[1]

The aim of this section was to analyze and summarize the main contributions, findings, and prototypes developed by research groups and universities from all around the world in relation to autonomous vehicles in agriculture, given the current interest of researchers, manufacturers, and stakeholders.

1.3.1. Research projects outside Europe

The first prototype described is developed by Massey University (New Zealand). As detailed in [9], the platform is an autonomous robotic four-wheel-drive vehicle designed for harvesting tasks in orchards. Using four robotic arms mounted on the platform, kiwifruit pickers end-effectors can reach the fruit hanging above the platform and harvest it (Figure 3).

[1] This work has contributed to the published paper: A Review of Current and Historical Research Contributions to the Development of Ground Autonomous Vehicles for Agriculture. - Rondelli, V., Franceschetti, B., Mengoli, D. - Sustainability (Switzerland), 2022, 14(15), 9221

Figure 3 - Kiwifruit harvesting prototype vehicle developed by Massey University (New Zealand)

With the help of a combustion engine, working as a generator, the UGV can move and steer exploiting its hydraulic system, driven by a hydraulic pump. The harvesting robotic arms are powered by electrical motors and the control system algorithms are executed by two dual-processor motherboards. Using a Differential Global Positioning System (DGPS) receiver, a digital compass, and a computer vision system, navigation and localization tasks are performed. A total of 10 cameras are installed in the prototype: two of them are used in the navigation algorithms to enable obstacle avoidance features, six are used for harvesting fruit (so to detecting and locate fruits and track them), and two are used for monitoring the stored fruits in the harvesting bins.

Monitoring both the machine's status and its environment allows the machine to operate continuously. As a result of the first algorithm, a full bin is unloaded at the end of a row, an empty bin is collected, and the fuel refill station is automatically accessed when required. As a result of the second monitoring algorithm, when wet conditions are detected, the vehicle automatically pauses operations and controls floodlights accordingly.

Bangkok University in Thailand created a three-wheeled platform, powered by a 500 W DC motor for each of the two rear wheels (Figure 4). This autonomous vehicle has several sensors, like a speed encoder/tachometer and a steering angle sensor, that help estimating the total distance traveled by integrating odometry data. Autonomous navigation is enabled by using a Proportional-Integral-Derivative (PID) controller to regulate speed and a proportional controller to manage the steering angle. The purpose of this vehicle is to sow rice directly on dry fields.

The vehicle can follow a series of waypoints to enable autonomous seeding operations. In order to enable this, and more generally autonomous waypoints navigation or path following capabilities, a GPS antenna and an Inertial Measurement Unit (IMU) are used for platform localization and attitude estimation [10]. An Extended Kalman Filter (EKF) is used to accurately estimate the overall system's position from the sensors information.

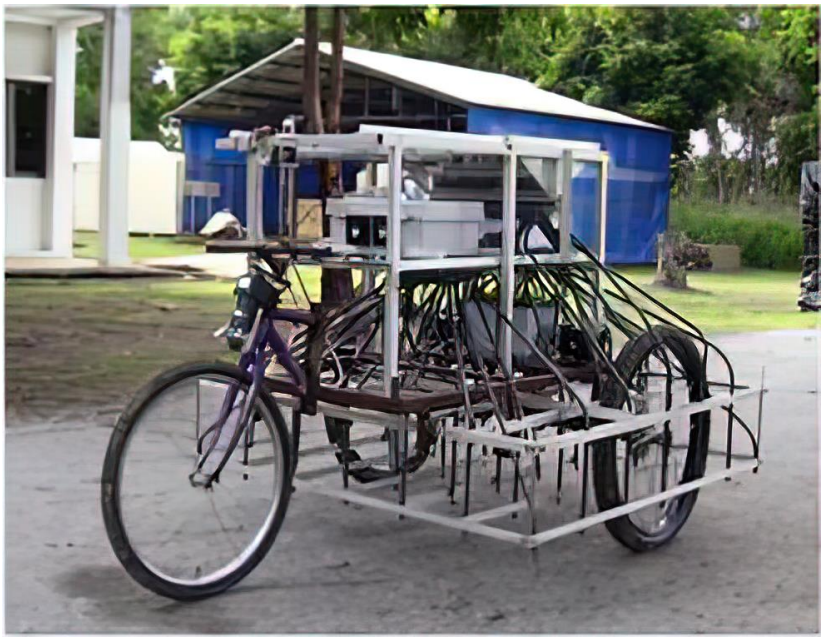

Figure 4 - Rice seeding robot developed by Bangkok University

The Australian Centre for Field Robotics (ACFR) at the University of Sydney has developed a multisensory UGV platform called "Shrimp". It can detect, trace, localize, and map mango fruits in orchards efficiently.

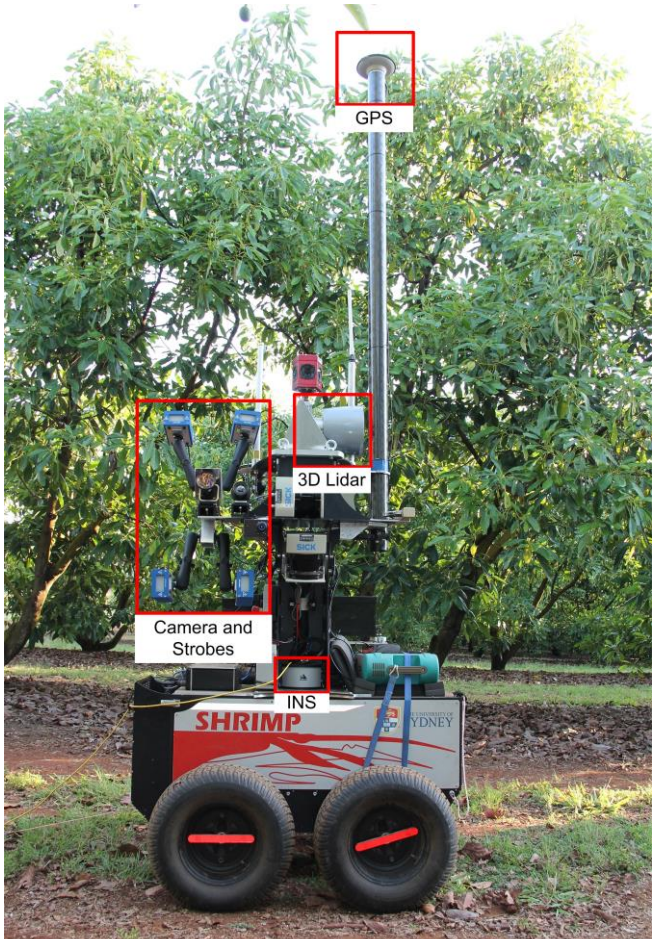

Figure 5 - Mango fruit detection robot developed by the Australian Center for Field Robotics

As illustrated in Figure 5, the electric vehicle is equipped with a color camera, stroboscope lights, a 3D Light Detection and Ranging (LiDAR) system, and a GPS/Inertial Navigation System (INS) suite. Using a multi-view approach, the system is able to track and distinguish each fruit autonomously without overcounting [11]. With the help of trajectory data provided by the navigation system, that relies mainly on a GPS/INS system, fruit pair-wise correspondences can be established between subsequent captured images. LiDAR components automatically create image masks for each canopy, allowing each fruit to be identified with its corresponding tree. The tracked fruit are then triangulated to locate them in the 3D space, allowing several spatial statistics per tree, row or orchard block.

1.3.2. Research projects within Europe

The Hortibot tool carrier was developed at the University of Aarhus in Denmark (Figure 6), which evolved from a remote-controlled mower and features a robust design [12].

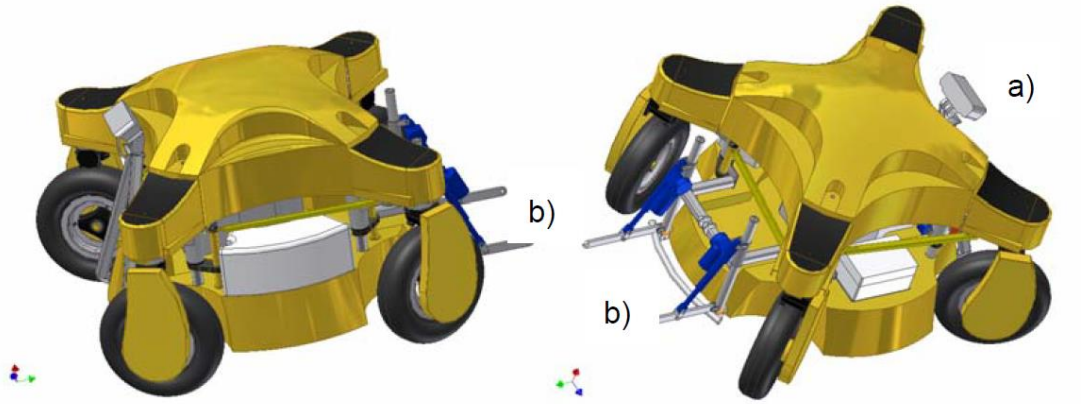

Figure 6 - Hortibot robotic platform developed by the Aarhus University

Through the use of a commercial row detection system developed by Agrocom Vision (formerly Eco-Dan Inc., Kvistgaard, Denmark), HortiBot can travel autonomously through several plots with visible rows, limiting the usage of GPS information.

The performances in farming operations of the HortiBot platform has been evaluated by Aarhus University and Vitus Bering University College Denmark in a feasibility study, by equipping the robot with weeding tools. Based on knowledge of horticulture, targeted performance was used to demonstrate the operation and the capabilities. The implemented tools for inter-and intra-row weeding consist of standard A-shaped hoes and bed ridges attached to each end of the implement toolbar. Using finger and torsion weeders, along with pneumatic nozzles, the five individual units carrying A-shaped hoes are used to weed intra-rows. In order to open and close the pneumatic nozzles, electronically controlled pneumatic valves are used. Computer vision algorithms are used to control both the autonomous navigation operations parallel to the rows and within them [13].

Wageningen University (The Netherlands) has an history of technologically advanced research project that have also produced several robotic platforms. In this overview, two of them are presented. Both were designed for weed control in open fields: the first was designed for arable crops, specifically maize crops, while the second was designed to control pasture weeds. A diesel engine powers both platforms, as they share the same four-wheels drive locomotion platform.

Through the use of mechanical weeding elements placed at the rear of the vehicle, the first vehicle is capable of autonomously controlling weeds in a cornfield by exploiting seeding navigation data coming from the planting tractor (Figure 7) [14].

Figure 7 - Maize weeding robotic platform developed by Wageningen University

Through a radio communication channel, a Real-Time Kinematic (RTK) correction signal is transmitted to the two GPS receivers onboard that exploit a base station to provide differential GPS (DGPS) for UGV autonomous navigation [15]. There are two parts to the platform control system: the high-level controller and low-level one. The high-level controller features with two Proportional-Integral (PI) controllers to minimize the lateral and orientation errors. The four-wheel angle parameters were determined by inversion of the kinematic model. The low-level proportional controller controls each wheel angle by exploiting a Smith predictor.

The second vehicle [16] also has a vertical axis weeding tool, that is in this case mounted on the front. Computer vision algorithms are used to detect the weeds by analyzing the camera images. A special arm holds the GPS antenna as well as the camera (Figure 8).

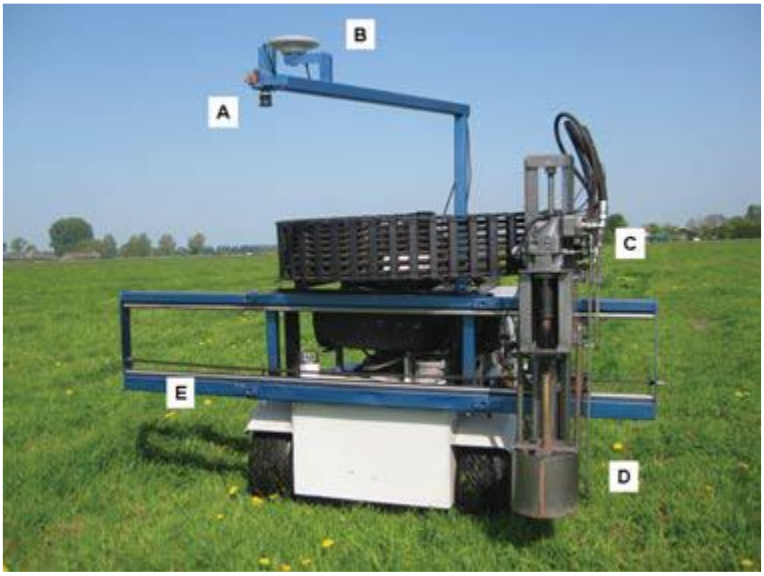

Figure 8 - Pasture weeds control robot developed by Wageningen University

The navigation system is basic, since its working environment is limited to meadows and pastures. Basically, the vehicle has to follow predefined trajectories. Security measures have been implemented to ensure the vehicle follows the imposed trajectory: it will stop if it loses the GPS signal. Navigation, path following, image processing, and actuation commands are handled by high-level controllers; sensors and hydraulics are handled by low-level controllers.

Aarhus University in Denmark created GrassBots [17], a UGV that harvests herbaceous substances to be used in biogas plants. It runs on two hydraulic tracks and is driven by a 74 kW diesel engine. The 3-metres wide platform is equipped with rotary encoders, an IMU and an RTK GNSS antenna for top-notch navigation: all done using open-source software designed for unmanned systems.

Figure 9 - GrassBots UGV developed by Aarhus University

This system was designed to operate with little or no human interaction and is capable of safely and efficiently covering all areas of the field on its own. A navigation algorithm defines parallel

lines (as a set of waypoints) to be followed on the field, as well as its boundaries, obstacles, line widths, driving direction, and end-of-field maneuvers.

Finally, the last analyzed UGV platform is the Armadillo from the University of Southern Denmark. The vehicle is a field robotic tool carrier that utilizes a versatile design to enable it to cater for a host of precision agriculture research projects. At around 425 kg, the two 18 × 80 cm footprint track modules are attached to one side of its interchangeable tool platform, adjustable in both width and clearance height. Each track integrates a 3.5kW electric motor, gear, and motor controller. For extended service, this machine is powered by a 48 V lithium battery capable of delivering 10 h of operation [18].

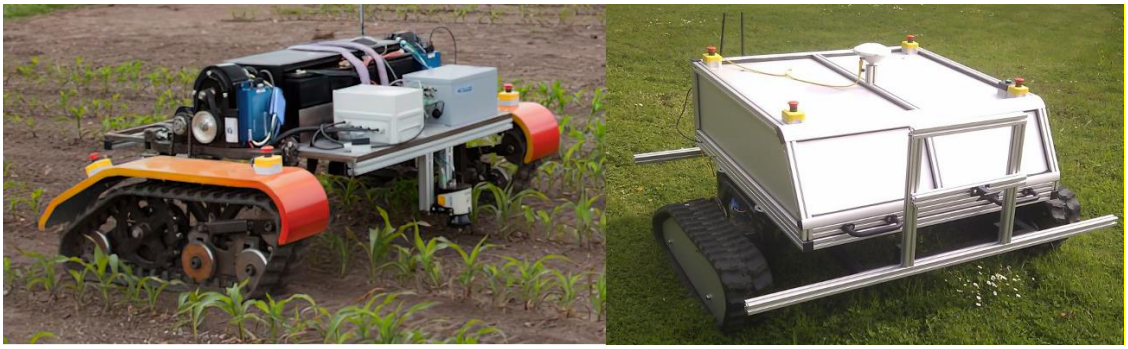

Figure 10 - Two vehicle fitting of the same Armadillo robotic platform developed by the University of Southern Denmark

A brushless motor hall sensor (625 ticks/meter) provides odometry feedback to the Armadillo robot, and a GPSnet.dk reference network is used to connect the IMU and RTK-GNSS receivers. In order to control Armadillo, an industrial dual-core PC is running the open-source ROS-based FroboMind software platform developed by the same university [19].

1.4. Motivation

Since 1960, the University of Bologna has been working enthusiastically to address the challenge of implementing automation for farming tasks, and thus developing autonomous navigation in agricultural machines. At that time, Professor Giuseppe Stefanelli led Pietro Bosi and Luigi Martelli to develop and build the first programmable unmanned tractor, called "BOPS-1960" [20]. However, the current range of autonomous ground vehicles differ in size and shape compared with standard or narrow track tractors; therefore, under existing European regulations [21], they cannot be classified as tractors. As a result, they typically feature light platforms limited to a maximum speed of 6 km/h and may include specially designed or commercially available agricultural implements.

This work aims to describe the new UGV developed within the Electrical, Electronic, and Information Engineering (DEI) department of the University of Bologna (UNIBO), from both a mechanical and software perspective. Details about the autonomous navigation algorithms implemented and mechanical features are provided, together with the design concepts that guided the robotic platform development. Finally, an overview of the precision agriculture applications enabled by using the robotic platform are reported, with the associated research activities accomplished.

The developed robotic platform was designed to specifically perform autonomous operations inside an orchard scenario. This environment was selected because of its complexity and scarcity

of widespread mechanization. Contrary to the vineyard's application, where also the high-value final product allows more investments by the farming companies, the fruit orchard environment often remains on the sidelines of technological developments that are instead being activated with regard to other areas, such as arable crops, in which mechanization is commonly used for a wide variety of applications.

Chapter 2. Robotic platform

In this chapter, the developed autonomous robotic platform is presented, together with the equipment and the sensor suite used. Also, a description of the customized implements is provided. The design concepts, human-machine interface and a system model are given, with some stability testing to prove the vehicle weight distribution.

The concept of using robots in agriculture is not new and has existed for quite some time. However, agricultural robots have just lately become more prevalent and are currently exploited for a range of jobs, such as mowing and spraying. Utilizing robots in agriculture has numerous benefits, including their capacity to navigate autonomously, their versatility, and their productivity.

One of the greatest benefits of agricultural robots is their capacity to roam the field independently. This allows them to work long hours without taking breaks, which boosts their productivity. Agricultural robots may also be outfitted with a range of accessories, making them versatile equipment that can be utilized for a variety of jobs. Some agricultural robots, for instance, may be equipped with GPS systems and sensors that enable them to avoid obstacles and apply pesticides or herbicides in a precise way.

In this context, the research work done within the CASY laboratory of the University of Bologna has produced a multipurpose autonomous ground vehicle that evolved from a hybrid vehicle to a full-electrified vehicle. The new platform features a modular implement mount that can support both legacy 3-point hitch tools and newly designed efficiency-friendly implements, so that the latter can take advantage of the electrical power storage of the machine and its electrical interface.

2.1. Prototype vehicle

The vehicle has been designed and developed using some guidelines in terms of modularity and size. As the different farming situations are highly unique, specially related to the soil in terms of ground flatness and possible obstructions, the locomotion platform has been selected to not rely on wheels, but instead to fit rubber tracks. In addition, the platform's dimensions must be compatible with existing orchards and farming activities within them. The platform is meant to target small/medium farms that constitute a significant economic and social entry market in terms of power and mechanical design. The resulting tiny and lightweight construction satisfies the requirement for a platform that is easy-to-maintain, cost-effective, efficient, easily transportable, and “low-weight” to enable field operations even in the face of harsh terrain. The rover utilizes the “weather independence” approach outlined in [22], augmented with an expandability concept based on replication of a basic “small” platform rather than making a single unit larger if more performance is required.

Adaptability and flexibility

The prototype operates by exploiting a core locomotion system that includes a rubber track and an electric motor to drive it. This single locomotion module can be flexibly combined by a mechanical frame that is specifically tailored to be adapted in terms of size and dimension. This concept can produce the simplest working layout made of two tracks in a *differential drive* or, more precisely, *skid steering* configuration. For instance, the inter-axis distance between the two tracks may be adjusted to fit any different row width. Using “plug-and-play” implements, the platform versatility is employed to complete demanding orchard management operations. The vehicle is also designed to be a "socket in the field": an easily available power source that

can provide energy to any interchangeable tools. Different implements can be attached according to the intended operation and working environment, by means of a common and simple data and power interface.

Scalability and Expandability

Although the mechatronic design is scalable to accommodate different situations in terms of size required, it naturally conforms to the notion of expandability through replication of identical simple and small modules. If a higher workload is required, the solution is to add more small-scale modules rather than create a larger one. This approach allows to maintain the fundamental platform's competitiveness (in terms of initial investment, maintenance, mobility, robustness, extra costs during expansion, etc.) while cloning it to boost work rates. The redundancy of platforms to enhance the precision of time-sensitive activities (such as spraying) or to deal with malfunctions and routine maintenance of single units is an additional significant advantage.

Versatility: Autonomous platform and "Facilitator"

The versatility of the rover to be employed in both fully autonomous scenarios and as a "facilitator" in situations where human employees are actively involved is another aspect. During operations like as shredding or spraying, the platform functions independently when fitted with the appropriate equipment and navigation algorithm. On the other side, the same platform may support harvesting labor if modified with the right instrument.

Implement design

Actual working platforms are superseded by the "motorized implement" concept, which replaces the oversized and overpowered traction system capable of *pulling* the implement with a new idea of a smaller platform that *carry* the implement. The ultimate objective is to optimize the entire platform in terms of weight, power efficiency, and mobility by integrating all the various components. Currently, two rover prototype-specific modified implements have been created. Both tools share an appropriate customized powertrain to drive the implements: the same mechanical interface may be fitted to both tools to facilitate a rapid tool change. The dedicated and personalized implements are:

i. *A branch shredder and lawn mower*: A commercial shredder/mower was modified to meet the rover's compactness in terms of power take-off. This enables shredding and mowing tasks in orchards, even on rainy or muddy ground.
ii. *An orchard sprayer*: A commercial multi-nozzle sprayer with turbine has been modified so that it may be installed directly on the implement support interface of the rover without the need for a separate tow trolley. The requested power is obtained using a belt with pulleys that give an appropriate reduction factor to reach the turbine's nominal speed. The nozzles include solenoid valves for selective spraying in accordance with the precise treatment map.

2.1.1. Mechanical design[2]

Static stability is essential for building an autonomous vehicle that must traverse an uneven terrain, such as that often observed in agricultural settings, especially when operating in hill areas with steep slopes. This is the reason why rubber tracks were selected over wheels: the larger contact area of crawler mechanisms enhances their capacity to adapt to uneven terrain, ensuring higher traction. In *Skid-Steering* vehicles, turning movements are performed by activating each side at a different speed or in a different direction, causing the tracks to slide, or *skid*, on the ground and forcing the robot to shear the terrain. Modeling these events is essential because it enables us to create a relationship between geometric vehicle design parameters, trajectory coefficients, and ground characteristics. For this reason, the vehicle design has been driven by the outcome of the pressure distribution model analyzed starting from the dynamical model of the vehicle.

Dynamical model

Tracked vehicles needs to shear and let caterpillars skid over the terrain while turning, hence the name Skid Steering Vehicles (SSVs). The effects of slip are analyzed and studied in detail and then exploited to obtain a model-based design tool for tracked vehicles.

Assuming a planar motion of the vehicle, the following kinematic model can be written with respect of an inertial reference frame:

$$\begin{aligned} \dot{x} &= V_G\, cos(\theta) \\ \dot{y} &= V_G\, sin(\theta) \\ \dot{\theta} &= \Omega_z \end{aligned} \qquad \text{Eq. 1}$$

where the $\dot{x}$ and $\dot{y}$ expresses the evolution of the position of the vehicle in inertial coordinates, according to the linear velocity V_G of the platform center of mass, expressed in body coordinates. This evolution can be computed by knowing the heading angle θ of the vehicle (in the inertial frame) and the angular turning velocity Ω_z. This model holds due to the non-holonomic feature of SSVs, as V_G and Ω_z can be computed using a simplified so-called unicycle model:

$$V_G = \frac{v_R + v_L}{2} = \frac{r}{2}(\omega_R + \omega_L)$$
$$\Omega_z = \frac{v_R - v_L}{2d} = \frac{r}{2d}(\omega_R - \omega_L)$$

In this representation, the ω_R and ω_L values are right and left motor speeds respectively. Similarly, v_R and v_L are the linear track speeds of right and left wheels (or tracks). r is the wheel radius, and $2d$ is the track width dimension of the robot.

To keep into account the tracks slippage typical of SSVs during operations, the most common model is the efficiency reduction one, as seen also in [23]:

[2] This work has contributed to the published paper: Design Concept and Modelling of a Tracked UGV for Orchard Precision Agriculture. - Tazzari, R., Mengoli, D., Marconi, L. - 2020 IEEE International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2020 - Proceedings, 2020, pp. 207–212, 9277577

$$V_G = \frac{v_R\left(1 - i_R(t)\right) + v_L\left(1 - i_L(t)\right)}{2} = \frac{r}{2}\left(\omega_R\left(1 - i_R(t)\right) + \omega_L\left(1 - i_L(t)\right)\right)$$
$$\Omega_z = \frac{v_R\left(1 - i_R(t)\right) - v_L\left(1 - i_L(t)\right)}{2d} = \frac{r}{2d}\left(\omega_R\left(1 - i_R(t)\right) - \omega_L\left(1 - i_L(t)\right)\right) \quad \text{Eq. 2}$$

In this representation, two terms $i_R(t), i_L(t) \in (0,1), \forall t \in \mathbb{R} \geq 0$ are added to the formula to model the time-varying slipping coefficients associated to the right and left tracks, respectively.

A non-linear relationship exists between i_R, i_L and v_R, v_L [24], [25], but this is limited to small steering radiuses and a complete general relationship that is also including the soil properties is still missing. In general, the slip coefficients can be expressed as:

$$i = 1 - \frac{V_t}{v_i} = 1 - \frac{V_t}{\omega r}$$

where i denotes a single track, so that v_i is the ideal track speed given by ω and r that are motor speed and wheel radius respectively. The V_t quantity denotes the actual ground/track contact point speed that is hardly measurable. So, the expression is able to relate track speed with the slip coefficient, but any soil parameters dependency is missing.

The traction force of the track with respect of the soil can be used, so taking into account physical terrain properties[26], [27]. The formulation is as follows:

$$F = F_{max}\left[1 - \frac{K}{il}\left(1 - e^{-il/K}\right)\right] \quad \text{Eq. 3}$$

In this case, l is the length of the contact surface between the track and the terrain, K is the soil shear deformation modulus and F_{max} is the maximum traction force that can be developed by the track. This force can be expressed as:

$$F_{max} = S_c \tau_{max} = S_c(c + p\ \tan(\varphi)) = S_c c + \frac{W}{2}\tan(\varphi)$$

where S_c is the track-soil contact surface, τ_{max} is the maximum shear strength of the soil, c is the apparent cohesion of the soil and φ is the angle of internal shear strength of the soil. p is the normal pressure acting on the track, and W is the total weight of the vehicle. By inverting Eq. 3, it is then possible to derive the slip coefficient of the track, given the soil parameters and the tractive forces applied by each one of them. Also in this case, the relationship found is incomplete as it does not relate both track speeds and soil characteristics into a single equation.

In order to develop the dynamic analysis of skid-steering tracked vehicles, it has been followed the work produced by Steeds [28], Bekker [26], [29] and Wong [27], [30]. Al-Milli [31] then extended it to handle the softness of the terrain and Willis [32], Keller [33] and Liu [34] have inspired the developed non-uniform Normal Pressure Distribution model.

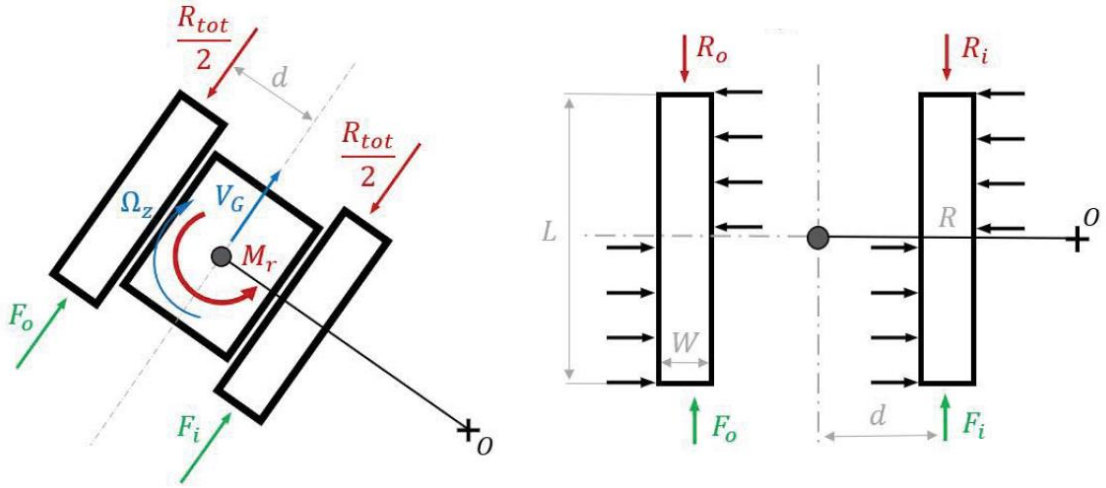

Figure 11 - Forces acting on a skid steering vehicle

The model of the forces acting on a tracked vehicle during the planar steering maneuver (Figure 11) is given by:

$$\begin{aligned} m\ddot{s} &= m\dot{V}_G = F_o + F_i - R_{tot} \\ J_z\ddot{\vartheta} &= J_z\dot{\Omega}_z = (F_o - F_i)d - M_r \end{aligned} \qquad \text{Eq. 4}$$

where s and V_G are the linear displacement and velocity of the Center of Gravity (CoG), and z denote the angular displacement and velocity around the center of rotation O (z-axis). m and J_z are the linear and angular inertia, respectively. The motor thrust forces generated by the inner and outer track are F_i (inner) and F_o (outer). R_{tot} and M_r are respectively the resistive force and torque, while, as already seen, d is half the distance between the longitudinal centerlines of the tracks.

Considering the above Eq. 4, when applying a stationary turning maneuver, it is possible to write the forces of the two tracks following Wong[27], [30] characterization:

$$\begin{aligned} F_o &= \frac{R_{tot}}{2} + \beta\omega_o + \frac{M_r}{2d} = \frac{f_r mg}{2} + \beta\omega_o + \frac{M_r}{2d} \\ F_i &= \frac{R_{tot}}{2} + \beta\omega_i + \frac{M_r}{2d} = \frac{f_r mg}{2} + \beta\omega_i + \frac{M_r}{2d} \end{aligned} \qquad \text{Eq. 5}$$

In this definition, f_r is the rolling friction coefficient along the longitudinal direction of motion, while β represents the viscous friction coefficient due to the damping effects of the mechanisms connected to the motors. The gravitational acceleration is g. The turning moment M_r depends on the footprint of the track on the ground and, clearly, on the terrain parameters that may affect friction. Considering the uniform normal pressure distribution along the track and imposing the lateral resistance per unit length of the track equal to $R_L = \mu mg/2L$, it can be obtained the resistant moment of rotation as:

$$M_r = 4\mu_t \frac{mg}{2L}\int_0^{L/2} x\,dx = \mu_t \frac{mgL}{4} \qquad \text{Eq. 6}$$

where μ_t represent the lateral friction coefficient and L is the length of the track contact surface. By applying these results, the Eq. 5 can be rewritten as:

$$F_o = \frac{f_r mg}{2} + \beta\omega_o + \mu_t \frac{mgL}{4 \cdot 2d} \approx \frac{mg}{2}\left(f_r + \mu_t \frac{L}{4d}\right)$$
$$F_i = \frac{f_r mg}{2} + \beta\omega_i + \mu_t \frac{mgL}{4 \cdot 2d} \approx \frac{mg}{2}\left(f_r + \mu_t \frac{L}{4d}\right) \quad \text{Eq. 7}$$

The above Eq. 7 has no dependency of F_o or F_i, neither about the actual trajectory (V_G, Ω_z and steering radius R'). Also, vehicle design parameters are not considered (such as CoG position, number of rollers in the tracks, …).

However, the motor thrusts are strongly dependent on these parameters, as empirical evidence have demonstrated: decreasing the turning radius R' lead to an increase in M_r. In fact, the effect of those parameters, as it will be detailed later, affect the coefficient of friction μ_t. Incorporating the above listed parameters into the expressions of F_o and F_i is therefore a crucial aspect.

According to [28], the shear stress along the track-ground contact surface abides by the Coulomb law of friction. This means that the resulting shear stress on a track element reacts in opposition to the relative motion between it and the ground. However, experimental data presented in [27] demonstrates that it is actually continuous in relation to shear displacement (Figure 12).

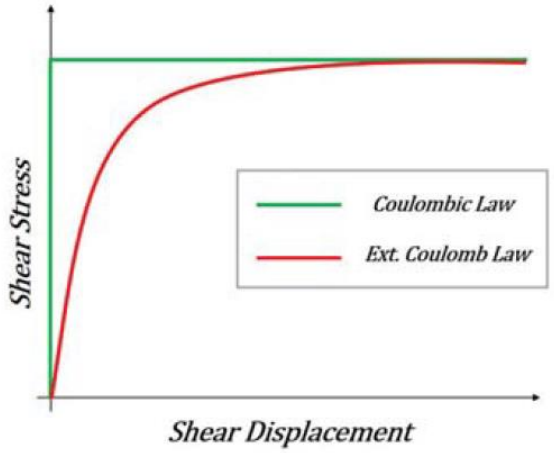

Figure 12 - Comparison between the standard Coulombic friction model and the Extended Coulombic friction model

So, it is more accurately approximated as an extended coulombic friction law of type:

$$\tau = (c + \sigma\mu)\left(1 - e^{-j/K}\right)$$

where τ is the shear stress, σ is the normal pressure, μ is the friction coefficient between the track and the soil and j is the shear displacement. With K is denoted the shear deformation modulus: it cannot be zero as with $K = 0$ a Coulombic friction model law will be obtained. The c term is a soil cohesion factor, and it was introduced in [31] to take into account both the soft and firm terrains. σ is assumed to be constant along the whole track contact surface, according to [27]. This condition, as it will be shown later, can be relaxed in order to take into account also a more realistic normal pressure distribution.

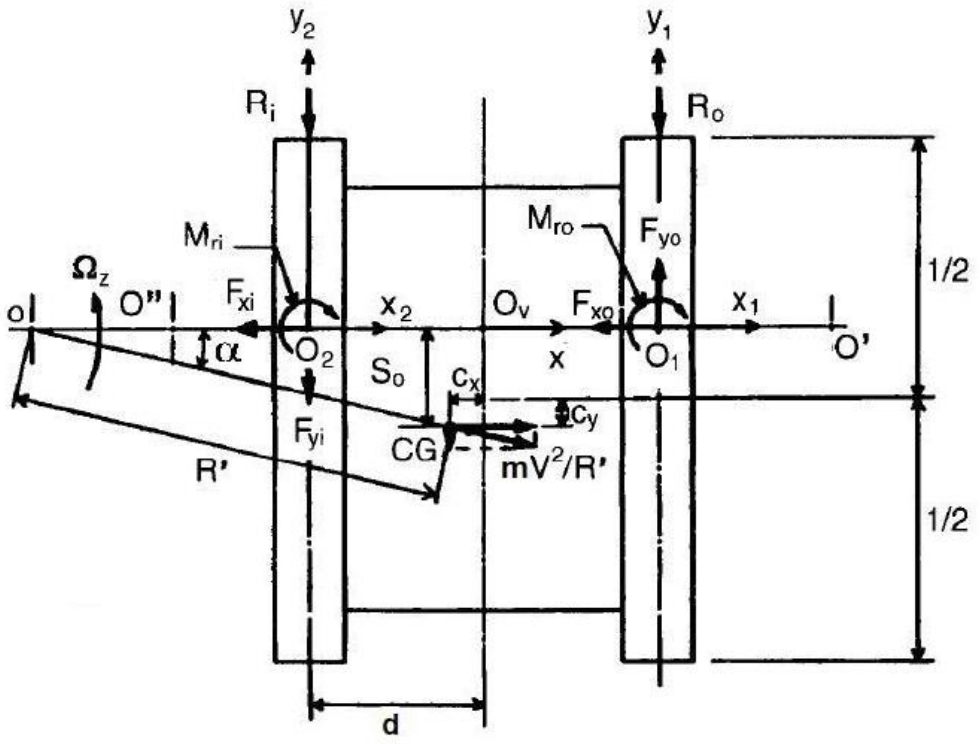

Figure 13 - Forces involved during a steady state turn within an SSV

The moments M_{r_o} and M_{r_i} depicted in the Figure 13, and caused by the lateral shear forces with respect of the two tracks O_1 and O_2, can be calculated as:

$$M_{r_{o,i}} = -\int_{-L/2+c_y-s_O}^{L/2+c_y-s_O}\int_{-W/2}^{W/2} y_{1,2}\left(c+\sigma_{o,i}\mu\right)\left(1-e^{-j_{o,i}/K}\right)\cos\delta_{1,2}\,dx_{1,2}\,dy_{1,2}$$

$$\cos\delta_{1,2} = \frac{-y_{1,2}\Omega_z}{\sqrt{\left(\left(R''+d+c_x+x_{1,2}\right)\Omega_z - r\omega_{o,i}\right)^2 + \left(y_{1,2}\Omega_z\right)^2}}$$

In which L and W are the length and width of the tracks contact surface (both inner and outer, assuming that both tracks are geometrically identical). The lateral and longitudinal distances between the CoG and the lateral and longitudinal axes of the vehicle frame are indicated by c_x and c_y respectively. s_O is the center of turn translation due to the slippage effect, wheel gear radius is r, and $(x_1, y_1), (x_2, y_2)$ are the coordinates of a generic point on the outer and inner track, respectively. Still considering outer and inner tracks, σ_o, σ_i are the normal track pressures involved, j_o, j_i are the shear displacement due to the tracks, and the angles δ_1, δ_2 are determined by the lateral directions of the points on the outer/inner tracks and the resultant sliding velocities. $R'' = \sqrt{R'^2 - s_O^2}$ represent the lateral CoG distance from the center of rotation.

Ultimately, the coefficient of lateral resistance μ_t can be computed by using $M_r = M_{r_o} + M_{r_i}$ from the Eq. 6 as:

$$\mu_t = \frac{4\left(M_{r_o} + M_{r_i}\right)}{mgL}$$

that is, in this case, a function of $W, L, c_x, c_y, V_G, \Omega_z, R', \mu, c$. So, the μ_t parameter is now expressed as a function of the vehicle design parameters W, L, c_x, c_y, the trajectory parameters V_G, Ω_z, R and the soil properties μ, c.

Gaussian Normal Pressure Distribution model

Assuming a uniform normal track pressure distribution, will not enable to account for the effects of soft grounds or the use of a different number of track rollers, which is one of the relevant design parameters, as involves the maneuverability of the vehicle. Therefore, assuming that the pressure distribution is a function of the longitudinal coordinate of the track (y), while remain uniform on the transversal coordinate (x), the sinusoidal case was proposed in [32] as:

$$\sigma(y) = \frac{m_h g}{A}\left[1 + \cos\frac{2n\,\pi\,y}{L}\right]$$

Here, $m_h g$ is half of the vehicle weight, considering one track, as $m_h = m/2$, and A is the contact surface of the track with respect to the ground. Given N the total number of track rollers, n is defined as $n = N - 1$, and y represents the longitudinal coordinate along the length (L) of the track. In this case, the pressure peaks values are independent on the number of rollers or their radius, which is quite unrealistic. Therefore, the proposed solution uses a Gaussian Normal Pressure Distribution (GNPD) model, composed of the sum of N Gaussian function, each centered in the mid-point of each roller, as follows:

$$\sigma(y) = \sum_{i=1}^{N} \frac{m_h g}{nA} \cdot \frac{1}{r_\omega\sqrt{2\pi}} \cdot e^{-\frac{(y-l_i)^2}{2(r_\omega L)^2}}$$

Where the y-coordinate of the wheel mid-point is denoted by $l_i = L/n(i-1)$, and the radius of the roller is represented by r_ω. A comparison between the three different approaches can be seen in Figure 14. As a result of the formula, peak values change with the number of rollers, resulting in lower peak values when the number of wheels is higher. Furthermore, increasing the variance of the Gaussians (by increasing r_ω), the Gaussian functions can be overlapping, resulting in nonzero pressure even far outside the rollers center positions. This can be used to model soft terrain and grouser sinkage. So, the firmness of the terrain can be considered, by rewriting the equation as:

$$\sigma(y) = \sum_{i=1}^{N} \frac{m_h g}{nA} \cdot \frac{1}{r_\omega k_f\sqrt{2\pi}} \cdot e^{-\frac{(y-l_i)^2}{2(r_\omega k_f L)^2}}$$

where the added k_f term, parametrizes the firmness coefficient of the soil.

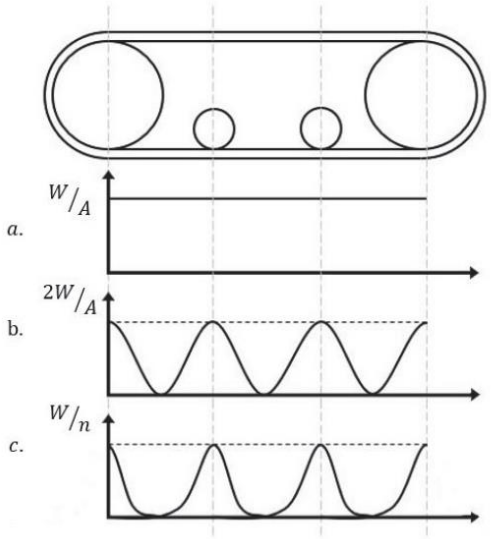

Figure 14 - Comparison between normal pressure distribution models: uniform (a); sinusoidal (b); Gaussian (c)

Using this pressure distribution, it is possible to recalculate τ and $M_{r_{o,i}}$ by considering the pressure variables σ, σ_o, σ_i not constants anymore, but instead as a function of the y longitudinal coordinate: $\sigma(y), \sigma_o(y), \sigma_i(y)$. By doing so, also M_{r_i} and M_{r_i} expression will become functions of the number of track rollers N. This is affecting track design and vehicle dimensions by analyzing the effect on μ_t by changing some parameters, such as:

- Vehicle total mass (m) or normal pressure (σ),
- Vehicle track width ($2d$)
- Dimensions of the tracks in terms of length and width (L, W)
- Number and radius of the track rollers and wheels (N, r_ω).

Vehicle design and tracks design

According to the concepts described in the Section 2.1 intro, the vehicle is conceived to be smaller and lighter with respect to traditional tractors, even smaller ones.

In order to enable on-the-go small adjustments, aluminum modular bars were used to build the vehicle frame as a prototype. As a result, the total weight of the final vehicle has not been fully optimized. Conversely, the CoG was instead carefully calculated to be as close as possible to the tracks' longitudinal midpoint and close to the lower plane of the frame. However, once the design of the platform will be finalized, a soldered frame can be used to optimize the weight. To take advantage of the analysis done previously, it is necessary to resolve the $M_{r_{o,i}}$ equations. These was solved numerically along the contact surface of the track, so obtaining:

$$M_{r_{o,i}} = -\int_{-L/2+c_y-s_O}^{L/2+c_y-s_O}\int_{-W/2}^{W/2} y_{1,2}\left(c+\sigma_{o,i}\mu\right)\left(1-e^{-j_{o,i}/K}\right)\cos\delta_{1,2}\,dx_{1,2}\,dy_{1,2}$$

$$\approx -\sum_{-L/2+c_y-s_O}^{L/2+c_y-s_O}\sum_{-W/2}^{W/2} f(\Delta L, \Delta W)\Delta A$$

Where $\Delta A = \Delta L\,\Delta W$ is the finite discretization of the contact track surface with the ground.

Since the mass determines the normal pressure, which, in turn, generates friction, keeping the vehicle mass as small as possible minimizes the depth of ruts caused by vehicle passage, as well

as reducing the overall resistive torque. It can be estimated that performing the appropriate optimization, the mass m of the vehicle may vary from 350 kg (empty weight, without implement attached) to 750 kg (full water tank, sprayer attached).

