SAPIENZA
UNIVERSITÀ DI ROMA

Autonomous and mobile manipulator robot for hydroponic greenhouse picking tasks

Ingegneria dell'informazione, informatica e statistica

Master's degree in Control Engineering

Candidate

Federico Rollo

ID number 1851121

Thesis Advisor

Prof. Giuseppe Oriolo

Co-Advisor

Dr. Nome Cognome

Academic Year 2018/2019

Thesis defended on 18 January 2020
in front of a Board of Examiners composed by:

Prof. Alessandro De Luca (chairman)
Prof. Roberto Belardi
Prof. Andrea Cristofaro
Prof. Paolo Di Giambernardino
Prof. Fabio Giulii Capponi
Prof. Giuseppe Oriolo
Prof. Leonardo Lanari

Autonomous and mobile manipulator robot for hydroponic greenhouse picking tasks
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: December 23, 2020

Author's email: rollo.f96@gmail.com

Dedicated to
my family
and all the people
who believe in me

Contents

List of Figures

List of Tables

Sommario

Un piccolo sommario in lingua italiana è stato introdotto nella tesi per facilitarne la comprensione a chi ha poca domestichezza con l'inglese.
La seguente tesi è basata sullo sviluppo di una piattaforma mobile con sopra annesso un braccio manipolatore che possa lavorare all'interno di un ambiente agricolo quale una serra. Lo studio di fattibilità svolto prevede una piccola parte progettuale e una parte simulativa finale per validare la fattibilità e la buona riuscita degli algoritmi, delle logiche e del progetto in generale. La tesi comincia con la presentazione della logica dell'algoritmo utilizzata, vedi capitolo 2, per passare poi ad analizzare i vari aspetti decisionali e tecnici delle strutture da utilizzare per la costruzione del robot, come la piattaforma mobile, vedi capitolo 3, e il braccio manipolatore, vedi capitolo 4. In questi capitoli diversi aspetti vengono tenuti in considerazione, partendo dalla cinematica delle varie componenti, arrivando fino allo studio dell'organo terminale e il sistema di visione necessario per il compimento dei compiti assegnati. La tesi poi continua con l'analisi delle diverse piattaforme esistenti sul mercato e la presentazione di alcuni nuovi prototipi da poter sviluppare per queste applicazioni, vedi capitolo 5 per poi finire con il capitolo 6 dove viene presentata la simulazione svolta con le piattaforme Matlab Simulink e Vrep. Il capitolo finale, ovvero il capitolo 7 conclude il progetto traendo le conclusioni del progetto e mettendo alla luce alcune implementazioni future che potrebbero migliorare il progetto stesso. Alla fine della tesi è stata aggiunta anche una appendice composta da tre capitoli dove sono riportati alcuni concetti o algoritmi utilizzati durante il lavoro. in particolar modo, nell'appendice A sono presentati gli angoli di Eulero, nell'appendice B viene presentato l'algoritmo K-mean e una sua applicazione per il riconoscimento dei fiori e nell'appendice C viene presentato l'algoritmo di classificazione e separazione dei vari fiori all'interno di un immagine, per poterli identificare e localizzare nello spazio.

Chapter 1

Introduction

The agriculture and, in general, food production are one of the most important and ancient jobs for human beings. Every day,due to the population growth, the necessity of producing food in large quantity and in a small time is increasing constantly. This high rate of production results in more costs for the business owners, more labor for the farmers and consequently more costs for the buyers. These costs cannot be faced by most of the people because the population rising decrease the common purchasing power. A good alternative to face this problem is the use of machines to perform human hard labours. This kind of automation decreases that production costs which would burden on the consumers preserving farmers health and resulting in a more sustainable and less costly product.
In general, it is proved that a small number of autonomous robotic workers can cover the labour of a bigger number of workers, more or less three or four farmers for picking tasks,[76]. This is not only due to the task accomplishment speed but even thanks to the robot ability to work constantly, needing only short rest times (charging, cooling, repair, ecc.) with respect to humans. Moreover, thanks to the rise of green energies these robots would be able to perform most of the task, if not all, consuming clean energy and without polluting the environment. Indeed, the most innovative producers, especially the hydroponic ones, are always associated with some clean energy companies which provides energy and invest in these small realities to make them bigger. In this way, all the stakeholders of the agricultural area, probably all the human beings, should be able to produce and consume good and fresh products at affordable prices.
There exist companies which already use autonomous robots for the entire growth of food. One of the most famous is the IronOx [24] startup which has built a fully autonomous hydroponic greenhouse where the human labour is confined in monitoring robot efficiency, maintenance and economical tasks. The automation of all the human heavy works is one of the control engineers' main objectives and

by using this approach the workers could be able to perform softer labors such as maintenance, monitoring and others. These reasons explain us why there exists so much studies and researches over this field. There are a huge number of papers, studies, applications, hypothesis, books and projects included this work, which cover lots of this task, from seeding to picking, from monitoring to weeding, from transportation to packing. A good treatment of the all this different tasks and their automation can be found in [18][27][26][62]. All these studies aim to simplify and improve the presence of robot in farming tasks presenting different solutions and giving other references to explore some specialized solutions.
Some of the most complete works, which can be compared with thesis, are presented in section 5.1 while some picking robots projects can be found in [13][9][76][19][64]. These papers aim to create an autonomous and mobile robot able to pick different fruits and store them in a collecting box. Other studies have been performed in order to improve the overall task by designing the end-effectors [78], the navigation techniques of the robot through greenhouses [31], some particular wheels suited for flat floors [42], the mobile platforms to use in closed environment [71][73] and some multipurpose robotic systems [69]. Lots of other works are not here reported due to the huge quantity of them but they deserve to be cited also if in a generic way. These studies can be found directly on web platforms such as IEEE [34], ResearchGate [59] and others academic websites.
The most challenging fields in autonomous picking tasks is fruit recognition, localization and pose estimation. For these task lots of different solution have been presented and most of them use machine learning techniques. Different useful fruit recognition reviews have been presented during the years starting from [16] where some visual servoing techniques are presented, [21] which relate some of the most used vision system with agricultural tasks and [72][77] that address different 3D localization and recognition methodologies for picking robots. Other good references are [70][32] where different vision and perception techniques and their application are presented and [10] which reports the most used machine learning methods. Also in this field there exists different separated works which study single portions of vision systems such as shape and object recognition [66][58][40], colours classification [45], pose estimation of an object [41]. One essential role is covered by the Cameras, especially for the 3D reconstruction of the environment. The cameras considered in this work are only RGB-D cameras, see section 4.3.1, but also different kinds of cameras can be suited in these tasks, e.g. stereo camera. The RGB-D camera results very useful because they allow a fast 3D object recognition and reconstruction [22][79][6], the pose estimation and localization [80][29] and the measuring and classification of different kind of fruits [57]. These cameras can be merged with some neural network

techniques, generally convolutional neural network for image recognition, in order to provide a fast and efficient solution for picking robots [30][44]. Moreover, the presence of this camera on the robot could improve also the motions through the greenhouses because different camera based guidance techniques can be applied [56] improving both robot localization and obstacle avoidance. Some application examples of these methods can be found in [8][52][53].
Another important part for a good harvesting is the monitoring of the plants' growth and quality. These observations can be done in different ways but the most common is using multiple robots which are connected between them with a communication system. Some works on this topic are [62] where different monitoring techniques are presented, [7] which use swarm robotics for monitoring tasks, [20] which combines collaborative robotics with control theory to detect plant disease, [36] which use a UAV monitoring for detecting plants water stress in order to control the irrigation, [65] where is presented a robot able to supervise the plants growth.
Other minor, but not less important, works for these robots and for automation could be the autonomous scattering of pesticides or fertilizers when the robot recognise that they are needed [37][75][67], the autonomous right irrigation of the plants thanks to underground sensors [17] or other techniques [63]. Also seeding tasks, which could be a heavy work, can be automated using autonomous single seeding robots [49][35] or a bunch of them [11] to accomplish the task. furthermore, some particular task for seeding, such as seed selectors [33] can be implemented on each robot for the right and best task performance. In [55] is proved that the use of seeding and planting robot introduce also economic benefits as well as a human hard work decrease.
Regarding the technological methods and items used in this work different project and papers are cited during the thesis; while for the robotic background a very good reference is [68] where many robotic topics are covered.

1.1 Work presentation

The structure of this work has been divided to separate the different components which compose the whole picking robot. In chapter 2 the algorithm for the picking task is presented with a full explanation of the process procedure. Moreover, there is an environment description introduced both for completeness and to clarify some of the choices done during the work. Chapter 3 present different kinds of mobile robots with their kinematic models and the motion planning techniques used to move the robot inside the greenhouse. In chapter 4 the manipulator arm is presented. This chapter is the largest because the fruit picking is a delicate task and comprehends the end-effector design, see section 4.2, the vision system, see section 4.3, and the

motion planning of the arm, see section 4.4. In chapter 5 different existing solution for picking robot and some new prototypes for this robot are presented. Chapter 6 is used to present the simulation used to evaluate the work feasibility of this thesis. Finally, in chapter 7 the project conclusions are presented along with some of the most important future improvements. At the end of this work have been added some appendix chapters which are useful for the deep comprehension of the thesis. In particular, in appendix A there is an introduction to Euler angles, in appendix B a k-mean algorithm for color clustering has been presented and in appendix C there is a cluster separation algorithm for the recognition of the single flowers and stems.

Chapter 2

Algorithm hierarchy and task specification

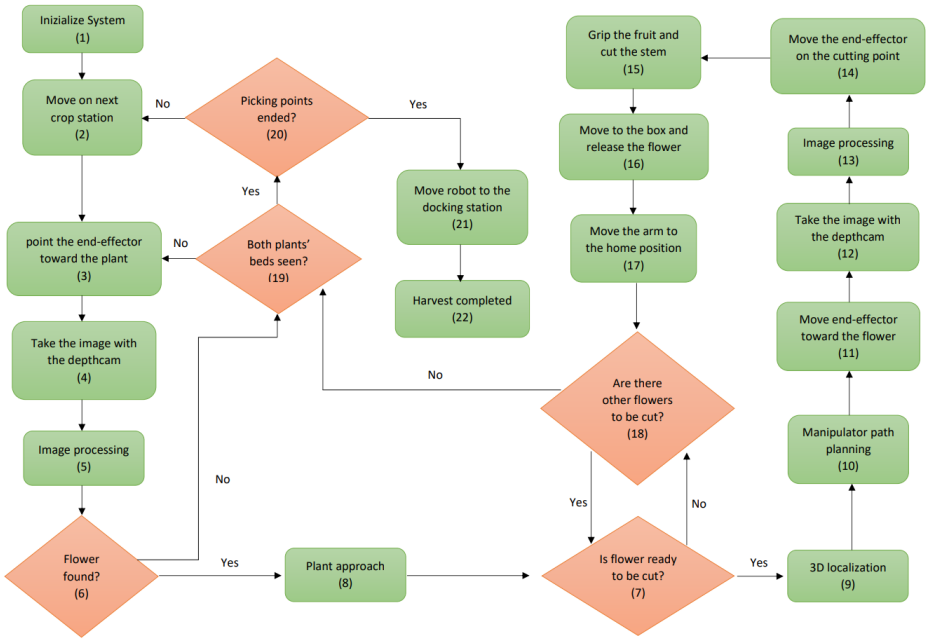


Figure 2.1. Hierarchical structure of the harvesting task

The main task of this robot is to pick some objects, in this case zucchini flowers or fruits, and then place them into the box over the carrier. But, to accomplish this task, different intermediate tasks are needed because the robot works in a large environment with the presence of multiple obstacles.

First of all, there is the charging station or docking station that is the point on the map in which the robot stations the most of the time due to its recharging and the

disuse, e.g. in Fig. 3.7 the robot is placed in its docking station. This charging station is the starting and ending point of the robot for the harvesting task. Indeed, when the robot is initialized, it is in the docking station and when the harvest is complete it returns there where it remains at rest after that the crop will be taken and brought to the packaging.
To simplify the comprehension of the robot picking task, a hierarchical flow chart has been reported in Fig. 2.1. In this way, the whole logical procedure followed by the robot is compressed in a single figure.

2.1 Process procedure

In block (1) the system is initialized, that is the robot is turned on and the whole system is set to start. block (2) concerns the movement of the robot in the greenhouse; despite it is a single block, can be seen in section 3.3 that it is not a trivial task. On the greenhouse map stored in the robot memory there are the plants points in which the robot must stop to take some pictures of the plant with the camera. Indeed, in block (3) there is the pointing of the camera toward the plant. The camera is positioned over the wrist of the robot, see section 4.3, in this way it is able to sense the zucchini flowers on the plant from different positions. The idea is to move the manipulator arm in different positions to have different points of view of the plants to maximize the possibility of finding a ready flower. In this work the different perspective points are not considered because the simplicity has been preferred despite of the completeness. For more information above the vision system have a look at section 4.3. Once that the camera mounted on the arm is pointing the selected plant, the depth cam takes an image of it, block (4), and then the image processing starts and it searches for some zucchini flower on the plant using the machine learning technique reported in appendix B, block(5). In block (6) there is an if-then logic: if at least one flower has been found, then continue with the harvesting task otherwise search on other plants, block (19). Continuing with the crop tasks, after that the robots has approached the plant, block (8), see section 3.4, in block (7) it takes into account one of the sensed flowers and decides if it is mature i.e. if it is ready to be picked; if it is, then the process continue, otherwise the robot considers if some other sensed flowers are ready or not, block (18). In some cases, the approaching movement can be neglected depending on the chosen trajectory for the robot or, most important, depending on the distance between the robot and the plant. In fact, if the manipulator arm is able to pick the flower directly remaining over its main trajectory, it could pick the flower directly from it. Anyway, this is not recommended because a trajectory of this kind could damage

both the robot and the environment due to its closeness to the plants. In block (9), the vision camera is pointed towards the plant and is ready to sense the 3D structure of the plant to find the position and orientation of the flower with respect to the end-effector, see section 4.3.4. With this information, in block (10), the path for the manipulator is planned to avoid the contact of the arm with the plant and to position the end-effector towards the flower, near the picking position, block (11). Once the end-effector is close to the flower, the camera takes other images with the depth cam, block (12), the robot processes the images, block (13), and with the new flower pose information the end-effector is moved exactly in the cutting point with the desired orientation for the picking task, block (14). Finally, to complete the picking task for the considered flower, the gripper grabs the stem of the flower and the scissors cuts it from the plant, block (15). Once the flower has been separated from the plant, the manipulator arm moves to the collecting box, releases the flower, block (16), and, finally, it returns to its home position, block (17). Now, if there are other flowers on the plant this picking process restarts with the new flower otherwise the robot looks for another plant to be checked on its row, block (19). If there are plants on the current row which has not been considered, then the robot restarts the procedure from block (3) with a new plant. Otherwise, it passes to block (20) where the robot checks if the picking point list has been completed or if there are other points on its trajectory. If there are other picking points, the robot returns on its trajectory and moves on the next crop station, block (2), to restart the harvesting process from block (3). Otherwise, if there are no more plant to be checked, the robot returns to its docking station following the prefixed trajectory, block (21), and completes the whole process, block (22).
In the charging station there will be a people or and automated machine which replaces the flower-full boxes with some empty ones and brings the picked flower to the packaging machine where they will be divided and properly packed.

2.2 process specifications

Once that the algorithm hierarchy has been decided there are also other characteristics that must be decided depending on different factors such as the plants nature and others.
The zucchini flowers are characterized by a fast maturity process, indeed, when they are ready, it remains open for at most 36 hours. The first impulsive choice could be picking the flower whenever it is ready, but this could weigh on the cost of production because the robot should do different turns during the day but, most important, in this way the flower fertilization process is completely nullified. In fact,

the zucchini flowers of interest are the male fertilization part of this process and, by cutting them when they are just blossomed, the female part, i.e. the zucchini fruit, cannot be fertilized provoking the fruit decomposition and slowing down the plant grown resulting in a smaller crop quantity during the season.
Considering that in general most of the flowers blossom in the morning or at most in the early evening, if the flowers are picked during the late afternoon or, even better, during the night, this problem is quite annulled. Moreover, the flowers during the night are fresher because they recover the dry out caused by the high temperature of the day.
The only problem is that the recognition of the flower is performed through an RGB camera sensor which needs a minimal brightness for the identification and localization of the flowers. To overcome this problem two possible choices can be considered for the picking scheduling:

1. take a monitoring turn during the day and pick the flowers in the late afternoon before the sunset

2. attach a flash to the camera

The first option is disadvantageous under many points. The electricity cost of the monitoring turn is an addition to the main picking turn and, depending on the size of the greenhouse, could also create problems for the charging scheduling of the robot which after the monitoring should be able to recharge the battery power to perform the picking turn in the evening. The flowers do not have time to recover the dry out and this will affect the product freshness. Moreover, in some cases could happen that the leaves of the plant shade the flowers and in low light condition the recognition task becomes quite impossible needing the flash in any case.
Instead, with the second option, i.e. the addition of a flash, the camera works always in an almost ideal brightness condition, being able to recognize the flowers also in low or absent light conditions. It is true that the power consumption of the flash is not negligible but, considering that in this case the monitoring turn is no more needed, the costs are almost equal. Moreover, as already explained, if the flower is cut during the night, it is fresher resulting in a high quality product which can be packed before the sunrise and ready for the early morning shipping.
Those little attentions could seem worthless under an implementation point of view, but its consideration is necessary to enter into the market and be competitive. Furthermore, some statistics, for example the ones in [76], highlight the savings gained due to the use of machines instead of humans under an economic point of view. Other improvements could be implemented on this project to optimize and maximize the production, the quality and the earnings, but there are too many

details to be considered and this could lead to a complete but confused work and the aim of this project is the modeling and simulation of an autonomous picking mobile robot. Some of these aspects are slightly introduced in section 7.1.

2.3 Environment description

The workspace of interest is characterized by an hydroponic greenhouse structure. The main peculiarity of these kinds of greenhouses is that the plants do not grow inside a common ground bed, but they are collocated inside a little tub with coir or directly with water. The solution chosen in this work is the first one, therefore the plants are put inside some pots with coir. using one of this two methods, the greenhouse not only can be rearranged in a very simple fashion, but it is also more resistant against diseases. There exist lots of advantages in using a hydroponic greenhouse with respect to a normal one. First of all, the water use decreases a lot because it is used only the one that the plant needs and it won't be drained by the terrain or evaporated by the heat of the greenhouse. Another advantage is the use of a smaller space where more plants can fit in. It is not uncommon to see a vertical plantation where there are two, three or more shelves used to place different plants. This is possible because a hydroponic greenhouse is composed by an artificial illumination which makes possible the plants photosynthesis, an automatic air conditioning which control temperature and humidity, an automatic sprinkler system for the irrigation and a controller which regulates the quantity of nutrients inside the water. These characteristics creates a suitable ambient for the grown of different plants and allows to obtain more food with less resources.
Anyway, this system has also some contraries; all these automatic controllers, the artificial illumination and the robot that is the subject of this thesis are current consuming objects and these costs are not negligible because they increase the production costs and, hence, the costs of the product. However, in the modern era the production of clean energy and the possibility of building greenhouses close to photovoltaic plants allows a smaller cost for the current production and transportation and consequently for the current cost, not to mention the advantages for the pollution.
Anyway, despite all these differences with an old-style greenhouse, the structure is quite similar. In general, there are two kind of hydroponic greenhouses, the ones completely automated, i.e. enclosed in a cemented environment such as warehouse and the ones which are an adaptation of an old greenhouse terrain to an hydroponic space. The second one is the choice selected because it is the most common due to the fact that if there is not a large amount of money to invest, then it is more suitable

to rearrange, in terms of time and costs, a predisposed terrain to a hydroponic fashion.
Indeed, that structure is the simplest one and can be seen in Fig. 2.2 where the light blue beds represents the tubs where the plants will be collocated and the brown is the normal soil. the differences with the normal greenhouses are that the plants are no more planted in the soil but in tubs with water or coir. D_x and D_y represents respectively the distance between the bed along x and y axis and it has to be decided depending on different choices.

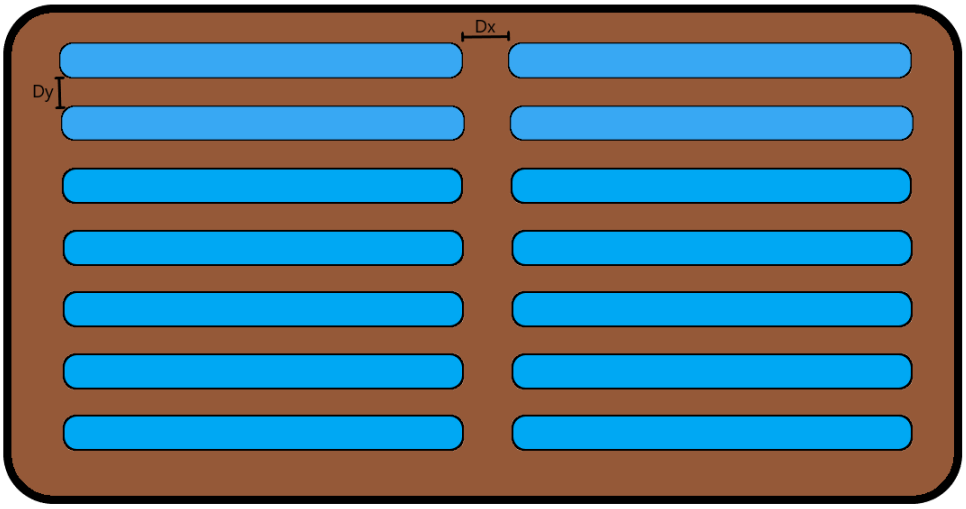


Figure 2.2. Structure of a generic hydroponic greenhouse. The organization is similar to the old-fashion greenhouses.

Anyway, once that the tubs are positioned the hydroponic greenhouse can be considered as a normal one. Obviously, before the positioning of the tubs, the soil should be flattened and leveled.
Another important characteristic, especially for this work, is that some rigid platforms will be positioned on the soil to uniform the terrain consistency and also to provide a catwalk to the people and to the robot, see section 3.1.
The last difference to consider is the altitude of the plants with respect to the floor. This distance depends on the positioning of the tubs on the soil and could affect the robot choice. If the tubs are only placed upon the terrain, the plant height would probably be different with respect to dig some holes and position the tubs at ground levels. This difference in altitude could seem an unimportant characteristic but, instead, is one of the most important because the picking area, i.e. where the flowers are present, should be inside the workspace of the robot or even better inside its dexterous space. If the robot can reach at most 1 m in altitude, the tubs are 1 m

tall and the zucchini plants are grown in height (some zucchini plants can reach and exceed 1 m of height/length), then the robot could be not able to perform its tasks and another robot structure should be chosen. Instead, if the tubs are at ground level the plants can be grown in height or can be left on the soil because the robot will be able to reach them without any problem. In this work, a middle way has been considered: the tubs are tall almost 30 cm and the plants are grown in height. in this way the zucchini plants can reach at most 1.2 m of height and probably this is not the case of the zucchini plant species chosen for a greenhouse environment. Growing the plant in height is a common technique used for different vegetables such as tomatoes, cucumber and others, especially in greenhouses. In general, some reeds are stuck in the soil and the plants is gradually attached to the rod during their growth. Another elegant solution, broadly used in greenhouses, is to tie some little strings of rope to the roof and wrap it around the plants. This second solution brings lots of advantages:

- more plants can be planted in a smaller place
- the robot view is clearer resulting in a more accurate sensing
- the plants are less exposed to diseases or rottenness because they are better ventilated
- the plants are more impact-resistant because if the robot collides with them, the plants will cushion the impact thanks to the wire softness.
- differently from the reed's solution, the plants is not suffocated by the wires which connect the plant to the rods.

In Fig.2.3 a simple representation of the plantings structure is presented. The plants are positioned inside some tubs or vessels containing some coir terrain and are positioned in row. The plants are grown in height and the maximal height of the plant is represented by h which could range from $30cm$ to $100cm$. The hydroponic vessels are represented in light blue to distinguish them from some normal plant beds or vessels.

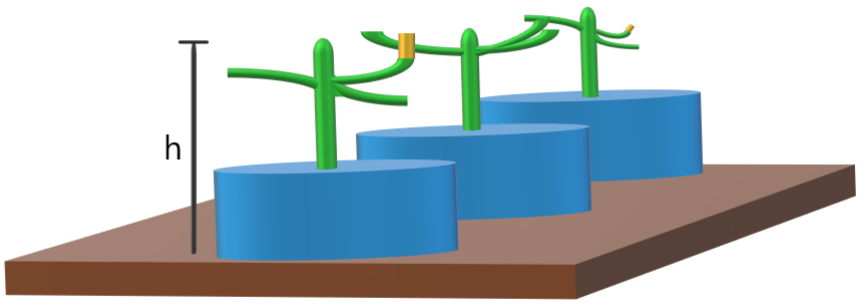


Figure 2.3. Plantation example for an hydroponic greenhouses where each plant has its own hydroponic vessel.

Chapter 3

Robot mobile platform

The choice of a good mobile platform for harvesting tasks is not straightforward. Different carriers have been invented to perform many tasks, but the construction requirement and the implementation choices always changes on the base of the information of the environment, the fruit to crop, the typology of the field, the space between the plants and others. Generally, these platforms must have the space for handling not only the harvesting manipulator arm but also to storage the cropped fruits, the battery which supply the electrical machines, the cameras used to detect a 3D approximation of the environment and the controllers needed to accomplish the task and synchronize the machines. In our case, the greenhouse environment is characterized by a fixed structure where it is difficult to encounter obstacles and where the field is almost smooth and flat. Thanks to the fixed structure of this environment the carrier wheels can be structurally simpler with respect to the ones used for open fields autonomous systems where the rugged terrain requires wheels with a good grip and a complex suspension system behind. These advantages allow to decrease the carrier costs and, therefore, use this savings for the manipulator, the end-effector or a specialized vision system for crop tasks.
As has been presentend in the introduction, Chapter 1, the autonomous robot needs to move in a structured greenhouse considering that the map of the environment is completely known. Different projects regarding greenhouses crop tasks uses carriers on rails to simplify the controller behind the movements, to limit the robot freedom and to decrease the possible damage to the plants or the environment. Anyway, this limitation could be costly for big areas and the low modifiability of this rails fixed structure could reflect into a problem when the greenhouses internal structure must be changed for any reason.
On the other hand, a wheeled robot is more suited for this task because the terrain smoothness and the knowledge of the environment map and the plants position allow to find simple paths which drive the carrier though the plant beds without

damaging anything. Another advantage for these free moving robots is the possibility of detecting a possible obstacle, e.g. a dropped plants onto the path, and overcome the problem by circling around them, founding another path or, whether there are no other solutions, reporting the problem to the central unit.
Now that the main problems encountered for the choice of the carrier have been expressed, the decision for the carrier and the motion algorithm can be presented.

3.1 Carrier choice

The first choice to do is what kind of carrier are preferred for this task. The answer is not simple, and it will affect the choice of the robot model. There are different kind of vehicles used for these tasks. dismissing the railed robots used in most of the picking task, the most famous models are the differential-drive robot and the car-like robot. The first one is characterized by two actuated wheels and generally one free carrier wheel introduced to improve the stability of the robot. This structure can turn around its center and it is able to perform all movements needed for our task but this kind of model has a big lack in stability. It is true that no huge stability issues are needed for our task but there could be always some drawbacks during the movement and the carrier should be able to overcome them as best as possible. An interesting differential-drive configuration has been presented in [31] where the wheels has been replaced by two tank tracks which increase the stability of the carrier on rough terrains. However, this is not suited for our case.
The second structure, the car-like robot, is characterized by four wheels which can have different actuation disposition. The possibility of the front wheel to steer allows to perform narrow movement around the plants' rows without needing too much space. Moreover, the disposition of the wheels enable the carrier to have a good stability.
There exists also another configuration very similar to the car-like robot that is characterized by for the presence of four Mecanum wheels [42] instead of the normal one and the non-stearing front wheels. These wheels enable the robot to move along its transversal direction which is not possible for the car-like robot which has a non-holonomic constraint exactly in that direction. There exist different greenhouses which already use this kind of wheels for their robots. one of the most famous is the Iron Ox start-up [24] which has a fully automatized hydroponic greenhouse where some mobile platforms are equipped with these wheels. The hydroponic greenhouses do not need the presence of soil and they are generally characterized by a well-structured floor that allows different free movement on a smooth surface. Indeed, depending on the floor characteristics the farmers could prefer to use a four

mecanum wheel robot for its omni-directionality or a simple car-like robot for its good stability and traction on rough terrains. A small comparison between these three mobile platform systems will be afforded in the next section 3.1.1

3.1.1 Car-like robot, Mecanum wheels robot or differential drive

In this subsection the models of three mobile platforms will be presented with a slight treatment of the pros and contraries of each system. let's start with the car-like robot.

Car-like model

The first structure considered is the car-like platform. This robot is composed by four wheels positioned in a rectangular shape, see Fig 3.1.

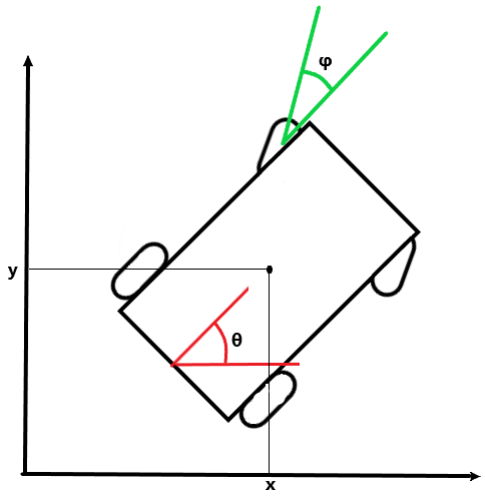


Figure 3.1. Car-like robot platform top view

The four wheels can be arranged with different actuation configurations: one motor for each wheel, one motor for the two rear wheels and one for the front wheels and finally use one motor to drive either the two rear wheels or the front wheels. In general, the last configuration is the preferred one because by using a single motor, the occupied space and the overall weight are reduced allowing to build a more compact and agile robot even if these choices depend on different factors. Despite of this, the strength of this robot is the possibility to steer due to the front steering wheels. Such as a real car, the robot can orientate the front wheels in the desired direction to perform a clean angular movement reducing the drifting behaviors which are one of the most common disturbances affecting these systems because the introduced error, once integrated in time, can grow till the infinity resulting in a wrong localization. To reduce these behaviors, different mechanical

and implementation improvements can be introduced in this structure. For the mechanical part, a particular machinery, called Ackermann steering mechanism, is introduced on the axle which connect the front wheels, the steering one. This machinery autonomously controls the direction of the wheels depending on the curve that the machine has to afford. If the steering wheels are oriented with the same angle, some drifting behavior naturally arise on the system and they will cause a wrong curve following. Instead, if the wheels can be steered with a different angle then the car is able to perfectly follow some generally circular path reducing the drifting error until almost neglecting them. One of the strong points of this mechanism is its completely autonomy, in fact, once it is positioned into our car-like robot it works independently to the task movements. This peculiarity allows to maintain the original model of the car-like robot without considering the presence of this machinery and more important the front wheels can be considered parallel into our model and this is a very strong assumption which simplify a lot the work. The second way to reduce these errors is implemented directly into the controller and need the help of some sensors to improve the localization system. This is necessary because the drifting behavior could arise not only because of the structure of the carrier but they can be caused also by other problems. If the wheels are not perfectly in contact with the floor, then their traction is a little bit noisy which means that the wheels could spin freely in some instants and this worsen the error integration in time. For these reasons, a good localization system is always needed for these tasks but this argument has been treated in section 3.2.
The kinematic model of the car-like robot can be approximated with a simple bicycle model which is characterized by four states and two input. The states are the x and y coordinates of the Cartesian plane, the θ angle, known as heading angle, which represents the angle between the bicycle and the x Cartesian axis and finally ϕ, known as steering angle, which is the angle between the sagittal plane of the front wheel with its rest position i.e. when the front wheel is aligned with the bicycle frame. The two input are the linear velocity v and the angular velocity ω of the whole structure (see the comparison between Fig. 3.1 and Fig. 3.2). This configuration can be used for our car-like robot because the front wheels are parallel and then they can be considered as a single coupled wheel positioned at the center of its axle and the same is valid for the rear wheels which are equally parallel. these strong assumptions allow to use the bicycle model as the approximation of the car-like robot providing a simpler control and application of the different algorithms.