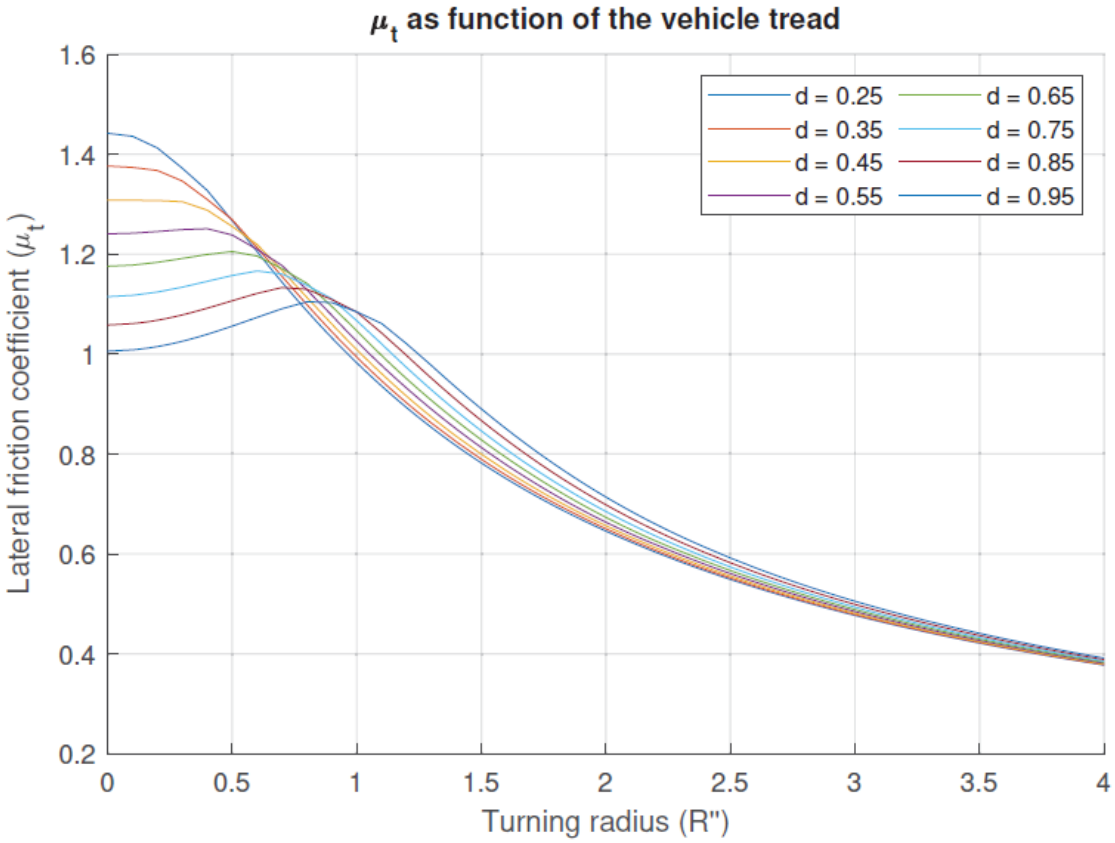

Figure 15 - Comparison of different values of d in terms of μ_t variation with respect to the turning radius

The above Figure 15 shows the impact of the track width design parameter, using half the track-to-track distance d. As can be seen by the graph, increasing the turning radius R'', the sensitivity with respect to d decreases, as the lateral friction coefficient tends to reach zero value, therefore the impact of the chosen width is minimized until it becomes irrelevant. The peak in the functions is reached when the turning radius is equals to d. This is explained by the fact that in this condition, the inner track is actually blocked, so creating the maximum slippage level with respect to the other values of R''. Using the simulations, the vehicle will tend to have the maximum allowable d value, so to minimize the friction coefficient μ_t and therefore increasing efficiency and maneuverability, but this dimension is constrained by the orchard dimension and vehicle size and weight to be compliant with the proposed concepts, so the chosen d value was selected at 0.5m, to ensure a proper tradeoff with size, and maneuverability among the orchard fields by ensuring enough margin for the maneuvers.

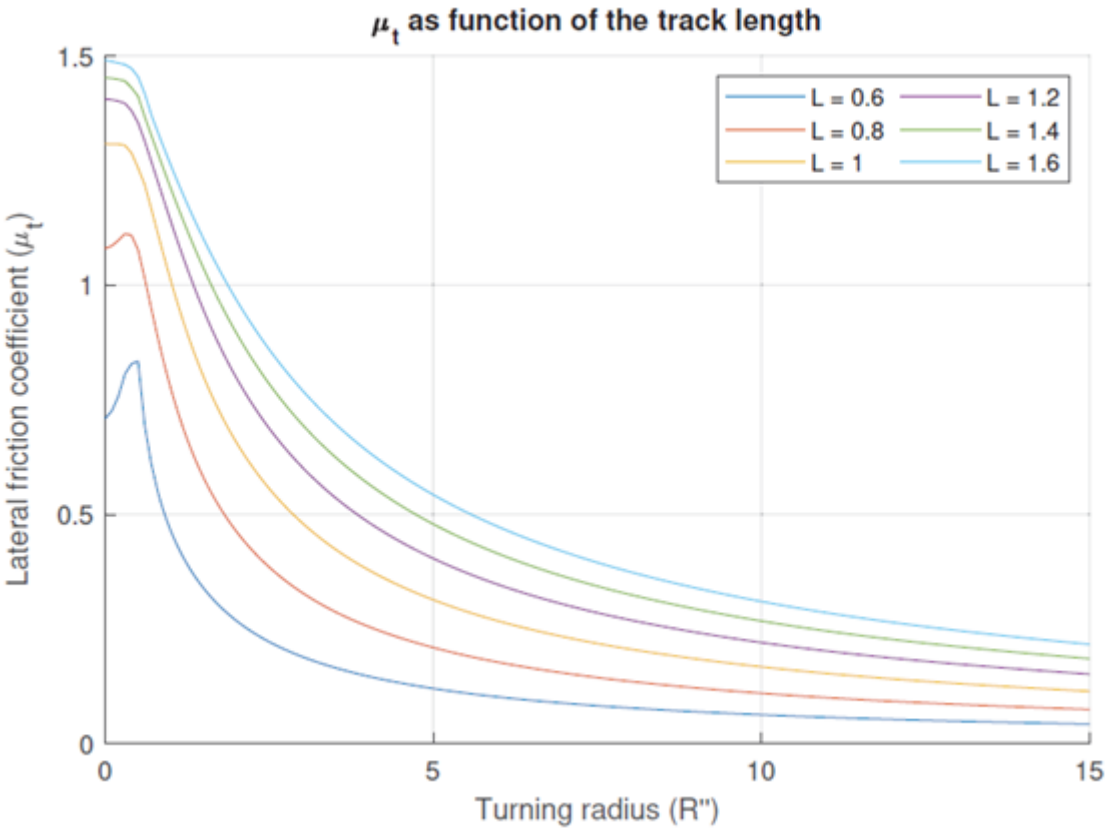

Figure 16 – Comparison of different values of track length L in terms of μ_t variation with respect to the turning radius

With respect to the track design and dimensions, one of most important parameters is the track length L, since (as it will be described in Section 2.3) has an impact on the static stability of the vehicle, together with d. Figure 16 shows the impact of the selected track length value L with respect of lateral friction coefficient μ_t and turning radius, similarly as what is computed above for the track-to-track distance. Excessively long tacks are expected to have a significant lateral friction coefficient even with high turning radii, while, on the other side, very short tracks may affect too much the vehicle longitudinal stability, acting similarly to having wheels. Also, having shorter tracks may lead to shorter vehicle frames with reduced space and volume to store tools or inputs. A good compromise between those aspects seemed to be the choice of a track length L equals to 1m. As the single-track width W has not proven to create significant changes in the μ_t coefficient, the choice has been made by selecting commercial off-the-shelf products with a width comparable in size to the overall vehicle dimensions. In the prototype platform case, W was selected equal to 0.3m.

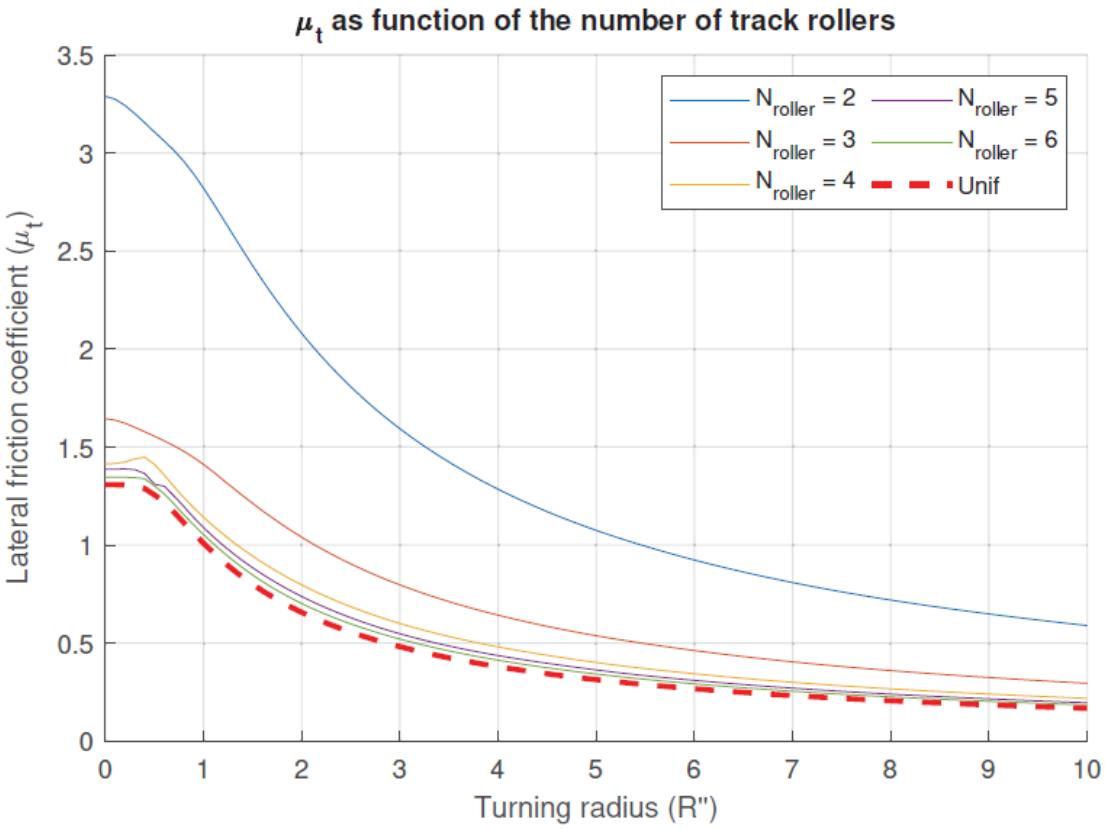

Figure 17 - Comparison of different values of track rollers N in terms of μ_t variation with respect to the turning radius. The radius r_ω was fixed at 7.5cm.

The next relevant design choice was the number of wheels (or rollers) inside the track. These are used to distribute the pressure along the whole track contact surface, thus trying to create a homogeneous uniform normal pressure distribution along all the track length. According to the simulations depicted in Figure 17, the best tradeoff between complexity (e.g., high number of rollers) and uniform distribution similarity, was selected to a number of rollers N equals to 4 for the prototype. The roller size r_ω was chosen to 0.075m for practical reasons.

Experimental results

Using the robotic platform prototype that has been build, it was possible to check the results that lead to the design choices on a real vehicle setup, by comparing real motor speeds and torques. To assess the accuracy of our setup, different tests were conducted along circular paths. Constant reference speed has been sent to the motors, which enabled the vehicle to traverse circular trajectories with three distinct radii: $R_1'' = 2m, R_2'' = 4m, R_3'' = 7m$. Each test was repeated clockwise (CW) and counterclockwise (CCW), in order to minimize the effect of asymmetries on the experimental platform. It is important to note that these are ideal turning radii computed from the kinematics of a differential wheeled robot, excluding skidding effects.

Based on the characterization detailed in [27], heavy clay soils are described by the following parameters: $f_r = 0.3, \mu = 0.35, K = 0.05m, c = 3.45kPa$. This is the actual soil type that best represents the terrain characteristics of the test field used for the trial. In the Figure 18 below, the planar trajectories performed for the three radii are depicted, together with the track forces, to compare measured forces with model estimated ones.

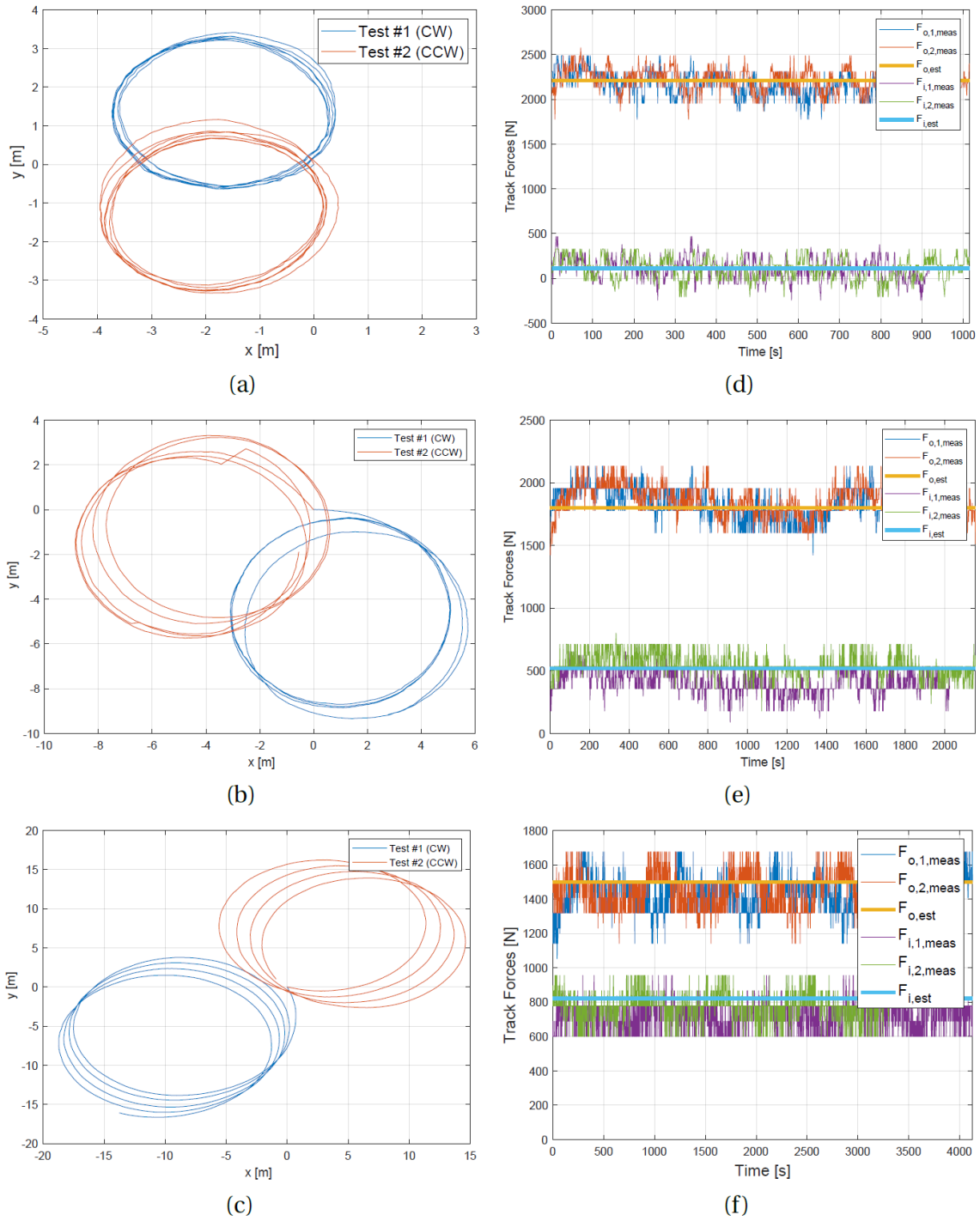

Figure 18 – On the left column are presented 2-dimensional graphs for the trajectories performed for: 2m radius (a), 4m radius (b) and 7m radius (c). On the right column the comparison between estimated track forces for clockwise turning and counterclockwise turning and model estimated one is presented for the three cases: 2m radius (d), 4m radius (e) and 7m radius (f).

Based on these comparisons, the model estimates real motor forces in a good approximation. More information about Root-Mean-Square (RMS) estimation errors can be found in TABLE I, which indicates that the data is approximated well, but there is a greater deviation (in percentage) from real values when the forces have a small absolute value, as it happens with small turning radius. In this case, the inner track produces a force that is close to 0 N, in which case the measurement noises increase, reducing the accuracy of the estimation.

Table 1 - Root Mean Square Error (RMSE) between real data and model estimation

Turning radius R''		RMSE (N)	RMSE (%)
2m	F_o	114.7	5.2
	F_i	91.9	80.7
4m	F_o	98.2	5.4
	F_i	88.2	17.2
7m	F_o	88.9	6
	F_i	83.6	10

2.1.2. Implement mechanical interface[3]

The customized implements are attached to the vehicle by means of a compact power take-off mounting frame that was specifically designed and built to fit the size specifications of the prototype.

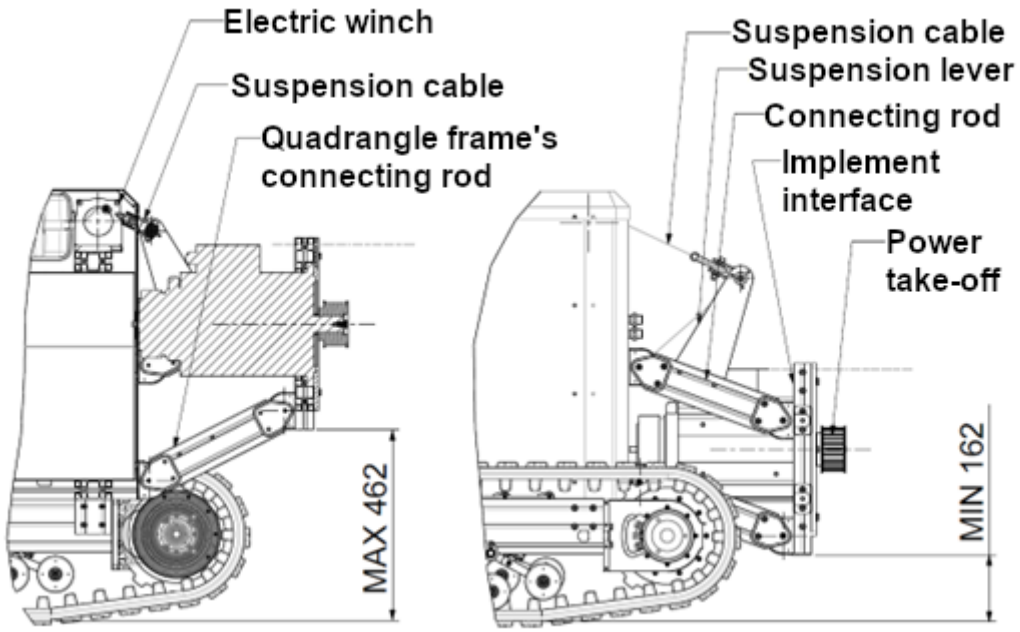

Figure 19 - Technical drawing representing the implement mounting mechanism

The implements are attached to the tractor by a pantograph connecting rod system. The front plate is equipped with a standard three-point hitch system that allows quick attachment of any implements having such an interface. The height of the implements relative to the ground is adjusted with a simple and inexpensive electric winch. The inclination of the implements' attachment plate can be changed by moving the eye of the lower or upper connecting rods with a lever centered on the body or plate.

[3] This work has contributed to the requested patent: Sistema di sollevamento di un veicolo agricolo per agricoltura di precisione e relativo veicolo agricolo. - Applicants: Alma Mater Studiorum – Università di Bologna - Inventors: Marconi Lorenzo; Mengoli Dario; Sala Andrea; Tazzari Roberto - Filing date: 07/09/2021

In the usual practice, the implements are connected to the tractor's Power-Take-Off (PTO) via a universal joint that transmits motion while allowing adjustment of the implements' working height. In some applications, such as turbines, a speed gear able to increase the rotation is placed downstream of the coupling because the PTO turns at a lower speed than is necessary for the implements to operate. Sometimes the coupling poses a danger to the operator if not properly maintained. In the proposed solution, the electric or endothermic motor that drives the rotation mechanical parts of the implements is attached to the same plate to which the implement is connected (Figure 19), and the PTO acts directly within it. The speed of the motor can be chosen independently of that of the traction motors, and this makes it possible to eliminate all mechanical speed-change devices and/or multiplication/de-multiplication systems. By doing so, the weight of the transmission can be reduced, while the efficiency is increased. Since there are no exposed moving parts, the solution is also safer for the operator.

According to the schema proposed, the implements can be moved indifferently by an electric motor or an endothermic motor. This is an advantage in commercial terms because two solutions having complementary characteristics can be provided.

The endothermic combustion engine solution can have the following advantages:

- It is more economical as it does not require the presence of high-capacity batteries, replaced by a simple tank.
- It can work on several consecutive shifts by simply filling the fuel tank.
- It can be easily integrated with the vehicle's automatic control system thanks to modern electronic injection control units.

The electric motor solution can have the following advantages:

- More sophisticated management of the implement and thus of the work being performed thanks to the electronic engine control system, which can be adjusted in torque or speed (revolutions per minute – rpm).
- It can be easily integrated with the vehicle's automatic control system.
- It is quieter with respect to conventional internal combustion engines.
- It gives less vibration compared to gasoline engines.
- It has no CO2 emissions and therefore can be used in closed environments such as greenhouses.
- It has a cheaper energy supply cost compared to petrol or diesel fuel.
- The energy supply chain can be more direct and controlled if it is self-generated, such as through photovoltaic panels.

The implement plate mount has also been designed to facilitate tool attach by providing a customized multipurpose pulley that can drive both belt-driven implements (such as the sprayer) and shaft-driven implements (such as the mulcher). The mechanical fixing on the frame has been secured by the help of a reduced number of bolts that ensure rapid tool changing within the platform.

The height of mounting platform interface can also be adjusted in height maintaining its vertical orientation, thanks to the articulated parallelogram mechanics. The whole system is supported and lifted using an electric winch by means of a stainless-steel rope that is attached an appropriate support on the interface frame.

The described implement interface has been developed by mimic the standard PTO of conventional tractors, thus minimizing the effort of adapting off-the-shelf traditional implements. To best fit on the developed vehicle, the available implements have been revised using a compacted design and minimizing their cantilevered size.

A simplified implement interface will be described in the next Section 2.2.1 where the prototype mechanical improvements are discussed.

2.1.3. Locomotion

The locomotion module of the vehicle has been designed to maximize ground footprint while maintaining a high grade of simplicity. As we'll see later, this has been then revised in the new prototype version, as some complexity and versatility with respect to the tracks were needed to improve vehicle guidance. The original design was composed of a single 1.5kW electric brushless motor per track with 1:32 reduction gearbox over a 14 teeth 185mm track drive gear, providing a maximum of 20 Nm of peak torque. As the motor features a maximum speed of 2500 rpm using a 48 V power supply, this configuration enables a maximum speed of 6 km/h of the vehicle at full throttle. Since some maneuverability is needed to ensure proper steering capabilities, the nominal cruise speed of the platform was reduced at 5 km/h. This velocity is compliant with the current regulations about autonomous or remote-controlled vehicles that enforce a human supervisor in the nearby that should be able to rapidly reach the robot to stop it in case of some unexpected situations arise. Each motor is also equipped with a 7 Nm stationary brake that can be useful to hold the vehicle on slopes and for parking safety measures. As the locomotion power consumption has been experimentally proven to be about 1.0kW while travelling on flat land, the expected operational time on the provided 6kWh batteries is up to 6 hours of continuous usage. Of course, the vehicle operability will be eventually reduced in accordance with the actual working conditions.

Brushless motor controllers

A Sevcon (now BorgWarner) motor speed controller (Gen4-Size4) manages the three motor phases using a PMAC (Permanent Magnet Alternate Current) controller with encoder input. Motor electrical information is configured in the driver and data from the sin-cos encoder is used to synchronize the phase current with the motor rotation. The motor controller communicates using a CANBus interface and can be used both in torque or speed mode. For the vehicle and project purposes, the speed mode was selected. This allows to handle velocity setpoints for each one of the two tracks to control the overall speed of the vehicle and the turning rate, without the need to close a velocity control loop on a higher-level controller. Conversely, a torque setpoint could be useful to synchronize multiple motors driving the same track (e.g., to increase available torque) or to give the operator a more throttle-like feeling while driving the vehicle. Since the whole platform was conceived to be mainly autonomous (and exploiting one motor per track), it seems more relevant to handle track velocity setpoints and eventually monitoring the commanded torque to adjust the speed while traversing steep slopes. Available commands include throttle value, enable switch and speed direction. Available telemetry includes actual speed, commanded torque, voltage and current information, drive status and fault codes.

Figure 20 - Sevcon motor controller

2.1.4. Sensors

The sensor suite mounted onboard is selected to enable both waypoints navigation, orchard in-row navigation, obstacle avoidance and safety requirements. The whole set is described as follows.

LiDAR sensor

LiDAR (Light Detection and Ranging) sensors measure the distance to a given point by emitting laser beams and recording their reflections from the surfaces around it. The resulting point cloud can then be used either for mapping purposes, localization, or obstacle avoidance algorithms.

Figure 21 - 3D LiDAR pointcloud example

The Velodyne VLP-16 Puck sensor was selected for the vehicle, as it provides some vertical information while maintaining a 360° field of view in a sufficiently dense point cloud for an entry-level not expensive sensor. As 3D LiDAR devices may have a price tag of several tenths of thousand euros, the cost factor is something that needs to be under control while developing a

potential commercial platform. Data collected by this sensor is used mainly for the in-row orchard navigation, thus exploiting the Hough Transform algorithm for tree rows detection.

Figure 22 - Velodyne VLP-16 Puck LiDAR sensor

GNSS Receiver

The primary function of GNSS receivers is to provide an accurate real-time location relying on many satellite constellations. High-grade GNSS receivers may implement RTK (Real Time Kinematic) and DGPS (Differential GPS) correction. High-precision measurements are no longer confined to big-name companies, allowing customers and OEM providers to consider more cost-effective devices without sacrificing data quality. Position estimation above Earth surface is obtained by the so called trilateration algorithm that combines together multiple satellite ranging measurements into one small position estimation area.

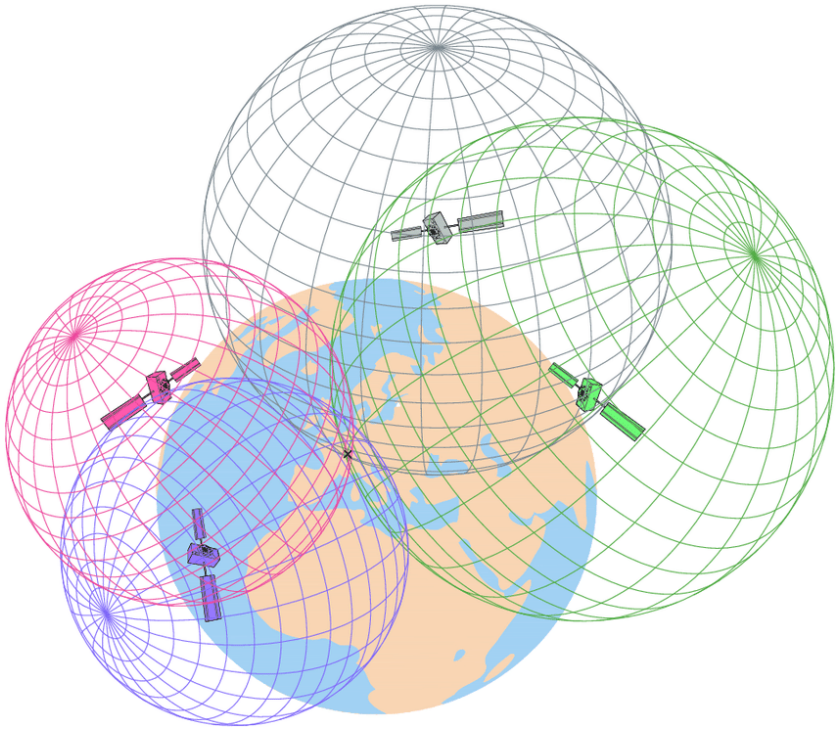

Figure 23 - Position computation using trilateration [35]

The GNSS receiver chosen for the robotic platform was the Trimble R8s receiver that features 440 channels over L1/L2 GPS/GLONASS frequencies, Galileo and BeiDou constellations. In order

to achieve sub-meter accuracy (up to 20cm of horizontal precision), a RTK improvement coming from existing topography network is enabled and correction information is exchanged using a built-in GPRS modem.

IMU

Using a combination of Gyroscope, Magnetometer, and Accelerometer, the IMU (Inertial Measurement Unit) sensor is able to compute and report the body linear forces, angular velocity, and body direction. It has been shown that when combined with GPS systems, the accuracy of navigation for UGVs can increase significantly.[36] The raw IMU data is often used in an Attitude and Heading Reconstruction System (AHRS) algorithm that is able to provide reliable attitude and heading estimation of a rigid body.

Inside the prototype, a XSens MTI-G-710 IMU sensor is used that is mainly exploited to obtain a reliable heading information for waypoint navigation, since GNSS position estimation is provided by a dedicated module.

Figure 24 - XSens MTI-G-710 IMU/GNSS Module

Motor Encoders

A motor encoder is a device that converts the rotational motion of a shaft or axle into digital or analog signals. Additionally, it can be used to measure the speed, position, and direction of the shaft or axle. Encoders typically employ optical, magnetic, or inductive sensing to detect shaft rotation without physical contact. Absolute and incremental encoders are the two most common types of encoders. Absolute encoders generate a unique code for every shaft position, making them ideal for applications in which the exact shaft position must always be known. Incremental encoders generate pulses that can be counted to determine the shaft's position, making them ideal for applications in which absolute positioning is not required. Typical automotive application motors use sin-cos type encoders, that provides accurate shaft position estimation at low cost. The information of speed computed by using encoder data can then be transferred to the control software application via the telemetry channel. Motor speed can then be used inside a Kalman filter to improve localization estimation.

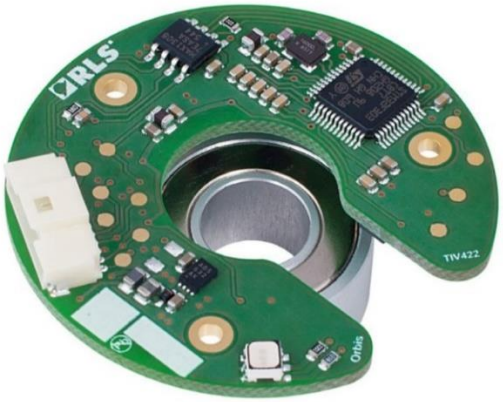

Figure 25 - Rotary encoder

2.1.5. Computational architecture

The logical architecture managing both hardware and software components of the robot is designed to provide optimal performance and reliable operation, even in the most demanding environments. It is divided into two parts, one for base low-level locomotion and I/O management, and the other one dedicated to the high-level autonomous navigation, sensor fusion and data processing.

The overall architecture can be represented in the following diagram.

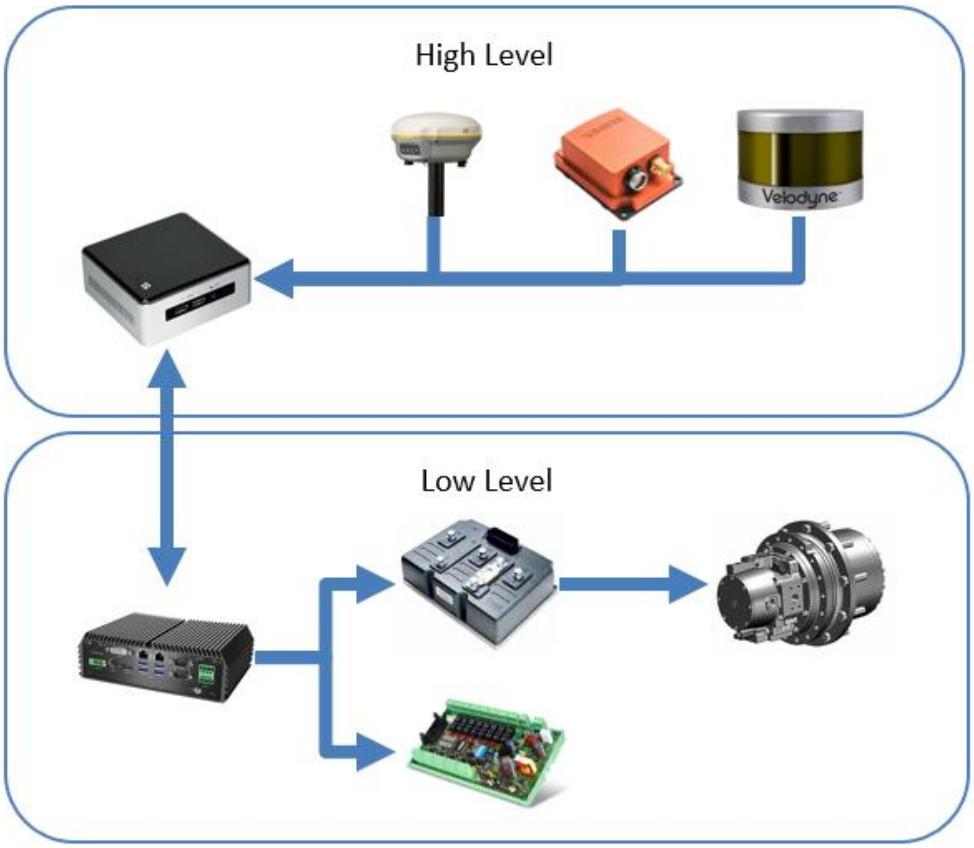

Figure 26 - Hardware architecture

The low-level computational unit is composed of a Cincoze DI-1000 fanless industrial PC that runs a basic C# application to handle the CANBus motor channel, the remote controller receiver, and the vehicle status. This scope separation is mainly due to safety and reliability concerns, since vehicle control must be always ensured even if the navigation stack may expect malfunctions (both in the hardware or in the software part). Also, the motor controllers have a failsafe operation that immediately halt the motors if commands are not received within a 100ms time frame. Furthermore, the remote controller (RC) receiver has its own failsafe status that is applied if the transmitter signal is lost due to interference or range limits.

The high-level navigation algorithms rely on the ROS (Robotic Operating System) framework and run on an Intel NUC D54250WYKH Core i5 mini-PC with Ubuntu Linux installed. The autonomous operation is triggered by a special switch on the remote controller and commands are then passed from the navigation algorithms described in the next chapter to the low-level computer and then to the motor controllers.

2.1.6. I/O management

A dedicated set of I/O modules is used to handle all the vehicle actuators and implements features (such as the sprayer electrovalves). The installed modules are Seneca ZE-4DI-2AI-2DO and Seneca ZC-24DO that can be controlled using ModBUS over TCP/IP and CANBus respectively. The digital output module is coupled with a relay battery of 16 elements that improve the signal driving capabilities of the module. All these subsystems are managed by the low-level hardware to ensure proper functionality and failsafe operation. Some of the basic handling includes parking brakes activation/deactivation, status LEDs operation on the information panel installed

onboard the vehicle, panel switches detection and management, implement mounting frame height adjustment and implements specific features and actuators.

2.1.7. Human Machine Interface

The prototype features both an onboard panel and a Graphical User Interface (GUI).

The onboard switches panel features a safety push button in the middle that is used to emergency stop the vehicle if needed. When activated, the button cuts off the contactor wire that provides electrical current to the motor controllers, thus deactivating the motors while the parking brake is engaged. On the right side, a blue button switch activates the battery so that power is delivered to the system and a key switch is used to power on all the onboard electronics (such as PCs, sensors, I/O modules, etc.). On the left side some status LEDs are used as visual feedback on the motors and parking brake status, and a red push button is used to power cycle the motor controllers after specific configuration setup or fault codes reset. Two knob selector switches are used to manually disengage the parking brakes to allow vehicle towing and to manually control the height of the implement interface by means of the winch direction operation.

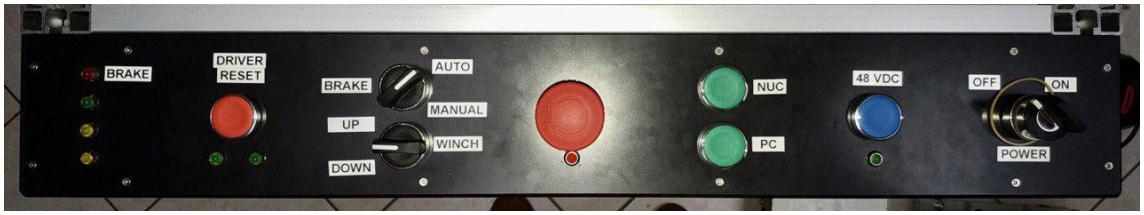

Figure 27 - Onboard switch panel

The main vehicle functions are operated by a C# application running on Windows operating system in the low-level computational unit. The main screen (presented in Figure 28 - Rover onboard GUI) contains all the vehicle parameters useful to monitor the platform state. The “Command” group-box is used to attach and enable the RC receiver data and to start the CANopen RPDO (Receive Process Data Object) messages flow to the motor controllers via the CANbus interface. The “Motors” group-box shows basic telemetry coming from TPDO (Transmit Process Data Object) frames sent by the motor controllers. Torque, speed, voltage, and current information is used also to provide feedback to the high-level control algorithms. The “Sprayer” group-box helps the user to visualize the status of the sprayer electrovalves that are controlled both by commands given from the RC transmitter and by orders coming from the high-level mission control algorithms. The “ROS” group-box enables connectivity to the high-level computer using Mavlink protocol [37]. Mavlink is a lightweight data exchange protocol that was specifically developed for onboard drone communication and ground station commands and telemetry transmission. The open-source implementation features a number of predefined messages and new custom messages can be added if needed. In our implementation only the following standard messages are used, with an indication to be received or sent to the high-level platform:

- heartbeat (communication alive – sent/received)
- local_position_ned (cartesian x/y coordinates from the high-level controller – received)
- home_position (origin position from the high-level controller – received)
- command_int (motor speed commands from the high-level controller – received)
- servo_output_raw (sensors state messages and flags – received)

- hil_controls (motor actual speed and torque telemetry – sent)
- rc_channels_raw (command flags for the vehicle status and control requests – sent)
- set_gps_global_origin (waypoint testing message in latitude/longitude – sent)
- mission_item_int (waypoint testing message in x/y cartesian coordinates – sent)
- gps_raw_int (actual GNSS position from the receiver – sent)

Although MAVLink protocol evolved over time, reaching version 2.3 at the present time, for compatibility purposes and because of the availability of the protocol when the original project started, the used and supported version for this data exchange is 1.0. Further improvements are planned but not at high priority at this time, since reliability and functionality are ensured with the version used both on the Windows and Linux applications. The “Next WP” group-box is used only for waypoint testing purposes without launching the high-level control subsystem, while the “GPS” group-box shows the position estimation data coming from the GNSS receiver, obtained by parsing standard NMEA messages over the serial data communication channel. The “Temperatures” group-box display the temperature sensor readings from inside the I/O enclosure and the PC enclosure. Lastly, the “Control Status” group-box can be used to request a specific control low to the high-level controller and the “Remote GS” group-box handles the ground station communication channel for the teleoperation capabilities described in the next paragraph.

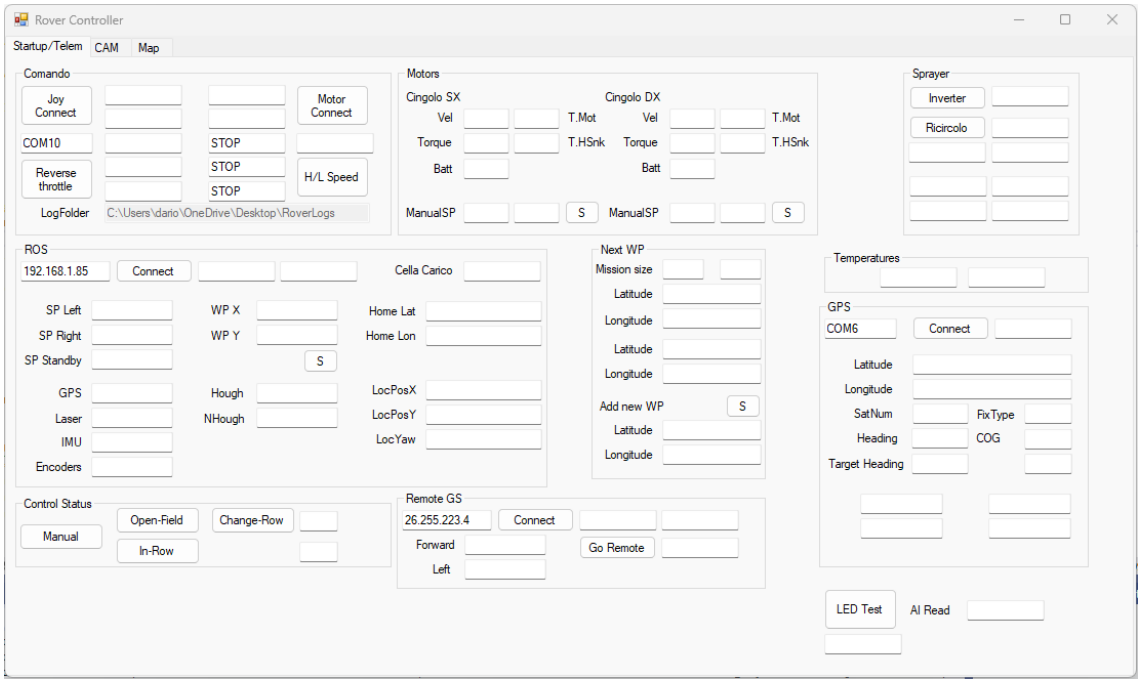

Figure 28 - Rover onboard GUI

2.1.8. Teleoperation capabilities

A remote teleoperation test has been conducted to prove real-time video stream capabilities and remote manual operation of the platform.

Figure 29 - Teleoperation PTZ color camera mounted on the prototype

The current implementation features a PTZ (Pan/Tilt/Zoom) camera mounted on the front right side of the rover, with the possibility to add another identical camera on the left side, so to potentially cover the full 360 degrees all around the vehicle. The remote user interface can show the satellite map of the surroundings and the camera video stream from the vehicle (as seen in Figure 30 as a picture-in-picture on the top left corner of the map). If needed, the two panels can be switched, so to show the map in the top left corner of the bigger video stream frame.

In the top section of the application there are some joypad information and management buttons and the parameters required to connect to the rover. Also, some status icons help the user to monitor some relevant status information such as parking brake activation, sprayer electrovalves actuation, autonomous navigation control status and auxiliary functionalities.

Figure 30 - Remote ground station user interface

The data connection stream is performed using UDP data packets over an Internet connection. This allows to minimize latency while prioritizing real-time control over actual ordering of packets or acknowledgment for reception. As the objective, in this case, is to guide the vehicle from a remote location, it is more important to get new commands as fast as possible with respect to be sure to receive all the information. Furthermore, the absence of an established connection channel, simplify the network management when internet connection is not fully reliable and may work intermittently.

The camera images are similarly transmitted back by means of a sequence of jpeg images. Again, the absence of a structured video stream allows for some missing frames and connection drops without any impact of previous or subsequent frames. This also simplifies the implementation because no particular video codec is needed, but every single frame (up to a maximum of 10 fps) is compressed into a jpeg image and sent over a single UDP datagram packet. Once the frame is received by the remote ground station, it is decoded and scaled up to fit the application window. In this context, some super-resolution algorithms can also be exploited to improve image quality on client side, without any impact to the bandwidth used. As the PTZ cameras allows some interaction with the user, also these commands can be sent to the vehicle to re-orient the camera to fulfill the user setpoints.

A full teleoperation test has also been conducted during a CAAB (Centro Agro-Alimentare di Bologna) event where the rover was remotely operated using a cellular 4G connection available in the room and onboard the rover.

2.2. New design and concept

During the last two years of research work, the rover prototype has faced several improvements that changed the profile and the design of the whole vehicle. The restyling activities have exploited the testing campaign conducted with the previous prototype iteration, by refining some aspects and improving the platform both in a mechanical and software perspective.

2.2.1. Mechanical improvements

As one of the main vehicle concept is to provide a multipurpose platform and flatbed to be fitted with specific tools for a variety of farming tasks, it seemed important to detach the structural water tank from the platform in order to allow a lower vehicle profile (that can be useful in certain scenarios such as trees with a bulky canopy) that can carry fruit harvesting bins or new generation implements that does not require a standard tractor PTO (power take-off), but can operate instead only with an electrical power supply. Meanwhile, about this aspect, as “legacy” implements cannot be ignored because probably are already owned by farm companies, a small size standard three-points hitch interface has been designed and constructed in order to make the whole robotic platform compatible with existing implements such as standard mowers or inter-row milling machines. The internal combustion PTO engine has also been swapped in favor of a 10 kW brushless electrical motor, thus making the vehicle fully electrified. This upgrade has also increased the energy requirements of the platform that has been fitted with a new generation of batteries provided by another UNIBO spin-off company (LiBER) that is specialized in creating high-performance Lithium battery packs with BMS (Battery Management System) integration. The new battery-pack module is providing 12 kWh of energy at a nominal voltage of 50 V, thus guaranteeing at least one hour of work with high demanding implements that can draw up to 10 kW of continuous power.

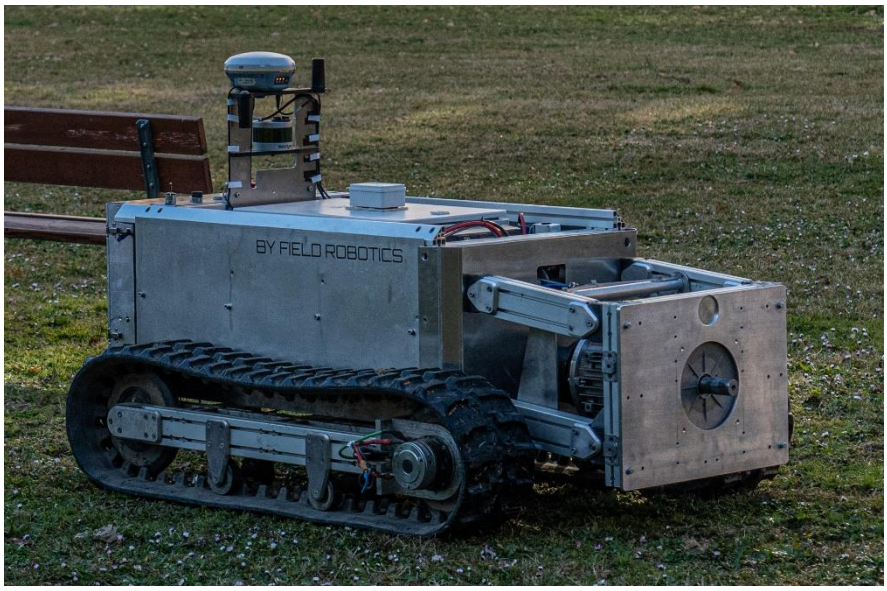

Figure 31 - Low-profile redesigned prototype

This platform has further been revised for the EIMA 2022 exhibition, where this concept has been further improved and stressed, producing a new platform that features a better captivating design that is more oriented to the market, while preserving and optimizing the concept features of the previous prototype iterations.

Figure 32 - Redesigned prototype presented at EIMA 2022

Innovative implements and simplified interface

As can be seen in Figure 32, the vehicle can be equipped with a standard three-point hitch to comply with "legacy" implements. In this case the PTO is preserved to be able to properly drive the attached implements using conventional equipment (yellow box frame in the picture). Nevertheless, the whole implement interface frame can be detached from the robotic platform to handle "new generation" innovative implements that does not require a PTO onboard the vehicle, since they have been co-designed according to the electrical power supply availability that the platform can provide. Therefore, smaller electrical motors of the proper size can be directly mounted on the implement, according to the specific task, and thus increasing the overall platform efficiency.

2.2.2. Upgraded locomotion

The locomotion tracked module has been improved with upgraded 5.3 kW motors that allow 1 ton of payload and increased towing capabilities. Also, some design solutions have been introduced to ease industrial construction and assembly while reducing production costs. A bigger 500 mm drive gear has been implemented that helps increasing efficiency by reducing track curvature radius. Furthermore, the idler wheel has been mounted on a suspension system to better control track tension. The two tracks are also mounted on a three-point suspension system that helps reducing the pitching of the vehicle while climbing over rocks or high steps.

2.2.3. Revised navigation sensor suite

As some vineyard testing, especially during periods with low vegetation, have highlighted some issues on row recognition because of the low number of points detected, the LiDAR sensor has been flanked with two stereocameras in the two front corners of the vehicle. The cameras selected are Intel Realsense D435i, that combines low cost with a reliable depth estimation and pointcloud generation given a distance range from 0.3 meters up to 3.0 meters. This choice has been selected taking into account standard and narrow orchards or vineyards dimensions which forms the vast majority of crops in the local area. The dense pointcloud generated from the RGBD (color + depth) camera systems may be fused together with the LiDAR data to improve lines detection in extreme conditions with only few narrow obstacles visible. Also, the Trimble R8s GNSS receiver has been replaced with more cost-effective options such as XSens MTi-680G or XSens MTi-670G IMU/GNSS receivers that combines both RTK satellite positioning systems and IMU data for automotive applications. Another option that is still under evaluation is the use of a commercial tractor autonomous navigation GNSS antenna such as the Trimble NAV900 sensor that can be supplied as an OEM part to be integrated into the vehicle architecture. The NAV900 receiver features high accuracy (<20 cm position estimation, up to 2 cm with special

RTK correction subscriptions) and an integrated inertial platform that can be useful also for heading estimation.

2.2.4. Low level hardware and software

The low-level computational unit has been replaced with a cheaper Raspberry PI4 single board computer or a Beckhoff industrial PLC with onboard Intel Atom computational unit. The two options aim to supply a better and more product-oriented implementation of the low-level control software that does not necessarily need a full-featured computer to operate the basic vehicle functionalities. As it can be expected, the Beckhoff PLC solution can be fitted in mission critical scenarios or high-end configurations, while the Raspberry PI solution is more oriented to entry-level configuration of the platform, while providing the same basic and locomotion functionalities. The whole low-level C# application has been rewritten to both fit the lightweight Linux operating system of the Raspberry PI board and the custom implementation of the Beckhoff PLC modules. At the moment, the new prototype relies on the Raspberry PI implementation running a python application to handle system timers, CANbus communication and an external 4x20 characters display to visualize vehicle status onboard.

2.2.5. Industrial radio-control interface

The RC transmitter has been replaced with an industrial-grade radio controller device that can be used for manual driving the prototype and activating auxiliary features and autonomous navigation controllers onboard. The selected device is an Autec *PJC transmitter* that is coupled with a CANopen enabled *CRP receiver* that is capable to command a maximum of 12 analog outputs and 64 digital inputs. The maximum range is 150 meters for safety reasons, as is required by actual regulations to maintain visual line of sight with the rover.

Figure 33 - Autec PJC remote transmitter example

2.3. Experimental stability testing

In order to assess the stability of the prototype, a testing campaign has been conducted for the first version of UGV (Unmanned Ground Vehicle) platform described, using different configurations:

- Vehicle only (Unladen-UGV)
- Customized Nobili Oktopus sprayer attached (full tank and empty tank) (UGV-sprayer)
- Adapted Concept Perugini CT mulcher (UGV-mulcher).

The tests have been performed at the official OECD (Organization for Economic Cooperation and Development) Test Station within the Laboratory of Agricultural Mechanics, Alma Mater Studiorum, University of Bologna. The ISO standards 16231-2 and 18497 were followed.

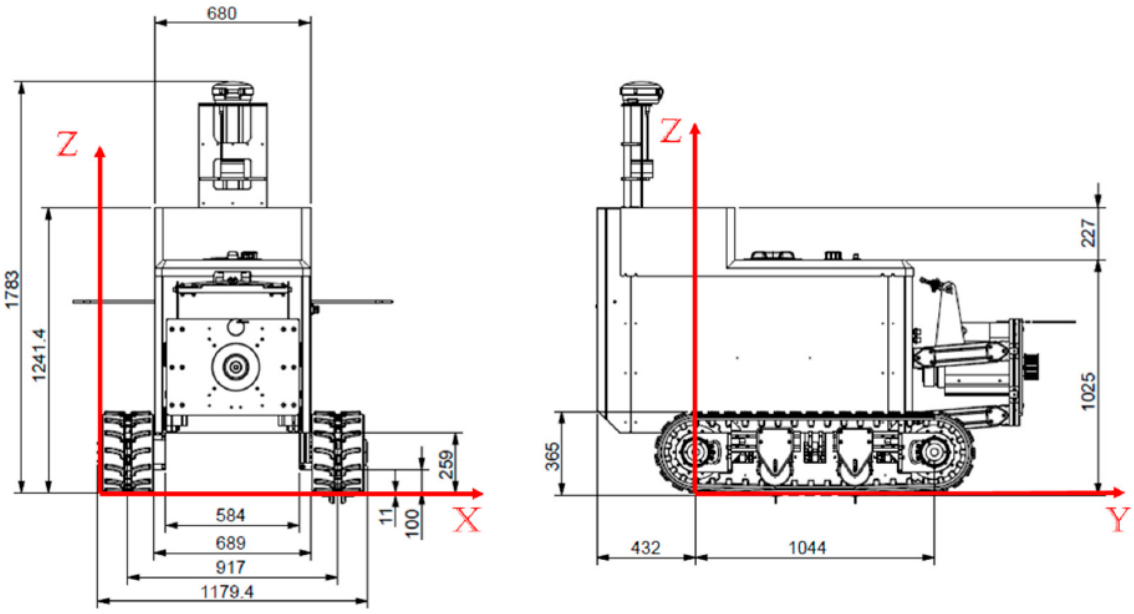

Figure 34 - Actual prototype dimension used for stability testing

The above Figure 34 shows the prototype employed for the tests and its relevant dimensions used for the analysis. A representation of the actual configurations with and without implements is depicted in the Figure 35 below.

Figure 35 - Prototype configurations assessed: (a) no implement; (b) mulcher; (c) sprayer

The prototype size and weight are likened to an agricultural tractor belonging to category C2 (EU Regulation 167/2013), which represent narrow-track tractors operating mainly in growing contexts. The main dimensions (Figure 34), such as length, width, height, wheelbase, and track width, are as follows:

- Total length: 1900 mm,
- Total width: 1179.4 mm,
- Total height: 1783 mm,
- Wheelbase: 1044 mm,
- Track width: 917 mm,
- Total empty weight: 630 kg.

The unladen UGV was the first UGV configuration considered for experimental testing (Figure 35a).

The UGV equipped with the mulcher was the second UGV configuration considered for the experimental tests (Figure 35b). The mower had a mass of 150 kg and was mounted on the implement interface of the vehicle. The installation of the mower produced an increase in length of 280 mm compared to the unladen configuration, but the overall width was not affected; the width of the mulcher was equivalent to the width of the unladen UGV.

The last configurations tested were the UGV with the air-powered sprayer mounted on the implement interface of the vehicle (Figure 35c). The sprayer had a mass of 100 kg. The 220-liter water tank is fitted in the central part of the UGV. Two different configurations were considered in the test: the UGV with the sprayer tank empty and the UGV with the tank full of water. The mass of water added to the tank was 190 kg. The installation of the sprayer produced an increase in length of 190 mm compared to the length of the unladen UGV and the overall with was not affected, similarly to the mulcher case. However, the total height increased to 1920 mm.

2.3.1. Experimental testing and Center of Gravity computation

Based on the three UGV configurations, the experimental tests performed are schematized in the following Table 2. The position of the center of gravity (CoG) of the three UGV configurations was determined, and lateral and longitudinal tilting tests were performed.

Table 2 - Tests description and planning

UGV Configuration		**Experimental Evaluations**		
Unladen UGV		Determination of CoG position	Lateral tipping test: - Upstream side force - Tipping angle	Longitudinal tipping test: - Upstream side force - Tipping angle
UGV-mulcher				
UGV-sprayer	Empty tank			
	Full tank			

The experimental test was repeated three times for each configuration to ensure data reliability.

Center of gravity position

The height of the CoG (Z position in Figure 34) for each configuration tested, was obtained indirectly by calculating the period of oscillation, which was measured with the vehicle fixed on a rocker platform at two different pivot heights, according to the parallel axis theorem (Figure 36). The longitudinal position of the CoG (X-Y positions in Figure 34) was obtained by using weight information obtained from four electronic scales placed under the tracks.

Figure 36 - Prototype placed on oscillating platform: (a) no implement; (b) mulcher; (c) sprayer

Lateral rollover tests

To perform the controlled side-tilt tests, the vehicle was placed on a tilting platform to bring the left track closer to the axis of rotation and trigger the lifting effect during the tilting action of the

platform, while the right track was on the far end with respect of the axis of rotation of the tilting platform. The longitudinal direction of the vehicle was parallel to the rotation axis of the tilting platform. To overcome the vehicle's imperfect grip during platform rotation, a dedicated structure was designed to prevent the vehicle from sliding downward. A load cell was placed under the right track to measure the lateral force upstream of the different UGV configurations. A digital level was used to monitor in real time the tilt angle of the UGV equivalent to the tilt platform angle.

Once positioned and secured to the tilt platform, the UGV was raised gradually to achieve a progressively increasing slope. Both the force on the load cell for estimating the unstable equilibrium phase and the tilt platform angle as digital level data were recorded at regular time intervals. The lateral tilt test was repeated for all UGV configurations and was stopped at the maximum achievable tilt equivalent to the unstable equilibrium condition. The experimental test was repeated three times for each configuration (Figure 37).

Figure 37 - Lateral tipping test: (a) initial position; (b) final position

Longitudinal rollover test

To carry out the longitudinal stability tests, the vehicle was placed on the tipping platform with longitudinal arrangement with respect to the direction of inclination of the base platform to simulate, and consequently hypothesize, the boundary conditions of vehicle stability both uphill and downhill on a sloping field. The longitudinal direction of the vehicle was perpendicular to the axis of rotation of the tipping platform. The vehicle's imperfect grip during platform rotation was overcome by means of a special structure that prevented the vehicle from sliding downward. A load cell was placed under each front track axle to monitor the stability status of the UGV in real time. As the inclined plane rose, the force on the front axle was gradually reduced to zero, corresponding to the unstable equilibrium condition.

Once positioned and secured to the tilt platform, the UGV was raised gradually to achieve a progressively increasing slope. Similarly to the lateral assessment, two types of data were recorded at regular time intervals: the force on the load cell and the tilt of the platform.

The longitudinal tilt test was performed on all UGV configurations until the maximum achievable tilt was reached. The experimental test was repeated three times for each configuration (Figure 38).

Figure 38 - Longitudinal tipping test: (a) initial position; (b) final position

Theoretical model

The CoG position change of the UGV and the stability condition during the progressive rotation of a tilting platform were predicted based on the initial position of the UGV prototype. In the first stage of the evaluation, the UGV was in a horizontal position (Figure 39a), that is, the coordinates of the CoG and the upstream track contact point with the ground (P_T) are:

$$CoG = \begin{cases} x = x_0 \\ z = z_0 \end{cases}; P_T = \begin{cases} A = W \\ B = 0 \end{cases}$$

where x_0 is the horizontal distance, z_0 is the height of the CoG relative to the axis of rotation, and W is the distance of the upstream prototype component in contact with the ground with respect to the rotation axis. Upon the rotation event (Figure 39b), after the initial position, the UGV polar coordinates in the plane are:

$$CoG = \begin{cases} x = \rho\ cos(\alpha + \beta) \\ z = \rho\ sin(\alpha + \beta) \end{cases}; P_T = \begin{cases} A = W\ cos(\alpha) \\ B = W\ sin(\alpha) \end{cases}$$

where $\rho = \sqrt{x_0^2 + z_0^2}$, α is the tilt angle, and $\beta = atan(z_0/x_0)$.

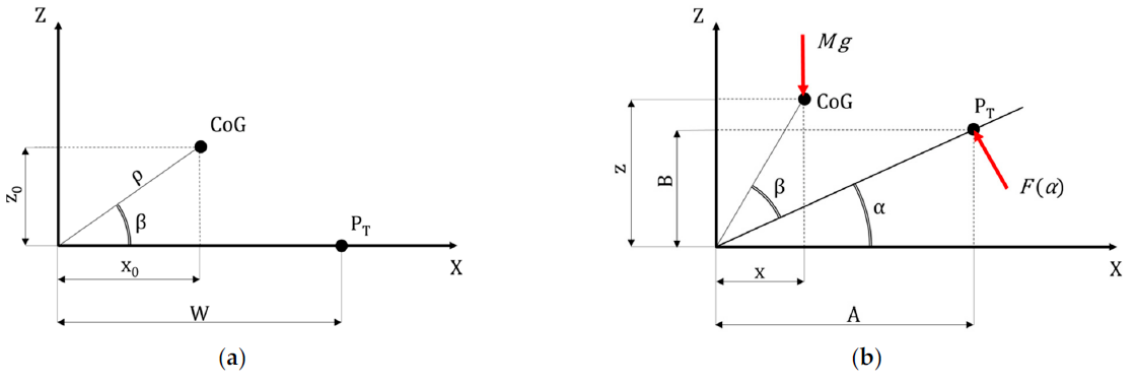

Figure 39 - Virtual lateral tipping analysis: (a) UGV horizontal position; (b) UGV tilted position

With reference to the mass of the UGV, the normal force acting on the track upstream of the prototype was calculated as a function of the tilt (α) to which the UGV is subject to, as follows:

$$F(\alpha) = \frac{M\ g\ x(\alpha)}{W} = \frac{M\ G\ \rho\ cos(\alpha + \beta)}{W}$$

where M is the UGV mass and g is the gravitational acceleration.

The approach used to analyze the case of lateral tilting was adapted to the case of longitudinal tilting by considering the y-coordinate instead of the x-coordinate.

2.3.2. Validation and experimental testing

Theoretical model data were compared with experimental results. The numerical model was developed using MATLAB®. The upstream lateral force (F) was calculated with respect to the inclined angle (α) of the inclined platform. Coefficient of determination (R^2) and root mean square error (RMSE) were calculated for comparison purposes using a dedicated MATLAB function to calculate the R^2 value based on actual experimental data and model data. The test results are given for the four UGV configurations (Table 2).

The results of the experimental tests were compared with the results of the predicted data, and the upstream lateral forces were superimposed to demonstrate the differences. The CoG mass and position values for the four UGV configurations are summarized in Table 3. The values of the upstream lateral force versus platform tilt angle (α) are shown in Figure 40.

Table 3 - Mass and CoG positions for the UGV configuration assessed, with data from lateral and longitudinal tests

UGV Configuration		Mass	CoG Position	Lateral Tests		Longitudinal Tests	
		M (kg)	x,y,z (mm)	R^2 (%)	RMSE	R^2 (%)	RMSE
Unladen UGV		630	(590,435,524)	99.7	0.56	99.4	0.58
UGV-mulcher		780	(590,672,459)	99.4	0.97	99.4	0.99
UGV-sprayer	Empty tank	730	(590,549,509)	99.5	0.80	99.7	0.71
	Full tank	920	(590,592,431)	99.8	0.46	99.5	0.92

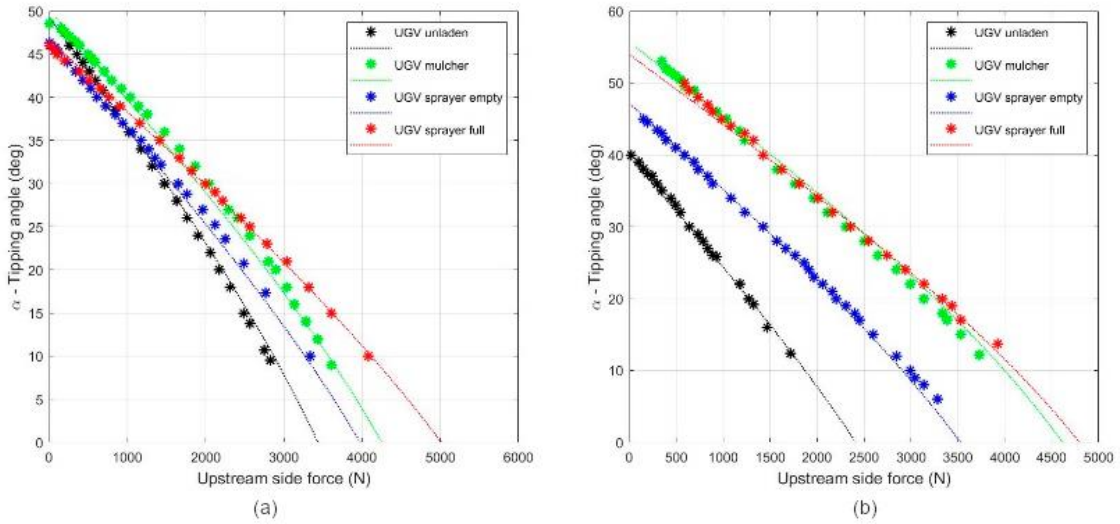

Figure 40 - Tilting angle compared to upstream force (for the different UGV configurations): (a) lateral tipping; (b) longitudinal tipping. Data represent the average of the three repeated tests

It is worth noting that the upstream force decreased as the angle of inclination increased. The evaluated behaviors can be attributed to the fact that increasing the slope angle affected the mass distribution, and the load was mainly on the downstream part of the UGV. Both R^2 and RMSE are shown in Table 3. The predicted model results show a statistically significant correlation ($R^2 > 95$). In addition, the RMSE was less than 1 degree. Model data obtained by combining the effects of fitted equipment are shown in Figure 41.

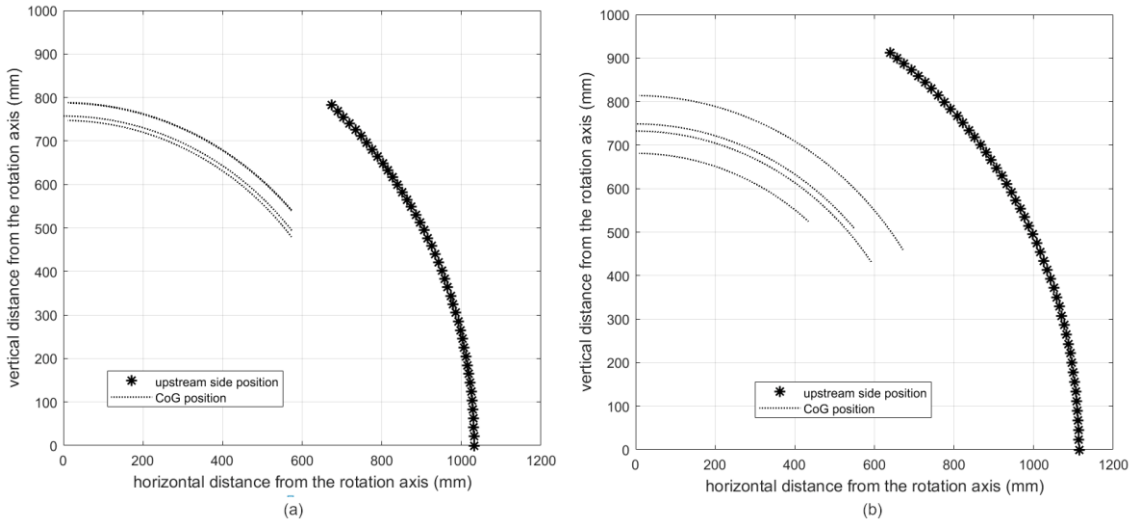

Figure 41 - CoG and upstream track contact point trajectory during tipping: (a) lateral tipping; (b) longitudinal tipping

2.3.3. Stability results

The lateral stability angle (α) was calculated, referring to the CoG coordinates in the initial ($P_0 = (x_0, z_0)$) and final ($P_f = (x_f, z_f)$) phases of the tilting action (Figure 39 and Figure 41):

$$cos(\alpha) = \frac{\overrightarrow{P_0}\,\overrightarrow{P_f}}{\|\overrightarrow{P_0}\|\,\|\overrightarrow{P_f}\|} \Rightarrow \alpha = arccos\left(\frac{x_0\,x_f + z_o\,z_f}{\sqrt{x_0^2 + z_0^2}\,\sqrt{x_f^2 + z_f^2}}\right)$$

The approach used to analyze the lateral stability angle was adapted to the longitudinal stability angle by considering the Y coordinate instead of the X coordinate. The following Table 4 shows the stability angles of the four UGV configurations.

Table 4 - Stability angles: lateral and longitudinal tipping limits

UGV Configuration		Lateral tipping angle (α)	Longitudinal tipping angle (α)
Unladen-UGV		49°	40°
UGV-mulcher		50°	56°
UGV-sprayer	Empty tank	47°	47°
	Full tank	47°	54°

Based on the study of the stability behavior (both lateral and longitudinal) of the UGV in different implement configurations, a theoretical model was developed to predict the stability angles of the UGV-implement system platform. The analysis evaluated the normal force sustained by the upstream side of the UGV in a tilted position. The type of setup, as well as its geometry and inertia, affected the stability angle. Due to the actual UGV conditions during the experimental test, especially when the tilt angle increased, some repeated tests failed to achieve the zero-

force value and therefore it was difficult to obtain an accurate measurement of the angle equivalent to the unstable condition.

A theoretical model was developed to predict the variation of the UGV CoG and the stable condition during the progressive rotation of the tilting platform. Based on the initial position of the UGV and considering rotation about an axis, the normal force acting on the track upstream of the UGV was also calculated. The model did not consider the effect of vehicle downward sliding or lack of grip on the platform, thus ignoring the ground-track interaction by simply considering rotation without translation. A comparison of the virtual and real results allowed the theoretical model to be calibrated and validated, showing that the model was reliable by means of a relative error less than 5% (R2 > 95%). This allowed a complete mapping of the stability angle, with respect to the position of the vehicle CoG, considering lateral and longitudinal contributions.

In the lateral tests, the mean value of the limiting stability angle was 48°, while in the longitudinal test it was 49°. These values are very close; however, the standard deviation increased from 1.5 degrees to 7.3 degrees, respectively. In the longitudinal test, the variability was greater, showing that the stability angle is more influenced by the implement in the longitudinal stability behavior compared to the in lateral stability one. In fact, the lateral test denoted that the stability angle showed no variability with respect to the implement. In contrast, in the case of the longitudinal test, the implement significantly influenced the stability angle of the UGV. This is due to the implement position at one end of the vehicle, thus greatly altering the CoG position in the longitudinal axis.

2.4. Tracks slipping estimation using Gaussian Processes[4]

A Skid Steering Vehicle (SSV) features a larger ground contact surface than the wheels, allowing a higher level of traction and stability. The main drawback of this locomotion system lies in the turning behavior: the two tracks are driven with different speeds causing the robot to skid on the ground. The ground friction coefficient can cause longitudinal or lateral displacement of the UGV or even hinder its progress.

An SSV should ideally behave like a differential wheeled robot (DWR), which is why the unicycle kinematic model is usually adopted, as seen also on Section 2.1.1 (Dynamical Model), and here reproposed for better readability:

$$\begin{aligned} \dot{x} &= V_G\, cos(\theta) \\ \dot{y} &= V_G\, sin(\theta) \\ \dot{\theta} &= \Omega_z \end{aligned} \qquad \textit{Eq. 8}$$

where the (x, y) pair expresses the position of the vehicle in inertial coordinates, while V_G is the linear velocity of the vehicle's center of gravity (CoG), in body coordinates. In addition, θ is the heading angle and Ω_z represents the angular velocity around the z-axis of the body.

Figure 42 depicts the main quantities involved in this discussion, along with both inertial (x, y) and body (x_b, y_b) reference systems. O_i represents the system inertial reference frame origin, while O_b is the body frame origin. θ and Ω_z are heading angle and turning velocity respectively

[4] This work has contributed to the published paper: Data-Driven Model Predictive Control for Skid-Steering Unmanned Ground Vehicles. - Gentilini, L., Mengoli, D., Rossi, S., Marconi, L.- 2022 IEEE Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2022 - Proceedings, 2022, pp. 80–85

(around the z-axis). Finally, v_L and v_R are the left and right track speeds and V_G is the linear speed of the CoG expressed in body coordinates.

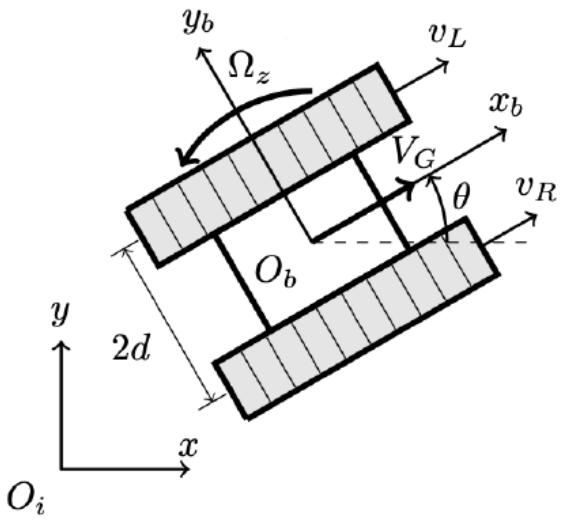

Figure 42 - Planar SSV model

The same kinematic model *Eq. 8* is used to describe DWR and SSV, with the only difference lying in the formulation of V_G and Ω_z with respect to the linear, or angular, velocities of the tracks. Specifically, in the case of DWR, V_G and Ω_z take the form:

$$V_G = \frac{v_R + v_L}{2} = \frac{r}{2}(\omega_R + \omega_L)$$
$$\Omega_z = \frac{v_R - v_L}{2d} = \frac{r}{2d}(\omega_R - \omega_L)$$

where v_L and v_R are the linear track speeds, while ω_L and ω_R are the track motor velocities, r is the track wheel diameter and d is half of the vehicle track width. This relationship holds in all those cases where slip is absent, so when the wheel speeds (whether angular or linear) are strongly correlated with the motion of the vehicle in the inertial system: this assumption is entirely valid in the case of DWRs, considering the small contact surface between the ground and the wheel (ideally one point), for good ground to wheel friction coefficient, as in most conditions. In contrast, in the case of SSVs, the slip is not negligible, so a more realistic and reliable model is needed. In the literature, several attempts have been made to successfully find a slip model for SSV. Kinematic slip has been modeled as a constant value plus an additive zero mean stochastic process and then estimated using Kalman filters [38]–[41]. Conversely, [24] and [25] analyzed the dependence of slip with respect to track speeds, turning radius, and soil properties. However, these relationships work only when the turning radius is kept sufficiently small. Among all the different models, it can be considered the efficiency reduction model also described in Section 2.1.1:

$$V_G = \frac{v_R\big(1 - i_R(t)\big) + v_L\big(1 - i_L(t)\big)}{2} = \frac{r}{2}\Big(\omega_R\big(1 - i_R(t)\big) + \omega_L\big(1 - i_L(t)\big)\Big)$$
$$\Omega_z = \frac{v_R\big(1 - i_R(t)\big) - v_L\big(1 - i_L(t)\big)}{2d} = \frac{r}{2d}\Big(\omega_R\big(1 - i_R(t)\big) - \omega_L\big(1 - i_L(t)\big)\Big) \qquad \text{Eq. 9}$$

where $i_R(t), i_L(t) \in (-1,1), \forall t \in \mathbb{R} \geq 0$ are slipping coefficients that are time-varying and associated to the right and left tracks, respectively.

In this context, $i_R(t)$ and $i_L(t)$ can be interpreted as efficiency coefficients associated with the control inputs ω_R and ω_L. Although Eq. 9 is a sufficient approximation for almost all applications, it fails when the SSV is required to perform precise maneuvers such as navigating within confined spaces (e.g., within narrow orchard rows) [42]. Furthermore, although it is known ([24]–[27]) that $i_R(t)$ and $i_L(t)$ depend on the type of terrain and the velocities of the tracks (both absolute and relative to each other), a mathematical model of $i_R(t)$ and $i_L(t)$ is not yet available in the literature. Therefore, the idea to propose a solution to this problem using a data-driven approach by extending the DWR model (*Eq. 8*) with nonlinear slip-dependent terms and recursively estimating them using the Gaussian Process (GP). The resulting model is implemented in a Model Predictive Controller (MPC) to better exploit the optimizations that can be made due to the prediction of the skid steering model. Experimental results were obtained by executing specific maneuvers with the prototype robotic platform developed.

2.4.1. Data-driven Gaussian Processes solution

As mentioned earlier, the DWR model (Equation (1)) can be extended to take into account the tracks slippage and skidding: the idea is to include some nonlinear terms describing slipping that depend on the input variables, so that they can be estimated using data-driven techniques such as Gaussian Process (GP). So, as ω_R and ω_L are the control inputs, the model can be rewritten as:

$$\begin{aligned}\dot{x} &= f_l(\omega_R, \omega_L) * cos(\theta) + f_t(\omega_R, \omega_L) * sin(\theta)\\ \dot{y} &= f_l(\omega_R, \omega_L) * sin(\theta) + f_t(\omega_R, \omega_L) * cos(\theta)\\ \dot{\theta} &= f_\Omega(\omega_R, \omega_L)\end{aligned}$$

where the unknown functions $f_l(\omega_R, \omega_L)$ and $f_t(\omega_R, \omega_L)$ represent the longitudinal translation and the traversal drift respectively. These contain both the time-varying tracks slipping effect and other possible friction disturbances. From a control law perspective, assuming the system state as $x = [x, y, \theta]^T$, and $u = [\omega_R, \omega_L]^T$ as input, the above model can be written as:

$$\dot{x} = f(x, u) = \begin{bmatrix} f_l(u) * cos(\theta) + f_t(u) * sin(\theta) \\ f_l(u) * sin(\theta) + f_t(u) * cos(\theta) \\ f_\Omega(u) \end{bmatrix} \qquad \text{Eq. 10}$$

The choice of implementing a MPC regulator is because it can be used directly as a prediction model without the need for ad hoc control law synthesis. In addition, the tracking signal is a trajectory in a real environment, where obstacles are often present: they can be easily mapped into state constraints, so they can be easily included in the MPC formulation and handled by the controller [43].

It is worth highlighting that to derive the formulation of Eq. 10, input-output pairs are needed for each function f_l, f_t and f_Ω. A sensor suite composed of motor encoders, IMU and GNSS receiver is sufficient to measure (or compute) the requested quantities, since tracks speeds (that form the u inputs) and body velocities (among x, y axis) are involved. The ability to collect these values online, while the vehicle is in operation, allows for very high adaptability across different terrains, weather conditions, and seasons.

Gaussian Process Inference

The key idea is to model the unknown functions $f_l(u)$, $f_t(u)$ and $f_\Omega(u)$ as a realization of three Gaussian Processes (GPs). GPs are widely used stochastic processes because of their high flexibility in modeling nonlinearities. In this framework, similar to most GP-based learning

problems, we assume the existence of the true inaccessible dynamics $f(x,u)$, which we measure as y through the following noisy process:

$$y = f(x,u) + w$$

where w is assumed to be a Gaussian distributed noise, with a time-invariant diagonal covariance matrix Σ ($w \sim \mathcal{N}(0,\Sigma)$). The diagonal structure of Σ allows to consider each dimension of y independently, thus leading to three separate unidimensional GPs.

A GP model is completely defined by its mean function $\mu(u)\colon \mathbb{R}^2 \to \mathbb{R}$ and by its covariance function $\kappa(u,u')\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ [44]:

$$f_{l/t/\omega}(u) \sim GP\big(\mu(u), \kappa(u,u')\big)$$

Among the many possible choices with respect mean and covariance functions, it is proposed to adopt a *zero-mean* function ($\mu(u) = 0$), and a *quadratic exponential kernel* covariance function:

$$\kappa(u,u',\tau) = exp\left(-\frac{\|u-u'\|_\tau^2}{2}\right)$$

where τ is a bidimensional vector of hyper parameters known as *characteristic length scale*. Assuming m as the number of training samples, the training inputs can be written as $u^{1:m}$ and the sampled results as $z^{1:m}$. The a-posteriori mean and variance functions should be:

$$\begin{aligned} f_{l/t/\omega}(u) &= \mu(u) + \kappa(u)^T[\mathcal{K} + \sigma_n^2 I]^{-1}\big(z^{1:m} - \mu(u^{1:m})\big) \\ \sigma^2_{l/t/\omega}(u) &= \kappa(u,u) - \kappa(u)^T[\mathcal{K} + \sigma_n^2 I]^{-1}\kappa(u) \end{aligned} \qquad \text{Eq. 11}$$

In which $\kappa(u) \in \mathbb{R}^m$ and is a column vector with elements $\kappa^i(u) = \kappa\big(u^i, u\big)$ with $i \in [1,m]$, and $\mathcal{K} \in \mathbb{R}^{m\times m}$ is the covariance matrix with elements $\mathcal{K}^{i,j} = \kappa\big(u^i, u^j\big)$ with $i \in [1,m]$ and $j \in [1,m]$.

In order to handle the unknown hyperparameters, a Bayes empirical approach can be followed. In this case the prediction steps are alternated with parameter estimation by maximum likelihood optimization:

$$\tau = arg\ \max_\tau p(z^{1:m}|u^{1:m};\tau,\rho)$$

By means of a Gaussian Process model, the quantity $p(z^{1:m}|u^{1:m};\tau,\rho)$ can be evaluated in closed form as a logarithmic approximation:

$$\log\big(p(z|u;\tau,\rho)\big) = -\frac{1}{2}(\mu(u)-z)^T[\mathcal{K} + \sigma_n^2 I]^{-1}(\mu(u)-z) - \frac{1}{2}\log(|\mathcal{K} + \sigma_n^2 I|) - \frac{m}{2}\log(2\pi)$$

where the notation $z^{1:m}$ and $u^{1:m}$ is left in favor of readability.

The sampled quantities $z^{1:m}$ can be derived from the measure $y = \big[y_{vx}, y_{vy}, y_\omega\big]^T$. Thus, by inverting Eq. 10, it can be computed z as:

$$z = \begin{bmatrix} z_l \\ z_t \\ z_\omega \end{bmatrix} = \begin{bmatrix} y_{vy}sin(\theta) - y_{vx}cos(\theta) \\ y_{vx}sin(\theta) - y_{vy}cos(\theta) \\ y_\omega \end{bmatrix} = \begin{bmatrix} -cos(\theta) & sin(\theta) & 0 \\ sin(\theta) & -cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} y$$

Model Predictive Control formulation

The Gaussian Process inference can be exploited within a MPC framework to enhance and adjust the state prediction. The predictive model can be described as:

$$\dot{x} = \bar{f}(x,u) = \begin{bmatrix} \overline{f_l}(u) * cos(\theta) + \overline{f_t}(u) * sin(\theta) \\ \overline{f_l}(u) * sin(\theta) + \overline{f_t}(u) * cos(\theta) \\ \overline{f_\omega}(u) \end{bmatrix}$$

where $\overline{f_l}$, $\overline{f_t}$ and $\overline{f_\omega}$ can be obtained by computing the GP a-posteriori mean using Eq. 11. Therefore, the MPC problem can be written as:

$$\min_u \tilde{x}(T)^T Q_f \tilde{x}(T) + \int_{t_0}^{T} \tilde{x}(t)^T Q \tilde{x}(t) + u(t)^T R u(t)\, dt$$

$$\begin{aligned} subj.to \quad & \dot{x} = \bar{f}(x,u), \\ & x(t_0) = x_0, \\ & \tilde{x}(t) = x(t) - x^*(t), \\ & u_{min} \leq u(t) \leq u_{max} \end{aligned}$$

where $x^*(t)$ is the state reference trajectory. In this basic formulation, the only external constraint is imposed to the input variable, because of actual physical limitations. However, as mentioned earlier, other hard constraint that may affect state variables can be added during navigation, by considering the presence of obstacles or other limitations. Consequently, it can also be necessary to relax these constraints under certain conditions to make the problem feasible using slow variables [43].

This controller has not yet been tested on the real vehicle, so all the design parameters such as the state terminal cost weight matrix Q_f, the state cost weight matrix Q, the input cost weight matrix R and the time horizon T of the MPC still need to be chosen depending on the desired performance and available computing power.

2.4.2. Experimental data

Although Input-output data can be gathered online while the vehicle is running within orchards, it has been chosen, to validate slip estimation, to collect a preliminary dataset by manually commanding the platform. The developed tracked prototype platform was used for this purpose. Figure 43 displays the x-y plane view of the motions set that have been executed to secure an initial comprehensive and trustworthy dataset. The collected body velocities and trace inputs are instead shown in Figure 44.

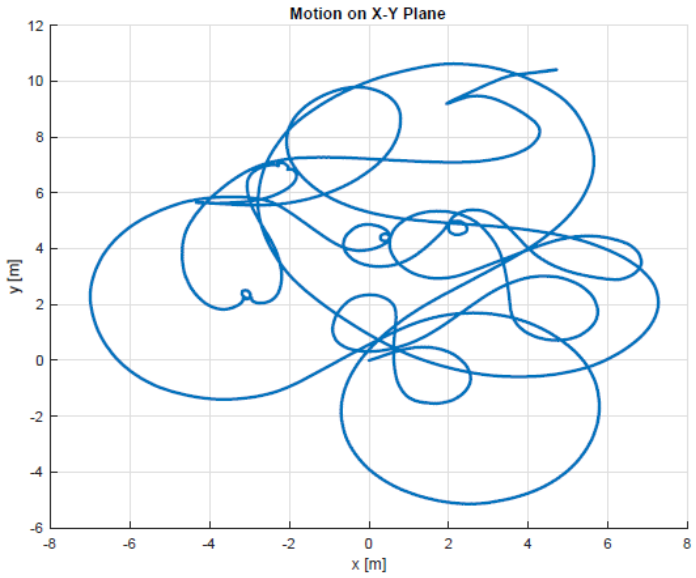

Figure 43 - Manual controlled motions represented on a x-y plane. Collecting different track speeds commands, together with the related body velocities, enables a more accurate estimation.

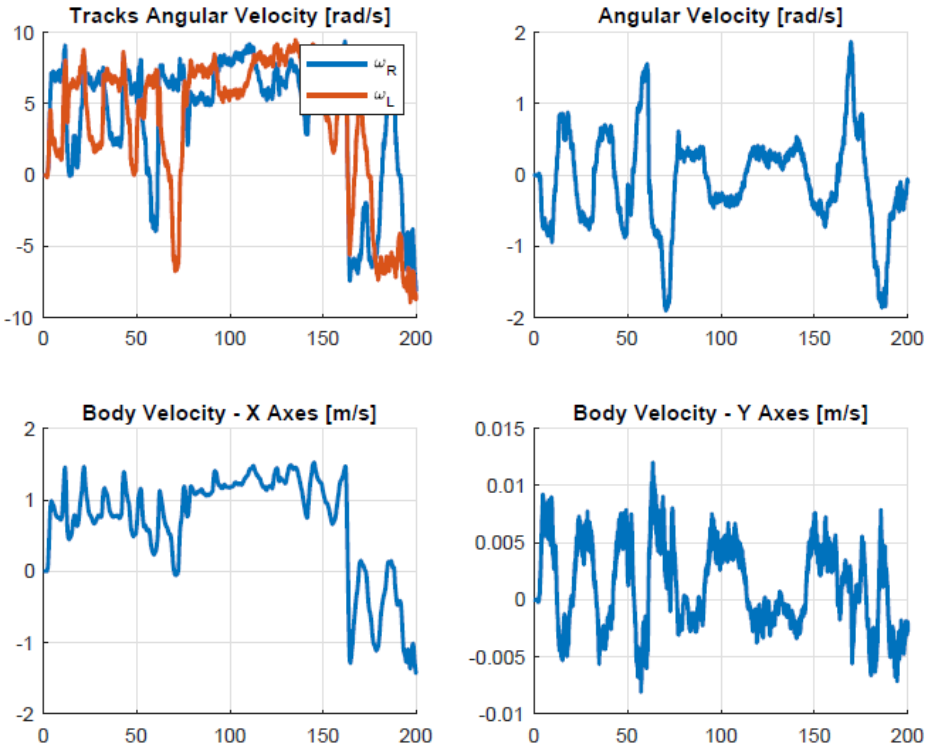

Figure 44 - Tracks and body velocities recorded during manual vehicle motion. The more the collected quantities span over the whole allowable range, the more accurate the GP regressor will be.

The motions performed were designed to generate a full range of trace velocities and relative velocities for both $(\omega_R + \omega_L)$ and $(\omega_R - \omega_L)$ space. Figure 45 displays the compilation of these datasets. As seen in Figure 46, comparing both the GP estimation model and the simple DWR model fed with the same input signals (ω_L, ω_R) into the prediction model (Eq. 10), it was found that its predicted output $(\dot{x}, \dot{y}, \dot{\theta})$ of the first GP model was closer to the measured output compared to that of the latter DWR model.