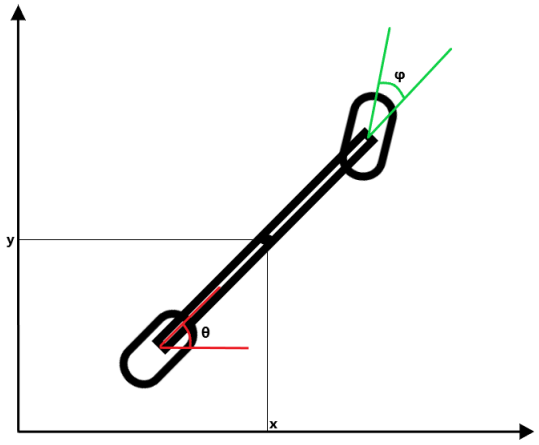
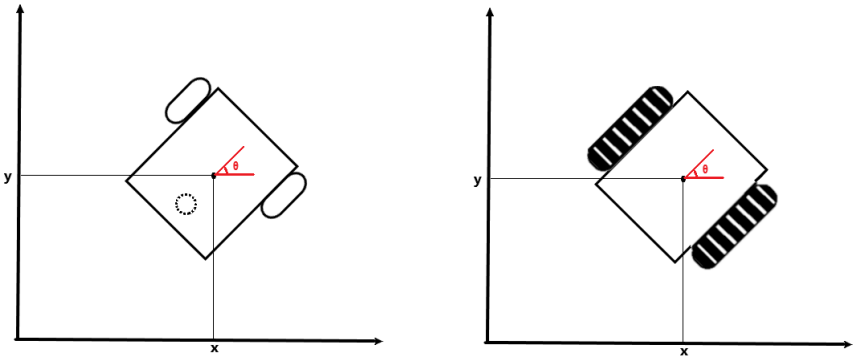


Figure 3.2. bicycle robot top view

for sake of simplicity, only the kinematic model of the rear wheel drive robot is presented in 3.1 but the front wheel drive can be obtained by multiplying a $cos(\phi)$ to the linear speed v:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ \tan\frac{\phi}{l} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.1}$$

Differential Drive robot



(a) differential drive robot with the support omni-directional wheel (dotted circle)

(b) differential drive robot with crawlers

Figure 3.3. Differential drive robots top views

As introduced, the differential drive robot can be composed by two actuated wheels and an omni-directional wheel necessary for the stability of the robot (Fig. 3.3a) or it can have two actuated crawlers (Fig. 3.3b). These configurations are characterized by a good mobility because the robot can turn around its center and, therefore, it can perform narrow movements without the necessity of performing maneuvers. The stability of the second differential drive configuration, i.e. the one with the crawlers, improves the performances of the robot because the increasing in stability allows different movements on rough terrain and reduces consistently the possibility of losing grip with the soil. Due to its simplicity in design, implementation and control, this configuration is widely used in many tasks that require motion. There exists also some experimental projects for agricultural tasks which use this configuration, in particular the one presented in Fig. 3.3b.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.2}$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2}(\omega_r + \omega_l) \\ \frac{r}{d}(\omega_r - \omega_l) \end{bmatrix} \tag{3.3}$$

Where ω_r, ω_l represent respectively the right and left angular speed of the wheels, r is the radius of the wheels and d the distance between them.
The kinematic model of the differential drive robot can be considered as a special configuration of the Unicycle one, indeed the kinematic model can be derived directly from it. By analyzing the Kinematics of the unicycle in 3.2 can be stated that thanks to the input transformation reported in 3.3 the differential drive kinematic model can be derived.

Four Mecanum wheels robot

let's firstly introduce the particular structure of the Mecanum wheels. The Mecanum wheel also known as Ilon wheel was invented in 1972 by the Swedish engineer Bengt Erland Ilon.

Figure 3.4. An example of Mecanum wheel for heavy tasks made by omni-robot [61].

They are essentially tireless wheels, composed by a sequence of rollers connected to the rim of the wheel in such a way that each roller have an angle of 45° with respect to the wheel sagital plane and also with respect to the axle line, see Fig. 3.4. Each Mecanum wheel is independent and its rotation generates a propelling force perpendicular to the roller axle which can decomposed in longitudinal e transversal component with respect to the vehicle [82].

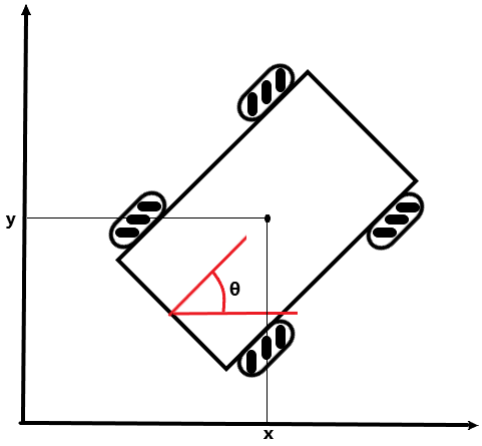


Figure 3.5. Mecanum wheeled robot top view.

This kind of robot is always a four-wheel robot but the particular choice of the wheels allows to have some simplifications on the structure. First of all, the front wheels do not steer anymore, the steering angle and angular velocity can be achieved by injecting a different rotation velocity at each wheel, see Fig. 3.6.

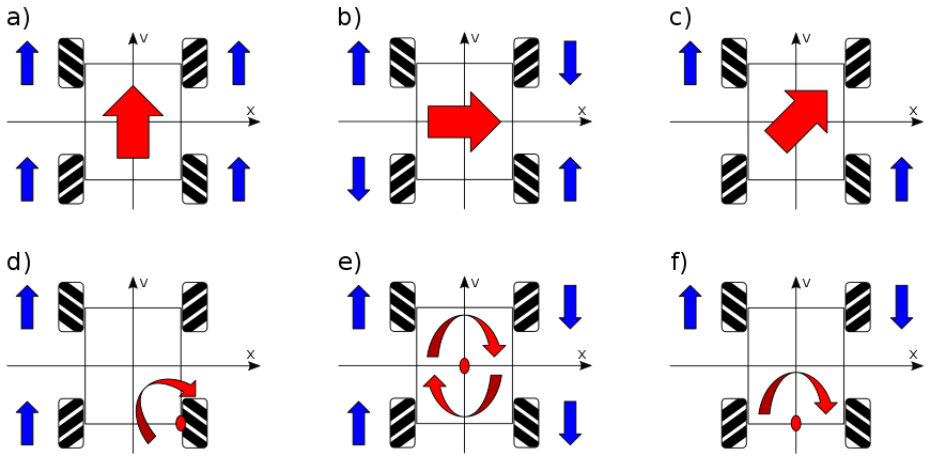


Figure 3.6. Different direction of the car can be obtained turning the Mecanum wheels in different directions. [82]

Obviously here the Ackermann steering machinery is no more needed because the front wheels orientation is fixed with respect to the robot frame. The actuation configuration is unique: each wheel needs exactly one motor because the four wheels must be controlled separately in order to perform the required movements. This seems to be a disadvantage in space, weight and battery terms but the possibility of having an omni-directional robot with good stability performances is a very strong pro. This robot does not have any non-holonomic or holonomic constraint and could move instantaneously in each direction. This characteristic is really important when one have to perform tasks in small environments or when the robot should pass through narrow passages because allows to cover each velocity direction without precluding a non-conventional shifting.
Anyway, the biggest contrary is the floor smoothness because the ground should be well structured otherwise on a rough terrain the wheels would have some drifting and traction problems.
The kinematic model of this platform will be now introduced. In this work, for kinematic models of a mobile robot is intended the connection between the velocity of the wheels and the linear and angular speed of the whole machine and not the connection between joint space and Cartesian space positions as it is used for the manipulator robots. This robot does not have front steering wheels and consequently the steering angle state is no more needed. The states of this system are only three: the x and y coordinates and the θ headling angle. Instead, regarding the inputs, now we have four wheels actuated separately by four motor and each angular velocity of each motor is now considered as one single input i.e. four input are present, one for each wheel angular velocity. The forward kinematic model [71] is:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \frac{r}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{(d_x+d_y)} & \frac{1}{(d_x+d_y)} & -\frac{1}{(d_x+d_y)} & \frac{1}{(d_x+d_y)} \end{bmatrix} \begin{bmatrix} \omega_{FL} \\ \omega_{FR} \\ \omega_{RL} \\ \omega_{RR} \end{bmatrix} \tag{3.4}$$

where FL, FR, RL and RR stands for Front-Left, Front-Right, Rear-left and Rear-Right to indicate to what wheel they are associated, r is the wheels radius while d_x and d_y are respectively the distance between the wheels on the x axis and on the y one.
Instead, the inverse kinematic model [71] [48] is:

$$\begin{bmatrix} \omega_{FL} \\ \omega_{FR} \\ \omega_{RL} \\ \omega_{RR} \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & -1 & -(d_x + d_y) \\ 1 & 1 & (d_x + d_y) \\ 1 & 1 & -(d_x + d_y) \\ 1 & -1 & (d_x + d_y) \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{3.5}$$

3.2 Localization system

The robot localization inside an environment is one of the most important tasks. If the autonomous mobile robot does not know its position or pose, then it is unable to accomplish most of the possible tasks. In our case, the map of the environment is already known from the robot because the general structure of the greenhouse is fixed and does not change until the next transplantation or some areas reallocation to introduce different plants. In this way the map can be considered entirely fixed and known, reducing in this way the possibility to encounter unknown obstacles or blocked passages.
The robot localization can be performed in different ways, the most common are:

- the odometric localization
- the Kalman Filter localization

The first method is characterized by using proprioceptive sensors to obtain, at each step, an integration of the kinematic model of the system that is used to compute the next probable robot pose. This method is widely used and it is integrated also in other localization techniques because it is the simplest and allows to obtain a good approximation of the robot position at the next integration step. The problems arise when the model of the system is highly uncertain or it is affected by some disturbances or noises which are not modeled in it. These errors could probably increase over time due to the open loop integration performed at each step, and they

could probably follow a divergent and unstable behavior for the localization system and, therefore, in the whole system.
The second method uses also some exteroceptive sensors and the Kalman Filter to recover the errors of the odometric localization. There exists different localization system based on Kalman Filter techniques, e.g. the landmark-based method, but one of the most know and used is the SLAM algorithm or Simultaneous Localization and Map-building method. With this technique the robot can build iteratively a map of the environment and automatically detects its position into that map. Also in this case, there exists different configurations and implementations of the SLAM algorithm depending on the task, the environment and the robot considered.
In this work the map is already well known and the localization system is assumed to be already present on the mobile platform. Anyway, it is advisable to implement a Kalman Filter localization system to recover some integration errors and for an online obstacle avoidance i.e. to detect and avoid obstacles which are not considered into the map. In fact, the most famous robots available on the market are all implemented with a SLAM method or some Kalman Filter based techniques for the obstacle avoidance and the right robot localization into the environment. Other solution there could be considered, e.g. the Particle Filter which, thanks to the sensing of the environment, is able to improve the localization of the robot recognizing the position of it in the map, using an iterative sensing of the environment and comparing it with the known map.
despite of these, the most famous mobile robot companies which sell autonomous mobile robots already implement a localization system on the robot and most of them provide an auto-mapping technique which allows the users a simpler setting without the needs to introduce any map into the carrier memory. For our purpose and in general, is always better to have an already drawn environment mapping, to provide to the robot a strong initial base for the localization system.

3.3 Motion planning

The motion planning of the robot is another delicate task of this work. The choice of the path and its following should be perfectly selected to avoid obstacles and provide the best initial picking position for the manipulator. Moreover, the camera mounted over the carrier should be able to recognize as best as possible the surrounding environment both for monitoring and for recognition. The environment conformation is simple and is composed by parallel axils as can be seen in Fig. 3.7.

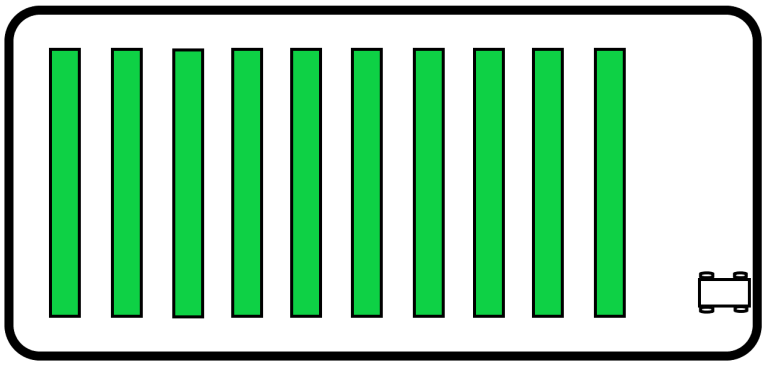


Figure 3.7. top view of the greenhouse displacement

Motion planning tasks are composed by three main points:

- Path planning: an obstacle-free path is found in the map to go from the starting point to the ending one
- Trajectory generation: a time trajectory must be created to assign a time sequence at the spatial path
- Trajectory tracking: the real controller which assures the robot trajectory following without errors

In Fig.3.8 can be see a logic scheme for these three motion planning passages.

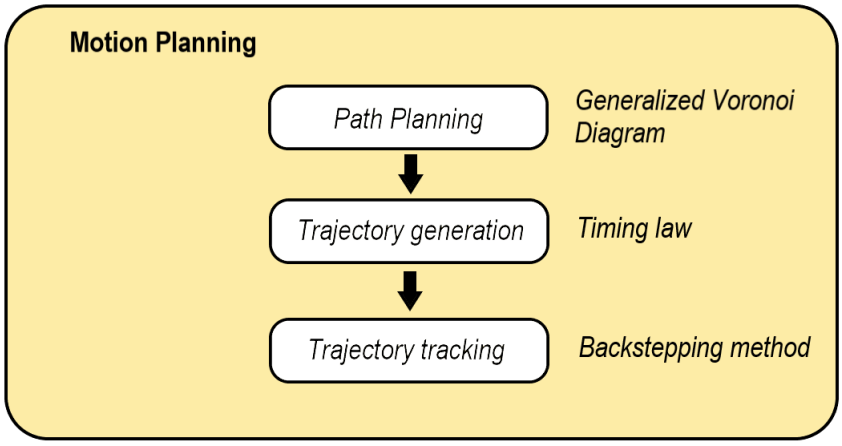


Figure 3.8. Motion Planning logic

Depending on the task and the movement to perform, a Voronoi diagram technique [31] can be used to find the path that the carrier must follow through the plant rows. This is possible because the map is really well structured and there aren't particular motions that the robot has to achieve. This algorithm should be

used preferably if the distance from the plant rows allows the robot to reach the plants from the center of the path because, in this way, it would be able to pick left and right zucchini flowers remaining on the Voronoi main path. Anyway, this is not strictly needed because an approaching movement can be implemented to overcome the distance between the manipulator and the plants and to maintain a good picking orientation between them. Other map-based techniques have been initially considered, e.g. the PRM [12] or RRT [39] algorithms could be used, but in this case a simpler algorithm is preferred because the task and the map of the environment are well known and the exploration of the environment is not a needed task. The final choice for this prject has been to use the Voronoi diagram algorithm and in next section 3.3.1 the reasons will be explained.