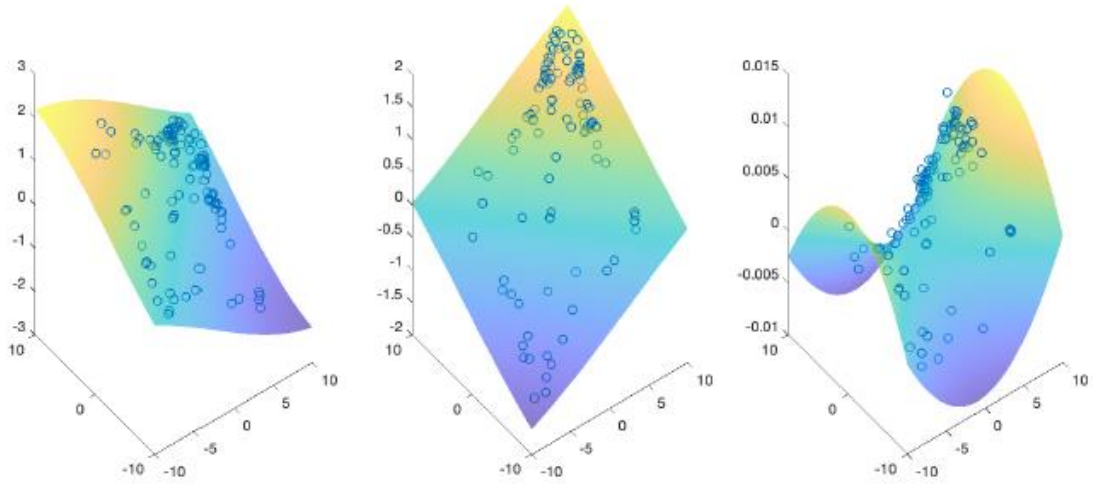

Figure 45 - GP interpolation of the collected data. The blue circles represent the samples, while the shaded surface is the GP estimation

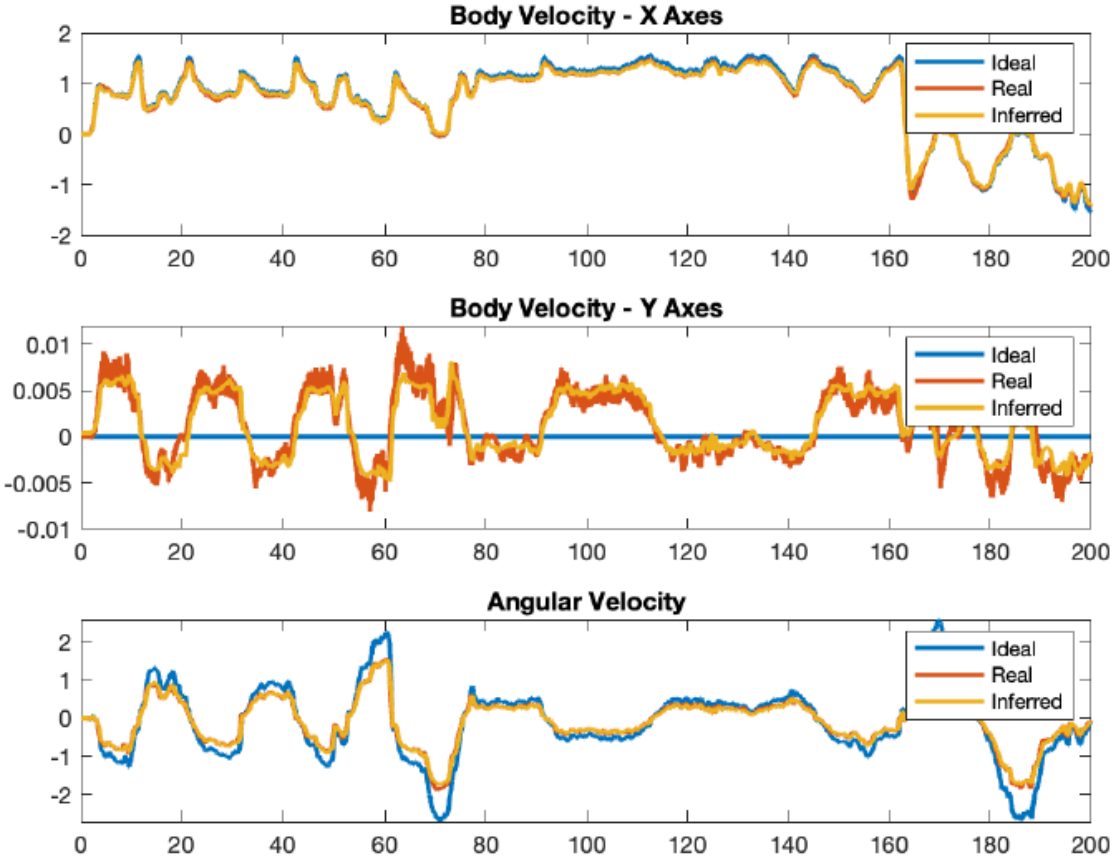

Figure 46 - Comparison between DWR and GP model, with the measured data along the manual driving test. The GP inferred quantities (yellow lines) approach the real data (red lines) closely with respect to the ideal DWR quantities (blue lines).

It can be seen that the angular velocity in the images, which indicates that rotation around the vertical axis, is the most affected behavior. Also, during high-speed maneuvers, there may be a slight transverse sliding motion (on the y-axis). Figure 47 shows the associated errors.

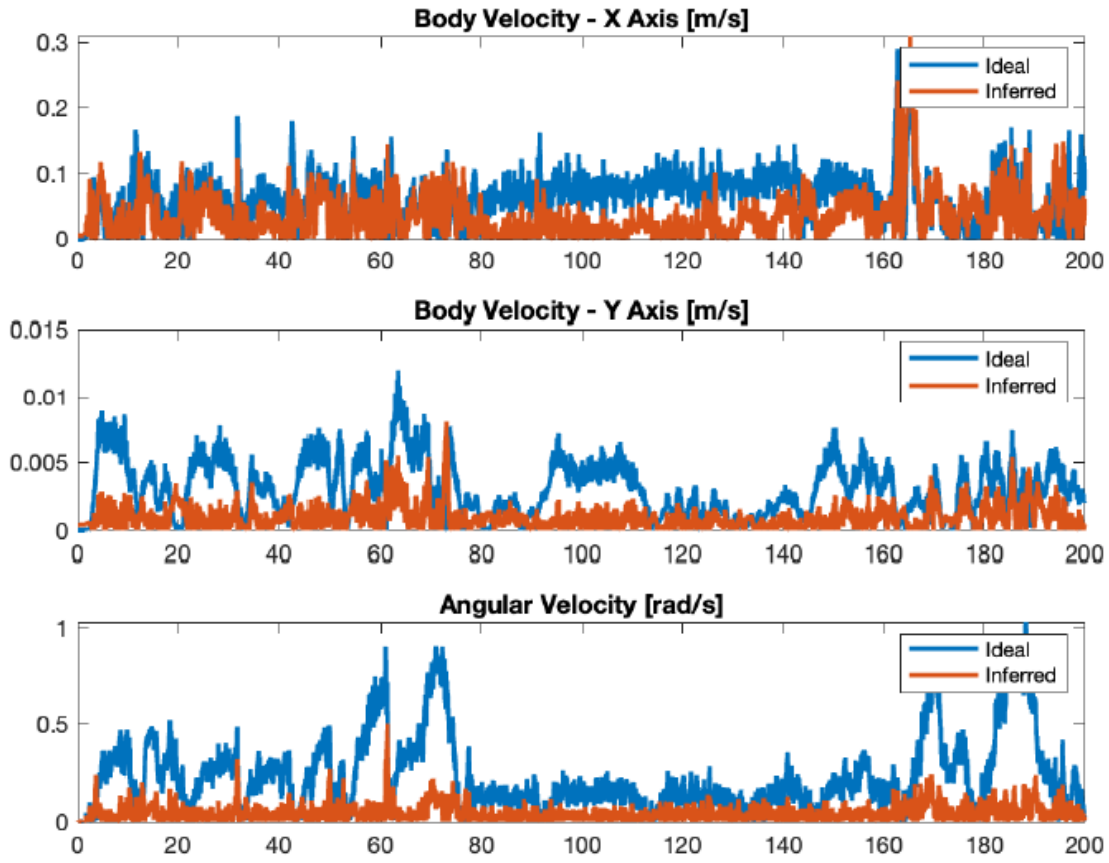

Figure 47 - Plot of the errors committed between the ideal DWR model (blue lines) and GP regressor model (red lines)

Two further experiments were conducted to move the robot on predefined paths: a circle at constant speed, the outcomes discernible in Figure 48; and an upside-down figure eight with velocities computed by a higher order trajectory planner, with showing presented in Figure 49. A condensed overview of all trial runs is displayed in Table 5, confirming that GP-optimized model outperformed the fundamental DWR model in each instance.

Table 5 - Root Mean Squared Errors (RMSE) obtained along the three experiments. The table compares the RMSE obtained by running the DWR model (left column) and the GP-based model (right column). Lower quantities are highlighted in bold for better visualization

RMSE data	Manual trajectory (DWR)	Manual trajectory (GP)
v_{x_b}	0.0374	**0.0361**
v_{y_b}	0.0020	**0.0008**
ω_z	0.1893	**0.0413**
	Circle trajectory (DWR)	Circle trajectory (GP)
v_{x_b}	0.0191	**0.0143**
v_{y_b}	0.0010	**0.0007**
ω_z	0.0416	**0.0290**
	Lemniscate trajectory (DWR)	Lemniscate trajectory (GP)
v_{x_b}	0.0321	**0.0294**
v_{y_b}	0.0011	**0.0005**
ω_z	0.0550	**0.0230**

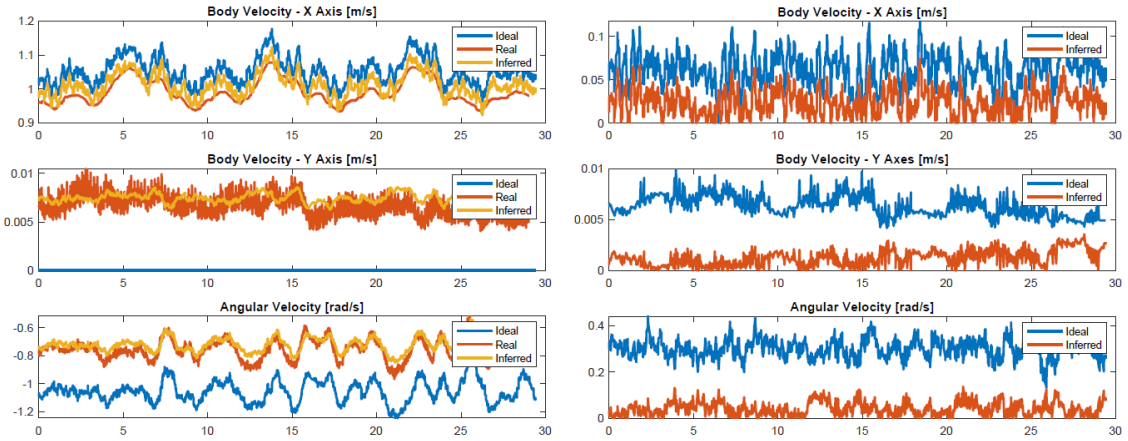

Figure 48 - Circle trajectory comparison between ideal DWR model, inferred GP regressor model and real velocities. On the left side are plotted the velocity values, while on the right side are plotted the errors for DWR and GP models

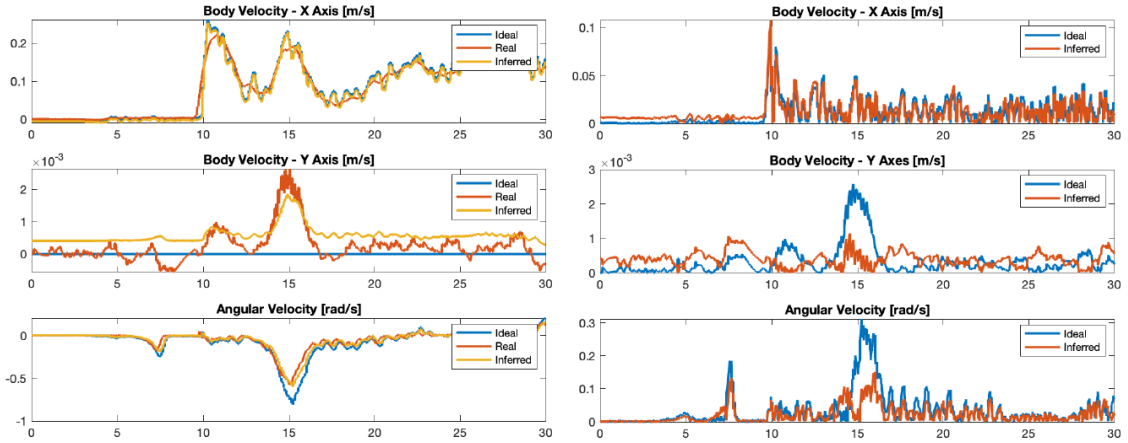

Figure 49 - Lemniscate trajectory comparison between ideal DWR model, inferred GP regressor model and real body velocities. On the left side are plotted the velocity values, while on the right side are plotted the errors for DWR and GP models

2.5. Conclusions

In this chapter it has been presented the mechanical design and concept underneath the implementation and the evolution of the autonomous ground rover prototype. The design choices and mechanical and software descriptions are detailed, together with some testing related to platform statical stability and track slipping estimation experiments. Further improvements will focus on refining the electronics design towards automotive standards, in order to fulfill current machinery regulations and thus creating a commercial platform that can enter the market of radio-controlled equipment at first, and then being recognized as an autonomous machine as soon as European regulations will be ready.

Chapter 3. Autonomous navigation features

In this chapter, the autonomous navigation features, algorithms, and architecture are presented. The platform features a full-featured navigation within and outside an experimental orchard with reduced inter-row width available at the Cadriano UNIBO experimental agricultural area. Furthermore, the simulation engines and scenarios used to help onboard software development are described.

The high-level computational layer presented in the previous chapter is in charge of executing and handling all the autonomous navigation algorithms, that are divided into different controllers:

- Open field navigation using waypoints
- In-row orchard navigation
- Change-row maneuver

The implementation is based on the ROS (Robotic Operating System) framework that provides built-in functionalities for data and messages exchange across different modules. As the ROS ecosystem is open source with a 3-Clause BSD License, it can rely also on a very active community that maintain and also improves a wide number of driver packages that can be used to gather data from the vehicle sensors. Also, permissive licensing allows also to use part of the software for commercial applications.

3.1. Modules architecture

The ROS implementation relies on a number of packages, each one solving a specific issue, such as reading data from a sensor, handling and processing information, implementing a single control law. Each package can collect several nodes, custom messages, or services to enable interaction within different nodes, while each node executes concurrently on the platform.

The list of developed packages can be grouped as follows.

Control packages:

- control_changerow – used to implement the change row maneuver controller
- control_inrow – used to implement the row following controller
- control_openfield – used to implement the open field waypoint navigation

Localization packages:

- robot_localization – standard package for extended Kalman filter localization using sensor data
- transformgps – used to inject frame transform information and compute and rotate odometry messages according to the preferred travel direction

Data gathering and processing packages:

- mapping_2d – used to create a two-dimensional map of the environment using LiDAR data and robot pose estimation
- mavros – used to exchange data between low-level and high-level computational layers using mavlink protocol messages
- realsense2_camera – standard package to retrieve Intel Realsense camera images and depth information

- realsense_merger – used to merge stereocameras pointclouds with LiDAR one, so to obtain a single comprehensive information set
- rover_hough_transform
- trajectory_server_v2 – used to compute a polynomial trajectory across the given waypoints with optimizations

Other tools:

- examon_bridge – used to send telemetry data to a centralized database (called Examon, described in Section 4.1)
- laser_launch – contains launch files used to startup system nodes
- poly_lib – ancillary library used to handle polynomial curves
- supervisor – used as the vehicle control supervisor to switch different control laws
-

The control algorithms will be detailed and explained in Section 3.3 and 3.4.

Each node communicates with others by means of ROS topics (debug messages or extra messages are omitted, message type between parenthesis):

- cmd_vel (geometry_msgs/TwistStamped) – used to set vehicle motor speeds
- gps/filtered (sensor_msgs/NavSatFix) – GNSS localization data filtered by localization node
- gps_trimble_raw (sensor_msgs/NavSatFix) – GNSS localization raw data coming from Trimble R8s antenna
- hough_left_line (geometry_msgs/PoseStamped) – left row line with respect to the robot, as detected by the Hough Transform line detection algorithm implementation.
- hough_perpendicular_line (geometry_msgs/PoseStamped) – perpendicular line with respect to the rows detected, as estimated by the Hough Transform line detection algorithm implementation. This detection is used during row-change maneuvers.
- hough_right_line (geometry_msgs/PoseStamped) – right row line with respect to the robot, as detected by the Hough Transform line detection algorithm implementation.
- hough_third_line (geometry_msgs/PoseStamped) – new row line estimated during row-change maneuvers, that forms the next corridor to be traversed, as detected by the Hough Transform line detection algorithm implementation.
- imu/data_body (sensor_msgs/Imu) – IMU data transformed according to the preferred motion direction of the vehicle
- imu/filtered (sensor_msgs/Imu) – IMU AHRS filtered data coming from the Xsens sensor
- map (nav_msgs/OccupancyGrid) – environment map
- mqtt/encoder_rpm_dx (std_msgs/Int16) – Right track motor actual speed to be transmitted to the centralized database
- mqtt/encoder_rpm_sx (std_msgs/Int16) – Left track motor actual speed to be transmitted to the centralized database
- odometry/filtered (nav_msgs/Odometry) – Vehicle odometry estimated by the localization node, centered with the vehicle startup initial position. This estimation does not take into account absolute position, so it is always guaranteed to be continuous
- odometry/filtered_map (nav_msgs/Odometry) – Vehicle odometry estimated by the localization node, centered with the map origin. This estimation serves as absolute robot position in the space, as it integrates GNSS information. Therefore, it is not guaranteed to be continuous.

- odometry/gps (nav_msgs/Odometry) – filtered GPS odometry information given by the localization node
- odometry/gps_yaw (nav_msgs/Odometry) – filtered GPS odometry information with orientation transformed according to the direction of motion
- odometry/raw (nav_msgs/Odometry) – vehicle motor encoders odometry estimation
- rover_supervisor/state (geometry_msgs/Vector3Stamped)
- scan (sensor_msgs/LaserScan) – 2-dimensional distance points filtered according to robot dimensions
- scan_raw (sensor_msgs/LaserScan) – raw 2-dimensional distance points information coming from the 3d pointcloud scan flattening node
- tf (geometry_msgs/TransformStamped) – used to apply conversions, such as translations and rotations between two different reference frames
- tf_static (geometry_msgs/TransformStamped) – used to apply conversions, such as translations and rotations between two different reference frames that are not time-varying
- velodyne_points – 3-dimensional distances pointcloud coming from Velodyne LiDAR sensor
- vs/commands (geometry_msgs/Vector3Stamped) – used to get commands from the user by radio-controller or GUI. Can be used also to force changing supervisor states
- vs/encoders (drover_msgs/RefSpeed) – contains actual motor speed, temperature and torque data retrieved from the CANBus telemetry

Several nodes also contain services to start-stop the different controllers, so to free computational resources when the specific regulator is not in use. Also, resetting behavior or other useful communication such as force changing supervisor state can be achieved to services:

- control_changerow/startControl – used to enable changerow controller
- control_changerow/stopControl – used to stop changerow controller
- control_inrow/statControl – used to enable changerow controller
- control_inrow/stopControl – used to stop changerow controller
- control_openfield/startControl – used to enable changerow controller
- control_openfield/stopControl – used to stop changerow controller
- plan_poly_trajectory – used to compute a new polynomial trajectory by the trajectory server
- start_in_row_hough – used to initialize Hough accumulators for in-row navigation
- start_row_change_hough_srv – used to initialize Hough accumulators for change-row maneuver
- supervisor/open_field_end – used to mark the correct end execution of the open field navigation controller
- supervisor/in_row_end – used to mark the correct end execution of the in-row navigation controller
- supervisor/change_row_end – used to mark the correct end execution of the row-change navigation controller

As the different regulators operates by writing motor setpoints in the same /cmd_vel topic, it is important for the vehicle supervisor to handle the navigation controllers correctly, according to the current operating status.

During nominal operation, the nodes involved for the execution of the different subsystems are as follows:

- /control_chengerow – autonomous change-row maneuver regulator
- /control_inrow – in-row autonomous navigation controller
- /control_openfield – state feedback open field trajectory tracking regulator
- /ekf_se_map – localization state estimate node at map frame (origin in the map origin)
- /ekf_se_odom – localization state estimate node at odom frame (origin in the vehicle startup point)
- /horizontal_tf_broadcaster – used to send horizontal ground plane reference
- /hough_supervisor_node – Hough Transform line detection algorithm implementation to work on 2d distance lidar data
- /mavros_vs – mavros high-level/low-level transmission bridge
- /navsat_transform – used to tansform GNSS information into an odometry message that uses a coordinates frame consistent with the robot's world frame. This value can be directly fused into the localization state estimate node.
- /pointcloud_to_laserscan – 3d to 2d LiDAR pointcloud compression
- /scan_filter_footprint_node – LiDAR 2d pointcloud filtering according to robot dimensions
- /supervisor – vehicle control supervisor
- /tgpsbroadcaster – frame transformation broadcaster, used also to invert preferred direction of the vehicle
- /trajectory_server_v2 – on-demand polynomial trajectory generation node
- /velodyne_nodelet_manager – Velodyne sensor management node
- /velodyne_nodelet_manager_cloud – Velodyne sensor management node
- /velodyne_nodelet_manager_driver – Velodyne sensor management node

The whole implemented ROS software architecture is detailed in the schema below (Figure 50):

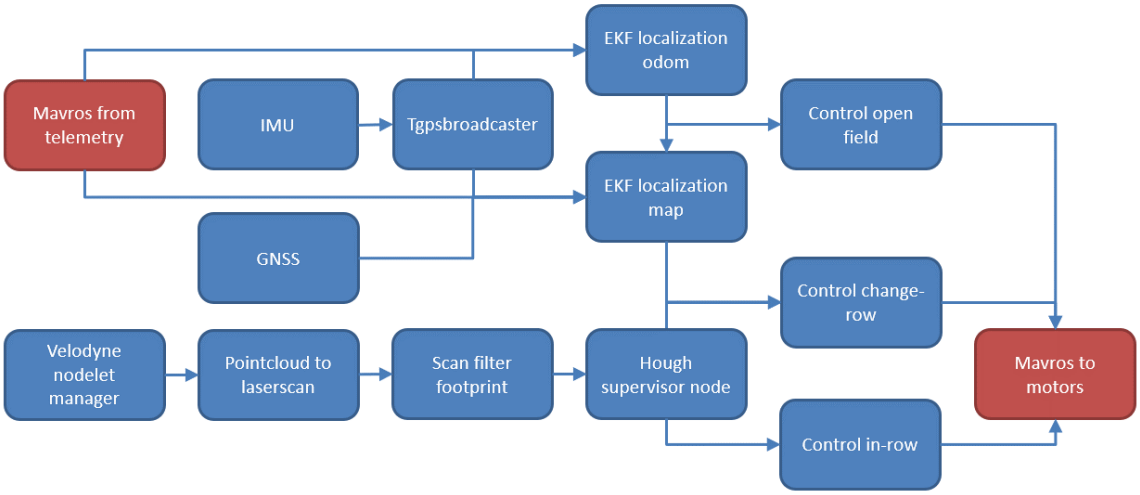

Figure 50 - Vehicle autonomous navigation architecture

3.2. Localization

The vehicle localization node makes use of the robot localization package [45]. The implementation features a multi-rate Extended Kalman Filter able to fuse different type of sensors in order to track a generalized 15-dimensional state of the robot/vehicle $(x, y, z, roll, pitch, yaw, \dot{x}, \dot{y}, \dot{z}, \dot{roll}, \dot{pitch}, \dot{yaw}, \ddot{x}, \ddot{y}, \ddot{z})$. In this notation x, y, z represent the position of the robot in the inertial frame, $\dot{x}, \dot{y}, \dot{z}$ are the linear velocities and $\ddot{x}, \ddot{y}, \ddot{z}$ the linear

accelerations. $roll, pitch, yaw$ are the three angles with respect to the three axes x, y, z that produce the attitude of the vehicle, and $\dot{roll}, \dot{pitch}, \dot{yaw}$ are the three angular velocities.

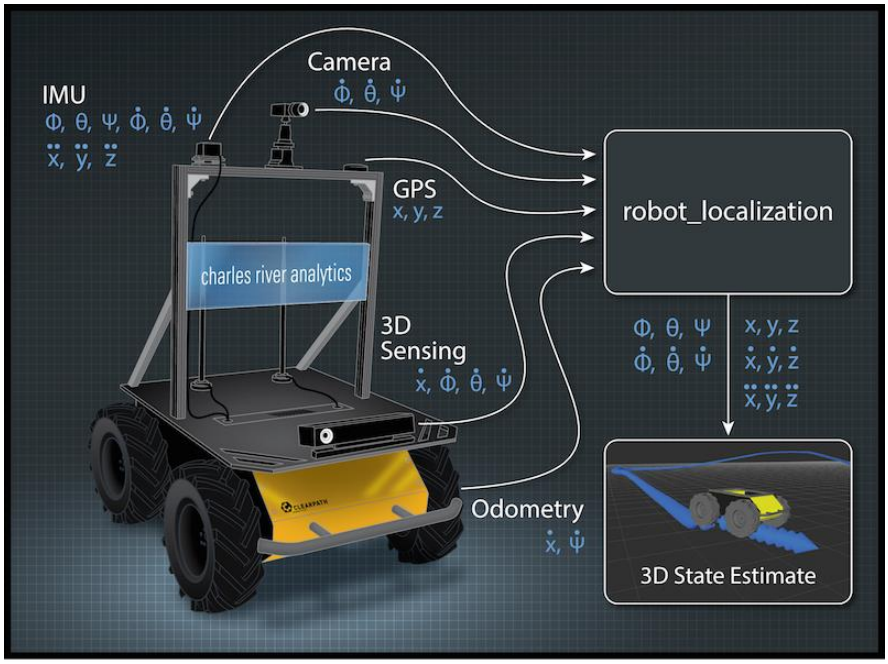

Figure 51 - Robot localization package estimation example

The robot localization node can be configured to read any estimation from IMU, odometry, pose or twist messages, produced by any sensor available, to create a new merged position estimation according to the covariance information given by each source (if available). Each sensor source can be customized to include or reject information for any of the state variables. As typically done for Kalman Filters, if no sensors provide updates for a period of time, the filter can still supply state estimations using an internal motion model.

The implementation onboard the UGV uses IMU data and motor encoder telemetry information to provide localization information in a local odometry "odom" frame. The GNSS estimation is then added to obtain a global position estimation in the global world "map" frame.

GNSS data can be fused with other sensors by transforming the WGS84 datum (latitude and longitude) information into a global cartesian x,y,z position with the origin in a preconfigured *home* position. This operation is in charge of the "navsat_transform" node that subscribes to raw GNSS data and republish odometry GPS information. The whole "robot_localization" package (in terms of EKF nodes and "navsat_transform" node) can be configured to perform 2D (planar) estimations (as in the case of ground vehicles) or full 3D pose estimations (as in the case of aerial vehicles). Obviously, from an agricultural autonomous rover perspective, the whole system has been configured to perform 2D estimations. Moreover, as the vehicle features non-holonomic behavior, the y-axis velocity (that according to the ROS convention traverse the vehicle laterally) has been forced to zero, since the robot cannot translate laterally (assuming that there is no track slippage). Some non-idealities may still occur, but this configuration helps to smoothen the estimation derived mainly from the noisy accelerometer sensors inside the IMU.

As the EKF state estimation node is able to merge any pose information, some preliminary testing has been made by adding a map-based particle filter localization output to the

"robot_localization" node. This allows for a position correction in case of GNSS data inconsistencies. As it may happen to travel near buildings (where multipath behavior is experienced) or under trees (where the water content in the leaves is dampening the GPS signal), the addition of a map-based localization estimation, such as the one provided by the "amcl" node, becomes crucial to allow a smooth navigation of the vehicle and a proper pose estimation in all possible conditions. It is worth noting that map-based localization and GNSS localization perform best in opposite scenarios. The GNSS information is best retrieved in very wide and open areas, without obstacles, while the map-based localization needs obstacle information to operate, so in is best suited for object dense scenarios.

3.3. Open field navigation and obstacle avoidance

The rover has been initially setup to take full advantage of the state-of-the-art ROS Navigation Stack [46]. By implementing this feature, a structured environment is created with a known map that is either available in advance or obtained through recognition across the farm. To determine a set of waypoints leading to the final goal, the robot utilizes standard graph optimization algorithms, such as A* [47], [48] using the provided static farm map. This map can be created exploiting LiDAR distance information together with global localization provided by the robot localization package described above, to obtain a 2D *costmap* of the farm area. As this is mainly required for the open field navigation, the GNSS information is reliable and decisive for the mapping accuracy. The *costmap* data structure is a quantized two-dimensional representation of the space where each pixel can be marked as free or occupied according to the obstacles sensed by the LiDAR and the current position of the vehicle. The *costmap* structure can also be layered to create *inflation* around occupied cells according to the robot dimensions or by overlapping multiple maps representing different obstacles semantics or already known structures or inaccessible areas.

The LiDAR data can also be used to enable obstacle avoidance features, by integrating a local *costmap* with updates given by lidar distances detected. This local temporary map is always centered on the vehicle, so it acts as a sliding window with respect to the global map and can be used to locally update any unknown or mobile obstructions. Using the ROS navigation stack planner is then possible to re-route the vehicle accordingly, thus performing evasive maneuvers around the detected occlusions.

Teach and repeat

Another option that has been implemented and it is typically used also by commercial platforms is the so-called teach and repeat mechanism. In this case, the vehicle is manually driven during a first-time operation, and, in the meanwhile, a list of subsequent waypoints is recorded according to a configured distance threshold. The robot is then capable to read the created list on the next operations, traversing each waypoint and therefore repeating the original path that was performed during the first manual operation. This approach can be useful to improve the vehicle usability, by giving the end-user a simple approach to produce and configure the vehicle with feasible paths for the autonomous navigation inside a known environment and with known obstacles. Using other type of sensors such as cameras or LiDARs, additional simplified obstacle avoidance features can be added to halt the vehicle if a detection is made under a certain distance from the robot. These should be all temporary or unknown obstruction of which the user can be notified exploiting the Internet connection onboard the UGV. The operation can then be resumed if the obstacle is removed.

3.3.1. Trajectory planning[5]

According to a list of given waypoints, it seems useful to develop a *trajectory planner* that can generate navigable trajectories from this list expressed in the world reference frame, and according to the rover's kinematic constraints. As its kinematics can be approximated with the unicycle model, this problem has already been studied extensively in literature. Path planning algorithms such as Dijkstra and A* [49], [50] have been proposed, though these create paths with sharp turns or zigzags due to their use of discrete search spaces. To improve upon this, much focus has shifted towards generating smoother and more efficient trajectories [51]. In [52], an algorithm was developed to produce smooth paths while also accounting for the robot's kinematics; it determines desired velocities according to the curvature radius of each segment. An alternate approach is taken in [53], where trajectories are created with circular arcs and straight lines; however, these pathways are sub-optimal as they were generated via heuristics rather than optimization methods. The implemented ROS node is able to generate trajectories with optimal acceleration, while allowing selection of each waypoint's position, velocity, and acceleration without sacrificing smoothness. Minimizing acceleration is of crucial importance when addressing electrically powered vehicles as overall locomotion efficiency relies also on the ability to maintain constant speed if no sharp maneuvers are necessary. This is the perfect job for the implemented trajectory planner.

Problem definition and regulator applied

In the domain of open-field navigation, it has been addressed the task of trajectory generation by means of a sequence of predetermined, collision-free waypoints $W = [w_0 \dots w_m] \in \mathbb{R}^{2\times m}$. It has been devised a trajectory optimization process that is devoted to producing a safe path going through those points while adhering to both kinematic and dynamic limitations specified by the UGV configuration. This approach is based on hard-constrained trajectory optimization, initially presented in [54] , with an exact formulation described in [55].

[5] This work has contributed to the published paper: Trajectory Planning ROS Service for an Autonomous Agricultural Robot. - Gentilini, L., Rossi, S., Mengoli, D., Eusebi, A., Marconi, L. - 2021 IEEE International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2021 - Proceedings, 2021, pp. 384–389

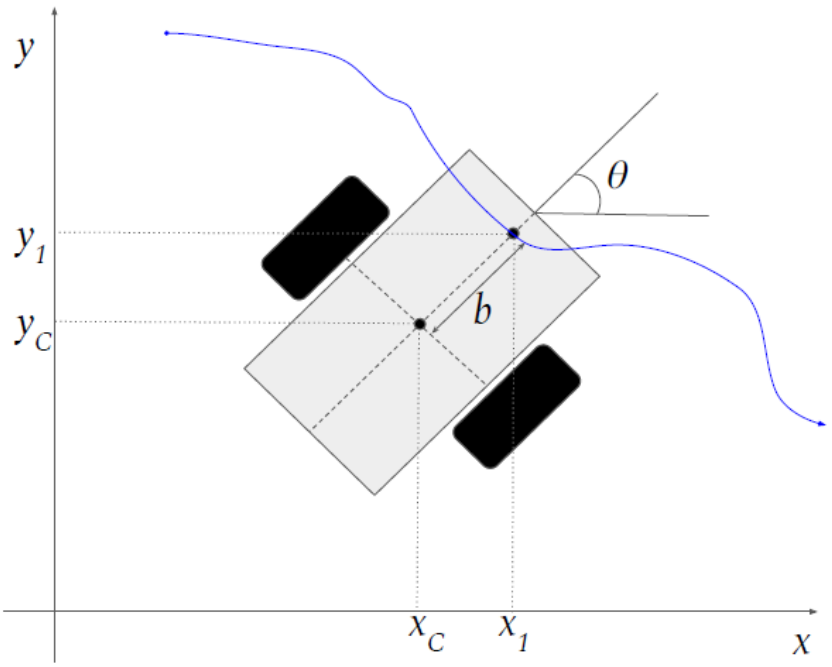

Figure 52 - Position of the linearizing point placed in front of the vehicle (x_1, y_1). The blue line represents a possible trajectory to be tracked.

As part of the planning process, it is essential to carefully design trajectories in order to match the low-level regulator implemented inside the UGV. A unicycle model of the vehicle has been approximated in this case, and the input-output linearization method [56] has been applied. As shown in Figure 52, the UGV model has been linearized with respect to an external point (x_1, y_1), positioned at a fixed distance b from the robot center of rotation, that is the intersection between the central point of the wheels axis and the sagittal robot axis. The obtained system is then defined as:

$$\begin{cases} x_1 = x_C + b * \cos(\theta) \\ y_1 = y_C + b * \sin(\theta) \end{cases}$$

If $b \neq 0$, it can be proved [56] that a valid transformation between a control input applied at this point $[u_1, u_2]$ and the velocity input of the unicycle $[v, \omega]$ always exist. Therefore, in order to control the position of this point, a simple linear controller can be designed as follows:

$$\begin{cases} u_1 = \dot{x}_{1d} + k_1(x_{1d} - x_1) \\ u_2 = \dot{y}_{1d} + k_2(y_{1d} - x_1) \end{cases}$$

using the two coefficients $k_1, k_2 > 0$, and where an appropriate trajectory will define the desired positions and velocities $[x_d, y_d, \dot{x}_d, \dot{y}_d]$ of the linearizing point (x_1, y_1) on the ground plane.

By exploiting the work of [54] developed to control quadcopters, and then extended by [55] to ensure numerical stability and speed up computation through both removing hard constraints and sacrificing the guarantee of asymptotic convergence to the global optimum, the implementation focused on a planner just for $\sigma = [x, y]$. This is because it was given importance to the vehicle position in the space, and each flat output's trajectory can be computed independently. The overall UGV trajectory $\sigma(t)$ is then defined as a piecewise polynomial curve of n order and composed by m different segments as follows:

$$\sigma(t) = \begin{cases} \sum_{i=0}^{n} \lambda_i^1 \left(\frac{t - T_0}{T_1 - T_0}\right)^i & t \in [T_0, T_1], \\ \sum_{i=0}^{n} \lambda_i^2 \left(\frac{t - T_1}{T_2 - T_1}\right)^i & t \in [T_1, T_2], \\ \vdots & \vdots \\ \sum_{i=0}^{n} \lambda_i^m \left(\frac{t - T_{m-1}}{T_m - T_{m-1}}\right)^i & t \in [T_{m-1}, T_m] \end{cases}$$

where the vector $\Lambda_j = \left[\lambda_0^j \ \cdots \ \lambda_n^j\right]^T$ is a set of $n + 1$ coefficients which summarize the geometry of the curve; the quantities T_{j-1} and T_j are respectively the start and end times of the j^{th} trajectory segment.

Implemented solution

The trajectory planner has been implemented by means of a ROS node able to process an indefinite number of waypoints defined as both the planar position of the generic waypoint i ($w_i = [x_i, y_i]$), and a vector of constraints ($C_i = \left[c_i^1, c_i^2, \cdots, c_i^{(n+1)/2}\right]^T$). In this vector, the j^{th} element ($c_i^j = \left[x_i^j, y_i^j\right]$) denotes the constraint applied for the j^{th} derivative on the considered waypoint i, and n is the polynomial curve order.

Specifying waypoints this way allows derivative constraints to be added only to certain points (e.g., the first and last waypoints can have a zero velocity and acceleration, as we want to typically perform a standing start from the initial point and to stop on the last waypoint). Additionally, the user is able to choose a subset of constraints ($C_i = \left[c_i^1, c_i^2, \cdots, c_i^p\right]^T$, with $p < (n + 1)/2$), that will apply to each given waypoint. This will allow to leave the remaining $(n + 1)/2 - p$ quantities to be freely determined by the trajectory optimization procedure in order to best optimize a defined trajectory cost.

In order to maximize the efficiency of the control effort and the trajectory length, a cost function should be constructed that takes into account both elements. For ground vehicles, motor commands are reflected in applied acceleration, so trajectories with less effort typically have low accelerations. However, minimizing the length of the path can result in near-zero accelerations, so we opt to penalize total travel time instead, thus obtaining the following cost function:

$$J(\mathcal{T}, \mathbf{\Lambda}) = k_T \underbrace{\sum_{j=1}^{m} \int_{T_{j-1}}^{T_j} \tau d\tau}_{J_T(\cdot)} + k_e \underbrace{\sum_{j=1}^{m} \int_{T_j - 1}^{T_j} \frac{d^p \left\| \sigma^j(\tau) \right\|_2^2}{d\tau^p} d\tau}_{J_e(\cdot)}$$

where $\mathcal{T}$ and $\mathbf{\Lambda}$ are two vectors of times (for each trajectory segment) and polynomial coefficients (again, for each segment). Obviously, the times vector $\mathcal{T} = [T_0, \cdots, T_m]$ is a vector of scalar numbers (e.g., time required to perform the specific segment), while the $\mathbf{\Lambda} = [\mathbf{\Lambda_1}, \cdots, \mathbf{\Lambda_m}]$ vector contains all the polynomial coefficient vectors for all the segments involved. The two coefficients $k_T, k_e \geq 0$ are useful to balance the two contributions of the cost function, allowing to weight more the time cost with respect to the effort cost or vice-versa. The value of the parameter p can be defined by the user, to allow p-grade derivative minimization. The proposed implementation use $p = 2$. The overall optimization problem can be summarized as follows:

$$\min_{T_1,\cdots,T_m,\boldsymbol{\Lambda}} J(\mathcal{T},\boldsymbol{\Lambda})$$

$$\begin{aligned} subj.to \quad & T_0 = T^*, \\ & A_f(\mathcal{T})\boldsymbol{\Lambda} = d^*, \\ & \|\dot{\sigma}(t)\| \le \dot{\sigma}_{max}, \\ & \|\ddot{\sigma}(t)\| \le \ddot{\sigma}_{max}. \end{aligned}$$

In which the mapping matrix A_f maps the vector of coefficients $\boldsymbol{\Lambda}$ to the vector of constrained derivatives (d^*). The two values $\dot{\sigma}_{max}$ and $\ddot{\sigma}_{max}$ represent the maximum allowed velocity and acceleration, respectively. According to [54], the defined optimization problem can be solved in two steps. First, the problem is completely reformulated as unconstrained quadratic programming (QP) and solved directly for endpoint derivatives as decision variables, instead of polynomial coefficients. The optimization problem becomes ill-conditioned as the number of trajectory segments m increases, making this reformulation of fundamental importance. A suboptimal solution of the overall optimization problem can also be obtained from this QP, which serves as an initial condition for the second optimization step, in which a gradient descend method is used to optimize the overall cost, including travel times as well.

Two-step optimization process

In order to calculate the cost function in the optimization steps, it is necessary to select a priori a baseline for the amount of time required to move from one waypoint to the next one. These traverse times were estimated by considering straight trajectories with constant velocities, and considering half the maximum velocity ($\dot{\sigma}_{max}$) of the UGV. Each delta time is then computed as $\Delta T_{i+1} = \frac{s_{i+1}}{\dot{\sigma}_{max}/2}$, where s_{i+1} is the distance between the waypoints as $s_{i+1} = \|w_{i+1} - w_i\|$. This first times vector will then be refined and adjusted in the last non-linear optimization step.

As a first optimization step, it is considered only the penalty on control effort $J_e(\cdot)$. This allows to reformulate the optimization problem as a quadratic programming (QP) problem. $J_e(\cdot)$ can be rewritten, in matrix form, as:

$$J_e(\cdot) = \underbrace{\begin{bmatrix} \Lambda_1 \\ \vdots \\ \Lambda_m \end{bmatrix}^T}_{\boldsymbol{\Lambda}^T} \underbrace{\begin{bmatrix} Q_1(\Delta T_1) & & \\ & \ddots & \\ & & Q_m(\Delta T_m) \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} \Lambda_1 \\ \vdots \\ \Lambda_m \end{bmatrix}}_{\boldsymbol{\Lambda}}$$

where $Q_i(\cdot)$ is the Hessian matrix resulting from the differentiation of the square of the polynomial according to its coefficients. Using the transformation between the curve coefficients and the end-point derivatives, the new cost function can be written as follows:

$$J_e(\cdot) = \begin{bmatrix} d_f \\ d_p \end{bmatrix}^T \underbrace{A^{-T} Q A^{-1}}_{R} \begin{bmatrix} d_f \\ d_p \end{bmatrix}, with \;\; A\boldsymbol{\Lambda} = \begin{bmatrix} A_f \\ A_p \end{bmatrix} \boldsymbol{\Lambda} = \begin{bmatrix} d_f \\ d_p \end{bmatrix}.$$

By using a set of travel times, like the baseline first estimation described above, that is compliant with the dynamic bounds of the vehicle, the optimization problem turns into:

$$\min_{d_p} J_e\left(\mathcal{T}, d_f, d_p\right)$$

$$\begin{aligned} subj.to \quad & T_i = T_i^*, \qquad \forall i = 0,\cdots,m \\ & d_f = d^* \end{aligned}$$

This describes an unconstrained quadratic programming problem that can be solved by differentiating $J_e(\cdot)$ and equalizing to zero.

$$J_e(\cdot) = \begin{bmatrix} d_f \\ d_p \end{bmatrix}^T R \begin{bmatrix} d_f \\ d_p \end{bmatrix} = \begin{bmatrix} d_f \\ d_p \end{bmatrix}^T \begin{matrix} R_{ff} & R_{fp} \\ R_{pf} & R_{pp} \end{matrix} \begin{bmatrix} d_f \\ d_p \end{bmatrix}, with\ d_p^* = -R_{pp}^{-1} R_{fp}^T d^*.$$

The last step is then to optimize the times vector, as the initial choice made is very conservative and does not fully exploit the UGV capabilities. Additionally, a conservative choice can significantly degrade the final solution because of the way times account into the overall cost. In order to solve the following optimization problem, an initial guess of segment times and d_p is used as a starting solution and the process will iteratively refine them using gradient descent method:

$$\min_{T_1,\cdots,T_m,d_p} k_e \begin{bmatrix} d_f \\ d_p \end{bmatrix}^T R \begin{bmatrix} d_f \\ d_p \end{bmatrix} + k_T \sum_{i=1}^{m} (T_i - T_{i-1})$$

$$\begin{aligned} subj.to \quad & T_0 = T^*, \\ & d_f = d^*, \\ & A\mathbf{\Lambda} = [\mathrm{d_f} \quad d_p]^{\mathrm{T}}, \\ & \|\dot{\sigma}(t)\| \leq \dot{\sigma}_{max}, \\ & \|\ddot{\sigma}(t)\| \leq \ddot{\sigma}_{max}. \end{aligned}$$

As a result of the aforementioned problem, a constrained non-linear and non-convex optimization problem is obtained, whose solution strongly depends on the initial conditions. Incorporating state limitations (i.e., maximum velocity or acceleration of the vehicle) can be applied with caution, as a wrong choice may cause an extensive decrease in optimization speed or result in an infeasible solution. An alternative is to set inequality constraints for discrete velocity and acceleration trajectories at each interval of time. Unfortunately, this inflation of inequalities might lead to a significant drop in optimization performance, especially when the trajectories are long. Therefore, it has been tried to evaluate the maximum induced velocity and acceleration at every optimization step, similarly to the method described in [57]. The maximum velocity can be estimated by equalizing to zero its gradient with respect to time, as detailed in the following equation:

$$\frac{d\|\dot{\sigma}(t)\|}{dt} = \frac{2\left(\sigma_x(t)\ddot{\sigma_x}(t) + \sigma_y(t)\ddot{\sigma_y}(t) + \sigma_z(t)\ddot{\sigma_z}(t)\right)}{-\sqrt{\dot{p_x^2}(t) + \dot{p_y^2}(t) + \dot{p_z^2}(t)}}$$

on which the Jenkins-Traub algorithm [58] can be applied to compute the real roots of the numerator.

Experimental results

The proposed results have been produced using the following parameters, detailed in Table 6.

Table 6 - Configuration parameters for the implemented trajectory generation ROS node

Curve Order (n)	5	Max Iterations	1000
Maximum Velocity ($\dot{\sigma}_{max}$)	1.0	Effort Gain (k_e)	1.0
Maximum Acceleration ($\ddot{\sigma}_{max}$)	0.8	Time Gain (k_T)	0.1
Derivative to Optimize	2	Soft Constraints Gain (k_{sc})	0.1
Relative Step Size	0.1	Soft Constraints Epsilon (ϵ)	1

The Figure 53 shows the computed trajectory by minimizing the overall acceleration, thus control effort. In this case, the corresponding velocity and acceleration values (in modulus) never surpass the maximum allowed bound. Furthermore, since we minimize the induced acceleration, its values result very close to zero.

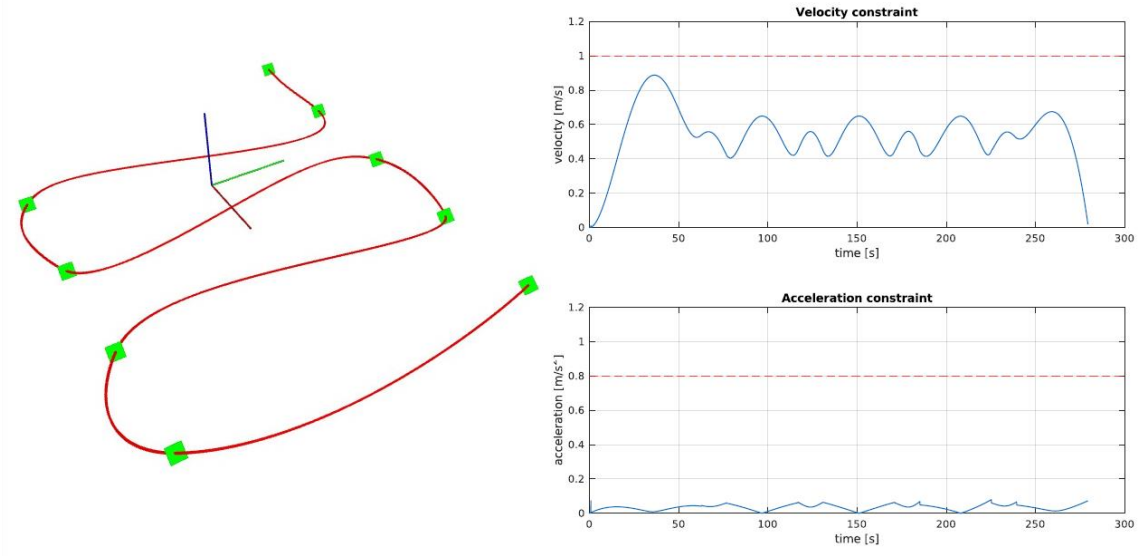

Figure 53 – Left side: off-line trajectory generation test (red line) according to a list of given waypoints (green dots). Right side: corresponding velocity and acceleration values

In order to better refine the solution, it is possible to tune and increase the maximum iteration parameter at cost of computation time. This applies only to the last step of the non-linear optimizer. To check the scalability of the implementation, the same parameters detailed in Table 6 were used to run the trajectory generation multiple times with a different number of waypoints, starting from two and then incrementing by one at a time. It is clearly showed in Figure 54 that the computational time grows exponentially with the number of waypoints. However, processing time is also related on the maximum permitted velocity and acceleration, in addition to the relative waypoint locations.

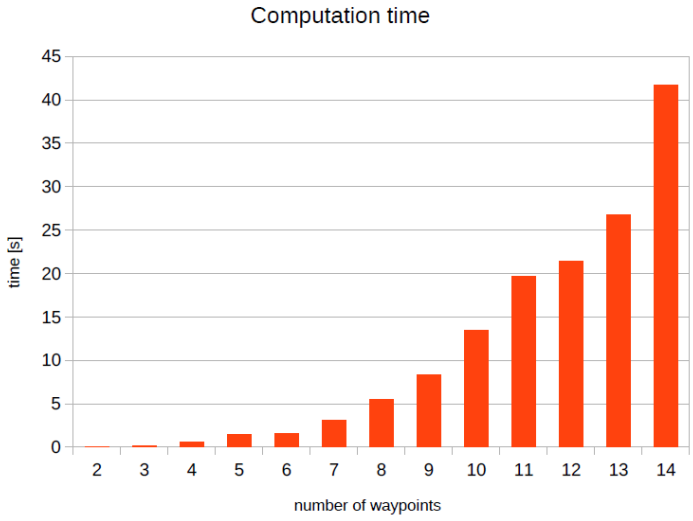

Figure 54 - Trajectory generation processing time with respect to the number of waypoints given

The trajectory generation has been also tested in simulation, using a series of waypoints in an obstacle dense forest scenario. Figure 55 shows the results of the trajectory optimization according to the mapped area. It is worth noting that the implementation imposes the trajectory to pass exactly in the given waypoints, thus, in this case, avoiding obstacles.

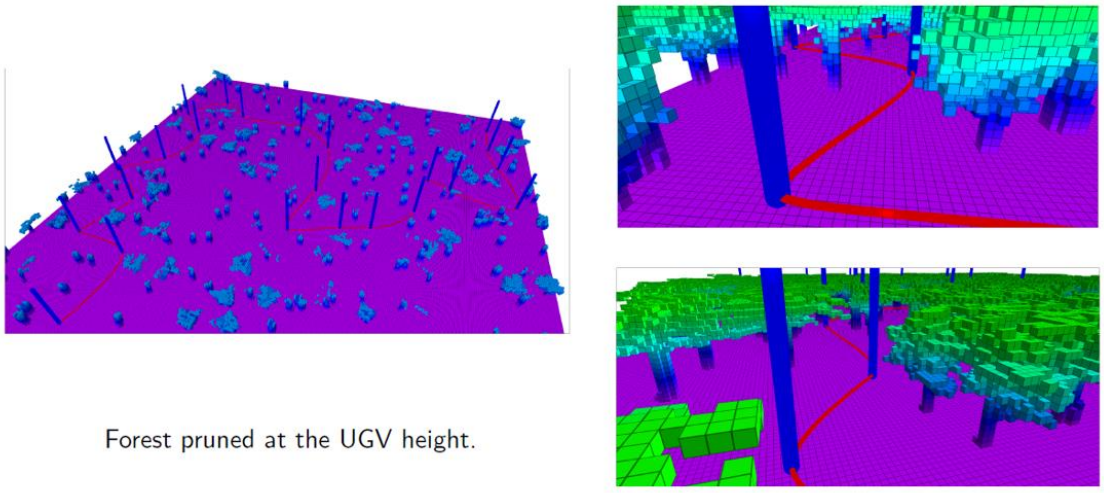

Figure 55 - Trajectory generation in a simulated cluttered scenario

The ROS node was also used to perform open-field navigation of the described UGV by reaching the orchard entrance from the recovery area of the vehicle. The vehicle path is computed onboard using yellow waypoints depicted in Figure 56. These are located along a narrow road leading to the first lane of fruit trees (white circle). Once arrived here, the vehicle should switch over to the in-field navigation controller to continue its mission. The starting point (red circle in the picture) is placed outside the storage shed used by the vehicle, to allow proper GNSS position estimation.

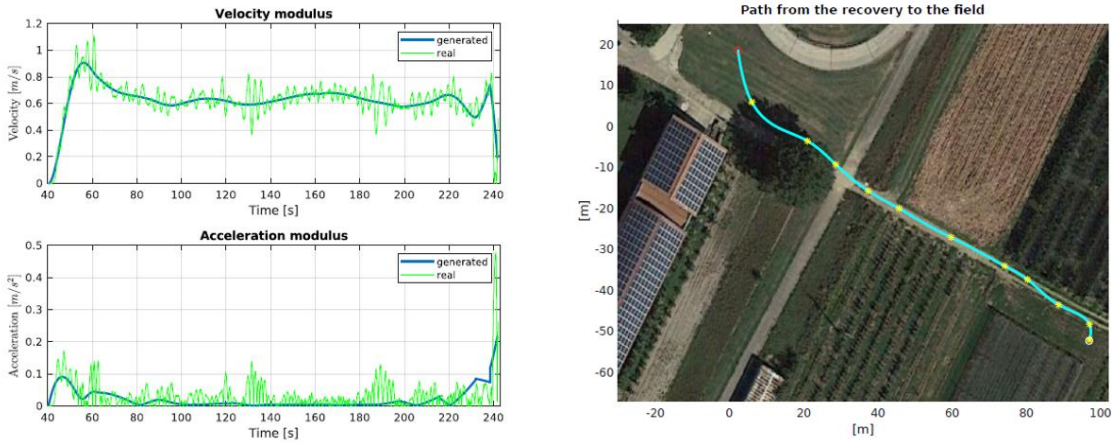

Figure 56 - Trajectory generated during real-world tests. Left side: commanded and real velocity and acceleration values. Right side: planned generated trajectory to reach the experimental orchard.

The processing time required to compute the trajectory is about 30-40 seconds using onboard high-level hardware, using the same parameters defined in Table 6, only altering maximum velocity and acceleration, according to the vehicle capabilities: $\dot{\sigma}_{max} = 1.2, \ddot{\sigma}_{max} = 1.5$. In the same picture, the commanded velocity and acceleration are compared to the vehicle's actual induced velocity and acceleration during the trajectory execution. As a result of the size of the

motors used for locomotion and the configuration given to the motor controllers that has then been adjusted in the next prototype versions, a little amount of oscillation can be seen in the real values obtained from the telemetry data and the localization filter. In order to manage this side effect, a more finely tuned regulator could be used to dampen this behavior. The current implementation has exploited some tuning of the regulator that has been tested in a simulation environment with the following parameters: $k_1, k_2 = 1.2$.

As a side effect of the used control technique described in this section, it is not possible to control the yaw angle θ during the navigation. This limitation can be accepted as only the last waypoint vehicle orientation is critical to the mission requirements, as from this position, by switching the onboard regulators to the in-row navigation, the system expect to have the first orchard lane in the front. To overcome this, multiple waypoints may be imposed in the last part of the trajectory, or an alignment maneuver can be made upon trajectory completion (thus rotating the rover in place to achieve the desired final yaw θ angle). For future developments, it should also be possible to integrate vehicle orientation into the developed implementation to also generate a trajectory reference for θ.

3.3.2. Marker navigation

Another option investigated and developed to perform autonomous open-field navigation is to exploit the fact that the farming environment is generally semi-structured, and a little amount of infrastructure can be placed or exploited to provide commands to the UGV. In particular, this can be done by installing some ArUco markers in the fields that contains a specific command according to the number that they represent. ArUco markers was primarily developed to provide fiducial points in the environment to autonomous vehicles, in order to improve localization capabilities. The same marker, properly placed in the field, and by implementing an appropriate dictionary onboard the vehicle, can be used to automatically and indirectly give commands to the autonomous platform upon marker detection with onboard cameras.

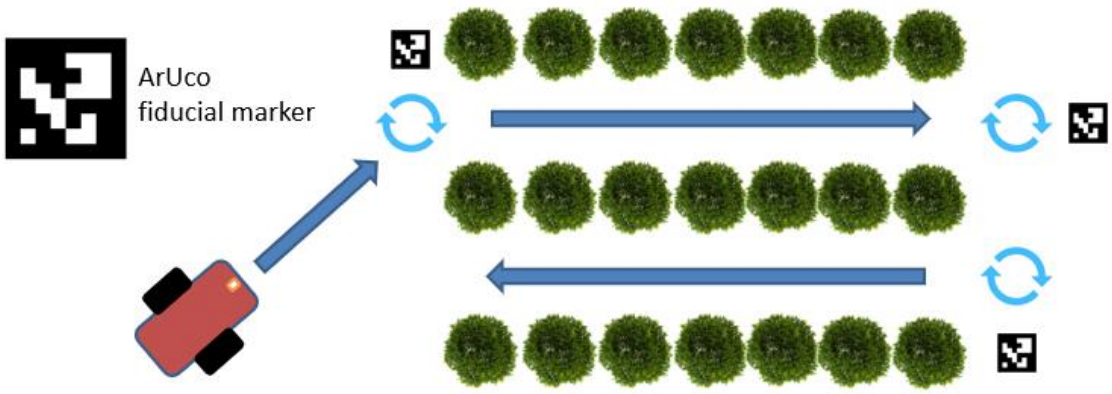

Figure 57 - ArUco maker navigation concept

The above Figure 57 shows the concept of the described autonomous driving. If the vehicle can detect an ArUco marker, it starts travelling towards it. Upon reaching the marker, it can execute the operation definite by its identification number (the ArUco number representation). Then, the vehicle continues to navigate in the commanded direction until a second marker is detected. The mission can continue as described above, by reaching the next marker and then executing the operation with respect to the id number retrieved from the marker itself. In this way, it is possible to traverse both open-field paths and orchard rows, as long as the markers are

sufficiently close one to each other to be detected by the vehicle camera after a reasonable distance from the previous one.

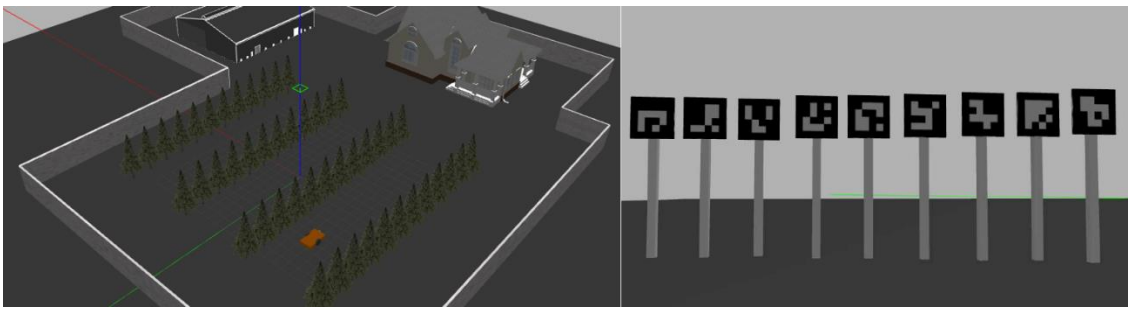

Figure 58 - Gazebo simulation (on the left) used for the ArUco markers navigation (on the right)

This method allows the farmer (as the end user) to place these ArUco markers across the field to instruct the vehicle motion without the need of any software configuration. Some simulation testing has been made to validate the solution (Figure 58), that can then be made as complex as will by increasing the number of markers involved. The implemented regulator is able to perform both in-place turning maneuvers upon reaching a marker (thus facing the direction commanded by the marker ID), and travelling towards a distant marker while also preserving an offset. The latter feature is particularly useful to not collide to the marker itself and to allow subsequent markers to be placed at a side of an orchard row.

The trajectory obtained while traversing the simulated orchard lanes is showed in Figure 59. This technique can also be used combined with the *teach and repeat* pattern, in which the teach phase is replaced by the deployment of the markers. Then, after a few navigation tasks, the platform is able to create its own map of the surroundings and store the paths traversed, thus being able to replicate the trajectory also without the help from the markers.

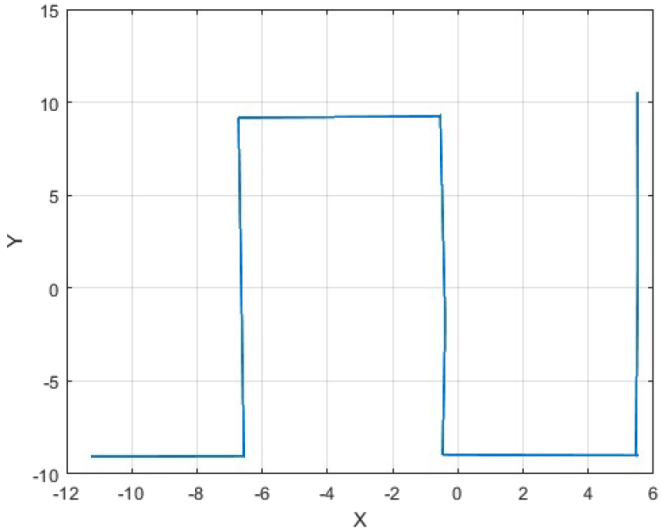

Figure 59 - vehicle trajectory performed while following orchard markers in simulation environment

3.4. In-row navigation

Technology today is mature enough to provide an automated robot platform that can also be used in other agriculture fields, such as orchards, where profit margins are not as high as in

vineyards, but automation is still crucial to increasing companies' competitiveness and sustainability.

It is in this context that Orchard Precision Agriculture (OPA) is developed, in which robots perform tasks based on environmental conditions, distributing fertilizer or water only where and when needed, thereby optimizing treatments [59], [60].

OPA also provides a pose estimation challenge. Open-field navigation is possible using GPS positioning but relying only on this sensor is not recommended as it cannot account for obstacles and would require high-end and costly receivers to be accurate enough. The research works in [61] and [62] have demonstrated that relative localization to trees is feasible and can be GPS independent, thus enabling accurate in-row navigation. Bergerman's group made use of specific markers at the ends of tree rows to detect lane terminations, which the solution described in this work plans to avoid by exploiting track motor encoders, and also to make end-of-row detection more reliable, and free of any added infrastructure.

Navigation in orchards, where trees are organized in rows, can be divided mainly into two parts:

- navigation within the rows,
- row-change maneuver.

These aspects allow fully autonomous navigation in the orchard during any agricultural task.

Navigation between rows can be studied [63] using the Hough Transform (HT) [64], [65] line detection algorithm, which, in the presented implementation, is able to detect the tree row line from a set of 3D LiDAR points. Furthermore, the LiDAR pointcloud can also be exploited in row-change maneuvers to detect the next lane entrance and providing a feedback for the turning controller.

3.4.1. Row detection and row-following[6]

Given the particular structure of the orchard, with its rows of trees visually defined by their alignment and uniform dimensions, there is no improvement in employing global localization systems, specifically when accuracy might not be as high as expected (such as with low-cost sensors) or considering the dynamic nature of the environment (for instance, foliage growth). On the other hand, an accurate local position is crucial for keeping the robot on route. Consequently, it follows that a two parallel lines model can be employed for the intra-row environment detection using a Hough Transform (HT) line detection algorithm [64]. Unlike other works that exploit the same line detection system from an UAV aerial image perspective [66], this method aims to apply the HT algorithm using a conveniently processed 3D LiDAR pointcloud.

The LiDAR 3D pointcloud is flattened to obtain bidimensional features image, where the HT line detection algorithm can be applied to each detection point. The HT line detection algorithm works by applying a vote on each possible line that pass for the specific point [67]. As the process is repeated for each feature/point in the image (or detection as in the specific LiDAR case), and a vote is applied for each different line that pass for the point by applying angle quantization,

[6] This work has contributed to the published paper: Autonomous Robotic Platform for Precision Orchard Management: Architecture and Software Perspective. - Mengoli, D., Tazzari, R., Marconi, L. - 2020 IEEE International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2020 - Proceedings, 2020, pp. 303–308, 9277555B

the algorithm then sum up all the occurrences in the so-called Hough space to find the most relevant line contained in the image (Figure 60).

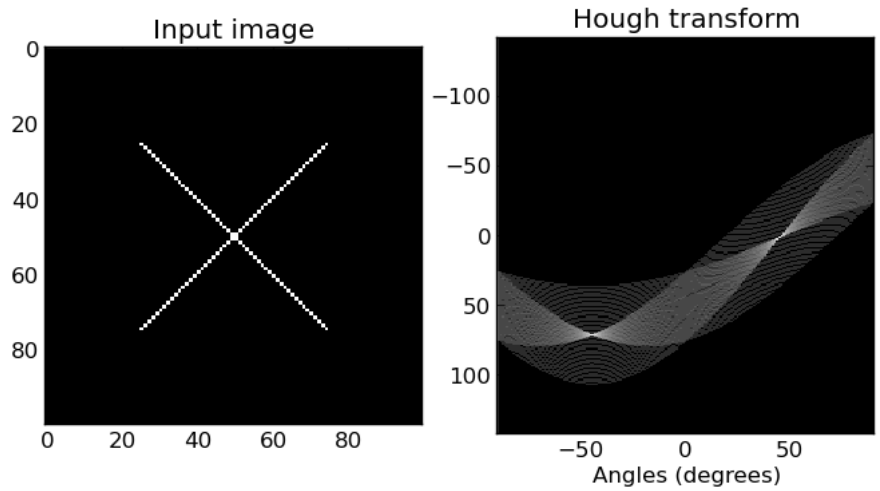

Figure 60 - Hough Transform line detection. A comparison between input image and votes in the Hough space

The Hough space is defined by using the Hesse normal form representation of lines, given by:

$$r = x\cos(\theta) + y\sin(\theta)$$

in which r denotes the distance from the origin and the line closest point, while θ is the angle between the x-axis and the orthogonal distance intersecting the line, as shown in Figure 61.

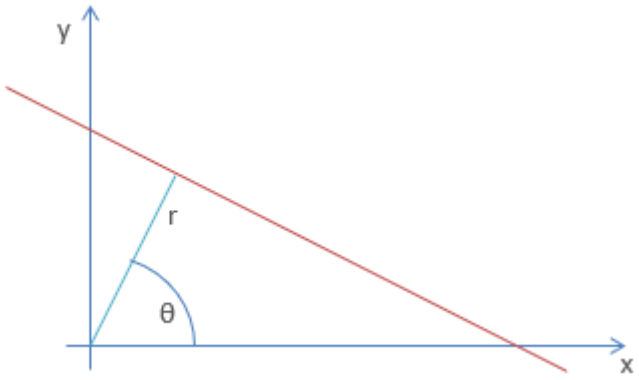

Figure 61 - r and θ definitions for line identification

In the prototype vehicle implementation, this algorithm has been completely rewritten to enable improvements and a better control on the voting system, so to have and extra degree of configurability and adaptability of the UGV with respect to different orchard training systems. This would have been impossible by using pre-defined and already available implementation of the line detection algorithm. Furthermore, considering that this estimation is the foundation of the autonomous navigation functions developed, it seems appropriate to directly acquire the knowledge behind the algorithm and applying a custom implementation of it.

By requiring parallelism and distance constraints, the algorithm looks for two distinct lines at the same time about the lane width. This redundant recognition, enforced by parallelism and distance constraints, increases the algorithm's robustness.

One of the improvements made on the algorithm shows that it is possible to optimize the computational complexity of the HT algorithm for line detection, and at the same time enhance its robustness and reliability.

Within the orchard, the row structure is kept throughout the entire row length, so after the initial HT runs on the whole space, the search grid can be diminished. The algorithm then looks to find lines that are similar to those before (e.g., during rover navigation thorough a row), as visible in Figure 62. This means that line estimate solutions do not require to scan the whole Hough space matrix, but only consider the cells that are nearby prior solutions. Additionally, the improved search span is supposed to improve robustness to misdetections due for example to missing trees along a row. In this case, one of the two lines would be strongly different from the previous one, so the reduced search span automatically eliminates those votes. This is valid not only for the very same point of view of the rover travelling inside the orchard lane, but also, more generally, for absolute tree line orientation with respect to north, since orchards are usually planted in straight lines, and trees does not move over time. The filter still allows for some adjustments that accounts for possible imperfections during the estimation process, and also for some adaptation if the tree rows are not deployed exactly in a straight fashion, but performs some bending and curves as seen in some vineyards.

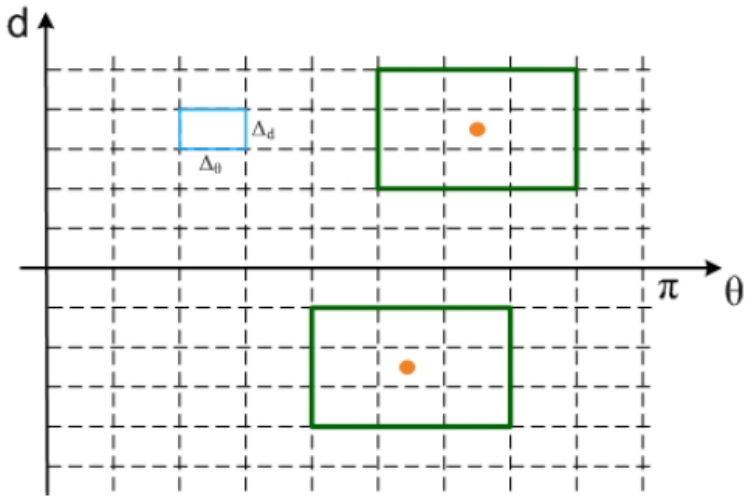

Figure 62 - Reduction of the whole Hough space by considering only votes in the nearby of previous solutions

As a plus, the HT line estimation data can be viewed as an additional heading and lateral displacement information in this navigation scenario, with respect to some *orchard-fixed* reference frame. Therefore, it can be used as an additional sensor to be given as input for the localization EKF. To obtain a refined position estimate, while traversing the orchard lane, the EKF can fuse together IMU, motor encoders, and HT estimates.

Reliable detection of the end of the orchard row is crucial for prompt lane-switching trajectory planning and the available LiDAR data can be used for this. As the end approaches, points belonging to lines behind the robot are detected instead of those in front and this triggers the row-change maneuver. However, there may be missing trees on either side of the robot which could lead to false positive detection. Previous research from Bergerman's group [62] attempted to tackle this by placing optical markers at the ends of each row but these are vulnerable to outdoor elements such as weathering, and products sprayed by farmers which may reduce their visibility. Also, any added infrastructure should be avoided in order to not occur in an increased

maintenance of the orchard. To resolve this, a solution not reliant on these markers is suggested and implemented in the vehicle.

As probably the orchard area is already defined, it is possible to compute the length of each lane using the GPS boundaries points that define the shape of each orchard field and that are used for open field navigation. In combination with the global position provided by the EKF, this data can be used to trigger end-of-row recognition, but only when actually approaching it. As an alternative, with less configuration effort, it has also been implemented a rough row length configuration as it will be presented in the next Section 3.4.2.

The onboard controller is then designed to keep the middle position within the row, by applying yaw speed reference setpoints while traversing the orchard, according to the estimated distances of the vehicle with respect to the left and right tree rows. The forward speed is controlled to be held at the fixed configured velocity according to the specific farming task. Furthermore, if an application requires two or more traversals of the same lane, such as example while using the mulcher for grass mowing in wide lanes, the distance can be adjusted pull the vehicle over to the left or right side as needed.

Experimental results

The lines estimation and navigation capabilities have been tested inside the experimental apple orchard in Cadriano (Bologna, Italy) using the developed UGV platform.

According to the navigation test performed, the robot was placed to start operations at the beginning of a row in the orchard. To benchmark algorithms, the distance between the UGV and the left and right tree rows is measured. In order to meet the requirements imposed by the mission type, measurements of the trajectory's smoothness are recorded. As shown in Figure 63, the output of the HT line detection algorithm is given by two lines (green and blue) extracted from the LiDAR data as an estimate of the tree rows.

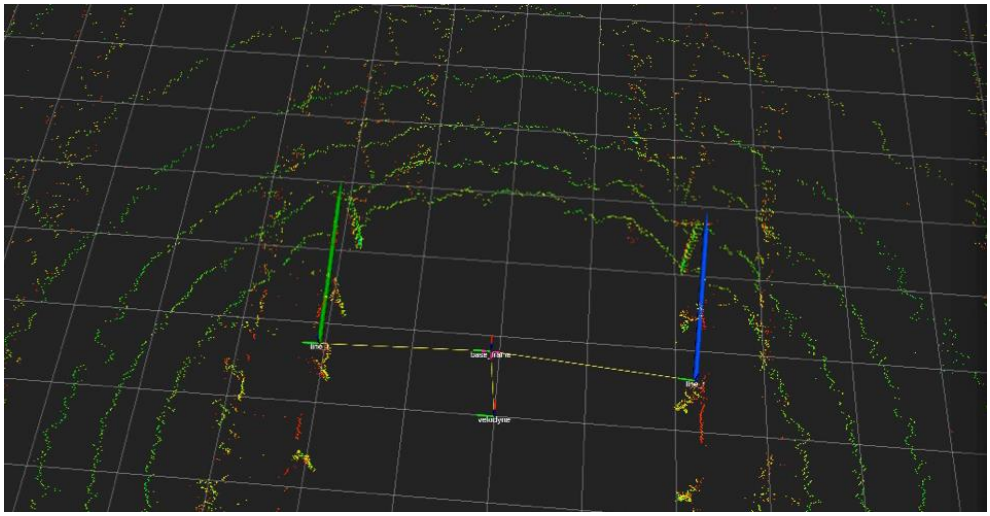

Figure 63 - Hough transform line detection algorithm output according to LiDAR data

It has been noted that some noise is observed in the row/line distance values during operation. This is primarily caused by the irregular vegetation of the trees (Figure 64). Moreover, the correct distance is maintained as the rover proceeded in the middle of the lane: the rows are about 4 meters wide in this case, and the average distance from both sides is 2 meters.

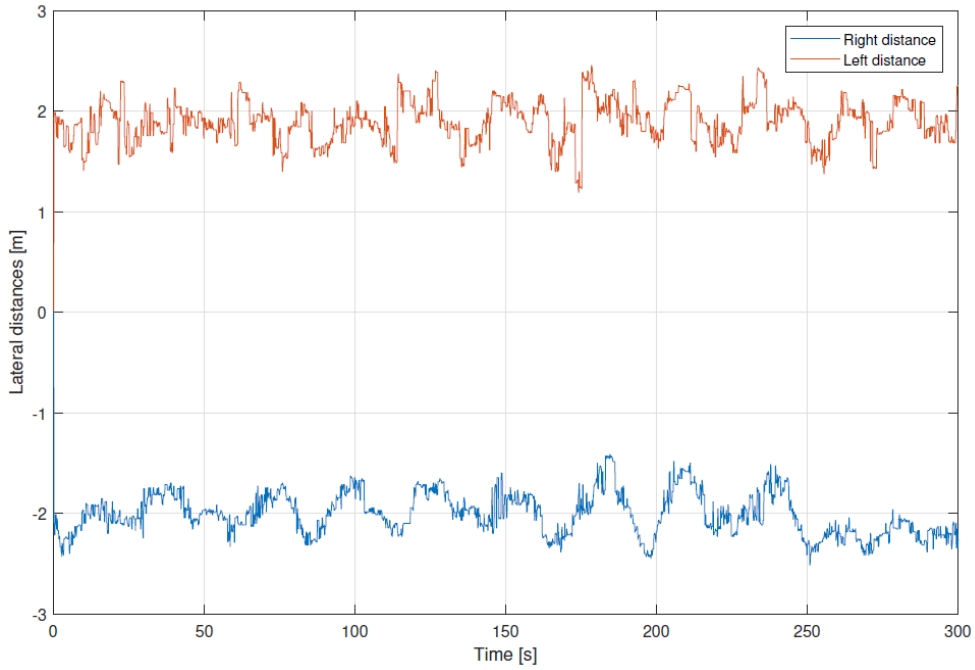

Figure 64 - Hough transform line estimation distance with respect to rover position during navigation

As the farming operation may be aware of vegetation and branches that protrude significantly from the main trunk, detecting foliage and possibly spurious invasive branches of vegetation while performing autonomous navigation is generally a good feature. The control algorithm adjusts the vehicle's distance from the canopy detected to compensate for this, similar to what a human operator would do.

HT line detection has been proven not to be affected by potentially missing or very thin trees, since multiple points are *voted,* and several trees are interpolated to find the correct orientation. Also, the reduced search span minimizes the likelihood of spurious vote peaks. Moreover, the intrinsic robustness of the closed-loop controller used to track the planned trajectory makes it less sensitive to glitches or temporary misplacement of the row line estimation.

3.4.2. Row-change maneuver[7]

To navigate autonomously, the robotic vehicle must go through two steps: navigation between rows and lane changing. Movement between rows can be governed by a navigation algorithm based on a LiDAR sensor and Hough Transform line detection algorithm, as described in the previous Section 3.4.1. On the other hand, as for the lane change maneuver, this task has been very challenging due to the angle of the sensor's viewpoint during the turn and the high accuracy required for insertion into narrow rows (whose width could be slightly larger than that of the robot) [61].

The lane change maneuver can be decomposed into the following steps:

1. detecting the end of the current lane while navigating
2. switching from the current lane to the next (adjacent) lane.

[7] This work has contributed to the published paper: Robust autonomous row-change maneuvers for agricultural robotic platform. - Mengoli, D., Eusebi, A., Rossi, S., Tazzari, R., Marconi, L. - 2021 IEEE International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2021 - Proceedings, 2021, pp. 390–395

3. correctly aligning the robot with the new lane and entering it.

To perform this maneuver, it is assumed that the robot is roughly pre-configured with respect to the structure of the orchard. In particular, the following parameters are required:

- lane width (+/-25% tolerance)
- expected lane length (approximately, as the system will search for an accurate row end. About 80/90% of the minimum expected length is recommended).

The robotic platform must autonomously detect the end of the current lane, then start estimating the beginning of the next one as it begins the turning maneuver that will finally end in front of the new lane, after which navigation within the row can be resumed.

Some of the main problems encountered during the maneuver are:

- the occlusion of LiDAR sensor row detectable points due to the presence of robot parts and tree canopy, especially when the sensor is aligned with the middle row,
- the reduced space at the end of the lane that severely limits the maneuvering radius,
- the accurate control of the distance between the rover and the end of the rows, to avoid bumping into the rows,
- the required sufficient accuracy in inserting into the new row, as the experimental tests carried out on 2 m wide rows with a 1.3 m wide robot.

In order to perform a robust change-row maneuver, without mapping or georeferencing each point or boundary of the orchard that would be complicated for the end user and requires an accurate field survey, the proposed solution aims to refine Hough Transform line detection algorithm described in the previous Section 3.4.1 to improve rows detection during the turning maneuver and implements multiple approaches to the maneuver by exploiting different turning shapes (i.e., circular trajectories, rectangular trajectories, etc.).

LiDAR points pre-processing

The row estimation system is based on the LiDAR sensor installed in the UGV prototype, which is capable of generate up to 0.3 million points per second with an accuracy of +/-3 cm. This huge amount of data needs to be pre-processed to relieve the computational load and to filter out those points that do not belong to the row, e.g., points detected on the ground plane or detections that belongs to object installed on the rover itself, such as the implement.

Similarly to what has been described in the row estimation section 3.4.1, the 3D point cloud obtained from the LiDAR sensor is flattened to a 2D distance scan, by means of the "pointcloud_to_laserscan" ROS package. By doing this process, most of the terrain points are discarded as an appropriate height interval is considered, according to the height position of the sensor and the size of the trees. Furthermore, the maximum visibility distance of each detection was limited to 20 m, while the minimum distance value was defined according to the size of the rover in the x-y plane. It is worth noting that the height conditions imposed are expressed in the sensor frame, that is solid with the robot frame. As the robot may tilt or change its attitude due to the harsh terrain, it is important to detect the orientation of the ground plane to allow a proper rotation of the LiDAR point cloud and ensure that the height threshold values are actually referred to the ground. Finally, it has been filtered out residual detected points that indeed belong to the rover itself: this is easily done by applying a rectangular mask according to the rover dimensions and discarding all those points that fall within it.

Thresholding for line detection

After filtering the pointcloud according to the height of the points in the world/ground frame, the Hough Transform algorithm can be executed to find the lines representing adjacent rows. To ensure that both estimated lines pass through the outer part of the tree canopy, a kind of linear filter has been implemented that increase the voting weight within the Hough accumulator to the lines that are closer to the vehicle, according to a configurable coefficient. Therefore, LiDAR detection closer to the vehicle will account more in each Hough accumulator with respect to farther detections. Consequently, the most probable detected line should ideally be the result of the nearest detections around the robotic platform. This was done because when traveling along rows with very large trees, the estimated lines could go through different parts of the tree; for example, the left line could cut through its trunk and the right one fit its outskirts instead: these uncoherent estimations may lead to an inaccurate row center position.

Finally, a parametric threshold has been introduced that defines a minimum number of points within the Hough accumulator to make the estimated solution acceptable. At the same time, it is also checked whether the lines found agree with the distance and orientation constraints, and if not, the current solution is discarded in favor of the previous interaction, which is then used as a starting point for the next one, by maintaining the neighborhood search space in the Hough accumulator. This further improvement helps when the number of detected points is very small because of occlusions, missing trees or canopy, or when approaching the end of the row.

The full processing pipeline, starting from the raw point cloud to the final estimated lines is shown in Figure 65. From the raw 3D point cloud given by the LiDAR sensor, a first transformation allows obtaining a point cloud with respect to the horizontal ground reference frame. Then, it is possible to flatten the 3D point cloud by retaining only those points that satisfy a height constraint. Knowing the size of the rover, it is also possible to filter out any readings that collide with the rover itself. Once the filtering is finished, the Hough Transform algorithm can be applied to detect relevant lines within the 2D point cloud. All the estimated relevant lines are then filtered based on the expected distance between adjacent rows, and non-parallel lines are removed. Finally, an orthogonal line will be selected to be used as the end of the rows. The final output of this process is then used as an input to the rover control system.

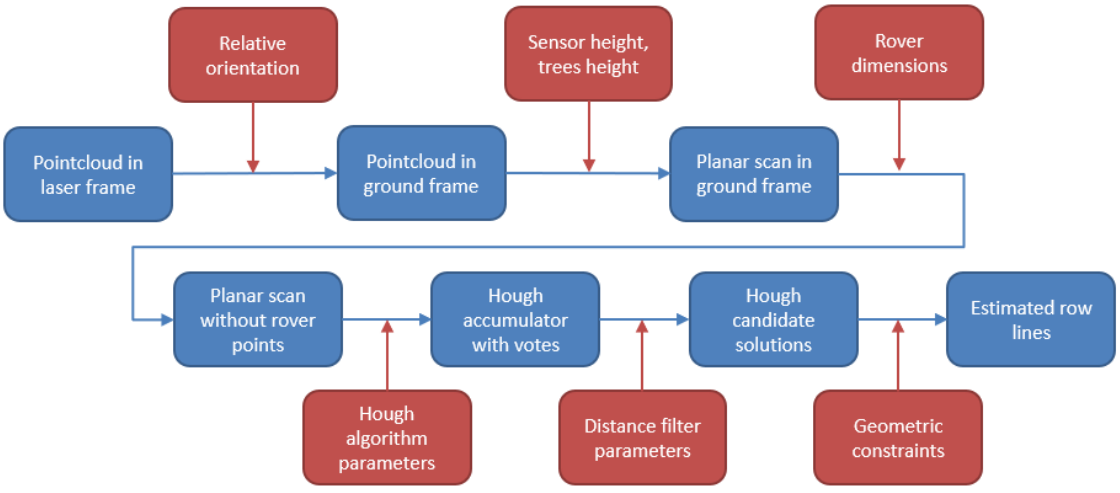

Figure 65 - LiDAR data processing pipeline

Lines estimation during row-change maneuver

The row-change maneuver is composed of autonomous end of lane detection and vehicle conversion from the current lane to the next one, that is, the actual maneuver. As seen for the in-row navigation, the navigation system during lane change is based on the lines estimated by Hough Transform's ROS node. This case is a bit more complicated as the following difficulties are present:

- During the initial part of the maneuver, the row to which the rover is turning is most likely hidden by the center row, thus the LiDAR sensor receives very little points information from that row.
- The same situation, but from the opposite side, occurs in the final part of the maneuver, where the row being abandoned is almost completely covered for the same reason.
- It is necessary to obtain an estimate of the distance of the rover from the beginning of the affected rows, in addition to the relative position with respect to the lines passing through the tree rows. This is for two reasons: first, because you want to avoid any collision with the trees themselves or crossing the boundaries of the orchard, and second, because this is critical information for executing a correct turning trajectory.

To overcome these problems, it has been developed a dedicated line detection system during lane change in which four lines are estimated. As shown in Figure 66, the following lines are detected:

- the two lines l_1, l_2 belonging to the old traversed lane,
- the line l_3 belonging to the new lane,
- the virtual line l_n passing through the end of the orchard rows.

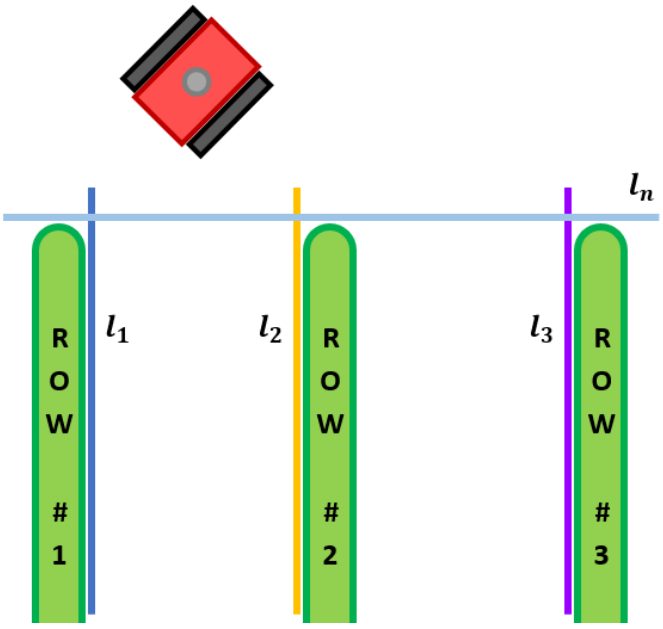

Figure 66 - Lines estimation during row-changing maneuver

These four lines are estimated by the same method described before, using the Hough Transform line detection algorithm processed on the filtered 2D pointcloud obtained from the LiDAR sensor.

For each estimated line, a solution is obtained consisting of a pair (θ, d_i) that expresses the orientation and distance of the i-th line with respect to the rover. It should be noted that d_n represents the distance of the rover from the virtual line passing through the ends of the rows: this is used for both for the control system and for the transition from the row change maneuver to the in-row navigation phase. The estimated values detected by the Hough Transform algorithm are depicted in the Figure 67 below.