3.3.1 Path Planning: Generalized Voronoi diagram method

The generalized Voronoi diagram method is a retraction path planning technique [68]. The idea is to construct a roadmap $\mathcal{R}$, subset of the free configuration space C_{free}, which can be used to connect each starting point q_s with each goal point q_g on the map. In this way, by connecting the starting and ending point to the roadmap, it is possible to find with an optimization method, the admissible paths which connects two general points of C_{free}. Thanks to the structure of the greenhouse environment some simplifying assumption can be done:

- the free configuration space is represented by a closed portion of a $\mathbb{R}^2$ space
- the free configuration space and consequently also the obstacle space are polygonal spaces

These assumptions allow to express the algorithm in a really simple way. First of all let's express the clearance $\gamma(q)$ as

$$\gamma(q) = \min_{s \in \delta C_{free}} \|q - s\| \tag{3.6}$$

where δC_{free} is the boundary set of the free configuration space which divides it from the obstacle configuration space. Moreover, define the set $N(q)$ which represents the point s belonging from C_{free} which are the nearest to configuration q:

$$N(q) = \{s \in C_{free} s.t. \, \|q - s\| = \gamma(q)\} \tag{3.7}$$

In this way, the generalized Voronoi diagram can be realized by simply collecting all the points of the map in which the neighbour set $N(q)$ has a cardinality bigger than one i.e. the points which are equidistant at least from two points belonging from

the obstacle configuration space. Now, a set $V(C_{free})$, which represents the Voronoi diagram i.e. the roadmap previously introduced, is found and it can be used for the path planning between two general points. This set has the characteristic of maximizing the clearance between the robot and the obstacles in the environment, providing a safety path which minimize the collision possibility. Once that the road-map has been built, the starting and ending point q_s and q_g has to be retracted to the roadmap. This can be done by simply taking $\nabla\gamma(q)$ which is always directed in direction of the roadmap. By continuing in the direction of the steepest ascent direction $\nabla\gamma(q)$ and assuming that the configuration q is not isolated from the free configuration space, the roadmap $V(C_{free})$ will be encountered and intersected in the point $r(q)$ which represent the connection point between the configuration point q and the Voronoi diagram.
It is important to specify that, when $q \in V(C_{free})$ then $r(q) = q$ otherwise some problems could arise with the connection of q to the roadmap. Finally, after the connection of q_s and q_g to the diagram, the whole path connecting them must be found. If the graph representing the roadmap is completely connected then this path always exists. There are different ways to find this path but one of the most used is the application of the A* algorithm because it provides the path with minimum length in a small computational time. It is necessary to assign to each arc of the graph a weight representing its length and then the path is founded iteratively by the algorithm.

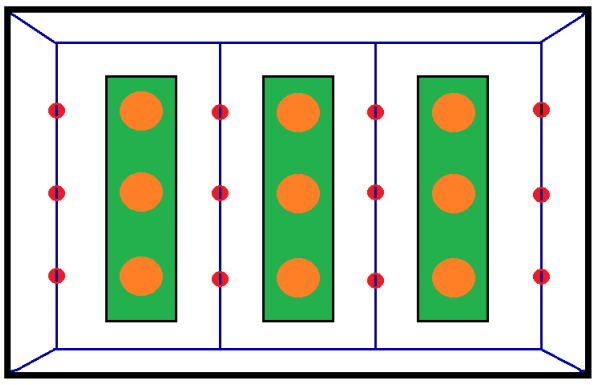


Figure 3.9. simple Voronoi diagram example with plant stopping points (red dots) on the path

Using this method, the distance from each plant bed is optimized and the robot is able to pick left and right zucchini flowers passing between the two plant beds and stopping in front of each plant; otherwise some approaching movements must been applied in order to move the carrier nearer to the plant. In practice each plant has its own configuration point on the map which can be reached directly with a path built with the Voronoi algorithm and then, after that the vision system has recognized the flowers, the approaching movement starts to bring the manipulator arm near to the plant to complete its task. This action simplifies the implementation of the motion planning algorithm and achieves always a good performance because the movements performed are really simple and they maximize the robot obstacle avoidance. In Fig. 3.9 a simple example of this path planning technique is presented: the orange points represents the zucchini plants inside their green beds, the blue line represent the Voronoi path and the red points represent the arrivals points for each plant. Obviously, this approach is possible only because an omni-directional carrier has been chosen; in this way the movements are not constrained by the structure of the robot and they are free to be realized in the most appropriated way.

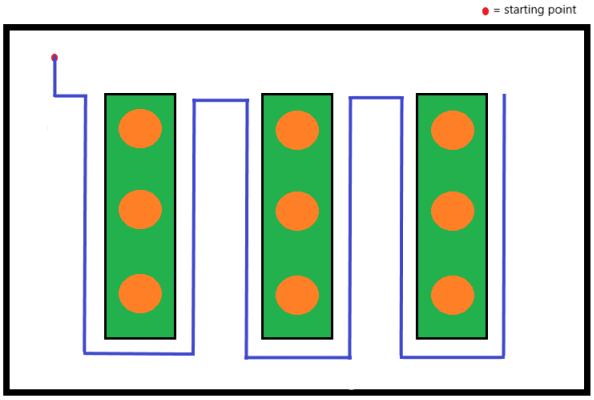


Figure 3.10. manual creation of a path for the robot movements into the greenhouse

This planning technique is not the optimal solution for this task because it does not minimize the distance traveled by the robot on the contrary it increase consistently the path length. This path length increasing burdens on the costs in terms of time and money but maximize the safety of the robot and the environment. This methods is suited for farmers which are not able to program or to impose a pre-stablished path to the robot but another simpler and elegant solution could

be the manual making of the path in a $(x(s),\ y(s))$ form in order to simplify not only the robot computations but also to fit the path at the particular form of each greenhouse, see Fig. 3.10. This manual method has been used for the simulation, see section **??**, but the approaching movement presented in section 3.4 is needed the same to increase the visual area of the flower recognition system.
Anyway, if the robotic arm can reach the plant remaining on the Voronoi diagram path then the first solution should be considered. Since the hydroponic greenhouse structure can be modeled in different ways during the organization planning, the farmer can choose the perfect distance between the plant beds, in such a way that the robot is free to move inside the beds and is able to crop left and right zucchini plants. This last choice is the best choice possible but depends on the structure decided during the greenhouse organization, indeed with some cooperation between farmers and engineers the robot can fit perfectly in the greenhouse.

3.3.2 Trajectory generation: assign a time law at the path

Finally, once the path has been found, a timing law must be assigned to it in order to have a trajectory in time to be tracked by the robot. In this case, except for the start and the stop of the robot, there is no need of acceleration or deceleration, in fact, depending on the cruise speed chosen for the robot, the trajectory must be rearranged. For example, if the robot speed is chosen equal to 0.5 m/s then the robot takes two seconds to move for one meter. By knowing that the path founded is a sequence of (x, y) points, the total distance from the starting point to the ending one can be calculated and, thanks to that, the time trajectory can be computed e.g. consider a 5 meters path and the robot speed equal to 0.5 m/s; the robot will take 10 second to accomplish the task, indeed the trajectory will finish at 10 s from the starting time and the path will be spread over this time. In this way, a trajectory of the form $(x(t), y(t))$ can be obtained and knowing that these are flat output the $\theta(t)$ angle trajectory can be computed as $\theta(t) = Atan2(\frac{y(t)}{x(t)})$. Now, the trajectory $[x_r(t)\ y_r(t)\ \theta_r(t)]^T$ has been found and the next step is to find a control strategy to let the robot follow this trajectory.

3.3.3 Trajectory tracking: backstepping approach

Trajectory tracking control strategy is a necessary step for the motion planning [28]. Once that the trajectory has been chosen the robot starts moving along it in open loop, i.e. without any measure. In this way, the possible initial position errors or integration errors caused by the dead reckoning are not considered. To overcome this problem a trajectory tracking control strategy must be applied to our robot

such that, thanks to the localization system and the trajectory generated, the car is able to perform the task imposed with almost zero final error. In this case the robot has to follow a 2-dimensional trajectory which implicitly comprise also the θ angle. First of all, let's define the error between the trajectory reference and the real position of the robot:

$$\begin{bmatrix} e_x(t) \\ e_y(t) \\ e_\theta(t) \end{bmatrix} = \begin{bmatrix} x_r(t) - x(t) \\ y_r(t) - y(t) \\ \theta_r(t) - \theta(t) \end{bmatrix} \tag{3.8}$$

Starting from this point and using the same technique of [28], i.e. the backstepping method, the control law is found. Is important to stress the following velocity relation before starting:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{3.9}$$

The backstepping method is a Lyapunov-based controlling design technique generally used with triangular systems, i.e. strict-feedback systems. It is an iterative procedure which aim to create a Lyapunov function for the whole system which assures the stability of the system. A similar approach for a car-like robot has been presented also in [1].
The results, after that the backstepping method has been applied on the nonlinear system, is:

$$v(t) = \sqrt{(\dot{x_r}(t) + c_1 e_x(t)) + (\dot{y_r}(t) + c_2 e_y(t))} \tag{3.10}$$

$$\alpha(t) = \arctan \frac{(\dot{y_r}(t) + c_2 e_y(t))}{(\dot{x_r}(t) + c_1 e_x(t))} \tag{3.11}$$

$$\omega(t) = \dot{\alpha}(t) + c_3(\alpha - \theta) \tag{3.12}$$

$\alpha(t)$ is a support function necessary for the development of the controller with the backstepping method and it represents the desired behavior for $\theta(t)$. In fact, by looking at 3.10, 3.11 and 3.12 its clear that if $e_x(t) = e_y(t) = 0$ then $v(t)$ follow exactly the references $\dot{x_r}$ and $\dot{y_r}$ and introducing this result in 3.11 then $\alpha(t) = \dot{\theta_r}(t)$ and the control 3.12 aim to stabilize the angle of the car exactly over its reference. Now that the controller for the velocity inputs $v(t)$ and $\omega(t)$ has been computed the

linear velocity on x and y axis can be decoupled with the equations in 3.9 and use the inverse kinematic of the robot in 3.5 to obtain the control inputs for each wheel. In this way, the robot is able to move from every admissible initial point to every admissible ending point in a two-dimensional space.

3.4 Approaching movement

As already introduced in 3.3.1, when the distance of the plants' beds is too high to Perform the picking task directly from the road center, some approaching movement must be performed to accost the manipulator arm to the plant. It is important to stress that, if the Voronoi diagram technique is not used in the planning stage then the approaching movement should be probably unnecessary because, by creating manually a path, the user could get rid of this approach building a path near enough to the plants.
Generally, it is possible to assign an ending point for the carrier exactly in front of the plant and let do all the work at the motion planner but in this way the final orientation is not always well considered because the Voronoi diagram technique works on a Cartesian plane which means that the θ states is not directly controlled but it is derived by the other two states, i.e. form the two flat outputs. For this reason, the final orientation of the robot may not be suitable for the picking task resulting in the necessity of a robot reorientation close to the plant which could create some damages either at the plants themselves or at the environment in general. Instead, using the approaching movement here presented, a safety translation of the robot through the plant can be assured.
In practice, when the robot stops in the ending point in the road center in front of the plants, it initially searches for ready zucchini flowers and then, if some of them are present, it starts the approaching movement which consists in a translation of the robot along its transversal axis. To be more clear, when the robot is arrived at its ending position, represented by one of the red dots in Fig. 3.9 its θ angle is exactly parallel to the plants' beds for construction of the Voronoi diagram and, thanks to its omni-directionality, it moves in an horizontal direction to get closer to the plants.
As can be seen in Fig. 3.11, considering the case of approaching the left bed, the robot moves through left until a good picking position is reached. In the figure has been highlighted the direction of rotation of each wheel and it's important to stress also that the velocity of each wheel must be the same to obtain a perfect transversal translation. This movement can be decided in two ways:

- the distance is defined manually depending on the distance from the plant ending point on the roadmap and the approach ending point near the plant
- the distance is automatically detected from the vision system or other proximity sensors

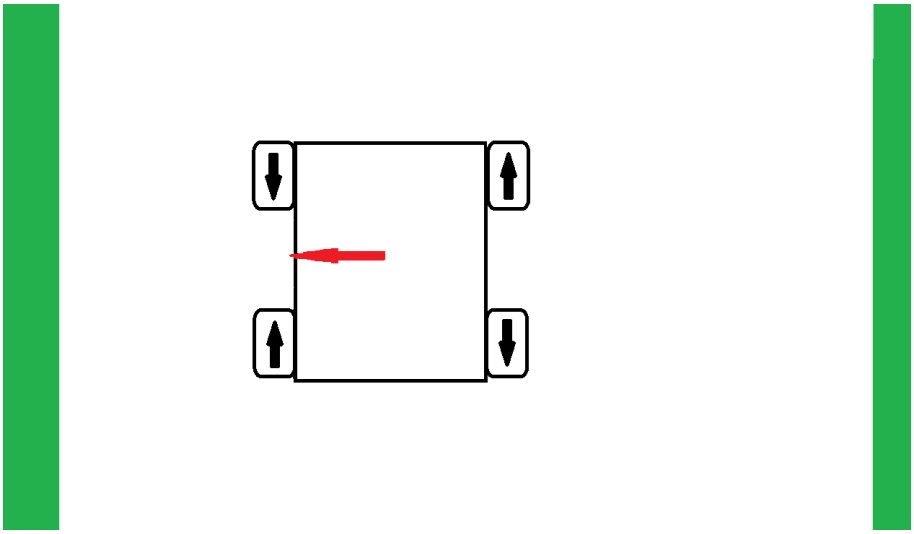


Figure 3.11. Approaching movement of the robot through the plant to accomplish the picking task

Both the choices have pros and cons. The first one must be added manually each time that the distance between the beds changes and its more sensitive with respect to many errors e.g. the beds distance can slightly vary if the plants rows are not well constructed. The second choice instead is more reliable because depending on the distance detected from the plant it changes the robot displacement, but it is more costly and difficult to implement.

Chapter 4

Manipulator arm

The second component analyzed is the manipulator arm. This part is crucial for the fulfillment of the picking task. The robotic arm should be able to move toward the flower (or fruit) without colliding with the sensible plant stalks to not damage it, approach the flower with the end-effector, separate the flower from the plant without causing any harm to it and bring back the arm to a safety position to place the picked flower into a collection box. To perform these movements the robot should be capable of sensing the flower and, more important, compute its position and orientation with respect to the end-effector. For this reason, the robotic arm must be incorporated with a vision system which is able to perform this localization. A treatment above the vision systems has been reported in 4.3. Another special attention is directed to the end-effector system which should be able to pick the flower without damaging it or the plant and, maybe, could also avoid some infection problem between the plants, as it will be specified in 4.2

4.1 Manipulator arm structure

In most of the existing applications a 6 DOF robotic arm or at most 7 DOF one has been used for the picking task. Due to the dexterity of these robots, they are necessary to work in an environment in which the possibility of collision is high. Using them, the manipulator can follow different trajectories and perform different poses to approach the goal object. These wide possibilities of actions, i.e. velocity directions available, improve the motion planner because the possibility of movement are many and the obstacle avoidance task is simplified. For our application, a 6 DOF manipulator arm composed by an anthropomorphic arm (3 DOF) plus a spherical wrist (3 DOF) has been chosen. The spherical wrist assigns to the end-effector the required dexterity for this spatial task. By adding another DOF the dexterous space of the robot is improved because its robot redundancy admits more configurations

for a single point then a 6Dof arm. Generally, the addition of this degree of freedom is not strictly needed although in lots of agricultural task it is used to ensure a smooth motion of the end-effector through the free configuration space reducing the possibility of obstacle collision which could damage the robot, the environment or the plants.
Anyway, as just said, the 6 DOF manipulator arm has been chosen for this work because it is slightly simpler than the 7 DOF one and because regarding the conformation of the plant it is enough for this picking task.