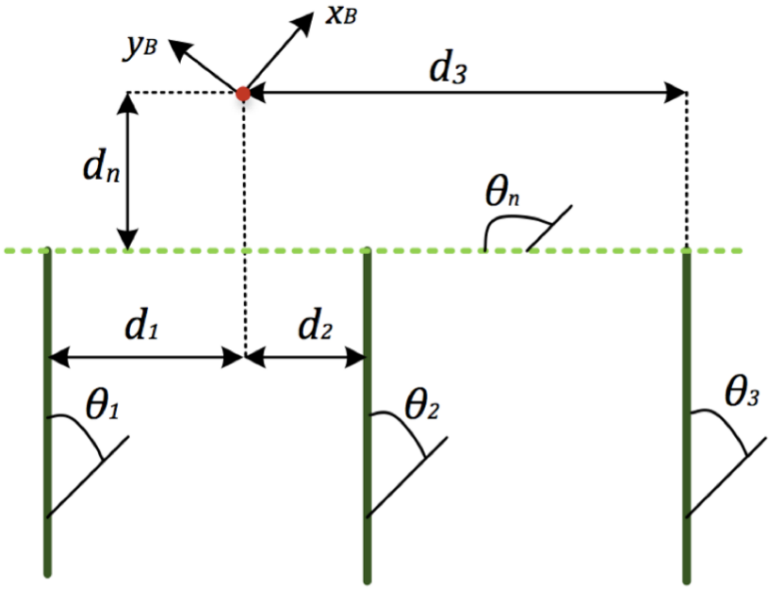

Figure 67 - Hough Transform estimated parameters with respect to the detected rows

In the case of lane changing, the implemented solution has been developed by exploiting the previous in-row navigation reasoning. First, the last solution given by the in-row navigation is exploited as a starting point for the row change line estimation system: this will reduce the search space for lines l_1 and l_2 to a neighborhood of their last known solution. Next, distance and parallelism constraints are imposed between the estimated lines, to further refine the applicable area in the Hough space for the voting mechanism also for l_3. Finally, orthogonality constraint is imposed to the row end line l_n. Further optimization can be adopted for subsequent iterations, by reducing the Hough search space for the estimated lines to a neighborhood centered on the solutions of the previous one. Figure 68 schematically shows Hough's accumulator with generic row-change solutions and the corresponding search envelopes.

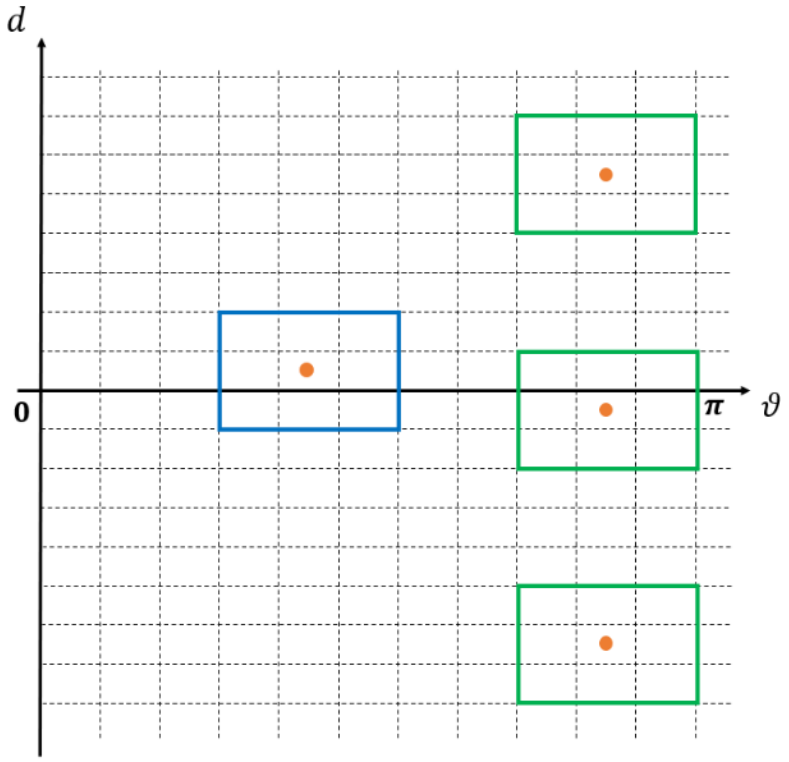

Figure 68 - Hough space search areas (green and blue boundaries) according to previous solution (orange dots). Notice the orthogonal end-row line with blue boundary

Since the early tests of row change maneuvering, it has been noticed that LiDAR data from any orchard or structure facing the working field negatively affect the line estimation system by adding unwanted points to Hough's accumulator that practically do not belong to the same orchard subject to operations. To overcome this problem, it is necessary to filter out all LiDAR points from objects outside the working field: this was done by exploiting obtained Hough's solutions as already described. It is indeed possible to exploit the orientation of the three row lines l_1, l_2, l_3 with respect to the rover to obtain the angular extent that face the working orchard. It has been implemented to filter out all LiDAR points whose angular value does not belong to the 180-degree range centered on the mean orientation value of the tree rows: $\theta_{mean} = (\theta_1 + \theta_2 + \theta_3)/3$.

Maneuver regulator

The row-change maneuver is a semicircular trajectory from the end of the outgoing lane to the beginning of the new lane. This operation is mainly based on the following information:

- the odometry data from the EKF localization node, as seen in Section 3.2,
- the lines estimated from Hough's line-change node, as seen in the previous paragraph,
- the widths of the affected rows, as taken from the last in-row estimation matched with pre-configured values.

The lines estimated from the Hough Transform node are exploited to compute in real time the center of the semicircular trajectory, called the pivot point, by calculating the point of intersection between the end-row line ln and the two straight lines passing through the outer rows: l_i and l_3, with $i = 1\ or\ 2$. The fact that either row 1 or row 2 can be used during computation is primarily due to expected row visibility from the vehicle perspective according to the execution of the maneuver. By selecting the rows involved, it is possible to compute the

pivot point either by extending the central row l_2, or by computing the parallel line equidistant from the external rows l_1, l_3. The latter method can be especially used when the LiDAR sensor is aligned with the central row l_2, such as when the vehicle is approximately in the middle position with respect to the turning trajectory. Figure 69 shows the pivot point calculated during the lane change maneuver.

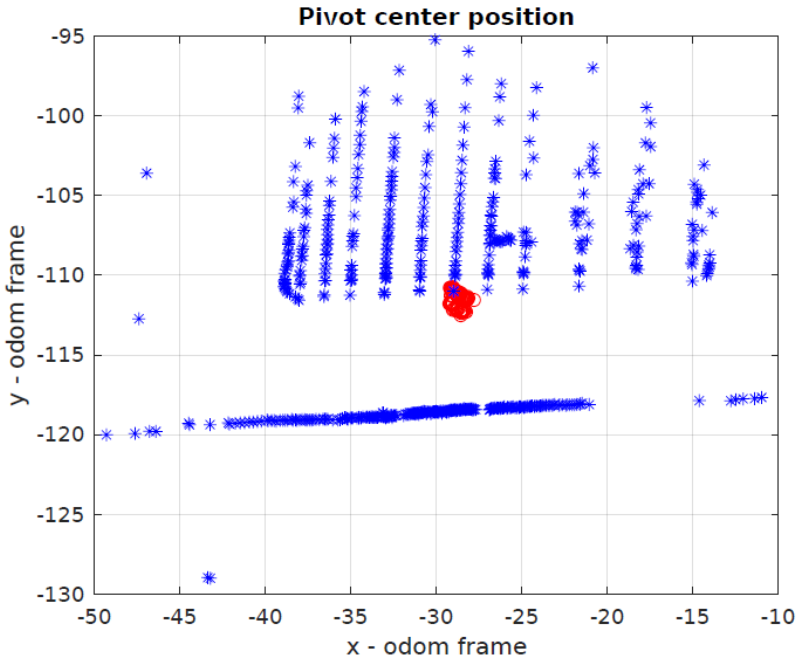

Figure 69 - Pivot position estimation (red circles) with respect to the LiDAR tree points (blue stars)

A speed control algorithm has been developed to perform this trajectory, which is based on two main terms:

- a feed-forward term calculated from the geometric values of the rows, that applies to both linear and angular velocities,
- a feed-back term based on the distance error from the pivot point.

To improve vehicle position accuracy when entering the next lane, the distance to the estimated orthogonal line can be checked while performing alignment maneuvers by reversing vehicle motion direction to better comply with the new estimated row line. This technique has proven to be crucial in the case of SSVs as the robot's speed tracking is not particularly accurate during the turn.

When the available turning space at the end of rows does not allow for a semicircular trajectory, a rectangular trajectory should be applied. This is done by taking some space from the last tree in the row by going straight, then making an in-place turn to align with the orthogonal line. Once this alignment is achieved, the vehicle can advance in a straight line to reach the projected center position between the second and third lines (center of the next lane), then make a turn in place to align with the new line. Once the robot is correctly positioned in front of the next lane, the mission supervisor can return to the navigation controller between the rows, thus ending the lane change maneuver.

Experimental results

The maneuver has been performed using the developed UGV prototype in the university experimental apple field in Cadriano (Bologna, Italy). Figure 69 shows the data obtained in the field to demonstrate autonomous navigation capabilities. Tree line estimation has proved to be very accurate, also taking advantage of previous estimates of the in-row navigation (Figure 70). The orthogonal end-of-row line estimation is more affected by noise, as the trees to be interpolated are sparser and the canopy creates an unclear linear trend in this direction (purple line in the graph).

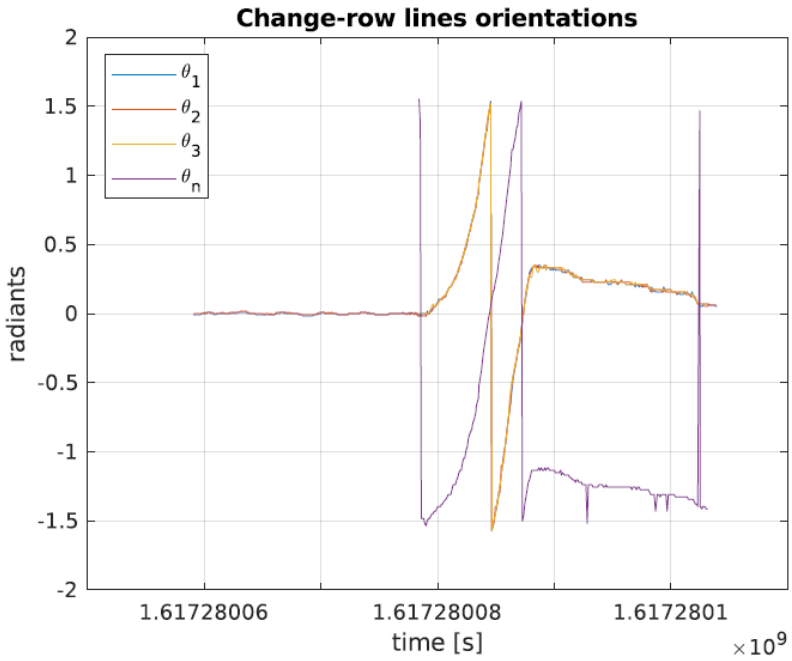

Figure 70 - Estimated lines orientation as a result of the Hough Transform line detection node

However, the estimation has been shown to be sufficiently good and reliable when controlling the direction and distance while approaching the beginning of the next lane to enter. This accuracy is then reflected by estimating the pivot point during the lane change maneuver (Figure 71) with an RMSE of (0.3429, 0.5840).

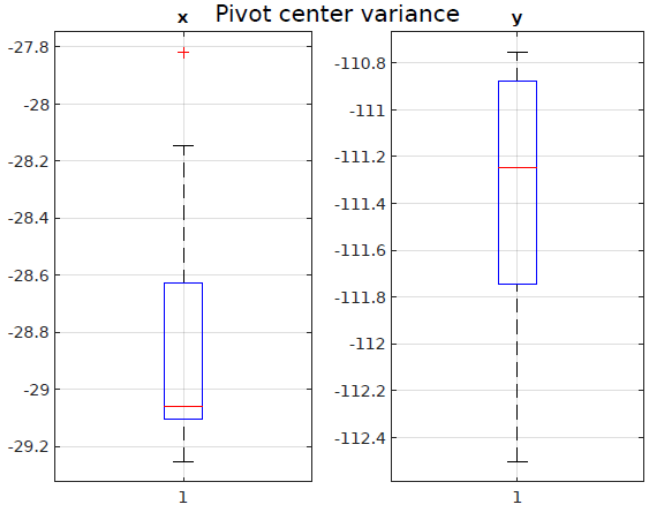

Figure 71 - Estimated pivot point position during the maneuver

3.5. Mission description and configuration

The whole mission description is configured by means of an XML file that defines the list of waypoints to reach and return from the orchard and the row width and length parameters for all the workable lanes within it. The last waypoint to reach the orchard is also used as first row entry point and it is the location in which the vehicle supervisor will switch from open-field navigation to in-row navigation controller. The number of rows to be traversed is also described, with the simple configuration detailed before for each lane. Also, the first turning direction is imposed, as the subsequent ones relies on an alternating rule. An example of the configuration XML file is produced below:

```
<?xml version="1.0"?>
<MissionData>
    <Field name="meleto">
        <open_field_andata>
            <waypoint id="1">
                <x>28.6</x>
                <y>-9.2</y>
            </waypoint>
            ... ...
            <waypoint id="12">
                <x>37.4</x>
                <y>-15.7</y>
            </waypoint>
        </open_field_andata>

        <open_field_ritorno>
            <waypoint id="1">
                <x>82.6</x>
                <y>-42.0</y>
            </waypoint>
            ... ...
            <waypoint id="9">
                <x>6.0</x>
                <y>6.0</y>
            </waypoint>
        </open_field_ritorno>

        <changeRowDirection>2</changeRowDirection>

        <Rows n="8">
            <Row id="1">
                <rowLength>100.0</rowLength>
                <rowWidth>3.3</rowWidth>
                <rowDeviation>1.2</rowDeviation>
            </Row>
            ... ...
            <Row id="8">
                <rowLength>100.0</rowLength>
                <rowWidth>1.95</rowWidth>
                <rowDeviation>0.4</rowDeviation>
            </Row>
        </Rows>
    </Field>
</MissionData>
```

Navigation supervisor

In order to execute the whole mission, a ROS supervisor node has been implemented, allowing both autonomous navigation algorithms to be used to reach first the working orchard from the parking shelter and then perform operations inside the orchard.

Using open-field navigation, the waypoints are defined to travel along existing pathways in the farm, and they are followed by entering the first workable row defined by GPS endpoint coordinates at the border of the orchard. When the robot reaches the goal waypoint, the supervisor switches to the in-row navigation logic, and when the task is completed, the robot switches back to open-field navigation controller to return home. The same node is in charge to start row-change maneuver when the end-row is detected and then relaunch in-row controller after the turning maneuver is finished. These regulator switches are critical, as each controller publish vehicle speeds on the same topic that is therefore transmitted to motor controllers to drive the vehicle.

3.6. Experimental results

The vehicle has sum up over 800 hours of testing in the field, exploiting the Cadriano orchards that includes apples, apricots, pear, peaches, grapevines, and others. The variability of the different fields helped in the development of a multipurpose and flexible vehicle able to autonomously navigate within the orchards with minimal configuration. Also, a demonstration with a *bench* orchard (Figure 72) has been made as part of the final event of the POR-FESR project S3O that aims to create a sustainable automated orchard for the future of farming.

Figure 72 - Customized orchard made of benches to demonstrate vehicle autonomous navigation capabilities and flexibility

With respect to open field navigation, the quality of the localization filter has been proven by the repeatability of the trajectories tracked to reach the orchard. The pathways present in the Cadriano field present several narrow sections (such as when crossing a trench and surpassing a road bar), that were tackled without difficulty by the vehicle. Also, numerous numbers of full

mission demonstrations have been made to *Precision And Sustainable Agriculture* students during the last years.

The alignment maneuver described at the end of the row-change maneuver has proven to robustify the solution, by giving the vehicle the space to correctly enter the narrower rows in the planar cordon apple field that have an inter-row width of less than 2 meters, that was autonomously entered with the prototype that has a width of about 1.35 meters, with basically no margin of error to enter the lane. Before entering the corridor, the vehicle is moving forwards and backwards trying to get the perfect centered position according to the adjacent tree rows, in order to overcome possible inaccuracies during the turning maneuver.

Figure 73 - UGV prototype entering narrow orchard lanes

3.7. Simulation tools and environments

In order to speed up the development of the vehicle and the software implementation process, several scenarios have been created using two of the most common simulation engines available that can be integrated with the ROS framework. The first scenario described relies on Gazebo and has been created mainly to test the robot navigation features; while the second uses the Unity game engine and has been devised to handle fruit picking application, by exploiting the game features of the engine.

3.7.1. Navigation scenario

Gazebo is a 3D multi-robot simulator, mostly used for outdoor environments, offering robust physics and rendering. It is the default simulation engine integrated within the Robot Operating System (ROS) framework. Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments.

When used with ROS, Gazebo can provide realistic simulations of sensor input and odometry for robot platforms. This has proven to be especially useful for testing different navigation algorithms and SLAM (Simultaneous Localization And Mapping) applications. Gazebo is also expandable by exploiting the World Plugin Interface: in this way it is possible to extend the simulation engine by adding custom sensors and other objects either to the robotic platform and the environment. Gazebo's flexibility makes it an ideal tool for rapid prototyping and experimentation with ROS-based robotic systems. It is also well suited for use in education and research, as it is freely available and open source.

In order to simplify the testing of the navigation algorithms of the prototype, and to check the high-level operations on the vehicle, a farm simulation environment has been created. This features a couple of buildings and a simplified orchard to test both open-field path planning and rows detection for the in-row navigation. Also, a border wall has been placed to limit the operational area and to help the mapping functions and the map localization algorithms (such as the Monte-Carlo localization [68], similarly to the one implemented in the community-available “amcl” ROS module [69]) by creating more environmental features. A graphical representation of the configured environment can be seen in the Figure 74 below.

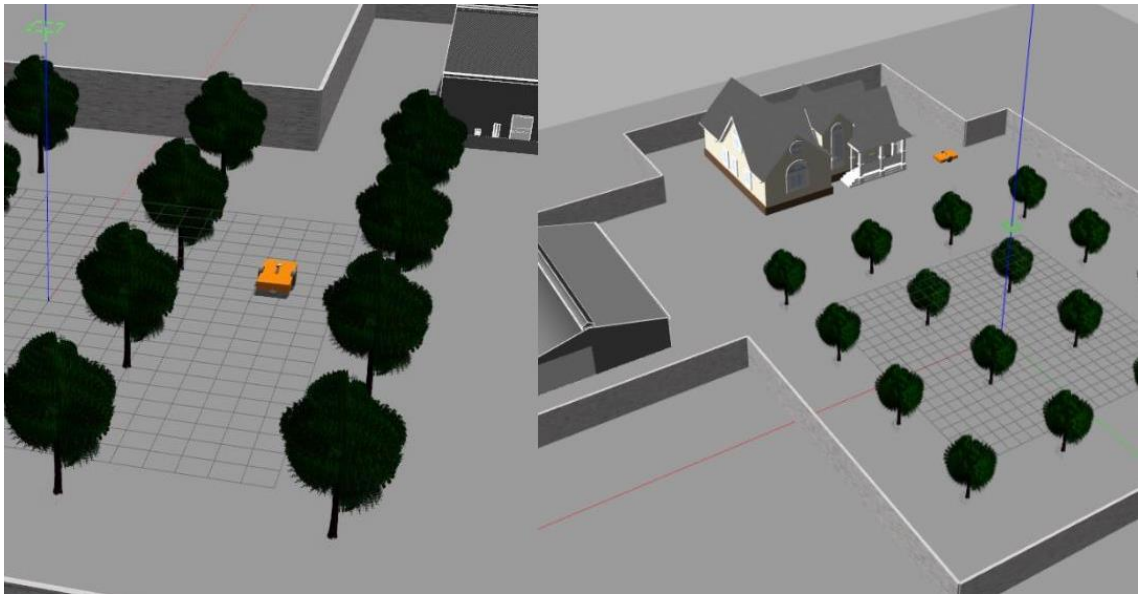

Figure 74 - Gazebo simulation environment

Two different world configurations have been produced, with different density of the orchard trees, so to enable testing of different rows configurations.

The robotic platform used within the simulation is a standard differential-drive wheeled robot that was provided by the Gazebo community, in which specific sensors have been installed, according to the ones available in the real vehicle. The choice of using standard Differential-drive Wheeled Robot (DWR) was made because, as it has been presented, it behaves similarly to a Skid Steering Vehicle. The purpose of having this simulation environment is mainly to test high-

level control algorithms and vehicle supervision logic, so possible inaccuracies in the trajectory tracking are not taken into account in this phase and are devolved to the real-world testing using the real prototype.

Gazebo software can be slow processing complex simulations, which can have an impact on the testing activities. Although Gazebo rendering engine has improved in the last year, the used version of Gazebo (linked with the version of ROS used for the vehicle navigation software implementation), had some lack of optimizations typical of game engines, that limited complex simulation environments on commonly used hardware. Therefore, a choice has been made to switch the simulation engine to a more game-oriented ecosystem, such as the Unity game engine for further developments in the harvesting fruit scenario.

3.7.2. Fruit harvesting scenario

Unity is a powerful 3D engine that can create realistic graphics and simulations. It also has a well-documented API that allows developers to create their own custom applications.

Unity is widely used in the game industry, but its potential for other applications is often overlooked. In particular, Unity is a great tool for developing robots and robotic systems. By using the built-in physics engine and creating custom scripts, it is possible to create realistic simulations of robotic systems. Its flexibility as a multi-platform tool, the provided user-friendly editor, and the ability to connect ROS system to the game engine by exploiting community available plugins makes the choice of using Unity effective on a mid-long term basis and for complex scenarios with multiple object interactions, physics simulation and possibly robotic swarms. Furthermore, Unity can simulate a wider range of scenarios than Gazebo alone. For example, Unity can be used to simulate environmental factors such as wind or rain and this can be helpful for testing how the robot will perform in different conditions.

The gaming nature of the engine makes it also an effective choice to test computer vision applications and object detection algorithms as the ones that are requested to develop autonomous fruit harvesting applications. The created environment features a kiwifruit row to represent a possible orchard scenario, with multiple fruits hanging from the top part of the trees (Figure 75). A simplified differential-drive wheeled robot has been used as a mobile platform on top of which a commercial robotic arm has been placed to enable fruit picking. The end effector is also a simplified version of what would be the final gripper tool, to allow fruit interaction and grasping. More refined models of the arm and the gripper would then be introduced as a final custom 3D model will be available during the ongoing activities.

Figure 75 - Unity simulation environment

The simulation has been used to validate high-level supervisor logic while attempting fruit picking and for testing different arm trajectories to detach the kiwifruit from the tree.

3.7.3. Gazebo skid steering plugin

In order to improve the vehicle trajectory tracking capabilities and to exploit the simulated environment, it was important to refine the differential-drive wheeled model available in Gazebo with a more accurate version with respect to the tracked nature of the prototype robotic platform that has been developed. By exploiting the work made on the vehicle model detailed in Section 2.1.1, a new Gazebo controller plugin has been implemented to take into account the tracks skidding forces the occurs during turning maneuvers.

The implementation computes the specific pressure of the tracks using cosine and Gaussian models, according to the number of wheels/rollers configured for the tracks. The integral is solved numerically by considering a quantized space along the length and the width of the tracks. The plugin can then be used to provide track forces estimation and by adding slippage during vehicle movement in the simulation environment. This can be especially useful for checking expected tracking performances of the low-level controller.

Computed forces have then been compared with the MATLAB simulations to validate the plugin implementation (Figure 76).

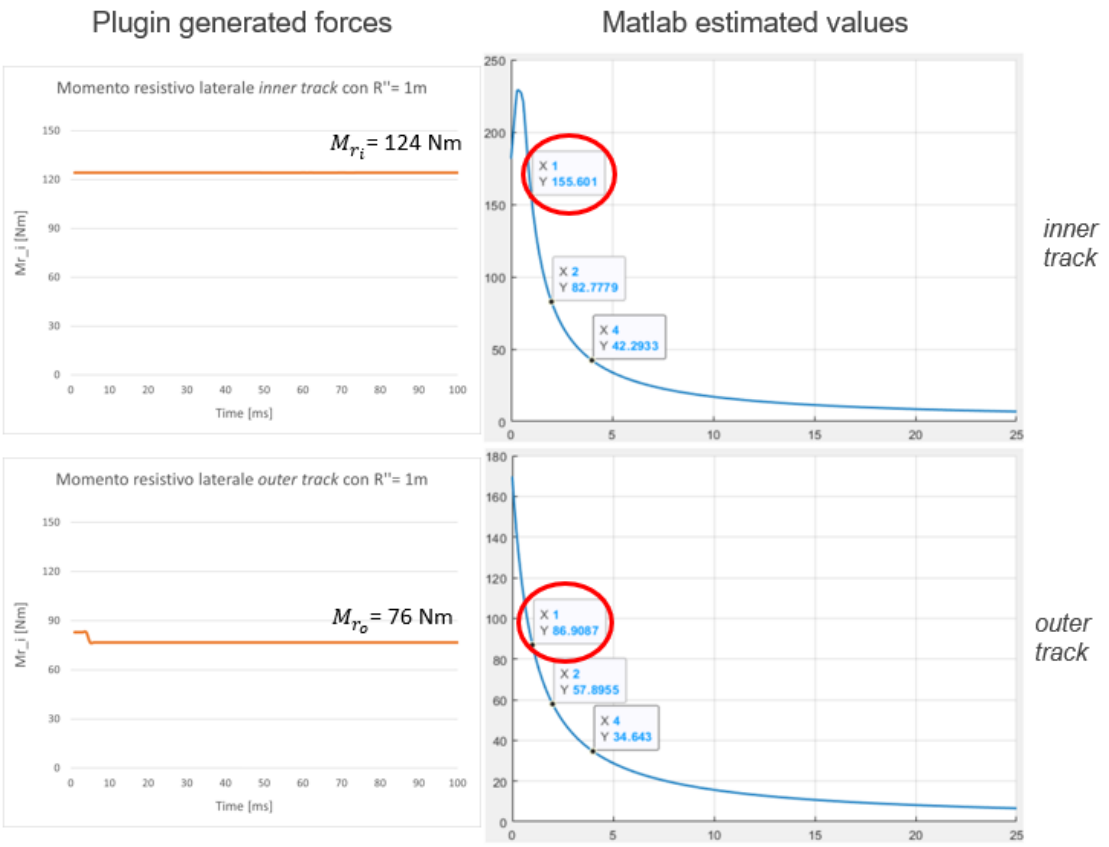

Figure 76 - Skid Steering plugin comparison with Matlab estimated values

3.8. Conclusions

In this chapter, the software architecture of the vehicle has been presented, as a description of the ROS node involved into the autonomous navigation system. The different types of regulators for the open-field and in-row navigation tasks have been detailed, together with the simulation environment used to validate the algorithms. Experimental testing within the Cadriano experimental orchard was also discussed, to prove the solidity of the developed solution.

Further developments will focus on improved open field navigation, that can be either map based or obtained by configuring some pre-defined routes across the field, that should work as priority pathways to reach any relevant location in the area. Moreover, a local sliding-window obstacle map can be implemented to give to the vehicle environmental awareness of unexpected or moving obstructions that should be avoided during navigation.

Chapter 4. Precision Agriculture applications

In this chapter, some of the precision agriculture techniques investigated exploiting the robotic platform are presented, together with the data collection systems that have been used in the field.

The availability of an autonomous robotic platform can be exploited on a series of Precision Agriculture projects that may benefit from automatic monitoring both coming from fixed in-situ sensors or other vehicle installed systems. To enable this, a centralized database is necessary as well as the computing systems able to run the algorithms for monitoring different orchard properties.

4.1. Time-series data collection system

Data collection and processing are essential needs for precision agriculture. In this project we aim at building an infrastructure to collect, store and process data gathered by various sensors in the field. The project will also explore the integration among optimization and machine learning algorithm to further improve the achieved results.

A centralized repository is needed to provide a reliable and standard way to store and retrieve data for the benefit of all the research groups involved in the project. This is actually composed by a time-series database (to store sensor data) and in the future can be expanded with a structured/indexed image database, possibly with annotations.

Examon database

A NoSQL database has been specifically deployed for the Precision Agriculture projects needs for the Cadriano testing field. It relies on KairosDB/Cassandra Tool to specifically store time series data. Unlike relational tables, NoSQL databases don't store data in tabular format. The types of NoSQL databases vary according to their data model. They offer flexible, easily scalable schemas with large amounts of data and high user loads, including document, key-value, wide-column, and graph. NoSQL data models allow related data to be nested within a single data structure, by applying properties on the data value that is stored.

Time-series data is any information that changes over time, weather readings or sensor measurements. Using a NoSQL database is an excellent alternative for timeseries data to traditional relational databases, as it can quickly adapt to ever-changing needs and handle storing large amounts of data without delays. Furthermore, it can also provide features that make it easy to query and analyze timeseries data, such as aggregation, and perform simple computation while aggregating, such as sum or average.

Real-time field gathered data is sent via GSM Internet connection to an MQ Telemetry Transport (MQTT) broker that acts as a centralized data collection interface. MQTT is a lightweight, yet powerful, publish/subscribe messaging protocol designed for machine-to-machine communications. The implemented broker is used to translate the MQTT topic into the tags and metrics to store a single measurement point (according to its timestamp). The measure value and the timestamp are sent as topic payload using a specific convention, while key-value pairs that characterize the sensor, together with the sensor metric, are transmitted by another convention on the topic key string. Grafana is then used as a data visualization tool, providing customizable graphs and dashboards.

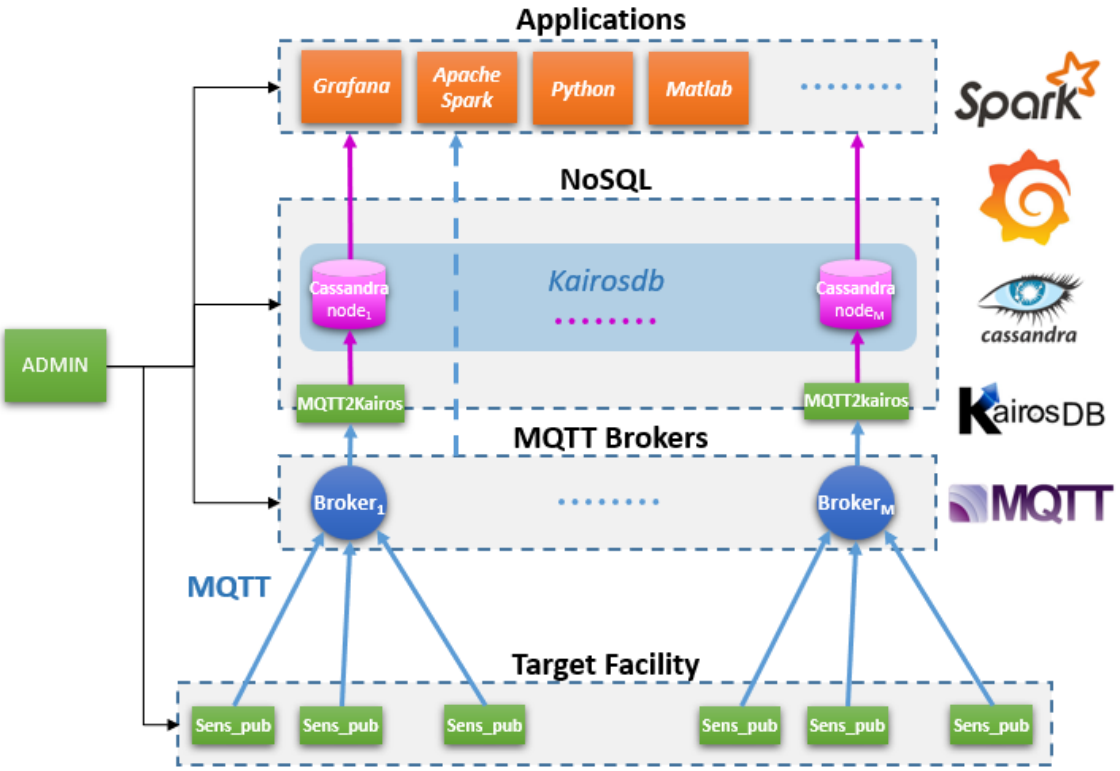

Figure 77 - ExaMon architecture and technology used

ExaMon (Exascale Monitoring) is designed to offer fine-grain and precise monitoring of power, energy, thermal and architectural elements across distributed and large-scale high-performance computing systems. Although the original purpose of ExaMon was to monitor supercomputers installation, the time-series nature of the relevant field information that can be gathered exploiting in-situ sensors, together with the flexibility of the technologies used within the ExaMon framework, have made it an ideal starting point to collect the amount of data coming from the experimental orchard.

In order to ensure some level of privacy, the public MQTT entry point has been secured by applying user/password authentication, and the connection has been encrypted. Also, the system is discarding all the messages that do not comply with the defined conventions in terms of topic key definition and payload.

QuestDB implementation

As the EU project that supported ExaMon has ended, and ExaMon support is limited, other time-series storage solutions have been investigated. Also, considering that ExaMon was developed more than five years ago, it was worth checking if other database solutions have been developed and were suitable for the intended application.

Among all the available options, QuestDB was selected. QuestDB is a high-performance open-source time series database. It is designed to handle large amounts of data with high throughput and low latency. QuestDB supports various query types including aggregations, filters, and transformations. It also supports various data sources including CSV, JSON, and InfluxDB.

Other open-source options similar to QuestDB that were evaluated are:

- InfluxDB: an open-source time series database written in Go. It's designed to be scalable and easy to use, with a wide range of features including SQL support, Cluster support, and Grafana integration.
- TimescaleDB: an open-source time-series database optimized for fast ingest and complex queries. It's fully compatible with PostgreSQL, so it's easy to get started with if you're already familiar with that database.
- Prometheus: open-source monitoring system that includes a time series database. It's designed for large-scale deployments and can handle billions of metrics per second.

The advantages of using QuestDB over other solutions are listed below:

- QuestDB is built on top of a columnar storage engine, making it very efficient for time series data. This also makes QuestDB very fast for queries.
- It supports a SQL syntax is specifically designed for time series data, making it easy to work with this type of data.
- It also supports SQL compatibility options to connect the storage engine as a standard datasource for Business Intelligence or graphical applications (such as Grafana).
- QuestDB supports various types of data sources protocols, including InfluxDB, Prometheus, OpenTSDB, and more.
- It also provides a web interface that makes queries and data visualization a straightforward process.
- The project is open source and free to use.

As the data capturing interfaces and ROS bridges were already implemented, the idea was to retain the MQTT interface of the ExaMon project and replacing the storage technology with the new choice. In this context, the MQTT interface was a plus, since it allowed to keep existing data ingestion logic, while only changing and implementing the new QuestDB connector to actually store the timeseries values in the new architecture.

Future uses and applications

A dedicated graphical web interface has been implemented to allow easy data visualization and help the farmer taking decisions according to the monitoring activities. This system is generic enough to be extended with other physical sensors, both in-situ or installed on vehicles. Furthermore, it is possible to implement other *virtual* sensors by exploiting the developed data ingestion mechanisms in such a way that by applying custom computations with respect to available data and inserting them with appropriate timestamps as new sensor types, more advanced or complex analysis can be performed and visualized on the same platform. The general idea is to conduct data analysis on the cultivation time-series data to investigate the effect of cluster activity. The average measured by sensors and their error drift will be processed to create homogeneous zones or clusters of similarity. Then, it will be possible to study the correlation between temperature, humidity, and other traces, in order to compensate sensors errors and to provide reliable information for Machine Learning (ML) anomaly detection algorithms.

4.2. Fruit detection and counting[8]

To ensure a successful production in agriculture, yield prediction and crop monitoring activities are essential. Technology has become increasingly important in these processes in recent years, as it can be evident in the adoption of decision support systems fed by data collected by autonomous vehicles.

In yield prediction, the role of autonomous robots in crop monitoring is a necessary step to take, in order to accurately predict how much fruit will be produced. Traditionally, this process was performed manually by workers who would walk through each field and count the fruits. However, this process is quite time-consuming and can often result in inaccurate counts.

The introduction of autonomous robots has made fruit counting faster and more accurate. These robots are equipped with sensors that can identify and count fruits on each plant. This data can then be used to predict the crop's yield. This information can be also used to determine the best time to harvest, and the amount of thinning required based on the growth stage of the plant.

Autonomous fruit counting tasks in yield prediction and crop monitoring applications require precision and accuracy. A robotic system must be able to count the number of fruits in a given area accurately. A variety of methods can be used to accomplish this, such as imaging systems or sensors that detect fruit presence.

4.2.1. Proposed implementation

ROS (Robot Operating System) framework has been exploited to implement real-time counting. The development version is ROS2 Humble Hawksbill, the latest ROS2 Long Term Support version at the time of the project. The processing pipeline is described by Figure 78: two camera nodes are in charge of acquiring frames from the left and right Intel Realsense D435i cameras, as the vehicle is capable of count fruit on both sides, by installing one camera on each side of the robotic platform. Every frame is then processed by a detection node, utilizing a modified YOLO [70] (v5 size S) neural network tailored for apple recognition. To boost performance, the images are rescaled to 640x480 resolution. The same node also applies the chosen Multiple Object Tracking (MOT) algorithm for keeping track of each detection, therefore making it possible to count fruits. MOT tracking data is then sent to a dedicated counting node that actually carries out object counting.

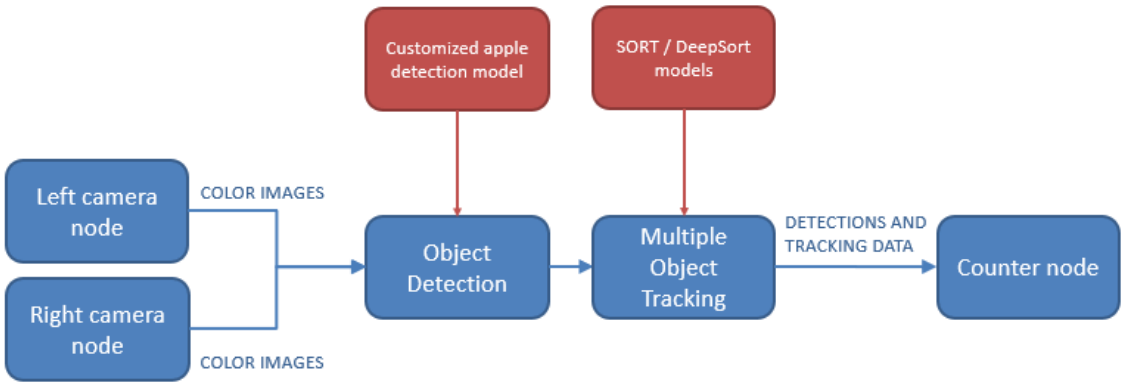

Figure 78 - Fruit counting architecture

[8] This work has contributed to the accepted paper: An online fruit counting application in apple orchards. - Mengoli, D.; Bortolotti, G.; Omodei, N.; Rossi, S.; Piani, M.; Bucciarelli, A.; Manfrini, L. - 14th European Conference on Precision Agriculture - ECPA 2023

Yolo detection

Object detection in the scene was performed by exploiting YOLO [70] convolutional neural network. YOLO is a state-of-the-art, real-time object detection system. It uses a single shot detection system that is able to detect multiple objects in an image with high accuracy, by exploiting a Convolutional Neural Network (CNN) that can directly take images as input and produce bounding box predictions as output. The choice of YOLO network was primarily due to its speed and flexibility, as well as its available tools and its open-source nature. Prior to YOLO, existing image classifier neural networks were able to classify single images according to the subject. In order to detect multiple objects into the same scene, the input image was cropped in smaller pieces and individual classifiers were executed on each fragment. YOLO surpassed this behavior by enabling multiple object detection at one, with only a single neural network execution (hence the name You Only Look Once). It was clear that YOLO quickly became increasingly popular due to its effectiveness and speed. The original network has evolved over the year, producing many versions and iteration, with an increase in accuracy and processing speed. For the purpose of this project, the YOLOv5 object detection network was selected because of its speed and availability for python code and torch framework at the time of initial implementations. Further improvements have been made in the recent years with YOLOX, YOLOv6, YOLOv7 and lately YOLOv8.

The community-available YOLO network has been retrained to fit specifically the fruit recognition on-field and inside tree rows. As YOLO features a supervised training, a good amount of data examples has been required to obtain proper detection accuracy after the network training process. To do so, an extensive labelling process was made.

The labeling process is the most time-consuming task in the object detection preparation phase, as it requires a high amount of manual work to be done. As typically object detection is a supervised learning method, the training dataset has to be manually labelled to teach the convolutional neural network (CNN) how to perform detection correctly.

Several software exists to facilitate this process, among the open-source options, the tool that has been used is Label Studio [71]. Label Studio is a web-based application that supports multi-user installation and allows the creation of different labelling projects according to the required image detection or image segmentation needed.

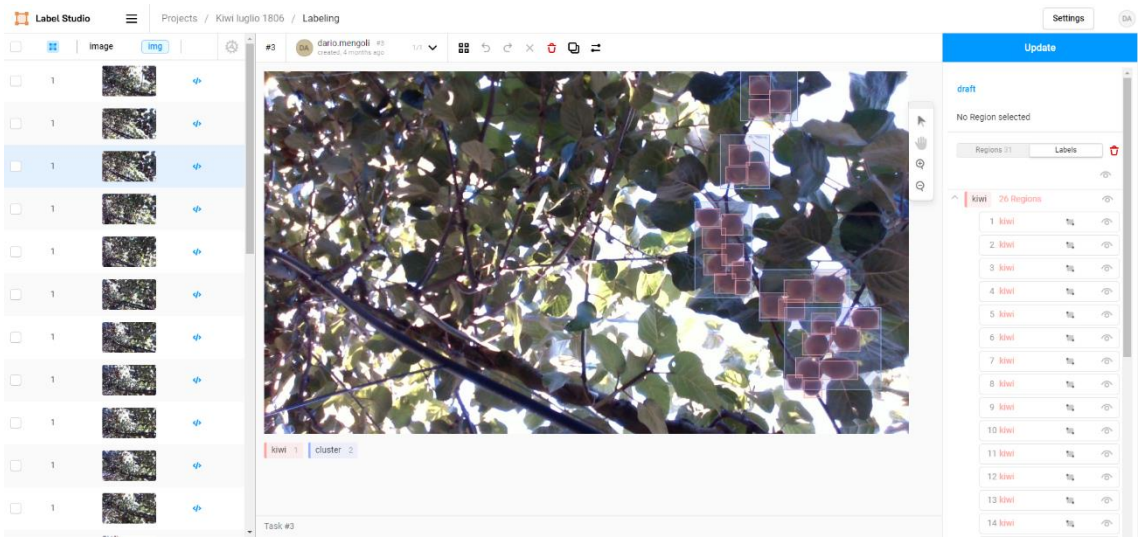

Figure 79 - Label Studio labelling interface. The example shows fruit and cluster labelling for kiwifruit detection

For research purposes, multiple fruit detection models were trained, such as apples, grapes, apricots, kiwifruit, peaches, and more. The appropriate model is then selected, by means of a ROS node configuration parameter, to enable the correct fruit detection according to the orchard analyzed.

Object Tracking

Fruit tracking over multiple frames is accomplished by executing existing state-of-the-art Multiple Object Tracking (MOT) algorithms. The two MOT algorithms selected were SORT [72] and DeepSORT [73], which is a deep-learning improved version of SORT algorithm. The first technique is purely algebraic and depends only on bounding box parameters and confidence, while the second includes an image feature matching layer that use several pre-trained neural networks to increase matching on future detections. The deep learning association network labeled "osnet_x0_25_msmt17.pt" was chosen for use in experiments using the DeepSORT method. It is a community-available, pre-trained network designed to tackle typical benchmarking scenarios for people tracking. Evidently, this association network proved to be rather general, since it also worked for our apple tracking scenarios. The type and dimension of the deep association metric network was mainly affected by the computational power required by the system to process multiple fruit detection: as speed is quite important within this context, a simplified and faster network was preferred to more complex but slower ones.

In order to avoid double counting of the same apple in subsequent frames, the tracking data from the MOT algorithm is used to perform the fruit counting, as it is marked with the same object ID number, so referring to the same object/apple. Considering that the detected apples have very similar shapes, once the previous object has left the video frame, the same ID number may be recycled and associated to newly seen apples. By checking the position of the object during the next detection, this condition can be detected easily. As the vehicle is travelling forward through the lane, the scene movement is expected to occur predominantly in one direction. Therefore, a threshold might be established to detect if an old id has been reused on a new item or not (so the previous object has maybe reappeared after an occlusion). In addition, because tree scanning is performed laterally while the robotic vehicle is in motion, the desired object matching should be performed horizontally. To improve this, the initial object confidence obtained by the YOLO detection is deleted and replaced with a virtual confidence tied to the vertical location of the item (rounded up to the second decimal place). This technique has shown to be useful, especially when executing the SORT MOT algorithm, to facilitate object matching.