4.1.1 Forward kinematics

The direct kinematics of this robot is derived directly by using the Denavit-Hartemberg convention. This convention establishes a standard procedure for the derivation of the direct kinematic of the robot needed to find a relation between the end-effector pose and the robot configuration, i.e. the joint angles.
In this case, the robot pose is described by six variables, three for the position of the end-effector and three for its orientation $x_e = [p_x\ p_y\ p_z\ \theta\ \phi\ \psi]^T$. Moreover, also the joint variables are six and they can be expressed as $q = [q_1\ q_2\ q_3\ q_4\ q_5\ q_6]^T$ where each q_i represents the angle θ_i of the joint i of the robot. Returning to the Denavit-Hartenberg convention, it mainly consist in finding four parameters, called DH parameters, that will be used to create some homogeneous transformation $A_i^{i-1}(DH_i)$ which will be used to describe mathematically the connection of the robot kinematic structure. DH_i represents the Denavit-hartemberg parameters for each joint i which are $DH_i = [a_i\ \alpha_i\ d_i\ \theta_i]$. It is important to stress that due to the rotatory structure of the joints the only DH parameter which can change with time is θ_i for each joint i, indeed, the homogeneous transformation can be rewritten as $A_i^{i-1}(\theta_i)$. Moreover, by noticing that the parameter θ_i represents exactly the configuration value q_i of joint i the final homogeneous transformation will be described as $A_i^{i-1}(q_i)$. The DH parameters are used to build the coordinate transformation between the reference frames attached to each robot link with the following one. In practice, with this coordinate transformation the reference frame attached to the joint i is translated and orientated in such a way to coincide with the reference frame of joint $i+1$. With this connection, by changing the value of the θ_i parameter, is always possible to describe the orientation of link $i+1$ with respect to link i for any joint variable value $\theta_i \in [0, 2\pi)(rad)$.
Once that all the connections, i.e. all the homogeneous transformations, have been computed, the direct kinematic $T_n^0(q)$ of the robot can be directly found with the

ordered matrix multiplication of the homogeneous transformation:

$$T_n^0(q) = A_1^0(q_1) * A_2^1(q_2) * \cdots * A_n^{n-1}(q_n) \tag{4.1}$$

. A full treatment of the Denavit-Hartemberg convention can be founded in [68] The robot direct kinematic has the form:

$$T_6^0(q) = \begin{bmatrix} n_6^0 & s_6^0 & a_6^0 & p_6^0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

where n_6^0, s_6^0, a_6^0, p_6^0 are four 3x1 vectors representing respectively the normal direction, the slide direction, the approach direction and the position of the end-effector with respect to the first joint reference frame. In practice, by looking at Fig. 4.1 can be said that n_6, s_6 and a_6 are respectively the vectors x_6, y_6 and z_6 of the end-effector and then the matrix $R_6^0(q) = [n_6^0\ s_6^0\ a_6^0]$ represents the rotation matrix of the first joint reference frame with this last one attached t the end-effector. Instead, p_6^0 is the vector which translate the origin of the first reference frame over the origin of the end-effector one.

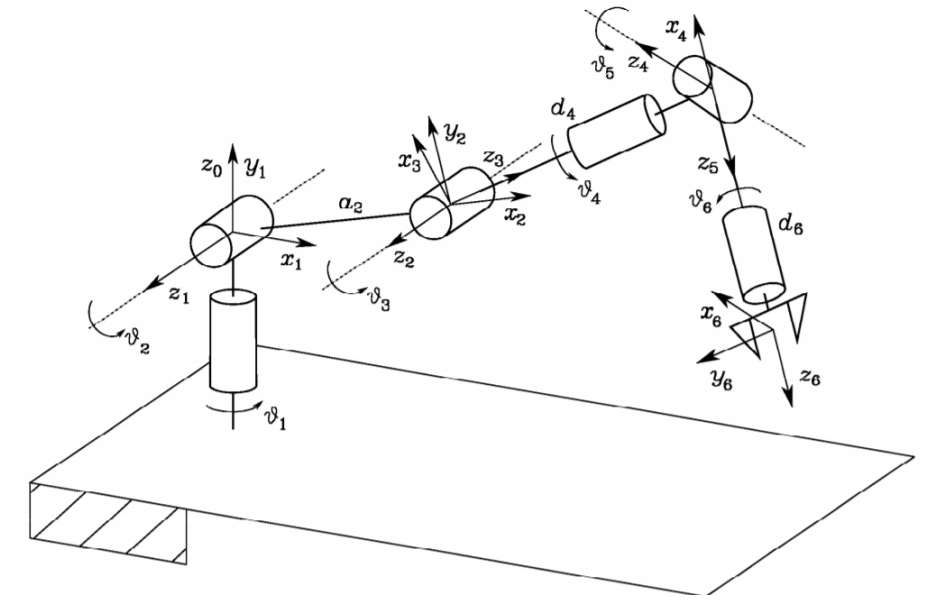


Figure 4.1. Anthropomorphic arm with a spherical wrist (6 DOF configuration) [68]

For the peculiar structure chosen, see Fig. 4.1, the DH parameter can be seen in Table 4.1. With these DH parameters the six $A_i^{i+1}(q_i)$ homogeneous transformations can be iteratively computed and thanks to equation 4.1 the following component

can be computed:

$$n_6^0(q) = \begin{bmatrix} c_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) + s_1(s_4c_5c_6 + c_4s_6) \\ s_1(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6) - c_1(s_4c_5c_6 + c_4s_6) \\ s_{23}(c_4c_5c_6 - s_4s_6) + c_{23}s_5s_6 \end{bmatrix} \tag{4.3}$$

$$s_6^0(q) = \begin{bmatrix} c_1(-c_{23}(c_4c_5c_6 + s_4s_6) + s_{23}s_5c_6) + s_1(-s_4c_5c_6 + c_4s_6) \\ s_1(-c_{23}(c_4c_5c_6 + s_4s_6) + s_{23}s_5c_6) - c_1(-s_4c_5c_6 + c_4s_6) \\ -s_{23}(c_4c_5c_6 + s_4s_6) - c_{23}s_5s_6 \end{bmatrix} \tag{4.4}$$

$$a_6^0(q) = \begin{bmatrix} c_1(c_{23}c_4s_5 + s_{23}c_5) + s_1s_4s_5 \\ s_1(c_{23}c_4s_5 + s_{23}c_5) - c_1s_4s_5 \\ s_{23}c_4s_5 - c_{23}c_5 \end{bmatrix} \tag{4.5}$$

$$p_6^0(q) = \begin{bmatrix} a_2c_1c_2 + d_4c_1s_{23} + d_6(c_1(c_{23}c_4s_5 + s_{23}c_5) + s_1s_4s_5) \\ a_2s_1c_2 + d_4s_1s_{23} + d_6(s_1(c_{23}c_4s_5 + s_{23}c_5) - c_1s_4s_5) \\ a_2s_2 - d_4c_{23} + d_6(s_{23}c_4s_5 - c_{23}c_5) \end{bmatrix} \tag{4.6}$$

Finally, substituting 4.3, 4.4, 4.5, 4.6 into 4.2 the complete direct kinematic formula is found.

Link	a_i	α_i	d_i	θ_i
1	0	$\frac{\pi}{2}$	0	θ_1
2	a_2	0	0	θ_2
3	0	$\frac{\pi}{2}$	0	θ_3
4	0	$-\frac{\pi}{2}$	d_4	θ_4
5	0	$\frac{\pi}{2}$	0	θ_5
6	0	0	d_6	θ_6

Table 4.1. DH parameter of the 6 DOF structure in Fig. 4.1

The forward kinematic form represented in 4.1 is not generally used because the homogeneous transformation is a matrix composed by a rotation matrix and a translation vector. Instead, the direct kinematic is generally preferred in another more compact form:

$$x_e = k(q) \tag{4.7}$$

Equation 4.7 is preferred with respect to 4.1 because its non-linear relation express directly the connection between the robot pose variables with its joint values. The

robot workspace is a 3D space and thus six component are needed to express the pose of an object in it: $x_e = [p_x\ p_y\ p_z\ \theta\ \phi\ \psi]^T$. Luckily, the first three components can be extracted directly form 4.1 because its last column represents exactly the homogeneous vector which connect the position of the first joint of the robot with the position of the end-effector.
Regarding the three angles θ, ϕ and ψ the computation is slightly more difficult. In fact, the angles need to be extracted directly from the rotation matrix in the homogeneous transformation and this is not foregone. Moreover, there exist different ways to express these three angles, e.g. in an Euler angles form and this can be done by analyzing directly the rotation matrix and extracting the angles from its entries. A simple way to express the angles θ, ϕ and ψ is in the ZYZ Euler form and the method for the extraction of this angles from the rotation matrix is presented in appendix A.
It's important now to stress that the direct kinematic expresses the relation between the reference frame 0 of the robot first joint to the reference frame 6, also known as e, of the robot end-effector. Once that the robot is placed in a fixed location another homogeneous transformation must be considered, the one which connect the robot base reference frame with the first reference frame of the robot. This operation is simple because, once the homogeneous transformation T_0^B is found, it has to be pre-multiplied to the direct kinematics of the robot to obtain the final connection. The same happens if there is the necessity to express the position and orientation of the robot with respect to the environment reference frame. In this case the transformation T_B^W express the roto-translation between the environment reference frame and the base reference frame of the robot. In this case, the T_0^B homogeneous transformation is composed by a simple translation of the base reference frame on the first forward kinematic one. In fact, the structure is:

$$T_0^B = \begin{bmatrix} 0 & 0 & 0 & x \\ 0 & 0 & 0 & y \\ 0 & 0 & 0 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

where x, y, z position depends on the position of the first direct kinematic reference frame with respect to the base one. In the simulations only the z component will be used because the robotic arm will be positioned exactly at the center of the mobile platform. This is because the mobile reference frame of the carrier will be considered as the base reference frame of the manipulator arm. Therefore, in the simulation the three translation values will be: $x = y = 0$ and $z = constant$.

4.1.2 Inverse kinematics

Although the robot forward kinematic could be useful for many tasks, another important connection between the joint variables and the robot pose is given by the inverse kinematic. This relation is the opposite of the direct one because starting from the robot end-effector position and orientation values, it is possible to compute the joint variable ones using the inverse kinematic. This relation is very useful because by giving a pose reference to the robot, i.e. a x_e reference (position and orientation in this case), the robot can adjust its pose using references in the joint space.
In general, most of the robots are already geared with some low-level controllers, which means that the feedback control loop for low-level tasks i.e. dynamic ones, is already implemented and this include also the inverse kinematic. In this way, the user can control the robot by directly giving the reference in a Cartesian fashion and the robot performs the work itself. The solution here presented has been obtained from [68] where a wide treatment of different kinematic structures is given.
Let's begin by saying that in general the 6 DOF manipulator arm structures are composed by a supporting structure characterized by 3 DOF and a spherical wrist attached to it, always composed by 3 DOF. The idea behind this concept is that, by looking at the structure under this prospective, the inverse kinematic problem can be decoupled into two sub-problems, considering, in this way, the position and the orientation of the robot separately.
To do this, it is important primarily to find the point p_W which separates the two kinematic chains, see Figs. 4.2 and 4.3, because the supporting structure joint variables will be computed with respect to it to obtain the inverse kinematic solution of an anthropomorphic arm with the end-effector centered in p_W (obviously this is the solution for the position of the 6 DOF structure). For the structure considered, see Fig. 4.1, p_W is centered exactly into the spherical wrist center of the fifth joint. Now, using the direct kinematic presented in 4.1.1 the separation point can be found as

$$p_W = p_6^0 - d_6 a_6^0 \tag{4.9}$$

Can be noticed how the equation in 4.9 is dependent only by the joint variables which are connected to the robot position. It is now possible to start with the inverse kinematic computation for the two different structures; the anthropomorphic arm solution will be firstly presented and then the spherical wrist one.

Anthropomorphic arm inverse kinematic solution

Consider a generic Anthropomorphic arm structure composed by three rotary joints such the one in Fig. 4.2.

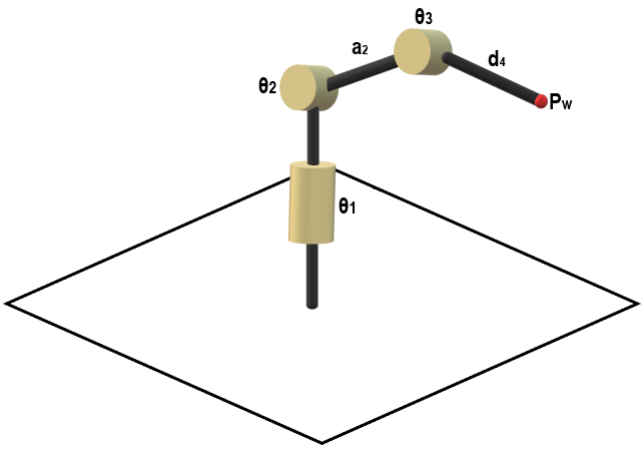


Figure 4.2. Anthropomorphic arm structure. The length of the last link has been called d_4 because this structure is related with the arm in Fig. 4.1

Its forward kinematic expressed for p_W is obtained always using the Denavit-Hartemberg convention or alternatively can be extracted from the kinematic of the 6 DOF robot substituting $d_6 = 0$ and θ_3 with $\theta_3 + \frac{\pi}{2}$.
Anyway in table 4.2 the DH parameters are presented while in 4.10 is expressed the forward kinematic of the robot.

Link	a_i	α_i	d_i	θ_i
1	0	$\frac{\pi}{2}$	0	θ_1
2	a_2	0	0	θ_2
3	d_4	0	0	θ_3

Table 4.2. DH parameters for the anthropomorphic arm structure of Fig. 4.2

$$T_3^0(q_{123}) = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1(a_2c_2 + d_4c_{23}) \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1(a_2c_2 + d_4c_{23}) \\ s_{23} & c_{23} & 0 & a_2s_2 + d_4s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

From 4.10 the relation between the end-effector position p_W and the first three robot

joints variables $q_{123} = [\theta_1, \theta_2, \theta_3]$ is

$$p_{Wx} = c_1(a_2c_2 + d_4c_{23}) \tag{4.11}$$

$$p_{Wy} = s_1(a_2c_2 + d_4c_{23}) \tag{4.12}$$

$$p_{Wz} = a_2s_2 + d_4s_{23} \tag{4.13}$$

By elevating to the power of two the equations 4.11-4.13 and then summing them the following result is obtained

$$p_{Wx}^2 + p_{Wy}^2 + p_{Wz}^2 = a_2^2 + d_4^2 + 2a_2d_4c_3 \tag{4.14}$$

which is dependent only by the cosine of θ_3, indeed

$$c_3 = \frac{p_{Wx}^2 + p_{Wy}^2 + p_{Wz}^2 - a_2^2 - d_4^2}{2a_2d_4} \tag{4.15}$$

Obviously, the solution is admissible if and only if $-1 \leq c_3 \leq 1$ otherwise the point p_W is out of range, i.e. it is not into the work space of the robot.
Now, by posing

$$s_3 = \pm\sqrt{1 - c_3^2} \tag{4.16}$$

the angle θ_3 can be computed with

$$\theta_3 = Atan2(s_3, c_3) \tag{4.17}$$

The equation in 4.17 provides two different solution depending on the sign of s_3:

$$\theta_{3,1} \in [-\pi, \pi] \tag{4.18}$$

$$\theta_{3,2} = -\theta_3, 1 \tag{4.19}$$

following a similar procedure, the solution for θ_2 can be found. Taking the sum of the square of 4.11 and 4.12 it possible to obtain

$$a_2c_2 + d_4c_{23} = \pm\sqrt{p_{Wx}^2 + p_{Wy}^2} \tag{4.20}$$

Equation 4.20 can be rearranged and thanks to some substitution it gives

$$c_2 = \frac{\pm\sqrt{p_{Wx}^2 + p_{Wy}^2}(a_2d_4c_3) + p_{Wz}d_4s_3}{a_2^2 + d_4^2 + 2a_2d_4c_3} \tag{4.21}$$

$$s_2 = \frac{p_{Wz}(a_2 + d_4 c_3) \mp \sqrt{p_{Wx}^2 + p_{Wy}^2} a_3 s_3}{a_2^2 + d_4^2 + 2 a_2 d_4 c_3} \tag{4.22}$$

Finally, with

$$\theta_2 = Atan2(s_2, c_2) \tag{4.23}$$

the second joint variable is found. It is important to be stressed that there are four possible solutions for θ_2 because θ_3 have two possible solution, each of which could have two solutions for θ_2. This solutions will be named $\theta_{2,1}$, $\theta_{2,2}$, $\theta_{2,3}$ and $\theta_{2,4}$. Finally, to find θ_1 a simple procedure can be followed, substitute the equation 4.20 into 4.11 and 4.12 obtaining the expressions

$$p_{Wx} = \pm c_1 \sqrt{p_{Wx}^2 + p_{Wy}^2} \tag{4.24}$$

$$p_{Wy} = \pm s_1 \sqrt{p_{Wx}^2 + p_{Wy}^2} \tag{4.25}$$

that, once resolved, give the two solutions

$$\theta_{1,1} = Atan2(p_{Wy}, p_{Wx}) \tag{4.26}$$

$$\theta_{1,2} = Atan2(-p_{Wy}, -p_{Wx}) \tag{4.27}$$

Now, that all the solutions for the joint variable values have been found, can be seen how the solutions have given four different robot configurations for the same point p_W:

- ($\theta_{1,1}$, $\theta_{2,1}$, $\theta_{3,1}$)
- ($\theta_{1,1}$, $\theta_{2,3}$, $\theta_{3,2}$)
- ($\theta_{1,2}$, $\theta_{2,2}$, $\theta_{3,1}$)
- ($\theta_{1,2}$, $\theta_{2,4}$, $\theta_{3,2}$)

. This group of possible solution is very useful for the obstacle avoidance because it increases the possibility of finding an admissible path.

Spherical wrist inverse kinematic solution

The spherical wrist, see Fig. 4.3 is used to improve the dexterity of the robot by increasing its orientation possibilities. Indeed, the inverse kinematic solution of this subsystem is found by looking directly at its forward kinematic and its orientation angles will be founded with the Euler angles method explained in appendix A.

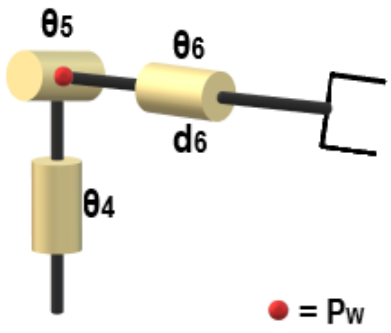


Figure 4.3. Spherical wrist structure

Let's introduce the direct kinematic of the spherical wrist:

$$T_6^3(q_{456}) = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 & c_4s_5d_6 \\ s_4c_5c_6 - c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 & s_4s_5d_6 \\ -s_5c_6 & s_5s_6 & c_5 & c_5d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.28}$$

From the expression in 4.28, by extracting the 3x3 rotation matrix $R_6^3(q_{456})$ which characterize the orientation of the end-effector, it is possible to obtain the values of $q_{456} = [\theta_4, \theta_5, \theta_6]$ by simply applying the transformation of appendix A.1. In fact, once that the rotation matrix $R_6^3(q_{456})$ is expressed as

$$R_6^3(q_{456}) = \begin{bmatrix} n_x^3 & s_x^3 & a_x^3 \\ n_y^3 & s_y^3 & a_y^3 \\ n_z^3 & s_z^3 & a_z^3 \end{bmatrix}$$

the solution can be found simply with the following expressions

$$\theta_4 = Atan2(a_y^3, a_x^3) \tag{4.29}$$

$$\theta_5 = Atan2(\sqrt{(a_x^3)^2 + (a_y^3)^2}, a_z^3) \tag{4.30}$$

$$\theta_6 = Atan2(s_z^3, -n_z^3) \tag{4.31}$$

if $\theta_5 \in (0, \pi)$, otherwise

$$\theta_4 = Atan2(-a_y^3, -a_x^3) \tag{4.32}$$

$$\theta_5 = Atan2(-\sqrt{(a_x^3)^2 + (a_y^3)^2}, a_z^3) \tag{4.33}$$

$$\theta_6 = Atan2(-s_z^3, n_z^3) \tag{4.34}$$

if $\theta_5 \in (-\pi, 0)$.

4.1.3 Differential kinematics

The differential kinematics expresses the relation between the joint velocity space and the Cartesian speed one. This relation is always linear with respect to the velocities and the matrix representing this transformation is known as Jacobian matrix. It is called Jacobian matrix because it is the full-fledged forward kinematic Jacobian of the robot. There exists two ways to derive this matrix, the first one is, as just said, by taking the derivative of the forward kinematic expressed as 4.7 and the other one is by using the robot geometric constructions to derive it. Generally, the first one, known as analytic Jacobian, is not equal to the second, the geometric one, but they could be both equally used without many problems. The difference between the two is in the angular velocity components but this argument is well treated in literature, for example in [68].
For our purpose, due to the simulative nature of the thesis, the analytic Jacobian will be used and it will be calculated directly on the programming platform by deriving the robot forward kinematics, obtaining a 6×6 matrix $J_A(q)$. Thanks to this Jacobian, the transformation between the two velocity spaces is possible. Indeed, the forward and inverse velocity relations are given by:

$$\dot{x_e} = J_A(q)\dot{q} \tag{4.35}$$

$$\dot{q} = J_A^{-1}(q)\dot{x_e} \tag{4.36}$$

The inversion of the Jacobian is possible because the matrix is square, but a special attention should be used for the robot structure singularites because in that case the jacobian is no more invertible and the velocities close to any singularity point are probably too much fast for the robot capabilities. The study and the handling of these singularities should be introduced in this work, but this will be left for future studies.

4.2 End-effector

Regarding the end-effector, there exists different solutions each one applied to a different picking task. There exist end-effectors composed by a four fingers hand for a good grip on the picked fruit and, with a small torsion, it allows the fruit collection without damages. In [78] this kind of hand has been used for the collection of tomatoes and indeed the structure of this hand fit perfectly with the spherical form of that fruit. There exists different end-effectors with this form mainly used to

pick tomatoes, strawberries and other small-shaped fruits.
Other kind of pickers are connected to a fruit collector, e.g. a small basket where the fruit can be housed until it will be placed into the collection box. To cut the fruit off the plant a linear saw can be used and it can be positioned in such a way that once the fruit stem has been cut, the fruit fall directly into the collector, see Fig. 4.4. This solution is generally used for high-weight fruits with a woody stem. As an example, a famous robot which uses this kind of end-effector is the SWEEPER robot [74], a robot recently built for the collection of sweet peppers. Anyway, the linear saw can be replaced with a scissors for that plants fruits which has a mild stem.

Figure 4.4. Stylized end-effector for picking different kind of friuts composed by a basket (in brown) and a linear saw (in grey)

The simpler and most common way for collecting some fruits or vegetables is the gripper end-effector. This one is the preferred for the zucchini flowers task because in this way the contact between the flower and the robot is constrained to be only on the stem, the non-edible part of the flower. In this way by applying a pressure with the gripper and turning the wrist to apply a torsion, the flower stem can be safety detached from the plant.
For a soft execution of this task, there exist other more elegant solution such as the use of a scissors annexed to the gripper. In this way, once the gripper has grasped the flower stem, the scissors can cut the stem to gently separate the flower from the plant, see Fig. 4.5.

Figure 4.5. Stylized end-effector for picking zucchini flowers composed by a gripper (in brown) and a scissors (in grey)

Another interesting way to separate the fruit from the plant is the one used in [76] where the gripper is equipped with a thermal cutting device which, not only separate the flower form the plant, but also prevents infection transfer between the plants of the same greenhouse. The zucchini or pumpkin plants, such as all the plants of the cucurbit's family, are inclined to have different fungal infections or other diseases which can be passed through the plants through the cutting devices. By heating up this device the illness transmission can be avoided because the overheating destroys all the bacteria, viruses and fungus which could be present on it. This is a smart idea that can prevent the diseases transmission trough plants inside the greenhouse and prevent also form the crop loss.

4.3 Vision system

For agricultural picking tasks one of the most important part is the recognition, maturity evaluation and the fruits localization. Indeed, for the task presented in this work, the camera, see section 4.3.1, is used not only for the picking task, but also for monitoring and identifying ones. The first action that the robot performs once it has arrived on its picking position, see section 2.1, is to turn the end-effector, on which the camera is mounted, towards the plant to identify some fruits, or flowers in this case, which are waiting to be gathered, see section 4.3.2. Due to the distance between the robot and the plant, initially the flower is identified through its color because its yellow-orange color generally is not present over the plant which is characterized by a light green hue. In alternative, a generic zucchini flower shape can be saved in the robot memory to identify the flower also from its shape during the image processing. The best recognition behavior in this visual recognition task should be to take different pictures of the plants using at least two point of view of the end-effector to improve the possibilities of finding the yellow or pale orange hue of the flower on

the RGB pictures. In this way, the robot should recognize the flower directly on its path. If the flower is recognized then the robot start the approaching movement, see section 3.4, and once the right picking position is achieved the camera take a nearest picture of the plant to identify the flowers which are ready to be picked, see section 4.3.3. Now, using this time also the depth sensors, the robot chooses a ready flower and localizes it in a 3D environment, see section 4.3.4. The robot starts the picking process by driving the end-effector near to the flower. Finally, with a last picture the robot identifies exactly the position and orientation of the flower with respect to the end effector and it accomplishes the picking task by grasping and cutting the flower stem.

4.3.1 Camera specifications

The first things to clarify is that, at least, an RGB camera is needed because this simplify the identification and recognition task otherwise the robot could not identify the presence and ripeness of the flower especially when it is far from the plant. The RGB cameras are equipped with a three-channel sensor which sense the intensity of the three primary colors (Red, Green and Blue) and analyze them to reconstruct the original image with some discrete values. The most common image sensors are the CCD sensor and the CMOS ones; The first one, the Charge Coupled Device, is characterized by a rectangular matrix composed by pixels. When a photon collides with the semiconductor, some free electrons are generated which charge a semiconductor depending on the illumination time integral over the pixel. The second sensor, Complementary Metal Oxide Semiconductor, is composed by a rectangular matrix of photodiodes. In this case, the photodiodes are pre-charged and they are drained when the photon collide with them. In this way, the sensor measures the quantity of the photons and not their volume and this reduce the blurring which instead affect the CCD sensors.
Anyway, the cameras are composed not only by these sensors, but other important components are the shutter, used to control the photon incidence time on the sensors, the lens, used to focus the light on the image plane and the signal conditioning analog electronics, used to convert the electric energy of the semiconductors in a digital color.
Different types of cameras have been created and are used to perform the 3D reconstruction of an object or environment [56]; the most common are:

1. stereo vision camera. Composition of two or more cameras (or point of views).

2. time of flight (ToF) camera.

3. structured light cameras
4. light coding cameras
5. laser triangulation cameras

The first one is characterized by the composition of two or more point of views of the same object. Generally, this is performed using two or more cameras, but, in some cases, the same camera can be used to take different picture of the surrounding. There exists different 3D reconstruction algorithm, the most common are the marker-based algorithms, where some markers are glued over the object to reconstruct, e.g. some high colored points which are simple to extract from the images, and the markerless algorithms which are characterized by the automatic feature extraction and tracking of some characteristic of the objects.
The second camera, Time of Flight camera or ToF camera, is composed by an infrared camera and a matrix of lasers. The lasers emit some pulses of light (visible or infrared) which are reflected by the objects and the environment and returns back to the infrared camera. The camera computes the phase shift of the sensed laser, i.e. compute the incidence time, which is dependent on the distance between the object and the camera. The general equation for the distance computing is

$$d = \frac{t_d * c}{2} \tag{4.37}$$

where d is the distance, t_d is the delay time and c is the light speed constant (299792458 m/s more or less). This Type of cameras allows very fast distance acquisition but there are also many problems regarding the disturbances, noises and resolution.
The structured light cameras are composed by a light projector and one or more cameras. The projector creates some fixed pattern light which will be reflected by the environment. The cameras sense this pattern, but the environment distorts it and thanks to some reconstruction techniques is possible to reconstruct the surrounding. The pattern can be either unique, one-shot techniques, or multiple, time-multiplexing techniques, anyway the patterns are generally characterized by a bar-code form, i.e. black and white consecutive vertical and/or horizontal lines.
The fourth camera type, known as light coding camera, is an evolution of the structured light cameras. In this case, the light source is continuously turned on and the infrared light source project a constant semi-random matrix of dots on the environment. This fixed dot pattern is collected by an infrared camera and then analyzed for the 3D reconstruction. This kind of cameras are innovative but there are some problems; the sensed map is not dense because the holes between the dots are not sensed, the near and far distances are generally not well sensed due to

distortion and attenuation. Moreover, some infrared waves emitted from the sun and reflected by the environment could disturb its infrared cameras. Anyway, these types of cameras are considered very valid for their low cost and functionalities in some robotic tasks and the industry improvement is giving birth to cameras with better resolution.
The last camera type considered is the laser triangulation camera. As the name suggest, these cameras are composed by a camera and a infrared light emitter. Due to the knowledge of the distance between the camera and the emitter, the angle of the emitter with respect the camera and thanks to the sensing of the infrared dot on the environment, it is possible to triangulate and compute the distance between the camera and the pointed object. More sophisticated sensors are composed by a line of emitters, i.e. a line of infrared dots, which are projected onto the surrounding. This type of camera is not able to reconstruct the environment using a single shot but need to be moved to obtain different measurements for the 3D reconstruction.
Once these cameras types have been presented, it is important to highlight that each camera type needs to be calibrated for the estimation of its intrinsic parameters, otherwise the various computations and sensing could be distorted and noisy. For more information about these [68] is a good reference and there are also different resources which treat this argument.

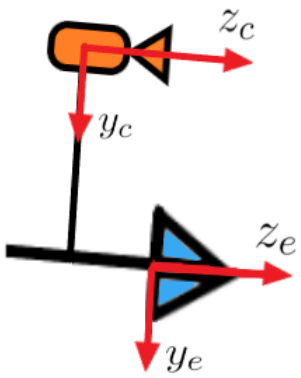


Figure 4.6. Camera position with respect to the end-effector one

The camera chosen to accomplish the task of this work is the Tof camera because allows fast and simple computations and have great resolution. One of the most known ToF cameras is the Microsoft Kinect camera [86] which is greatly used for 3D recognition and localization task in literature.
The last thing to say about the camera is its position with respect to the end-effector.

The camera is mounted directly on the robot, therfore it is an eye-in-hand camera. It is positioned directly over the wrist and, in particular, over the last joint connected to the end-effector. In this way, the camera moves exactly as the end-effector and points always in its same direction. In fact, the reference frame of the camera with respect the end-effector is characterized by only a translation, see Fig. 4.6.

Since both the camera reference frame and the end-effector one are on the same ZY plane the translation can be performed only on that allowing a translation of

$$X_t = \begin{bmatrix} 0 \\ y_e^c \\ z_e^c \end{bmatrix} \tag{4.38}$$

Where y_e^c and z_e^c values depends on the effective position of the camera with respect to the end-effector and generally they are both negative because the camera is behind and above the end-effector, see Fig. 4.6. The final homogeneous transformation is:

$$T_e^c = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_e^c \\ 0 & 0 & 0 & z_e^c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.39}$$

4.3.2 Flower identification

The flower identification is the first block of the algorithm in Fig. 2.1 in which the camera is involved. The robot points the end-effector towards the contemplated plant and takes one or more photos with different points of view. Those photos are then analyzed by a color recognition algorithm which on the base of the presence or not of the characteristic yellow hue of the zucchini flower decides if the picking task must start on that plant.
The RGB camera takes a photo and extract the three matrix layers which represents the intensity matrices, generally represented with a vector with range $[0, 255]$. Indeed, the pixels are composed by three values all ranging inside that vector, in general

$$p_i \in [0, 255] \times [0, 255] \times [0, 255] \tag{4.40}$$

With this resolution the camera is able to recognize $256 * 256 * 256 = 16777216$ different colors which are useless for our task but most of the camera on the market use this configuration and change from a resolution to another one is not difficult because it is enough to normalize the color here presented by dividing the pixel intensity x by the maximal intensity, i.e $\frac{x}{256}$, and then multiply for the new maximal intensity and round the number to the nearest integer. Generally, the maximal

intensity number is a power of two because depends on the bits assigned to each pixel.
Anyway, using the chosen configuration a yellow pixel p_{i_y} is characterized by the triplet $(255, 255, 0)$ because it is composed by the two fundamental colors red and green. The zucchini flowers color is like this triplet but has a slight orange hue, in fact a general triplet for representing its color is $(255, 220, 0)$ and it is represented in Fig. 4.7. The only problem is that the intensity of these colors on the camera depends on a bunch of factors such as the brightness, the distance, the flower maturity, the interference on the camera and others. Due to this generally also the camera works on a range of colors both for the RGB values.