4.2.2. Experimental results

Planar and spindle types were addressed to highlight the differences between a robotic-friendly orchard (the planar type) that allows all fruits to be exposed without occlusions and a more traditional cultivar type (the spindle type) that, by nature, will result in more fruit occlusions given by the canopy. Experimental testing was made by exploiting the developed robotic platform prototype described in Chapter 2. The UGV was set to traverse the orchard rows at a speed of about 0.6 m/s. This was selected according to the performances of the onboard computational power, made of a dedicated Cincoze DS-1301 automotive PC equipped with a Nvidia RTX A2000 Graphics Processing Unit (GPU). The YOLO detection achieved 90/100 frames per second (fps), that decreased to 20 fps when applying also the DeepSORT algorithm. This computational unit was specifically installed on the autonomous rover prototype to enable GPU acceleration of deep-learning algorithms. As the processing power needs to be split between left and right cameras, the cruise speed was configured to ensure proper overlapping between subsequent frames with a rate of 10 frames per second. It is worth noting that the computational

burden of MOT algorithms tends to increase when there are a lot of objects to track in the scene (i.e., when more than 50/60 fruits are simultaneously visible).

As a result of the different cultivar types and expected fruit occlusions, the configured confidence threshold for spindle cultivar types was set to 0.46, while the configured confidence threshold for planar cultivar types was set to 0.54. The configured Intersection over Unit (IoU) threshold was set to 0.5. The fruits of each tree row were also scanned on both sides, so that a comprehensive view could be obtained. In the planar type, canopy occlusion is not expected to happen for apple fruits, so the two counts have been averaged. On the contrary, in the spindle type, the canopy is expected to provide heavy occlusion from one side to the other, so the two counts of both sides are added to obtain a total value.

Depending on the actual fruit quantity, a calibration coefficient may be required to best fit the predictions, based on the expected occlusions (especially on the spindle architecture) or apples detected on adjacent rows (especially on planar type).

Default SORT parameters were kept, while DeepSORT parameters were adjusted to enable camera motion compensation, a minimum of 2 frames to initiate tracking, a maximum age of 30 frames, and a maximum distance of 0.8 for both matching and gating IoU thresholds.

The counting results obtained applying both SORT and DeepSORT MOT algorithms with the same YOLO object detection are represented in the following Table 7.

Table 7 - Apple fruit counting results comparing different MOT algorithms with manual assessment

Method	**Spindle type**			**Planar type**		
	Side 1	Side 2	Total (sum)	Side 1	Side 2	Total (avg)
SORT	1375	1108	2483	2983	2790	2886
DeepSORT	730	686	1416	1778	1880	1829
Manual	-	-	2298	-	-	1689

Because of the MOT algorithm performances may vary according to the detections coming from YOLO network, the same parameters have been applied for both solutions, thus obtaining the same bounding boxes for the two selected MOT algorithms. The results are then compared with a manual fruit counting value used as ground truth.

The Figure 80 depicts two example images of the tracked apples, according to the two orchard architectures: the spindle type is on the left side, while the planar type can be seen on the right side. A couple of blue bounding boxes can be seen in the images. These do not have an id number associated because are YOLO apple detections that have not been correctly tracked yet by the MOT algorithm. It is good to mention that during all the field testing executed, using the aforementioned configuration, every object detection has been tracked by the MOT algorithm sooner or later in the video stream. As it can be inferred by the configuration parameters, the MOT algorithms require a valid match for a specific number of subsequent frames to validate a tracked object. As YOLO detections may be intermittent due to the complex nature of the scenario or the camera shaking that creates motion blur in the image, some time may be required to stabilize the apple tracking with an ID association.

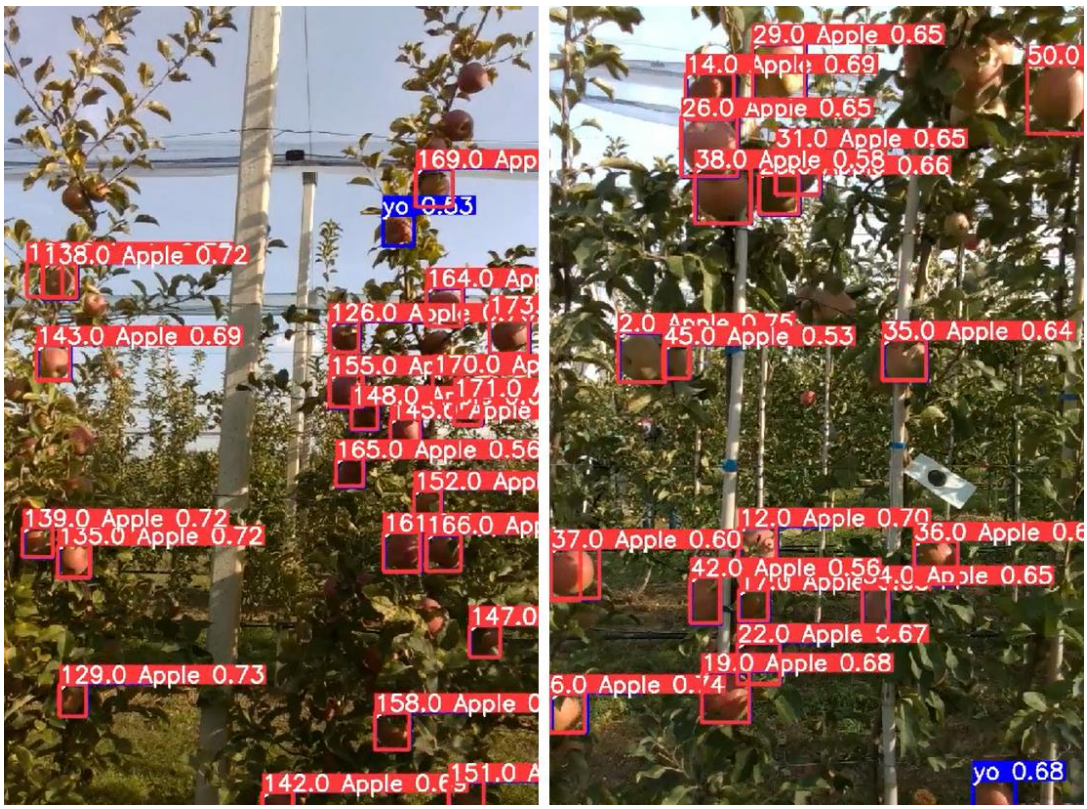

Figure 80 - Fruit detection and tracking examples for spindle (left) and planar (right) orchard types

As expected, DeepSORT outperforms SORT in terms of occlusion handling and ID re-assignment when leaves or canopy obscure a detection intermittently, making the SORT algorithm overestimate the total number of occurrences. The spindle type is also expected to hide more fruits due to its canopy shape, resulting in the obtained undercounting of the DeepSORT algorithm. To map the automatic counting to the actual number of apples based on the cultivar type, a calibration coefficient may be relevant and necessary in this scenario.

The results given by the field testing have highlighted an acceptable fruit counting, at least in the more robotic-friendly planar type. Although there has been some overcount in this cultivar type, it is mainly due to some object detection on the background rows that is responsible for the overcount. Even under those conditions, the total error committed remains less than 10%. In addition, it is important to note that YOLO performance is crucial for this kind of application, and that global system performance is highly dependent on actual fruit detections. In the presented scenario, when using the DeepSORT MOT algorithm, several manual checks insisting on limited number of apples showed that the overall detection is accurate, in terms of a precision and recall above 90% accuracy, and that the MOT given ID number remains stable throughout the entire frame traversal process of the single apple.

4.3. Fruit sizing[9]

In order to improve the quality and quantity of the final harvest, harvest estimation is of paramount importance in order to intervene during the growth season. To manage all required

[9] This work has contributed to the published paper: On-line real-time fruit size estimation using a depth-camera sensor. - Mengoli, D., Bortolotti, G., Piani, M., Manfrini, L. - 2022 IEEE Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2022 - Proceedings, 2022, pp. 86–90

operations within the short harvesting timeframe, harvest labor, storage, and transport logistics must be prepared in advance [74]. Fruit crop load estimation is particularly important for fruit crops since plant crop load affects fruit size and quality [75], and is especially important for those fruit crops with a short fruit ripening window and storage period [76]. Together with tree density and orchard size, fruit weight plays a major role in yield estimation. Therefore, the possibility of collecting data about the weights of a large number of fruits in the orchard increases yield estimation reliability. The effectiveness of mathematical models that convert fruit size into fruit weight has been evaluated in recent years. To predict yields, manual data collection with calipers and automated/continuous fruit gauges have been tested. There are several drawbacks to these methods, including that they require human labor, are repetitive, take a long time to complete, and have a limited sample size of 20 to 200 fruits per hectare. Recently, several automated applications have been investigated, both by exploiting available IoT configurations and wireless sensor networks [77], [78]; and using vision-based fruit counting systems[79]. However, the first method has its main drawback on relying only on a parcel-based estimation due to their fixed position nature that is typical of in-situ sensors, so the research presented in this work has focused on vision-based systems, exploiting the autonomous vehicle developed and the low-cost stereo camera sensors available in the market.

4.3.1. Camera used and system architecture

The trial consisted of evaluating an adaptive computer vision algorithm for sizing apple fruits in field and in real time. An Intel RealSense D435i consumer-grade depth camera (RGB-D) was used. This camera was chosen based on its specifications as well as preliminary field test results indicating that it outperformed other Intel RealSense cameras (i.e., D455 and L515) during testing in the specific environmental conditions and with respect to the allowable distance from the subjects. In the study, data were collected by recording RGB-D videos of a tree row in an apple orchard in which a variable number of apples were tagged and measured by a digital caliper for their maximum equatorial diameters. In addition, tennis balls were used as dimensions to evaluate the system's sizing capacity. The apple orchard selected was the experimental Planar Cordon [80] field in Cadriano (Bologna, Italy), that presents a robot-friendly architecture [9] that favors fruit visibility. The sizing algorithm implementation was applied on both apples and tennis balls, appropriately inserted within the orchard. This was due mainly to compare fruit sizing with an ideal spherical object of known size such as the tennis ball.

For the purpose of identifying fruits in the scene, a YOLOv5 CNN algorithm was employed. The YOLOv5[70] size L algorithm was selected based on accuracy performances as well as computational requirements. The customized apple detection model was trained on 123 apple tree images with an approximate of 5000 highly visible fruits, that covers different fruit dimensions and colorations. In this case, the fruit visibility was stressed more than fruit detection and presence, since for measurements purposes it is important to have a high percentage of fruit exposed from the camera point of view. The dataset is divided into three image sets: the train set, the validation set, and the test set with a proportion of 70-20-10 percent respectively. Dataset augmentation was obtained my altering colors, rotating, cropping and slightly blurring the images. The increase with data augmentation was about six times the original dataset. In order to train the model, a total of 500 epochs were processed on a dedicated Workstation with two NVIDIA RTX2080super GPUs. On the test set, the trained model scored 0.85. Another model was trained for detecting tennis balls and achieved a score of 0.93. The trained YOLOv5 CNN algorithms were then applied to the RGB image to detect fruits (and tennis balls).

After each YOLO object detection, a customized circle detection algorithm, based on the OpenCV [81] *HoughCircles* function, was applied to each detected fruit area of the bounding-box (bbox), in order to determine which circle best fitted the fruit (Figure 81). The RGB captured image was converted into grayscale to operate the Hough Transform circle detection algorithm in order to get its center (pixel coordinates) and radius (pixels). Thresholding and filtering have been applied to the input images and output results, to keep the best detection possible among all the circle detection function results.

In order to obtain the circle dimension in millimeters, two methods exploiting depth (z-dimension) information from the RGB-D camera were used. For both, before starting fruit detection and sizing, the depth-map was resized and aligned to the color image.

The first fruit sizing method (squared-bbox; Figure 81: Squared-bbox) involves extracting the mean fruit distance from the camera, based on averaging all valid (i.e., non-zero) depth points enclosed in a square-shaped Region Of Interest (ROI), placed in the center of the detection box and with a size of half the original detection bbox. The real scene dimension (at mean fruit distance) that is fitted in the camera frame can then be obtained based on the trigonometric relations between camera intrinsic parameters, such as the Field of View (FoV), and the distance from the fruit/object, as defined in the following equation:

$$Scene_{dim} = 2 * (mObj_{dist} * \tan(FoV/2))$$

where FoV is the camera field of view expressed in radians, and $mObj_{dist}$ represent the extracted mean object distance in mm. Then, in order to find the real object size, the real scene dimension was divided by the RGB image resolution, and the square root of that result was used as the real object/fruit resolution (Obj_{res}) value expressed in mm/pixel.

The second fruit sizing method (circle-bbox; Figure 81: Circle-bbox) is obtained by estimating the fruit distance following the same previously described method, in this case not using the square-shaped ROI, but instead averaging the depth information from the area projection gathered by halving the radius of the circle estimation returned from the *HoughCircles* algorithm, centered on the detected circle's center point in the frame. This second method computes the real object resolution by utilizing the camera's intrinsic f_x, f_y parameters, which describe the focal length of the image as a multiple of pixel width and height, respectively.

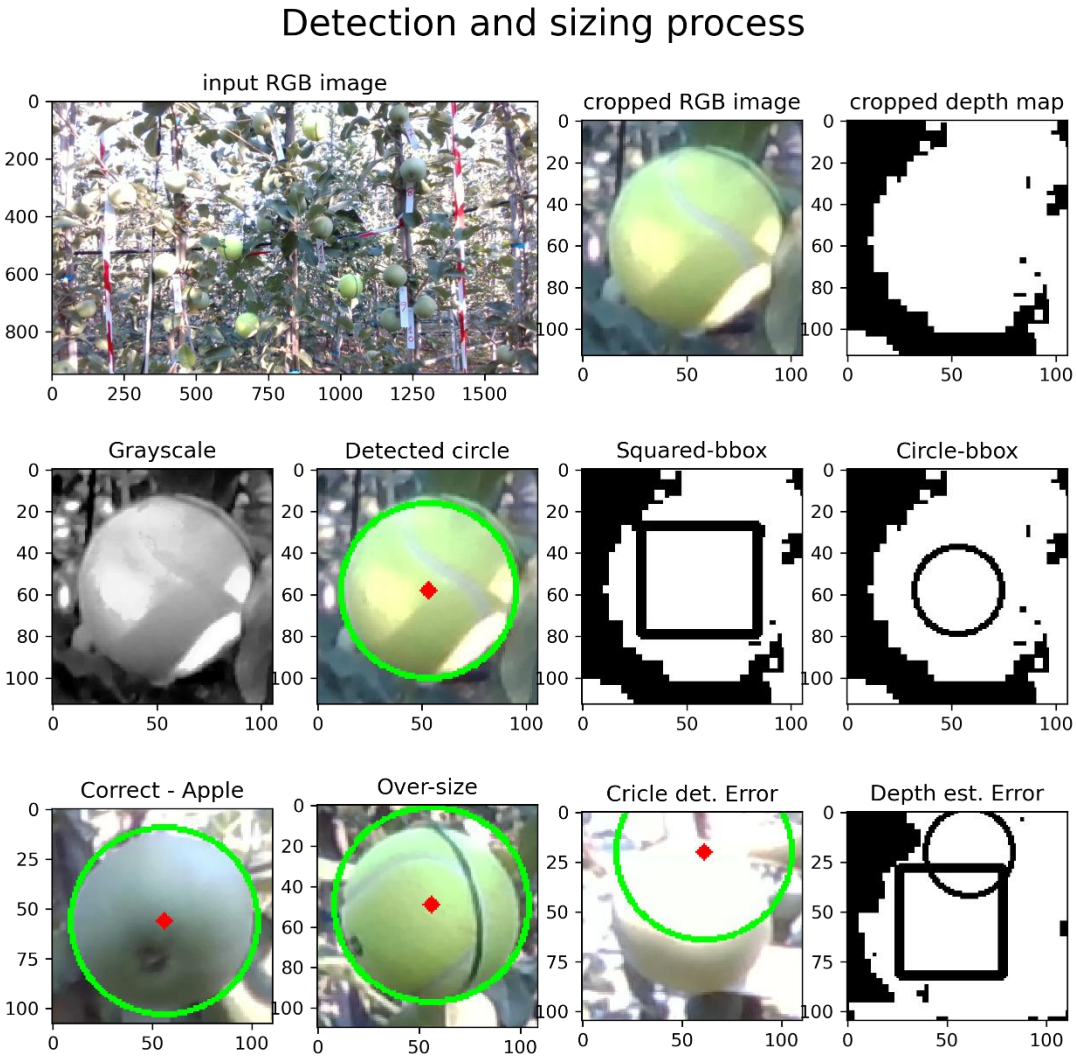

Figure 81 - First line: image of the scene and cropped RGB and depth after CNN object detection application. Second line: circle detection and sizing process. Third line: examples of correct circle detection, oversized circle detection, erroneous circle detection due to excessive light. The last picture shows the difference between Squared and Circle bounding box methods while estimating depth

Given this, the real fruit diameter (F_{size}) was estimated for both fruit distance methods, by taking the diameter of the detected circle in pixels (d_{px}), and multiplying it by the object resolution (Obj_{res}), as reported in the following formula:

$$F_{size} = d_{px} * Obj_{res}$$

The same process was applied also to the tennis-balls included in the images so that a more accurate interpretation of the system performance could be obtained as these objects exhibit a size variability of less than 1 mm. For both apples and tennis balls, absolute Root Mean Squared Errors (RMSE) and relative (RMSE %) were computed based on the vision-system diameter compared to the manually measured diameter.

4.3.1. Experimental results

The developed autonomous vehicle was used to perform a field survey in the Cadriano experimental apple orchard. A preliminary dataset composed of a total of 156 apple fruits and 73 tennis balls was collected and analyzed by matching the manually tagged and measured apples with the deep-learning YOLO detections. The developed detection and sizing processes

(method 1 and method 2) appeared to be promising in determining the dimensions of fruits and tennis balls. Preliminary results showed in Table 8 reveals that by using a squared-bbox (method 1), the accuracy of the estimated object size improved, as the RMSE for apples (7.2 mm) and tennis balls (7.9 mm) was lower than when running the circular-bbox method (8.6 mm and 9.4 mm, respectively). This can be mainly due to the fact that the circle detection is not always accurate during the real-time testing. As highlighted in Figure 81, if the circle detection is wrong, the whole sizing estimation is affected, as the radius will be incorrect and also depth information will be inaccurate. Both methods seem to better estimate apple size with respect to the tennis balls. Considering that tennis balls have a relatively regular shape and a lower size variability, this seems counterintuitive. Based on RMSE % values, method 1 (squared-bbox) and method 2 (circular-bbox) seem to perform equally on both types of objects (apple, tennis ball), with values of 12% and 14% respectively.

Table 8 - RMSE and RMSE % results obtained for both depth methods computed for the apple and tannis ball datasets

	Apple dataset		**Tennis ball dataset**	
	RMSE	**RMSE %**	**RMSE**	**RMSE %**
Squared bbox	7.23	12.0	7.9	12.06
Circular bbox	8.63	14.0	9.42	14.0

With regards to the effectiveness of the customized circle detection algorithm, it was able to identify "best-fitting circles" on 141 apples out of 155 apples (90 percent); for tennis balls, the algorithm was able to extract a "best-fitting circle" on each of the 73 tennis balls. According to the real-time application of the system, the entire process, from loading the images to saving the results, took 0.55 seconds (0.34 seconds for YOLOv5 model application, 0.21 seconds for fruit size estimation using both methods) per image, on a laptop with a 16-GB of RAM and an Intel i7-9750H CPU.

The results showed that the importance of object detection cannot be understated since it has been shown that errors can sometimes just be related to a mis-sized object detection bbox. The circle detection algorithm might not work correctly in that case or might even miss the subject entirely. In the case of regular shapes like tennis balls, the system seems to generally increase the estimation size of the object -- evidenced by finding the smallest RMSEs and mean error (2.4mm +1.3mm) when substituting the detected circle size in pixels with smallest bounding box dimension (e.g., smallest between x and y sizes). Supporting this, is the fact that all of the mean errors computed in relation to the actual dimension of tennis balls are greater than zero. Again, due to the non-regular shape of fruits, the algorithm has a negative mean error (i.e., -1.3 mm; RMSE = 4.7mm -the smallest-) when substituting the minimum bbox dimension for the diameter of the detected circle. Therefore, the system tends to overestimate size for fruits, but with a greater degree of variability for each individual case. The maximum errors achieved in fruit size estimation were -9.2 mm and - 10.4 mm when underestimating, while +24.4 mm and +26.9 mm when overestimating, for method 1 and 2, respectively.

Issues with RGB-D camera sensors due to high levels of incident radiation were also identified. The optical sensor can be saturated, making bright objects undetectable by the CNN, while the infrared sensor used for distance data is also affected by radiation in the same waveband which impacts depth estimation [82]. High illuminance can also cause errors in circle detection (Figure 81: Circle and Depth errors). For this reason, it is advised that images needed for detection and sizing should be acquired during morning or late afternoon when direct light interception and

high intensity lighting are less of an issue. Moreover, intensive shading of the orchard environment helps reduce these technological problems too. Knowing these issues is useful for improving the vision system's efficiency; either through making changes at the right time or calibrating the infrared/RGB sensor's sensitivity. Additionally, recordings can also be taken during night-time using artificial illumination, thus providing a more standardized image/video collection and better overall system performance.

Despite the interesting results, this system still needs some improvement to be in-line with the maximum allowable error range for fruit size estimation (i.e.,1-2 mm). However, large scale data collection that may be enabled by using autonomous vehicles and automated systems, together with algorithm improvements, could lead to average sizing error falling in the suitable 1-2 mm range.

The system is also being improved so that it may run on high-powered processing units dedicated to image analysis (i.e., GPU accelerated systems), which are reported to reduce processing time by as much as 50 percent. Results obtained using an Nvidia RTX-A2000 GPU, showed an average processing time of 20 milliseconds per frame that resulted in an application frame rate of more than 30 frames per second (approximate 50/60 frames per second), thus demonstrating the possibility of receiving online real-time size information about detected objects using conventional cameras capabilities.

4.4. Autonomous fruit harvesting

Harvesting tasks are the one of the most challenging applications to automate, due to the complexity of the scenario involved and the required interaction with the fruits. However, this is also one of the most demanding tasks in terms of labor force and cost associated to it. Given the short applicable harvesting period of some cultivars, it became also difficult to acquire the required personnel to perform a proper harvesting in the appropriate time window.

Some preliminary tests have been made to reach full autonomous fruit picking and harvesting in kiwifruit orchards. The idea is to traverse each orchard lane with an autonomous vehicle, such as the one developed ad described in this work, equipped with several robotic arms by mimic the research described for a similar project in [83]. The kiwifruit is one of the most suitable fruits to develop and test this application, since the targeted harvesting phase is made when the fruit has not yet reached full ripening, therefore it is very compact and robust. This this makes the type of fruit particularly suitable for mechanical handling.

In order to start investigating the process, a commercial low-cost RL-DP-5 robotic arm from Igus featuring 5 Degrees of Freedom (DoF) has been acquired (Figure 82 – left) to create a laboratory setup to start developing fruit grabbing and detaching trajectories. A customized prototype *grasping hand* has been designed and printed using a commercial resin 3D printer (Figure 82 - right). To simplify the grasping task for the first on-field testing campaign, an off-the-shelf Shunk parallel gripper has been installed exploiting the gripper interface on the robotic arm, to enable a linear clamping motion. The grasping tool has been tested with and without soft padding to check fruit response, possible damage and correct slipping while gaining the correct grasping position.

Figure 82 - Igus robotic arm with the customized end effector

The setup was initially tested in laboratory environment, by hanging a kiwifruit to test different possible approach and detach trajectories (Figure 83). In order to speed-up the development process, all the trajectories were initially predetermined and implemented relative to the fruit position that was manually acquired.

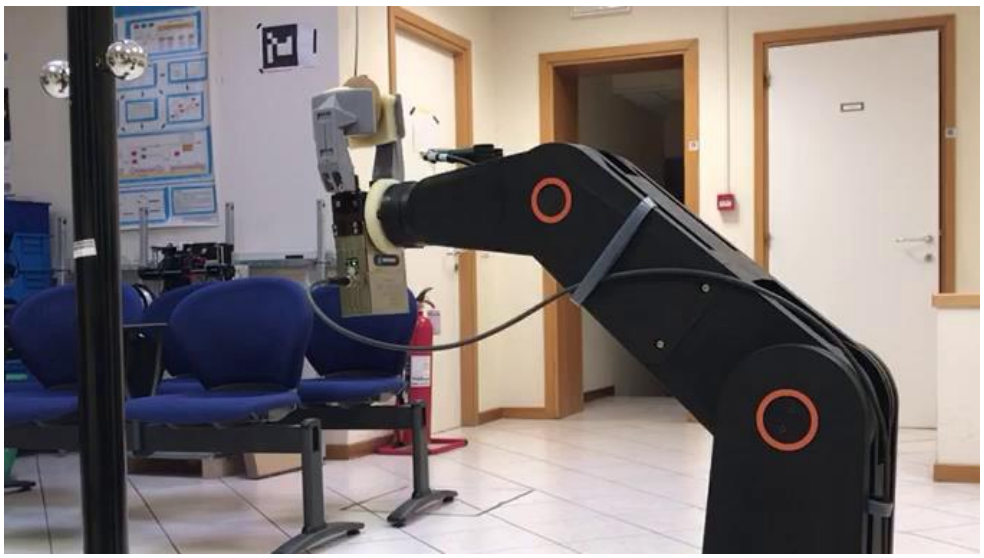

Figure 83 - Kiwifruit grasping in laboratory setup

The process have been configured as follows:

- The kiwifruit position is manually retrieved and saved in the arm workspace
- The kiwifruit approach position is manually retrieved by moving down in the z axis from the fruit position
- The arm start position is randomly acquired in the proximity of the approach position

- The robot starts moving with a computed trajectory from the starting position to the fruit approach position
- From there, the arm moves up to the fruit position
- Upon reaching fruit position, the grasping clamp closes, thus grabbing the fruit
- The detach maneuver begins, that is tilting the end effector to reach approximate 70 degrees
- The arm performs a downward motion to detach the fruit
- The robot moves to a delivery position in order to store the fruit to the appropriate container/storage

All the trajectories are computed according to the kinematics capability of the robotic arm, using both the provided Igus Robot Control software and the community available ROS package MoveIt motion planning framework [84] that provides inverse kinematics and joint position solver among other useful tools.

4.4.1. Preliminary on-field testing

The described kiwifruit picking process has been tested in a real orchard located in the Romagna region. The robotic arm was mounted on a table frame to reach the height of the hanging fruits within the arbor structure orchard. Fruit position and approach position were identified and stored, so that the whole system was able to compute the appropriate trajectories in the joint space. The setup has been successfully picked all the kiwifruit selected without damaging them.

Figure 84 - Real world kiwifruit semi-autonomous picking

Figure 84 above shows the detach moment of a real kiwifruit within a real orchard scenario. The robot movements were autonomous, based on the predefined fruit position taken before the tests within the workspace of the robot.

In order to fully automate the process, kiwifruit detection and tracking is necessary to enable visual servoing control of the end effector to autonomously reach the targeted fruit and initiate grasping procedures. A YOLO object detection neural network was specifically trained to

recognize kiwifruit, based on an image dataset taken in the orchard. Similarly to what has been already described in Section 4.2, the implemented ROS node was able to recognize kiwifruit from a camera mounted on the end effector gripper to allow fruit tracking. Figure 85 shows an example of fruit tracking taken from in-field camera footage.

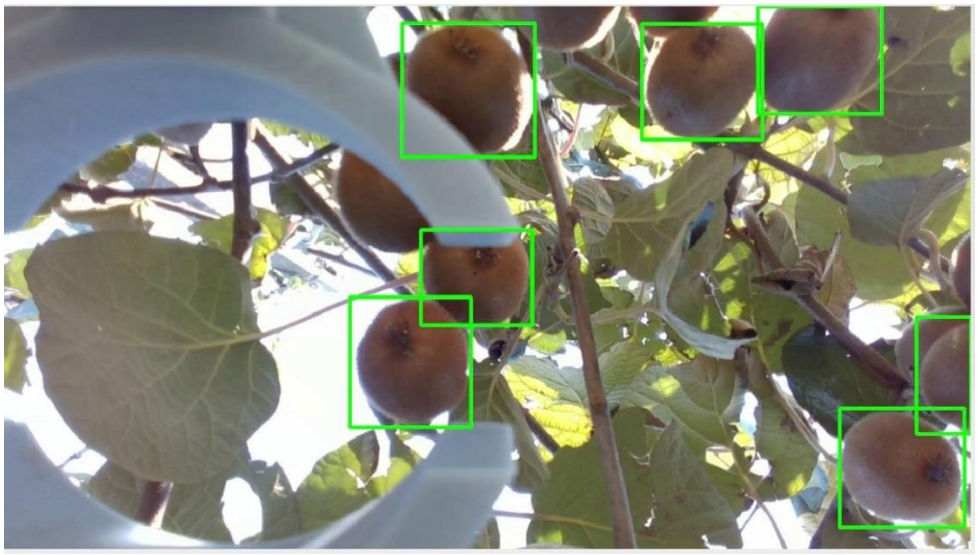

Figure 85 - On-field kiwifruit detection and tracking

Since this research project is still ongoing, only preliminary tests and experiments have been presented, as a full autonomous task execution has not yet been performed. Future improvements will merge together the whole implementation into a single ROS-enabled system to test the fully autonomous application and then boost the robotic arm performances to achieve a commercially useful autonomous fruit harvesting application by also equipping the developed autonomous vehicle with multiple robotic picking arms.

4.5. Canopy estimation for precision spraying

Precision Agriculture relies also on Variable Rate Application (VRA) of inputs, such as fertilizers, herbicides, or other types of chemical inputs.

Spraying is a key feature of orchards and vineyards, with the purpose of protecting crops from fungi and diseases [85]. However, its operation can affect both to the human operator health [86] and the environmental safety [87]–[89], due to the chemical pesticides used, which may drift away from their target. So, when addressing spraying applications, it is crucial to minimize droplets drift due to wind conditions or incorrect setup of the implement. Precision farming and autonomous vehicles offer an opportunity for farmers to efficiently protect their crop, save on labor costs and uphold regulations. This type of spraying applies ultra-low rates of chemicals exactly where they are needed and when they are required. Research has been undertaken to examine the effectiveness of low and ultra-low doses to control pests and diseases ([90]–[92]).

In order to reduce water consumption and chemical dispersion in the environment, plant trait estimation, such as canopy attributes [93] and features are significant to properly adjust and control new generation spraying implements.

A huge number of manual and time-consuming operations were required to obtain reliable data on plant structure (i.e., canopy and woody characteristics) before the diffusion of image analysis

and sensors. Due to the advent of depth sensing technology in the last few years, it has become increasingly easier to extract information relating to the structure of trees. The plants 3D reconstruction (generally as a point cloud) has been improved to a satisfactory level, from which the desired information can be extracted and modeled remotely, without going directly into the field if not for collecting the 3D reconstruction. As evidence of this, previous research activities calculated canopy volume in single vineyard rows employing various techniques such as 3D occupancy grid, convex hull, oriented and axis-aligned bounding boxes on its 3D point cloud reconstruction generated by Intel Realsense R200 depth camera [94]. Another example, based on vine measurement, is provided in a study which highlights a robust correlation ($r2 > 0.9$) between canopy volume estimated with a Microsoft Kinect camera and hand measured canopy volume [95]. Furthermore, the same study observed that camera volume is also correlated with leaves number, leaf area and Leaf area Index indicating the potential of data estimated using 3D sensor detectors. The research work presented in [96], described a method to evaluate Leaf Area Density and also Leaf Area Index for *big* trees, such as magnolia, that can make data collection feasible at a speed, thus enabling large scale data collection. It utilizes a 3D LiDAR scan with point cloud segmentation and a voxel-based model.

The major disadvantage of the presented examples is their reliance on point cloud analysis, which necessitates a great amount of computational power for both 3D model creation and plant structure data extraction. As a result, this data cannot be used immediately, but instead is usually incorporated into prescription maps pertaining to the addressed farming task. These maps are quite useful for improving efficacy; however, they can only remain valid for short periods of time (i.e., when the scan was performed). This proves problematic in orchards, as these systems are always changing. thus, a real-time process is needed to account for any changes that have occurred between 3D scanning and operational execution. To address this issue, alternatives that require fewer computational resources but be investigated.

It became interesting to use canopy porosity, as the percentage of the canopy's external surface that is not covered by leaves (e.g., "light holes"), to improve variable rate spraying and fertilization. Although real-time spraying rate systems have already been investigated [97], they are usually complex by integrating many inexpensive sensors, or using only a few expensive sensors for high canopy resolution.

4.5.1. Developed solution

The presented work aims to investigate the possibility of using a commercially available low-cost RGBD camera to estimate canopy porosity through a computationally lightweight image processing algorithm that exploits the depth reconstruction capabilities typical of stereocamera sensors.

A dedicated canopy estimation ROS node has been developed to be integrated in the UGV prototype developed. By exploiting the depth image information coming from two Intel Realsense D435 stereocameras, mounted on the two sides of the vehicle, a canopy porosity percentage can be computed.

The evaluation process takes the provided scene depth image as input to perform some thresholding according to the expected maximum distance of the canopy. This can be done according to the orchard row width parameter and the expected canopy width. As the rows behind are eventually seen at a further distance, the threshold is sufficient to investigate the nearest tree canopy without occurring in other lines noise. An example of the acquired depth can be seen in Figure 86.

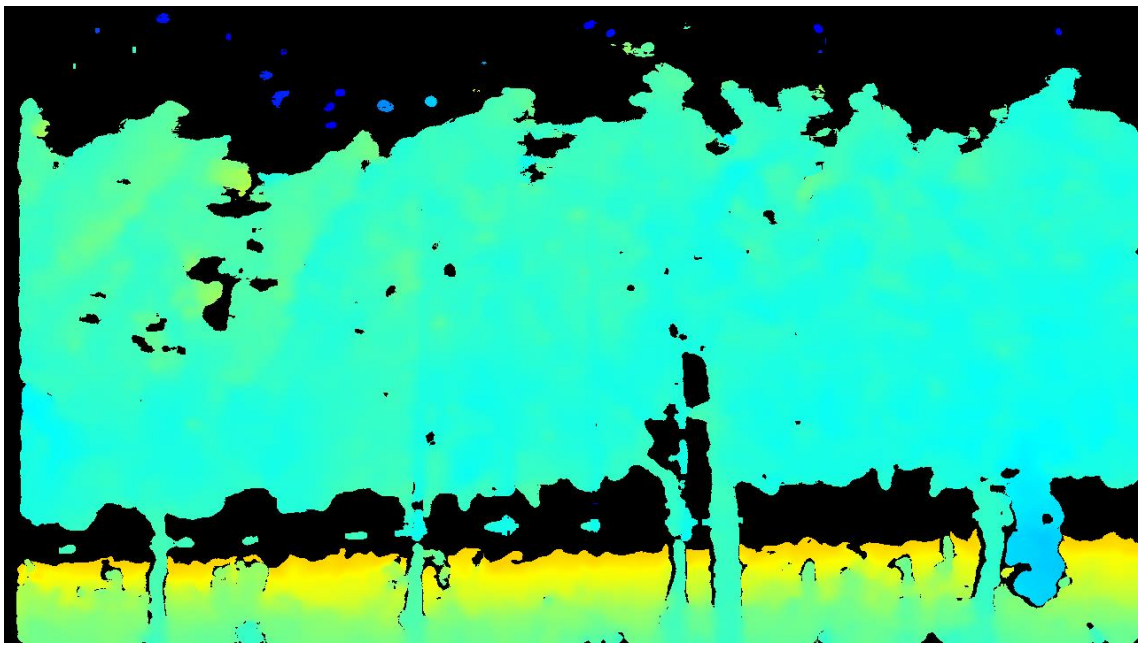

Figure 86 - Depth estimation on a Planar Cordon apple orchard

The canopy pixels in the scene are then retained using the following formula, by exploiting the segmented binary depth image obtained after the thresholding:

$$P_{\%} = (Tp_B/Tp) * 100$$

where $P_{\%}$ denotes the porosity percentage, Tp is the total number of pixels and Tp_B is the total number of black pixels in the scene.

According to the binary depth image, it is also possible to retrieve the canopy-colored segmentation, by aligning the depth image to the color one. The process can be made either manually, by adjusting depth image size according to the color frame dimensions, or by exploiting the available functions of the Intel Realsense camera, that automatically produces an aligned depth image with respect to the color image properties.

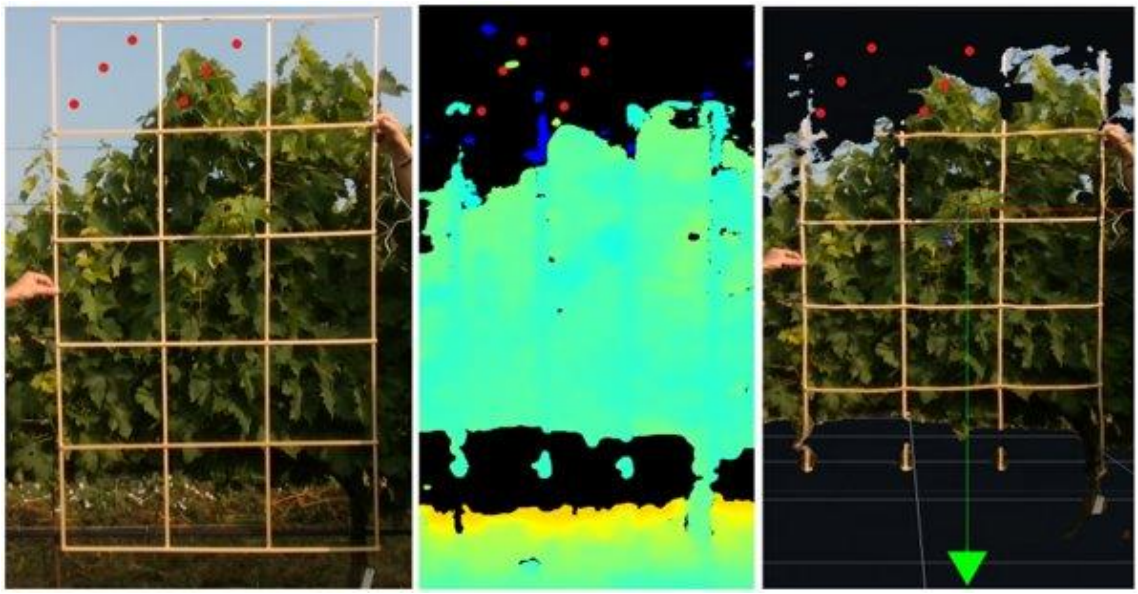

Figure 87 - Image segmentation process applied to compute canopy porosity

To enhance debug capabilities and provide a user feedback on the processed results, the colored canopy image is also processed, and an example can be seen in Figure 87. The same picture shows also the grid used for manual assessment of the porosity value. This value can then be used as the ground truth to compare algorithm results. As the operation is quite labor intensive, the validation process is still undergoing, but the accuracy of the obtained results can be easily checked by means of the colored segmented image.

In addition, Figure 88 highlight different Region of Interest (ROI) configuration that can be applied to the computation in order to match the capabilities of the sprayer used. As commercially available sprayers may be equipped with a different number of electrovalves and may have a different number of nozzles, the porosity computation node can be configured to divide the whole frame (as it is supposed to cover the whole sprayed area) in different region of interest according to the number of nozzles installed on the implement. Typically, the nozzles are equally distributed across all the sprayer height, therefore the frame is equally divided into different areas and the computation is performed according to the pixels present in the specific area. The overall output is then not only a number, but a vector of percentage values that can then be sent through CAN or Serial communication to modern sprayers to integrate the porosity system with the Pulse Width Modulation (PWM) controlled electrovalves, able to adjust the water flow accordingly. Obviously, it is expected that the nozzle could be completely shut-off if no vegetation is detected in the area.

Figure 88 - Different ROI configuration examples according to sprayer capabilities

Together with porosity estimation, other canopy traits can be computed by taking advantage of the depth estimation of the stereocamera. Therefore, in the same ROS node it is also provided an estimation of the canopy volume. By keeping all the distances obtained from the canopy pixels, the overall distance average can be computed. According to the configured row width, and assuming that the camera Point of View (PoV) is placed in the middle of the lane, the estimated canopy extension can be calculated as:

$$Canopy_{ext} = (row_{width}/2) - Canopy_{dist}$$

where $Canopy_{ext}$ is the mean canopy extension, $Canopy_{dist}$ is the mean canopy distance obtained from the depth map and row_{width} is the configured row width value. Then, the real scene dimensions according to the camera FoV values and at half row distance, can be computed as:

$$H_{scene} = 2 * \frac{row_{width}}{2} * \tan\left(\frac{FoV_H}{2}\right)$$
$$V_{scene} = 2 * \frac{row_{width}}{2} * \tan\left(\frac{FoV_V}{2}\right)$$

In which FoV_H and FoV_V are respectively the horizontal and vertical camera field of view expressed in radians.

The overall scene area is then computed by multiplying the two horizontal and vertical scene dimensions according to the above computation: $A_{scene} = H_{scene} * V_{scene}$. The same area can be adjusted according to the computed porosity to adjust canopy volume with respect of actually canopy occupancy: $A_{scene\_adj} = A_{scene} * \big((Tp - Tp_B)/Tp\big)$, as $\big((Tp - Tp_B)/Tp\big)$ can be observed to be the percentage amount of *occupied* canopy pixels in the scene. The canopy volume is then computed according to the mean canopy extension $Canopy_{ext}$, as $Canopy_{vol} = A_{scene} * Canopy_{ext}$, and also adjusted with respect to the computed porosity as $Canopy_{vol\_adj} = A_{scene\_adj} * Canopy_{ext}$. In this case, the canopy is observed disregards of the sprayer ROIs, as the whole foliage spans through the whole frame. Nevertheless, in case ROI specific volume is required, the same principles apply considering only the ROI area.

4.6. Post-harvesting estimations[10]

Since fruit value on the market is primarily function of size, color and absence of defects, being able to have information regarding these parameters directly in the field could help growers increase profitability by being able to sell top-quality fruit to those consumers that are willing to spend more for a better product, and by also taking advantage of being able to perform direct sales through a local shop or marketplace. In recent times, computer vision has been applied both to the evaluation and selection of fruit quality before storage [98]–[100] and to the detection of post-harvest disorders/diseases [101].

4.6.1. Developed framework

The computer vision system currently still under development consists of a small computing unit connected to a depth camera (RGB-D) and a standard-resolution GPS module. The experimental validation was conducted in Ravenna, Italy, in the 2021 season during the harvest of one hectare of peach (Prunus persica L. Batsch) orchard.

On the 33 bins filled within the field and representing the entire production for the season, it has been manually measured the average size of 40 fruits selected from the first visible layer of each container. Each fruit was measured with a digital caliper and then put back into the container making sure to keep it in the top layer for later image collection. For each container, latitude and longitude coordinates relative to the field were also retrieved for future evaluation.

Two different types of Intel Realsense depth cameras (model D435i and model D455) were tested under both natural (n) and artificial (a) light conditions. In order to position the camera properly, a wooden parallelogram frame was created to fit the top of the loading platform, and both cameras were firmly fixed in its upper central part so as to get a privileged point of view for the bin surface (Figure 89).