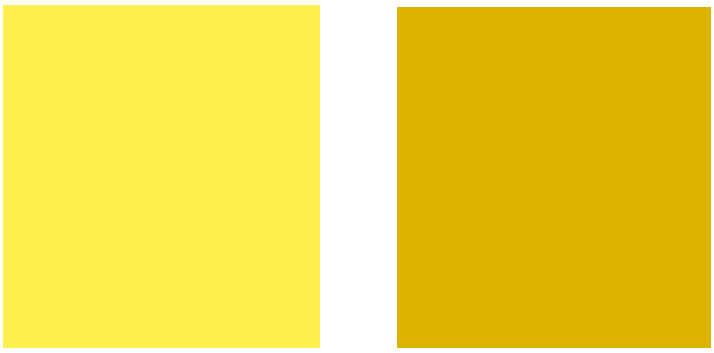
Figure 4.7. The right ripeness flower color. RGB value $(255, 220, 0)$

Since the aim of this first task is to detect the flowers presence on the plant, a perfect identification is not needed and, therefore, has been decided to increase slightly the range of yellow accepted hues. Moreover, the distance, the light and the camera resolution could affect these values and by increasing this range also the robustness of the identification is improved. The range to be considered for the RGB values of the pixels are:

- a value bigger than 220 for the R component
- a value contained in $[180, 240]$ for the G component
- a value inside $[0, 80]$ for the B component

Another important constraint to be considered is $R - G > 15$ this is because if the green intensity is close to the red one or even higher the pixel color become more green than yellow resulting in the identification of the not yet ripe flowers or also plants parts. In this way, a group of possible pixel colors have been selected for the flower identification. In particular the colors vary from the color in Fig. 4.8a to the

one in Fig. 4.8b. The colors in 4.8 are the boundary colors for the identification task, obviously the optimal color is inside these boundaries as can be seen from Fig. 4.7.



(a) Maximal RGB color for the right identification of the flower on the plant. RGB value $(255, 240, 80)$

(b) Minimal RGB color for the right identification of the flower on the plant. RGB value $(220, 180, 0)$

Figure 4.8. Minimal and maximal RGB colors for flower identification

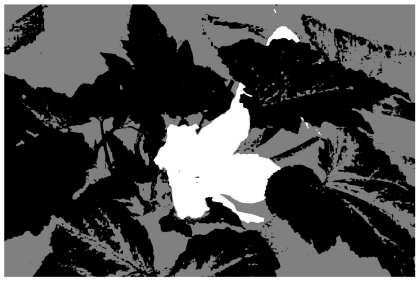
Once that the color specifications have been identified, a clustering algorithm is needed to separate the colors of the image into different clusters characterized by the same or a similar color. In this way, different groups distinguished by their color can be obtained and used to identify the flower inside the multitude of different shapes which are present in a greenhouse.
The chosen algorithm for accomplishing this task is the K-mean algorithm, treated in appendix B, and some results are here presented using an algorithm developed using the Image Processing Toolbox of Matlab, see algorithm B.1.
The idea behind this separation of colors is to find a cluster where the flower colors is widely present in a dense way. Actually, this method can be used also for disease or parasite recognition, but this is out of the aim of this work and, therefore, it will be postponed for future searching.
By using this algorithm, it is possible to extract from the image in Fig. 4.9 different pixel clusters based on the different colors in the image.
Before applying this algorithm, the image has to be transformed from the RGB space to the CIE XYZ trimulus values one [81]. This image space is derived by the LMS trimulus values space which is like the human eye sensing. In practice, the eyes are composed by cones that can sense an electromagnetic wave which will be processed as a color by the brain. The eyes' cones feel three kind of wavelength: Long, Medium and Small (LMS) which generally can be associated respectively to red, green and blue colors. These trimulus values are well suited for human

beings, but some wavelengths ranges overlap i.e. the three wavelengths influence each other and they are not linearly independent. This fact could be useful for the colors reproduction because it is possible to create the same sensed color for the human eye by using different wavelength, but this could cause problems to a computer system because the linear dependence of the trimulus values creates some computational difficulties. For this reason, the International Commission of illumination (CIE) has created the XYZ trimulus image space which is composed by three linearly independent values. In particular, the Y axis represents the radiance or luminance, i.e. the brightness, the Z axis characterize the most of the blue wavelengths while the x axis is correlated to the other wavelengths. In this way, the XZ plane is responsible for the color tonality while the Y axis regulate the brightness.

Figure 4.9. Image of a zucchini plant with flowers

This fact introduces also other pros for our work, because, first of all, the colors can be searched into a two dimensional space which do not consider the brightness of the color. Therefore, the same color with low and high luminance could be perceived in almost the same way by the camera; this is not true in the reality because if the brightness gap is too big the pixel colors could be sensed as different. The second and more important pro is that by using this image space, the colors are more compact. Indeed, for example, as reported from [45], the orange hues expressed in the RGB reference frame has a sparser composition than the one expressed in the XYZ one. Based on these strengths, the XYZ trimulus values have been chosen to simplify the computational work.
Returning now to our problem, once that the colors has been converted, the K-mean

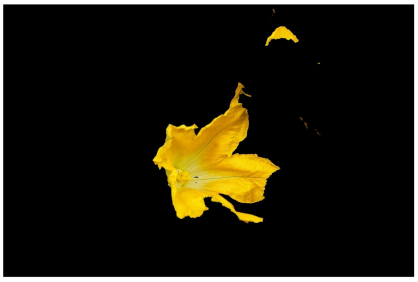
algorithm is applied. It divides the images, e.g. Fig. 4.9, into a priori decided number of different color regions. As can be seen in Fig. 4.10 the number of region decided is three and generally for a greenhouse with a small color noise it could be sufficient, but in section 6 this will be refuted. It is possible to see how the original image (Fig. 4.9) is decomposed into three different region labeled with a different color in Fig. 4.10a. This labeled image allows to separate the clusters on the original image by using a support mask which is a $1,0$ entries matrix used to choose the pixels which will be considered (if $p_i = 1$) despite of the ones that must be discarded (if $p_i = 0$). Using these masks the algorithm is able to divide the three clusters as can be seen in 4.10. In particular, for this explanatory example, the three clusters are: Fig. 4.10b where the green part of the plant is present, Fig. 4.10c where the environment is depicted with other unrecognized color objects and, finally, Fig. 4.10d where the zucchini flowers are perfectly represented. It is important to be noticed how in Fig. 4.10b there is a unripe zucchini flower which is rightly assigned in cluster 1 (leaves) and not in cluster 3 (ripe zucchini flowers).



(a) Zucchini plant image divided into labeled regions. Each color represents a different cluster region on the original image

(b) First cluster extracted from Fig. 4.9 characterized by the leaves and green part of the plant

(c) second cluster extracted from Fig. 4.9 characterized by the environment and other unrecognized color regions



(d) Third cluster extracted from Fig 4.9 characterized by the zucchini flowers

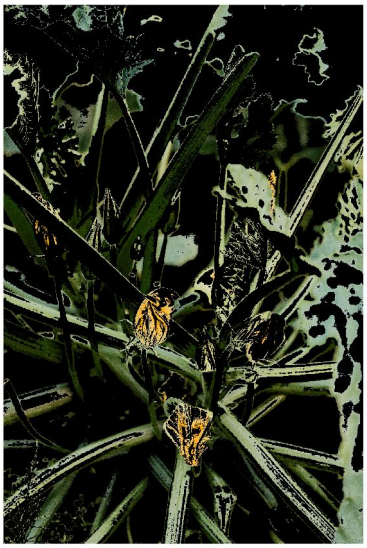
Figure 4.10. Zucchini image processing with clustering decomposition in **three** different clusters

Figure 4.11. Color noisy zucchini plant image

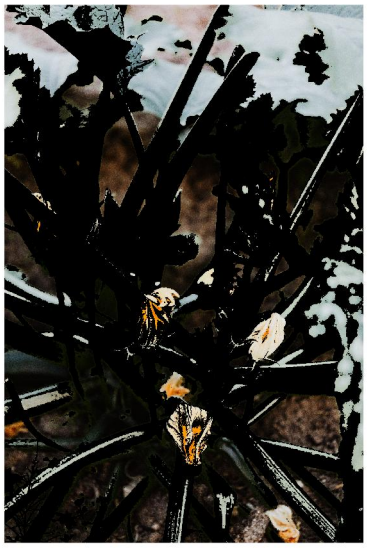
For a color noisy greenhouse, the choice of three clusters could be not enough but in that case the clusters number can be increased depending on the quality of the photos that the robot is able to take. See for example the image in 4.11, the brightness conditions and the different green hues of that plant create some problem for the three groups clustering in Fig. 4.12 while the choice of five clusters presented in 4.13 seems to work properly.

In general, the duty of the farmer, who monitors the greenhouse, is to set the plant growth in such a way that the picking task is simplest as possible or, at least, affordable for the robot. There exist different techniques already used in greenhouse agriculture where the plants are forced to grow in such a way that the pickers' work is simplified. The same approach can be used in this case to increase the possibility of avoiding situations in which the robot encounters some difficulties that could impede the gathering task.
Once that the clusters are all obtained, it is enough to analyze each of them looking for the colors depicted in Fig. 4.8. This operation is simple, the robot scans each

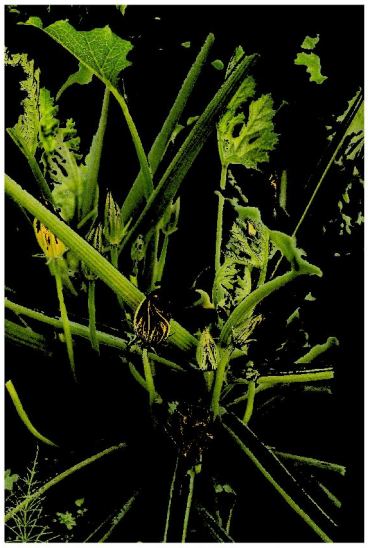
(a) Zucchini plant image divided into labeled region. Each color represents a different cluster region on the original image



(b) First cluster extracted from Fig. 4.11

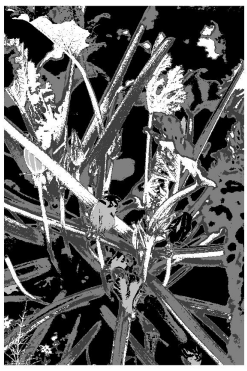


(c) second cluster extracted from Fig. 4.11

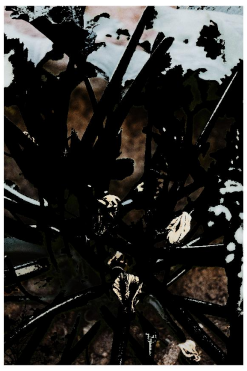


(d) Third cluster extracted from Fig 4.11

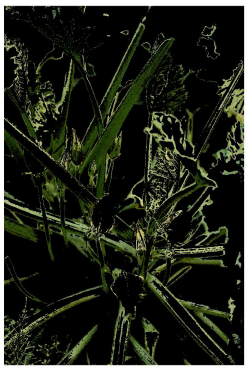
Figure 4.12. Color noisy zucchini image processing with clustering decomposition in three different clusters



(a) Zucchini plant image divided into labeled regions. Each color represents a different cluster region on the original image



(b) First cluster extracted from Fig. 4.11



(c) second cluster extracted from Fig. 4.11

(d) Third cluster extracted from Fig 4.11

(e) fourth cluster extracted from Fig 4.11

(f) fifth cluster extracted from Fig 4.11

Figure 4.13. Color noisy zucchini image processing with clustering decomposition in **five** different clusters

cluster by comparing the images RGB values with the ones presented before. If in one cluster there is a big pixel percentage belonging from the chosen RGB range, then there is an high possibility that at least a flower is present. This step is even more useful for the task in section 4.3.3 because in this way the robot is able to identify the flower cluster and use it for the ripeness control and separation.
For this task only the presence check is needed, indeed this algorithm is enough and the robot can pass to the next step of the algorithm which is the approach and the ripeness identification.

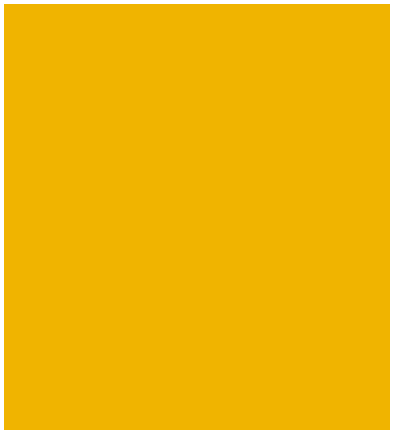
4.3.3 Flower ripeness

The algorithm for the ripe flowers selection is like the one in section 4.3.2 because also the flower ripeness can be detected trough the color. The only difference is that now the camera is very close to the plant and can take more precise pictures of the surrounding flowers. Moreover, when the distance between the plant and the camera decreases, the efficiency of the flash improves, resulting in a more limpid and shining image. Indeed, the range of the pixel intensity values for the flower ripeness must be re-tuned. This time the RGB values are:

- a value bigger than 240 for the R component
- a value contained in $[180, 255]$ for the G component
- a value inside $[0, 40]$ for the B component

In this case, the constraint between the red and green values does not have to be added because if the R component assumes very high values it does not matter if the green component exceeds with respect to the red one, the yellow intensity will be always acceptable and the maturity of the flower is assured.

(a) Maximal RGB color for right ripeness evaluation of the flower on the plant. RGB value $(255, 255, 40)$



(b) Minimal RGB color for right identification of the flower on the plant. RGB value $(240, 180, 0)$

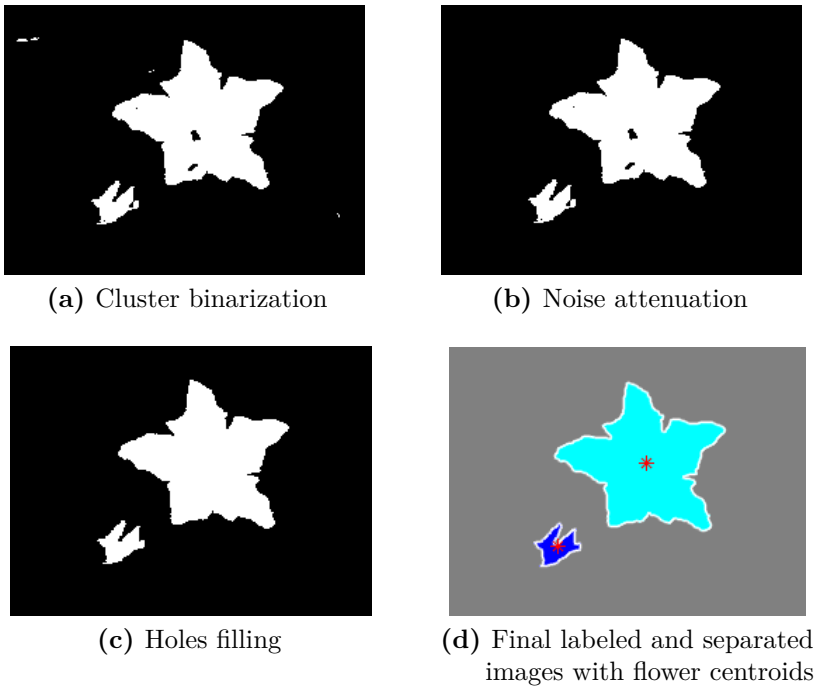
Figure 4.14. Minimal and maximal RGB colors for flower ripeness

As can be seen with the comparison between Fig. 4.8 and 4.14, the colors depicted in the first images are paler while the second ones are brighter. This choice depends mainly on the distance between the robot and the plant because with a greater distance the quality of the photograph is slightly worst and it can be misleading for some colors. However, when the robot is close to the plant, the photos are more like the reality. This sharpness allows to identify the colors and then to separate the yellows hue of the flowers by some possible plant yellow leaves. In this way, the flower misidentifying is improved and the robot can save time and battery for the next flower or plant.