[10] This work has contributed to the published paper: A computer vision system for in-field quality evaluation: preliminary results on peach fruit. - Bortolotti, G., Mengoli, D., Piani, M., Grappadelli, L.C., Manfrini, L. - 2022 IEEE Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2022 - Proceedings, 2022, pp. 180–185

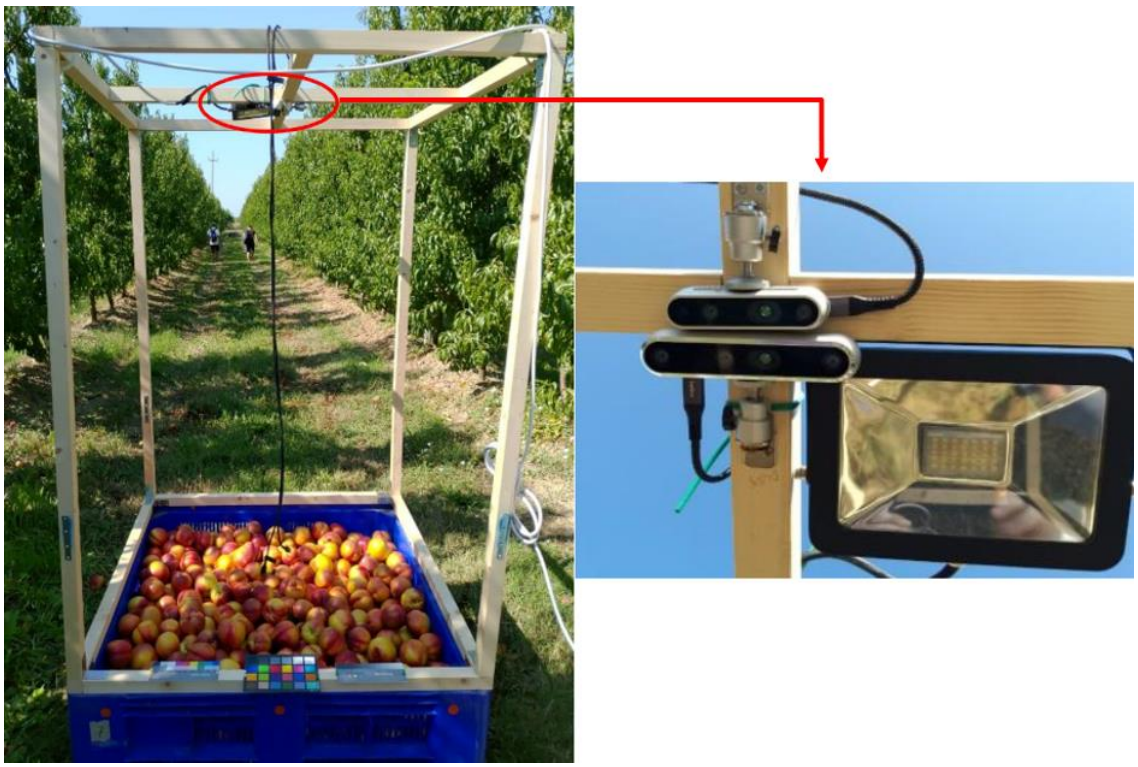

Figure 89 - Bin setup with wooden structure for cameras and illuminator. The right side is a zoomed view of the red circled area for sensor mounting.

The height determined for the wooden frame is 1.75 m. This was set according to the lowest field of view (FoV) of the cameras (given in this case by the D435i sensor), in order to ensure that both of them framed the entire surface of the container. To test the different lighting conditions, the wooden frame was left uncovered for the natural light condition, while, for the artificial light condition, it was completely covered with 100% shading plastic sheeting and a 30w LED lamp was used.

RGB-D data collection is given by stationary recording of the container surface of both cameras for each light condition. For each container, a color image (RGB) and a depth map (composed of a matrix with the depth information of each RGB pixel) were extracted from these recordings. As the whole system and infrastructure is still under development, the supplied Intel Realsense Viewer software running on a standard laptop was used for the data collection. The aligned color images and depth maps were extracted using a custom developed Python script, exploiting Intel Realsense SDK2.0 libraries.

A Convolutional Neural Network (CNN) was used to detect and identify the fruits in the top layer of the container in order to automate data extraction. The YOLOv5 [70] size “s” model was selected as object detection neural network, because of its speed and variable size. The custom peach model was trained for 500 epochs, on the obtained images, using two NVIDIA RTX2080super GPUs.

The Python script implemented to automatically extract the average fruit size operates as follows:

- The custom trained peach detection YOLOv5 model was applied on the RGB image of the container, to identify the most visible fruits in the upper layer.

- The coordinates of the RGB fruit identification pixels were projected onto the corresponding depth map.
- In this, for each identified fruit, the pixel dimensions (D_{minp}, D_{maxp}) of the detection bounding box (bbox) were extracted.
- The average distance of the individual fruit from the camera was computed as the average of all depth values in the fruit detection bounding boxes.
- The actual dimensions of the scene (RSD_v – vertical and RSD_h – horizontal) were calculated from the equation $real_{SD} = 2 * (FD * \tan(\frac{1}{2} FoV))$, in which $real_{SD}$ is the considered dimension of the framed scene (in mm), FD is the mean fruit distance and FoV is the size of the field of view according to the selected camera.
- The resolution of the real object (mm2/pixel) was obtained by dividing the area of the real scene ($RSD_v * RSD_h$, mm2) by the resolution of the image (1280 * 720, pixels). The size of the real object (D_{minr}, D_{maxr}) was then obtained by multiplying D_{minp} and D_{maxp} by the square root of the resolution of the real object.
- The size of the single real object, i.e., the diameter of the fruit, was estimated as the average of D_{minr} and D_{maxr}.
- The mean fruit size extracted from RGB-D bin was finally obtained as the average of the estimated single fruit diameter, for all fruits detected by the YOLOv5 CNN.

The Root Mean Square Error (RMSE) between the RGB-D extraction and the average manual measurements of the fruit bins was calculated by considering the manual survey as the reference value. This process was performed for each light/camera combination: D435n, D435a, D455n, D455a.

4.6.2. Experimental results

The YOLOv5 model trained for detecting visible fruit in the first layer of the bin presented interesting performance, obtaining a value of 0.81 on the F1 score at the 500th epoch. This result is in line with other studies using CNN algorithms for fruit detection in the field [102], [103], although better results are possible, as stated by [104]. It is worth noting that the application of the YOLOv5 detection in this case is different from the “in-field on-the-tree” scenario: the fact that the model should slightly underperforming can be attributed to the difficult background situation, where all the fruits are stacked together without a clear background, and where the CNN model should differentiate the most visible fruits (Figure 90, 'RGB'). In fact, this situation is more complex than differentiating fruits from a leaf-based background. The model performances can be improved by increasing the number of images used for the training set, as evaluated by [105]. In addition, the use of artificial images can be useful to extend the dataset size with no or minimal effort in terms of labelling time.

The goal of the fruit detection model was to recognize enough visible fruits (i.e., more than 40), so to have a reliable sample on which to estimate the average size. The average fruit detections per container were 64, 44, 43, and 38 for the D435a, D435n, D455n, and D455a situations, respectively. The best results obtained for the D435 camera, under both light conditions, are probably due to its higher digital resolution according to the bin area, with respect to the D455 camera, precisely 0.30 vs. 0.13 pixels/mm2, at 1.7 m distance. This difference is due to the different camera FoVs that create different object image sizes according to the same subject distance (Figure 90 'RGB' and 'bbox').

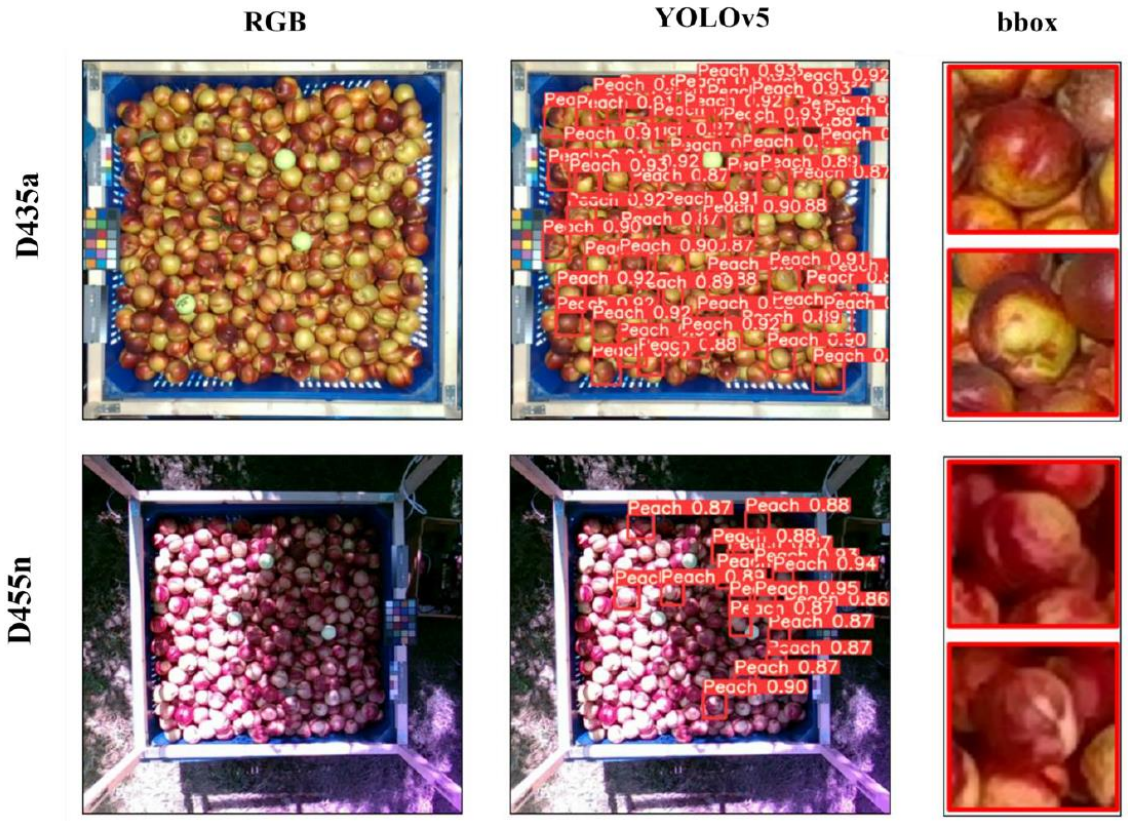

Figure 90 - Comparison examples of two opposite treatments (D435a, D455n) for RGB image collection (RGB), YOLOv5 fruit detection results (YOLOv5) and a zoomed detail of the detected fruits (bbox). On the YOLOv5 column it can be seen that natural lighting is negatively affecting fruit detections. On the bbox column, it is highlighted the different object resolution and image quality for the two camera sensors. None of the treatments allows for a proper bounding box (red contour) adhesion to the detected fruit perimeter.

Regarding light condition, the D435 camera gets better results in artificial light condition than in natural light condition, while the opposite case occurs for the D455 camera. There is no clear explanation at the moment for this behavior, although the artificial light condition (a) should have favored fruit detection for both cameras over the more variable natural light condition (n), in which the two cameras provided similar results.

The YOLOv5 section of Figure 90 highlight the differences in fruit detection results: natural illumination, with the presence of light/shadow areas, negatively affected fruit detection for both cameras, while the artificial light condition favored fruit detection for both cameras, as RGB sensors are particularly sensitive to brightness and light changes [106], [107]. The results of D435n and D455a are omitted in the Figure 90. In the depicted examples, it can be seen that most of the detections, for the daylight situation, are in the shaded part of the image, which leads to the view that overexposed images are not suitable for YOLO detection with RGB sensors. Nevertheless, it was also investigated whether the application of the CNN algorithm could overcome this problem by using deep learning methods to obtain more reliable results [104].

About fruit sizing performances, Figure 91 presents scatter plots between the average fruit size of the RGB-D detection and the manual measurement value. It is also stated the RMSE value, and the average fruit size computed among all bins (mFS). As can be seen, the D435a treatment again produce the best results, i.e., stronger correlation with r = 0.74 and lower RMSE value of 17.91 mm than any other situation. The D435n case provided the second-best condition for

RMSE value (21.11 mm), but not for correlation where it has a slightly lower value than D455n, r = 0.586 vs. r = 0.630.

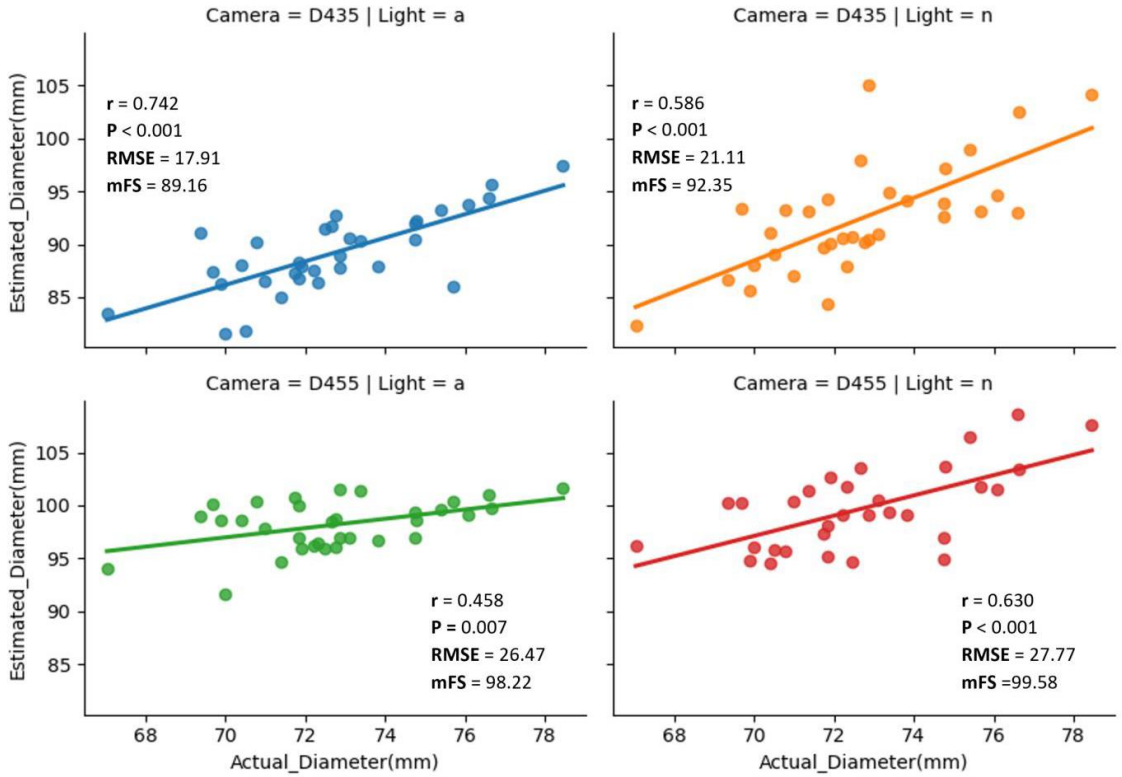

Figure 91 - Scatterplots of actual vs estimated (RGB-D extracted) mean bins fruit size per each treatment. Each graph represents Pearson's correlation r and p values, standard error (SE - mm), root-mean-squared error (RMSE - mm) and mean fruit size among all the bins analyzed (mFS - mm).

With respect to fruit sizing application, thus considering only RMSE values, D455a is not the worst case, as is happens for fruit detection results, but it turns out to be D455n. The differences in values between artificial and natural light conditions are greater for D435 camera than for the D455 model. For D435, artificial illumination reduced the RMSE by 15.2 percent, while for D455 the RMSE was reduced by only 3.8 percent, suggesting that, for the latter camera, sizing is not affected by illumination. This may be due to the smaller number of overall fruit detections that probably accounts for better bounding boxes results. Then, considering both cameras in the same light condition, it can be seen that using D435 reduces RMSE more than D455, 32.4% vs. 24%, for artificial and natural light, respectively. As for the average effective size (given by manual measurements) of the fruits in all the containers analyzed, it has been found to be 72.7 mm. The results obtained for each situation, regarding this last parameter (Figure 91, mFS values), clearly show that the developed system tends to largely overestimate fruit size. The closest value to the average actual fruit size is 89.16 mm, scored by D435a. Therefore, these results emphasize that there is a link between camera and light conditions for the system sizing performance.

The main motive for the high RMSE in size estimation is the poor accuracy of the YOLOv5 model in detecting peaches, partly because the bbox dimensions are largely oversized relative to those of the fruit (Figure 90 'bbox'). In fact, to obtain the correct fruit dimensions, the detection bounding box should be as close as possible to the fruit perimeter according to how the size is computed. In order to achieve this, the fruit detection model needs to be improved or other techniques should be applied to single fruit detection such as circle detection, blob detection,

threshold, or others, as seen in Section 4.3.1. These methods are likely to increase the detection accuracy, thus reducing the sizing error under the 2 mm threshold.

Another drawback concerns the bright light condition, as may happen in the case of natural illumination. This may alter the RGB-D in distance estimation, thus affecting the accuracy in fruit sizing. In fact, to estimate distance, the Realsense stereocamera that have been used, utilize the principle of active stereovision, taking advantage of infrared (IR) sensors [108]; intense sunlight can alter the performance of the sensors, by severely limiting them, due to the presence of IR wavelengths that interfere with distance estimation. This suggests that we should avoid high light intensity conditions when using IR sensors in RGB-D cameras on the field [109].

4.7. Conclusions

In this chapter, some of the investigated precision farming applications that are enabled by the exploitation of the autonomous capabilities of the constructed vehicle are presented. These accounts for in-field data collection systems and databases, fruit detection and sizing and canopy properties estimation. Further development will focus on improving fruit detection and size estimation to enable not only better monitoring and yield prediction across the growth season, but also to enable fully autonomous harvesting for different fruit types. In this last scenario, also fruit ripening assessment can be of valuable importance to select the appropriate fruit to harvest and/or directly perform in-field classification and selection according to both size and weight attributes and quality parameters.

Chapter 5. Final notes

The presented work was conceived to be a comprehensive guide for the developed ground robotic platform for precision agriculture. The design process from both hardware and software perspectives has been detailed to show the basic ideas behind the UGV and the autonomous navigation features implemented on the vehicle. The tools and the precision farming applications investigated by exploiting the rover have also been presented as part of the more general agricultural robotics project.

The UGV development is the result of the work of several PhD students that were involved into the agricultural robotics project born within the autonomous control group of DEI department, and which has recently evolved into a university spin-off company called Fieldrobotics. The research trials described have been made possible thanks to the fruitful cooperation with the department of agricultural and food sciences (DISTAL) groups of mechanical engineering and arboriculture and tree physiology.

The current development status should not be seen as a final achievement, but should instead serve as a starting point for both create a modern and innovative commercial platform to be used within real farming work, and also a research tool to continue developing and implementing other Precision Agriculture applications, enabling even more groundbreaking concepts and solutions as briefly described at the end of each chapter. Who knows what the future holds for us.

Bibliography

[1] M. Roser, H. Ritchie, E. Ortiz-Ospina, and L. Rodés-Guirao, “World Population Growth,” *Our World in Data*, 2013.

[2] H. Ritchie and M. Roser, “Environmental Impacts of Food Production,” *Our World in Data*, 2022.

[3] V. Rondelli, C. Casazza, and R. Martelli, “Tractor rollover fatalities, analyzing accident scenario,” *J Safety Res*, vol. 67, pp. 99–106, Dec. 2018, doi: 10.1016/J.JSR.2018.09.015.

[4] M. L. Myers, H. P. Cole, and S. C. Westneat, “Injury severity related to overturn characteristics of tractors,” *J Safety Res*, vol. 40, no. 2, pp. 165–170, Jan. 2009, doi: 10.1016/J.JSR.2009.02.007.

[5] D. Facchinetti, S. Santoro, L. E. Galli, and D. Pessina, “Agricultural Tractor Roll-Over Related Fatalities in Italy: Results from a 12 Years Analysis,” *Sustainability*, vol. 13, no. 8, 2021, doi: 10.3390/su13084536.

[6] S. J. Reynolds and W. Groves, “Effectiveness of roll-over protective structures in reducing farm tractor fatalities,” *Am J Prev Med*, vol. 18, no. 4, pp. 63–69, May 2000, doi: 10.1016/S0749-3797(00)00142-2.

[7] R. Gebbers and V. I. Adamchuk, “Precision Agriculture and Food Security,” *Science (1979)*, vol. 327, no. 5967, pp. 828–831, Feb. 2010, doi: 10.1126/science.1183899.

[8] R. Bongiovanni and J. Lowenberg-Deboer, “Precision Agriculture and Sustainability,” *Precis Agric*, vol. 5, no. 4, pp. 359–387, 2004, doi: 10.1023/B:PRAG.0000040806.39604.aa.

[9] A. J. Scarfe, R. C. Flemmer, H. H. Bakker, and C. L. Flemmer, “Development of an autonomous kiwifruit picking robot,” in *2009 4th International Conference on Autonomous Robots and Agents*, 2009, pp. 380–384. doi: 10.1109/ICARA.2000.4804023.

[10] P. Ruangurai, M. Ekpanyapong, C. Pruetong, and T. Watewai, “Automated three-wheel rice seeding robot operating in dry paddy fields,” vol. 9, pp. 403–412, Jan. 2015.

[11] M. Stein, S. Bargoti, and J. Underwood, “Image Based Mango Fruit Detection, Localisation and Yield Estimation Using Multiple View Geometry,” *Sensors*, vol. 16, no. 11, 2016, doi: 10.3390/s16111915.

[12] R. Jørgensen *et al.*, “HortiBot: A System Design of a Robotic Tool Carrier for High-tech Plant Nursing,” *CIGR J. Sci. Res. Dev.*, vol. IX, Jan. 2006.

[13] C. Sørensen *et al.*, “Hortibot: Feasibility study of a plant nursing robot performing weeding operations – part IV,” Jan. 2007. doi: 10.13140/2.1.1551.1040.

[14] T. Bakker, K. van Asselt, J. Bontsema, and E. J. van Henten, “Robotic weeding of a maize field based on navigation data of the tractor that performed the seeding,” *IFAC Proceedings Volumes*, vol. 43, no. 26, pp. 157–159, 2010, doi: https://doi.org/10.3182/20101206-3-JP-3009.00027.

[15] T. Bakker, K. van Asselt, J. Bontsema, J. Müller, and G. van Straten, “A path following algorithm for mobile robots,” *Auton Robots*, vol. 29, no. 1, pp. 85–97, 2010, doi: 10.1007/s10514-010-9182-3.

[16] F. K. van Evert *et al.*, “A robot to detect and control broad-leaved dock (Rumex obtusifolius L.) in grassland,” *J Field Robot*, vol. 28, no. 2, pp. 264–277, 2011, doi: https://doi.org/10.1002/rob.20377.

[17] P. Peças, G. M. Fonseca, I. I. Ribeiro, and C. G. Sørensen, “Automation of Marginal Grass Harvesting,” 2019, pp. 106–146. doi: 10.4018/978-1-5225-5909-2.ch006.

[18] S. H. Nielsen *et al.*, “A Low Cost, Modular Robotics Tool Carrier For Precision Agriculture Research,” in *11th International Conference on Precision Agriculture*, International Society of Precision Agriculture, 2012.

[19] K. Jensen, M. Laursen, H. Midtiby, and R. Jørgensen, “Autonomous Precision Spraying Trials Using a Novel Cell Spray Implement Mounted on an Armadillo Tool Carrier,” Jan. 2013.

[20] G. Stefanelli, “La trattrice a programmazione senza pilota <BOPS-1960> nel quadro mondiale delle trattrici d’avanguardia,” *Macchine e motori agricoli, n.9*, pp. 49–64, 1960.

[21] “Regulation (EU) No 167/2013 of the European Parliament and of the Council of 5 February 2013 on the approval and market surveillance of agricultural and forestry vehicles Text with EEA relevance.” https://eur-lex.europa.eu/eli/reg/2013/167/oj (accessed Jan. 22, 2023).

[22] S. Blackmore, “New concepts in agricultural automation,” in *HGCA conference: precision in arable farming - current practice and future potential*, London: HGCA, 2009, pp. 127–137.

[23] G. Yamamuchi, K. Nagatani, T. Hashimoto, and K. Fujino, “Slip-compensated odometry for tracked vehicle on loose and weak slope,” *ROBOMECH Journal*, vol. 4, Nov. 2017, doi: 10.1186/s40648-017-0095-1.

[24] K. Nagatani, D. Endo, and K. Yoshida, “Improvement of the odometry accuracy of a crawler vehicle with consideration of slippage,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 2752–2757.

[25] D. Endo, Y. Okada, K. Nagatani, and K. Yoshida, “Path following control for tracked vehicles based on slip-compensating odometry,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 2871–2876.

[26] M. G. Bekker, “Off-the-road locomotion,” *Research and development in terramechanics*, 1960.

[27] J. Y. Wong, *Theory of ground vehicles*. John Wiley & Sons, 2008.

[28] W. Steeds, “Tracked vehicles,” *Automobile Engineer*, vol. 40, p. 526, 1950.

[29] M. G. Bekker, “Theory of land locomotion,” 1956.

[30] J. Y. Wong and C. F. Chiang, “A general theory for skid steering of tracked vehicles on firm ground,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 215, no. 3, pp. 343–355, 2001.

[31] S. Al-Milli, L. D. Seneviratne, and K. Althoefer, “Track–terrain modelling and traversability prediction for tracked vehicles on soft terrain,” *J Terramech*, vol. 47, no. 3, pp. 151–160, 2010.

[32] B. M. D. Wills, “The measurement of soil shear strength and deformation moduli and a comparison of the actual and theoretical performance of a family of rigid tracks,” *J Terramech*, vol. 1, no. 2, pp. 83–88, 1964, doi: https://doi.org/10.1016/0022-4898(64)90068-0.

[33] T. Keller and J. Arvidsson, “A model for prediction of vertical stress distribution near the soil surface below rubber-tracked undercarriage systems fitted on agricultural vehicles,” *Soil Tillage Res*, vol. 155, pp. 116–123, 2016.

[34] W. Liu and K. Cheng, “An Analytical Model for Predicting Ground Pressure under a Rigid-Flexible Tracked Vehicle on Soft Ground,” *Math Probl Eng*, vol. 2020, 2020.

[35] F. Peyret, P.-Y. Gilliéron, L. Ruotsalainen, and J. ENGDAHL, *SaPPART White paper : Better use of Global Navigation Satellite Systems for safer and greener transport*. 2015.

[36] D. A. Grejner-Brzezinska, C. K. Toth, T. Moore, J. F. Raquet, M. M. Miller, and A. Kealy, “Multisensor Navigation Systems: A Remedy for GNSS Vulnerabilities?,” *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1339–1353, 2016, doi: 10.1109/JPROC.2016.2528538.

[37] “MAVLink Developer Guide.” https://mavlink.io/en/ (accessed Dec. 31, 2022).

[38] B. Zhou, Y. Peng, and J. Han, “UKF based estimation and tracking control of nonholonomic mobile robots with slipping,” in *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2007, pp. 2058–2063.

[39] T. M. Dar and R. G. Longoria, “Slip estimation for small-scale robotic tracked vehicles,” in *Proceedings of the 2010 American control conference*, 2010, pp. 6816–6821.

[40] F. Rogers-Marcovitz, N. Seegmiller, and A. Kelly, “Continuous vehicle slip model identification on changing terrains,” in *RSS 2012 Workshop on Long-term Operation of Autonomous Robotic Systems in Changing Environments*, 2012.

[41] M. Burke, “Path-following control of a velocity constrained tracked vehicle incorporating adaptive slip estimation,” in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 97–102.

[42] D. Mengoli, A. Eusebi, S. Rossi, R. Tazzari, and L. Marconi, “Robust autonomous row-change maneuvers for agricultural robotic platform,” in *2021 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, 2021, pp. 390–395. doi: 10.1109/MetroAgriFor52389.2021.9628694.

[43] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, “Review on model predictive control: an engineering perspective,” *The International Journal of Advanced Manufacturing Technology*, vol. 117, pp. 1–23, Jan. 2021, doi: 10.1007/s00170-021-07682-3.

[44] C. E. Rasmussen, “Gaussian Processes in Machine Learning,” in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, U. and R. G. Bousquet Olivier and von Luxburg, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71. doi: 10.1007/978-3-540-28650-9_4.

[45] D. Moore Thomas and Stouch, “A Generalized Extended Kalman Filter Implementation for the Robot Operating System,” in *Intelligent Autonomous Systems 13*, N. and B. K. and Y. H. Menegatti Emanuele and Michael, Ed., Cham: Springer International Publishing, 2016, pp. 335–348.

[46] ROS.org, “ROS Navigation Stack.” [Online]. Available: http://wiki.ros.org/navigation

[47] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, “The office marathon: Robust navigation in an indoor office environment,” in *2010 IEEE international conference on robotics and automation*, 2010, pp. 300–307.

[48] ROS.org, “ROS Global Planner.” [Online]. Available: http://wiki.ros.org/global/_planner

[49] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[50] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968, doi: 10.1109/TSSC.1968.300136.

[51] F. Kamil and K. N, “A Review on Motion Planning and Obstacle Avoidance Approaches in Dynamic Environments,” *Advances in Robotics & Automation*, vol. 04, Jan. 2015, doi: 10.4172/2168-9695.1000134.

[52] C. Guo, Z. Sun, Y. Chen, Y. Xie, S. Li, and H. Qian, “Trajectory Tracking of Unicycle-type Robots with Constraints,” in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2018, pp. 1700–1705. doi: 10.1109/ROBIO.2018.8665273.

[53] J. Kim and Y. Choi, “Trajectory generation of wheeled mobile robot using convolution method,” in *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2011, pp. 371–374. doi: 10.1109/URAI.2011.6145999.

[54] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525. doi: 10.1109/ICRA.2011.5980409.

[55] C. Richter, A. Bry, and N. Roy, “Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments,” in *Robotics Research: The 16th International Symposium ISRR*, 2016, pp. 649–666. doi: 10.1007/978-3-319-28872-7_37.

[56] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[57] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, “Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1872–1878.

[58] M. A. Jenkins, “Algorithm 493: Zeros of a real polynomial [c2],” *ACM Transactions on Mathematical Software (TOMS)*, vol. 1, no. 2, pp. 178–189, 1975.

[59] C. Lehnert, A. English, C. McCool, A. W. Tow, and T. Perez, “Autonomous Sweet Pepper Harvesting for Protected Cropping Systems,” *IEEE Robot Autom Lett*, vol. 2, no. 2, pp. 872–879, 2017.

[60] F. A. A. Cheein and R. Carelli, “Agricultural robotics: Unmanned robotic service units in agricultural tasks,” *IEEE industrial electronics magazine*, vol. 7, no. 3, pp. 48–58, 2013.

[61] M. Bergerman, S. Singh, and B. Hamner, “Results with autonomous vehicles operating in specialty crops,” in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 1829–1835.

[62] M. Bergerman *et al.*, “Robot farmers: Autonomous orchard vehicles help tree fruit production,” *IEEE Robot Autom Mag*, vol. 22, no. 1, pp. 54–63, 2015.

[63] D. Mengoli, R. Tazzari, and L. Marconi, “Autonomous Robotic Platform for Precision Orchard Management: Architecture and Software Perspective,” in *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, 2020, pp. 303–308. doi: 10.1109/MetroAgriFor50201.2020.9277555.

[64] R. O. Duda and P. E. Hart, “Use of the Hough transformation to detect lines and curves in pictures,” *Commun ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[65] S. Singh *et al.*, “Comprehensive Automation for Specialty Crops: Year 1 results and lessons learned,” *Intell Serv Robot*, vol. 3, pp. 245–262, Dec. 2010, doi: 10.1007/s11370-010-0074-3.

[66] L. Comba, P. Gay, J. Primicerio, and D. Ricauda Aimonino, “Vineyard detection from unmanned aerial systems images,” *Comput Electron Agric*, vol. 114, pp. 78–87, Jun. 2015, doi: 10.1016/J.COMPAG.2015.03.011.

[67] L. G. Shapiro and G. C. Stockman, *Computer Vision*, 1st ed. USA: Prentice Hall PTR, 2001.

[68] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, “Robust Monte Carlo localization for mobile robots,” *Artif Intell*, vol. 128, no. 1, pp. 99–141, 2001, doi: https://doi.org/10.1016/S0004-3702(01)00069-8.

[69] “amcl - ROS Wiki.” http://wiki.ros.org/amcl (accessed Jan. 17, 2023).

[70] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection.” arXiv, 2015. doi: 10.48550/ARXIV.1506.02640.

[71] “Open Source Data Labeling | Label Studio.” https://labelstud.io/ (accessed Jan. 15, 2023).

[72] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple Online and Realtime Tracking,” *CoRR*, vol. abs/1602.00763, 2016, [Online]. Available: http://arxiv.org/abs/1602.00763

[73] N. Wojke, A. Bewley, and D. Paulus, “Simple Online and Realtime Tracking with a Deep Association Metric,” *CoRR*, vol. abs/1703.07402, 2017, [Online]. Available: http://arxiv.org/abs/1703.07402

[74] G. P. Moreda, J. Ortiz-Cañavate, F. J. García-Ramos, and M. Ruiz-Altisent, “Non-destructive technologies for fruit and vegetable size determination – A review,” *J Food Eng*, vol. 92, no. 2, pp. 119–136, 2009, doi: https://doi.org/10.1016/j.jfoodeng.2008.11.004.

[75] S. Serra, R. Leisso, L. Giordani, L. Kalcsits, and S. Musacchi, “Crop load influences fruit quality, nutritional balance, and return bloom in ‘Honeycrisp’ apple,” *HortScience*, vol. 51, no. 3, pp. 236 – 244, 2016, doi: 10.21273/hortsci.51.3.236.

[76] W. S. Qureshi, A. Payne, K. B. Walsh, R. Linker, O. Cohen, and M. N. Dailey, “Machine vision for counting fruit on mango tree canopies,” *Precis Agric*, vol. 18, no. 2, pp. 224–244, 2017, doi: 10.1007/s11119-016-9458-5.

[77] B. Morandi, L. Manfrini, M. Zibordi, M. Noferini, G. Fiori, and L. C. Grappadelli, “A low-cost device for accurate and continuous measurements of fruit diameter,” *HortScience*, vol. 42, no. 6, pp. 1380 – 1382, 2007, doi: 10.21273/hortsci.42.6.1380.

[78] L. M. Peppi, M. Zauli, L. Manfrini, P. A. Traverso, L. Corelli Grappadelli, and L. de Marchi, “A Low-Cost and High-Accuracy Non-Invasive System for the Monitoring of Fruit Growth,” in *2020 IEEE International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2020 - Proceedings*, 2020, pp. 18 – 23. doi: 10.1109/MetroAgriFor50201.2020.9277571.

[79] A. Syal, D. Garg, and S. Sharma, “A Survey of Computer Vision Methods for Counting Fruits and Yield Prediction,” *International Journal of Computer Science Engineering (IJCSE)*, vol. 2, pp. 346–350, Jan. 2013.

[80] D. S. Tustin, B. M. van Hooijdonk, and K. C. Breen, “The Planar Cordon – new planting systems concepts to improve light utilisation and physiological function to increase apple orchard yield potential,” *Acta Hortic*, vol. 1228, pp. 1 – 11, 2018, doi: 10.17660/ActaHortic.2018.1228.1.

[81] G. Bradski, “The openCV library.,” *Dr. Dobb’s Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[82] J. Gené-Mola *et al.*, “Assessing the performance of rgb-d sensors for 3d fruit crop canopy characterization under different operating and lighting conditions,” *Sensors*, vol. 20, no. 24, p. 7072, 2020.

[83] H. A. M. Williams *et al.*, “Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms,” *Biosyst Eng*, vol. 181, pp. 140–156, May 2019, doi: 10.1016/J.BIOSYSTEMSENG.2019.03.007.

[84] “MoveIt Motion Planning Framework.” https://moveit.ros.org/ (accessed Jan. 27, 2023).

[85] P. Lagacherie, G. Coulouma, P. Ariagno, P. Virat, H. Boizard, and G. Richard, “Spatial variability of soil compaction over a vineyard region in relation with soils and cultivation operations,” *Geoderma*, vol. 134, no. 1–2, pp. 207–216, Sep. 2006, doi: 10.1016/J.GEODERMA.2005.10.006.

[86] C. Damalas and S. Koutroubas, “Farmers’ Exposure to Pesticides: Toxicity Types and Ways of Prevention,” *Toxics*, vol. 4, p. 1, Jan. 2016, doi: 10.3390/toxics4010001.

[87] M. Komárek, E. Čadková, V. Chrastný, F. Bordas, and J. C. Bollinger, “Contamination of vineyard soils with fungicides: A review of environmental and toxicological aspects,” *Environ Int*, vol. 36, no. 1, pp. 138–151, Jan. 2010, doi: 10.1016/J.ENVINT.2009.10.005.

[88] A. Hildebrandt, M. Guillamón, S. Lacorte, R. Tauler, and D. Barceló, “Impact of pesticides used in agriculture and vineyards to surface and groundwater quality (North Spain),” *Water Res*, vol. 42, no. 13, pp. 3315–3326, Jul. 2008, doi: 10.1016/J.WATRES.2008.04.009.

[89] E. Besnard, C. Chenu, and M. Robert, “Influence of organic amendments on copper distribution among particle-size and density fractions in Champagne vineyard soils,” *Environmental Pollution*, vol. 112, no. 3, pp. 329–337, May 2001, doi: 10.1016/S0269-7491(00)00151-2.

[90] R. Reuveni, G. Dor, and M. Reuveni, “Local and systemic control of powdery mildew (Leveillula taurica) on pepper plants by foliar spray of mono-potassium phosphate,” *Crop Protection*, vol. 17, no. 9, pp. 703–709, Dec. 1998, doi: 10.1016/S0261-2194(98)00077-5.

[91] J. A. S. Bonds, “Ultra-low-volume space sprays in mosquito control: A critical review,” *Med Vet Entomol*, vol. 26, no. 2, pp. 121–130, Jun. 2012, doi: 10.1111/J.1365-2915.2011.00992.X.

[92] A. Chaskopoulou, M. Miaoulis, and J. Kashefi, “Ground ultra low volume (ULV) space spray applications for the control of wild sand fly populations (Psychodidae: Phlebotominae) in Europe,” *Acta Trop*, vol. 182, pp. 54–59, Jun. 2018, doi: 10.1016/J.ACTATROPICA.2018.02.003.

[93] S. A. Pfeiffer, J. Guevara, F. A. Cheein, and R. Sanz, “Mechatronic terrestrial LiDAR for canopy porosity and crown surface estimation,” *Comput Electron Agric*, vol. 146, pp. 104–113, Mar. 2018, doi: 10.1016/J.COMPAG.2018.01.022.

[94] A. Milella, R. Marani, A. Petitti, and G. Reina, “In-field high throughput grapevine phenotyping with a consumer-grade depth camera,” *Comput Electron Agric*, vol. 156, pp. 293–306, Jan. 2019, doi: 10.1016/J.COMPAG.2018.11.026.

[95] F. Marinello, A. Pezzuolo, F. Meggio, J. A. Martínez-Casasnovas, T. Yezekyan, and L. Sartori, “Application of the Kinect sensor for three dimensional characterization of vine canopy,” *Advances in Animal Biosciences*, vol. 8, no. 2, pp. 525–529, Jan. 2017, doi: 10.1017/S2040470017001042.

[96] S. Li, L. Dai, H. Wang, Y. Wang, Z. He, and S. Lin, “Estimating Leaf Area Density of Individual Trees Using the Point Cloud Segmentation of Terrestrial LiDAR Data and a Voxel-Based Model,” *Remote Sens (Basel)*, vol. 9, no. 11, 2017, doi: 10.3390/rs9111202.

[97] J. Seol, J. Kim, and H. il Son, “Field evaluations of a deep learning-based intelligent spraying robot with flow control for pear orchards,” *Precis Agric*, vol. 23, no. 2, pp. 712–732, 2022, doi: 10.1007/s11119-021-09856-1.

[98] C. S. Nandi, B. Tudu, and C. Koley, “Machine Vision Based Techniques for Automatic Mango Fruit Sorting and Grading Based on Maturity Level and Size,” in *Sensing Technology: Current Status and Future Trends II*, A. Mason, S. C. Mukhopadhyay, K. P. Jayasundera, and N. Bhattacharyya, Eds., Cham: Springer International Publishing, 2014, pp. 27–46. doi: 10.1007/978-3-319-02315-1_2.

[99] S. Cubero *et al.*, “Optimised computer vision system for automatic pre-grading of citrus fruit in the field using a mobile platform,” *Precis Agric*, vol. 15, no. 1, pp. 80–94, 2014, doi: 10.1007/s11119-013-9324-7.

[100] Z. Zhang, A. Pothula, and R. Lu, “Development and Preliminary Evaluation of a New Bin Filler for Apple Harvesting and In-Field Sorting Machine,” *Transactions of the ASABE (American Society of Agricultural and Biological Engineers)*, vol. 60, pp. 1839–1849, Jan. 2017, doi: 10.13031/trans.12488.

[101] A. Bonora, E. Trevisani, K. Bresilla, L. C. Grappadelli, G. Bortolotti, and L. Manfrini, “Convolutional Neural Networks for Detection of Storage Disorders on ‘Abbé Fétel’ pears,” in *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, 2020, pp. 1–5. doi: 10.1109/MetroAgriFor50201.2020.9277561.

[102] R. Kirk, G. Cielniak, and M. Mangan, “L*a*b*Fruits: A Rapid and Robust Outdoor Fruit Detection System Combining Bio-Inspired Features with One-Stage Deep Learning Networks,” *Sensors*, vol. 20, p. 275, Jan. 2020, doi: 10.3390/s20010275.

[103] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. Mccool, “DeepFruits: A Fruit Detection System Using Deep Neural Networks,” *Sensors*, vol. 16, p. 1222, Aug. 2016, doi: 10.3390/s16081222.

[104] A. Koirala, K. B. Walsh, Z. Wang, and C. McCarthy, “Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of ‘MangoYOLO,’” *Precis Agric*, vol. 20, no. 6, pp. 1107–1135, 2019, doi: 10.1007/s11119-019-09642-0.

[105] K. Bresilla, G. Perulli, A. Boini, B. Morandi, L. Grappadelli, and L. Manfrini, “Single-Shot Convolution Neural Networks for Real-Time Fruit Detection Within the Tree,” *Front Plant Sci*, vol. 10, May 2019, doi: 10.3389/fpls.2019.00611.

[106] J. Gené-Mola *et al.*, “Fruit detection and 3D location using instance segmentation neural networks and structure-from-motion photogrammetry,” *Comput Electron Agric*, vol. 169, p. 105165, Feb. 2020, doi: 10.1016/J.COMPAG.2019.105165.

[107] A. Gongal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, “Sensors and systems for fruit detection and localization: A review,” *Comput Electron Agric*, vol. 116, pp. 8–19, Aug. 2015, doi: 10.1016/J.COMPAG.2015.05.021.

[108] R. Kerry *et al.*, *Sensing Approaches for Precision Agriculture*. 2021. doi: 10.1007/978-3-030-78431-7.

[109] J. Gené-Mola *et al.*, “Assessing the Performance of RGB-D Sensors for 3D Fruit Crop Canopy Characterization under Different Operating and Lighting Conditions,” *Sensors*, vol. 20, no. 24, 2020, doi: 10.3390/s20247072.

List of figures

List of acronyms

PA – Precision Agriculture

PTO – Power Take Off

ROS – Robot Operating System

BMS – Battery Management System

PC – Personal Computer

HL – High-level

LL – Low-level

EKF – Extended Kalman Filter

GUI – Graphical User Interface

IMU – Inertial Measurement Unit

AHRS – Attitude and Heading Reference System

LiDAR – Light Detection And Ranging

INS – Inertial Navigation System

GNSS – Global Navigation Satellite System

GPS – Global Positioning System

DGPS – Differential GPS

RTK – Real Time Kinematics

RGBD – Red/Green/Blue/Depth

PTZ – Pan/Tilt/Zoom

I/O – Input/Output

UGV – Unmanned Ground Vehicle

AGV – Autonomous Ground Vehicle

PI – Proportional-Integral

PID – Proportional-Integral-Derivative

CoG – Center of Gravity

FoV – Field of View

PoV – Point of View

DoF – Degrees of Freedom

SSV – Skid Steering Vehicle

DWR – Differential-drive Wheeled Robot

ML – Machine Learning

AI – Artificial Intelligence

CNN – Convolutional Neural Network

HT – Hough Transform

RMSE – Root Mean Square Error

VRA – Variable Rate Application

IR – Infra Red

RPM – Revolutions Per Minute

PMAC – Permanent Magnet Alternate Current

CAN – Controller Area Network

SLAM – Simultaneous Localization And Mapping

NMEA – National Marine Electronics Association

RC – Radio Controlled

PDO – Process Data Object

PLC – Programmable Logic Controller

LED – Light Emitting Diode

UDP – User Datagram Protocol

BSD – Berkeley Software Distribution

OPA – Orchard Precision Agriculture

IoT – Internet of Things

PWM – Pulse Width Modulation

GPU – Graphics Processing Unit

MQTT – Message Queuing Telemetry Transport