Figure 4.15. Original flower cluster extracted with the K-mean algorithm

The ripeness recognition algorithm is like the identification presented in section

4.3.2. The only difference is that now the flower recognition has a more important role because now, it is necessary to distinguish the flowers from the plant and the environment to extract more information from the selected plant.



(a) Cluster binarization

(b) Noise attenuation

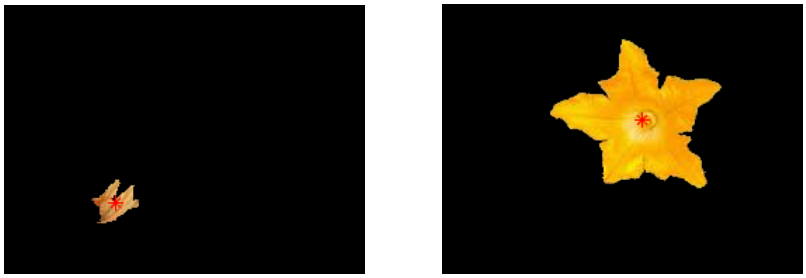
(c) Holes filling

(d) Final labeled and separated images with flower centroids

Figure 4.16. Segmentation of cluster in Fig. 4.15

Once that the flower cluster has been founded (in the same way of the flower identification), the block starts with the second operation, the segmentation and the listing of the possible flowers' centroids. In this way, a list of (x, y) values of the camera image plane are obtained and they will be used for the 3D localization of each single flower. For this, a flower segmentation algorithm has been implemented to separate and to identify each flower, see algorithm C. At each flower is assigned a centroid and an area inside a border, called mask. This algorithm is used to obtain a flowers separation to simplify their 3D localization.
The algorithm, like the Euclidean clustering algorithm, is presented in appendix C while here some results and observations are reported.
The segmentation algorithm is applied to the clustered flower image extracted with the K-mean algorithm (B.1), see Fig. 4.15. It starts with the image binarization, see Fig. 4.16a, to obtain a black and white image. Black and white images, due to their simplicity, improve the computations and consequently also the computational time of the segmentation algorithm. Once that the image has been binarized the noise is attenuated i.e. the outliers under a certain size range in the image are cancelled, see

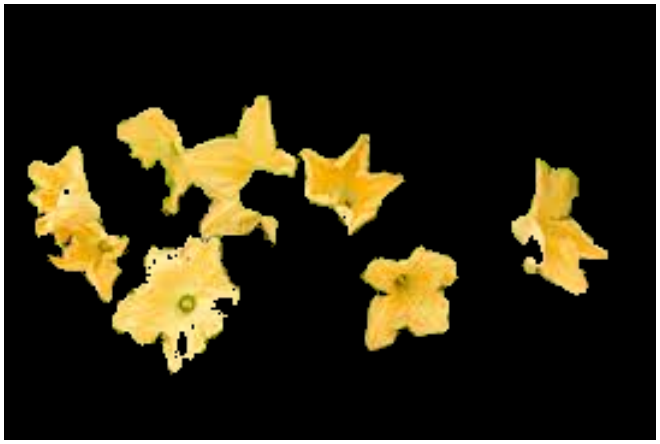
Fig. 4.16b. Subsequently, the holes inside the image closed bound are filled to have different flat surface without black holes inside of them, see Fig. 4.16c. Finally, the different white surfaces present in the image are labeled and separated and their centroids are computed with a simple pixel position mean.



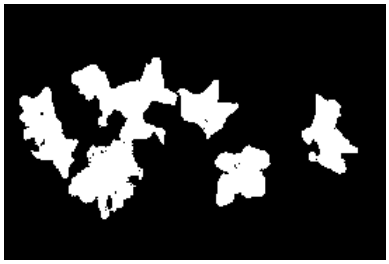
(a) First separated flower **(b)** Second separated flower

Figure 4.17. Flower separation

With this algorithm, it is possible to count the number of present flowers on a plant and separate them also extracting their centroids. This is the first real separation of the flowers and the algorithm can then continue with the 3D localization of the flower stem. In Fig. 4.17 can be seen the final separation of the flowers with the respective centroids.
Even in this case some problems could arise when the image is too much noisy or there are some flowers in the images which are super imposed or too close each other. For example, in the not common but possible case in Fig. 4.18a there is the presence both of superimposed and near flowers.
The results obtained by applying the separation algorithm to this image can be founded in Fig. 4.18. As can be seen from Fig. 4.18b, 4.18c, 4.18d, 4.18e, 4.18f the first two labeled groups are composed not by a single flower but from different of them while the flowers in Fig. 4.18g, 4.18h,4.18i are well separated. To overcome this problem either a post-processing algorithm or an Euclidean segmentation algorithm can be applied to each separated cluster to verify if it is composed by a single flower or a group of them. If the cluster is composed by a group of flowers some identification techniques can be applied to separate them and allow the robot to consider them singularly [53]. This post-processing separation is not considered in this work because the environment is considered ideal, i.e. the flowers are not superimposed or in large groups such as in Fig. 4.18. This is because the flowers are picked almost every day by the robot and it is rare that in just 24 hours the flowers on a single plant flourish all together.

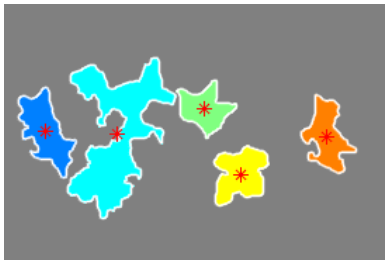


(a) Original clustered image



(b) Cluster binarization

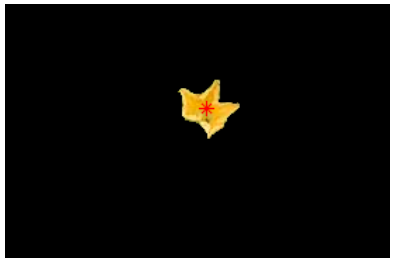
(c) Holes filling



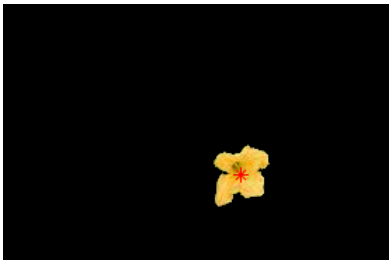
(d) Color labelled image for separated flowers

(e) first flower separated

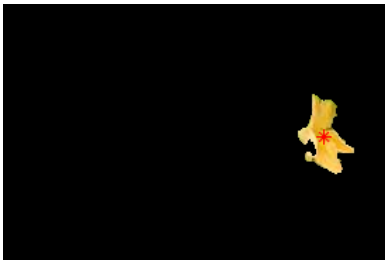
(f) second flower separated



(g) third flower separated



(h) fourth flower separated



(i) fifth flower separated

Figure 4.18. Flower separation

Anyway, this post-processing techniques can be introduced afterward in the code. In the case that the post-processing methods have some difficulties in identifying and dividing the flowers, another solution could be to capture some photos with different point of views and use some more sophisticated techniques to reconstruct the scene and to obtain a perfect recognition and separation, but this would probably be costly in time and money.
Now, the cluster has been separated, the centroid of each flower has been computed and saved in a (X_I, Y_I) array and the robot is ready to localize the flower in the 3D space and accomplish the harvesting task.
Here a single flower ripeness control could be implemented to check each single flower maturity. In this way the robot is sure that the selected flower is ready to be cut but this will also increase the execution time, therefore, this skill should be chosen from the costumers depending on time and cost behaviors. Anyway, the control is the same implemented before: if the pixel colors of the image which are inside the RGB ranges depicted above exceed a chosen percentage, then the flower is ready to be cut.
It is important to be stressed that, in this work, the color of the flower has been considered as a good parameter for the classification of the flowers and to distinguish them from the leaves and the environment. In some cases, this could be not enough and some better classification methods must be used for the classification of the fruits. As an example, there exists in literature many cases where some machine learning algorithm learns different information above fruits and surrounding and they are then able to distinguish and classify them in different classes, e.g. fruit, leaf, stem and others. For our purposes, the simpler approach proposed could be considered enough because also this modification can be made at a later stage.

4.3.4 3D localization

The information used until now for the flower identification and raw localization was based only on RGB data. As introduced in section 4.3.1, an RGB-D camera has been used to extract also depth information of the surrounding. For the flower 3D localization, these information are essential because the projection of the 3D space onto the 2D image surface implicitly cause a loss of information, especially regarding the world space depth. Thanks to the depth information the robot can approximately reconstruct the 3D environment or, at least, sense that information which are essential for the picking task.
The flower 3D localization is used in two following steps; the first one is the manipulator approaching movement through the plant which is characterized by the obstacle recognition and avoidance and the approximately placement of the

end-effector close to the flower. The second, instead, is the real picking task, i.e. the robot end-effector is pointing the flower, the image processing is creating a 3D shape of the flower assigning to it a position and orientation in the 3D space and then, once the final pose has been found, the robot accomplish the task.
The first step starts with the extraction of one of the centroids from the list created in the previous step, see section 4.3.3. Each centroid (X_I, Y_I) is associated with an area enclosed by its border. The aim of this step is to approach the flower in a close enough standardized position from which the robot can detect the flower and localize the stem. Thanks to the depth sensors and the centroid knowledge the robot can localize the flower position in the 3D environment, see section 4.3.4. First of all, the RGB images resolution must be shrunk to fit the resolution of the depth cam (generally, the resolutions are different, e.g. RGB camera in full HD while depth camera is in low resolution). This passage is necessary because, otherwise, the different images resolutions will cause a wrong estimation of the flower localization. Once that the image has been shrunk a simple 3D position estimation can be applied using the formulas:

$$z_\epsilon = D(X_I, Y_I) - \eta \tag{4.41}$$

$$x_\epsilon = \frac{z_\epsilon (X_I - C_x)}{f_x} \tag{4.42}$$

$$y_\epsilon = \frac{z_\epsilon (Y_I - C_y)}{f_y} \tag{4.43}$$

Where $D(X, Y)$ is the depth value sensed from the depth cam in the pixel position (X, Y), C_x and C_y are the coordinates of the principal point on the 2D image plane, generally these coordinates represent the image center and f_x and f_y are the camera focal lengths. In general, C_x, C_y, f_x and f_y are estimated using the camera calibration introduced in section 4.3.1.
The real flower position estimation is the one with $\eta = 0$, but in this first part the camera should only go close to the flower and not to completely approach it. Therefore, η is set to be 30/40 cm before the 3D estimated position, thus the camera is quite close to the flower, but it is also able to perform the last 3D localization.
Once that the position error for the camera has been found, $X_\epsilon = [x_\epsilon \; y_\epsilon \; z_\epsilon]^T$, also the desired orientation of the camera with respect to the flower is computed. Since the vector X_ϵ point at the identified flower center, the most intuitive way of doing it is to rotate the camera in such a way that the z axis of the camera is directed in the same direction of the flower vector. To do this operation a simple way is to transform the vector Cartesian coordinate in its spherical ones [83], see Fig. 4.19 and then rotate the camera reference frame over the vector using the angles just

found.
The Cartesian vector $[x\ y\ z]^T$ is transformed in (r, θ, ϕ) where r is the magnitude of the vector, i.e. its length, θ is the angle between the vector and the z axis of the reference frame of application and ϕ is the angle between the projection of the vector over the XY plane and the x axis of the reference frame.
The two variables of interest are θ and ϕ because performing a rotation around the z axis of ϕ radiant and then performing another rotation around the new reference frame y axis of θ radiant the z axis of the reference frame is perfectly aligned over the considered vector. The spherical coordinates are computed with the following formulas:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{4.44}$$

$$\theta = arctan\left(\frac{y}{x}\right) \tag{4.45}$$

$$\phi = arccos\left(\frac{z}{r}\right) = arctan\left(\frac{\sqrt{x^2 + y^2}}{z}\right) \tag{4.46}$$

Now the rotation to be performed are:

1. The rotation of ϕ around z axis

$$R_z(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.47}$$

2. The rotation of θ around the new y axis named y'

$$R_{y'}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{4.48}$$

With these two rotation matrices the final rotation can be found performing the matrix multiplication

$$R_\epsilon(\phi, \theta) = R_z(\phi)R_{y'}(\theta) = \begin{bmatrix} \cos\phi\cos\theta & -\sin\phi & \cos\phi\sin\theta \\ \sin\phi\cos\theta & \cos\phi & \sin\phi\cos\theta \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{4.49}$$

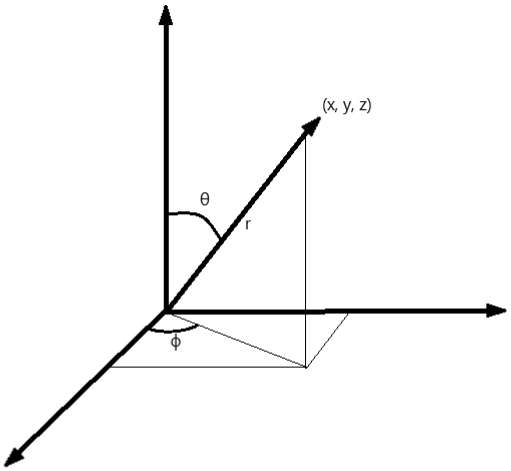


Figure 4.19. Spherical coordinate of a Cartesian space vector

Finding in this way the final homogeneous transformation:

$$T_\epsilon = \begin{bmatrix} R_\epsilon & X_\epsilon \\ 0 & 1 \end{bmatrix} \tag{4.50}$$

The homogeneous transformation in 4.50 represent the error that the camera has to recover to be in the position and orientation desired and it can be applied directly to the end-effector for two main reasons:

- The error vector X_ϵ is a free vector and can be applied to every point in the space

- The camera and the end-effector has always the same orientation in the space

Then, the error homogeneous transformation T_ϵ can be applied directly to the end-effector.
Now that the approaching position and orientation are known, the robot move close to the flower with the kinematic control presented in section 4.4.2. To use this control technique, the position and orientation information inside the homogeneous

transformation T_ϵ must be reduced in a minimal representation of the form

$$\bar{x}_\epsilon = \begin{bmatrix} x_\epsilon \\ y_\epsilon \\ z_\epsilon \\ \theta_\epsilon \\ \phi_\epsilon \\ \psi_\epsilon \end{bmatrix} \tag{4.51}$$

The first three component of 4.51 are exactly the same of X_ϵ while the others should be extracted from R_ϵ matrix using the ZYZ Euler angles as explained in section A.1. The error state $\bar{x}_\epsilon$ has a bar over the x only to avoid confusion between end-effector state and its desired coordinates.

This kind of control can be used because, as explained in section 4.3.2, the robot works in an ideal environment, also thanks to the farmers which set up the plants to avoid obstacles between the flowers and the robot.

Once that the robot has approached the flower, the camera capture another image, like the one in Fig. 4.20a, which is used to extract the centroid of its stem and its borders, see Fig. **??**. This is the second task of the 3D localization. It is the most difficult task of 3D localization because now the robot has a small error range due to the delicate picking task. There exists different solution in literature most of which use convolutional neural networks and other techniques to extract from the image the stem position and orientation. For simplicity, in this work, the stem position estimation is simplified by highlighting the flower stem on the picture, see Fig. 4.20b. In this way, the robot is able to extract the position information from the image, using the RGB-D camera in the same way of the flower clustering in section 4.3.2. Indeed, by applying the same algorithm B.1, the robot is able to recognize the flower stem and extract the information of its centroid, see Fig. 4.21.

(a) Zucchini flower image after the manipulator arm approach on plant in Fig. 4.9

(b) Approached plant of Fig. 4.9 with the stem highlighted in red

Figure 4.20. Highlighting of the flower stem to simplify the estimation of the stem position and orientation

(a) Labeled clustering of Fig. 4.20b

(b) Segmented cluster containing the stem of the flower in Fig. 4.20b

(c) computation of the centroid and the borders of the zucchini flower stem in Fig. 4.20b

(d) Final stem extracted with its centroid in Fig. 4.20b

Figure 4.21. Segmentation of the zucchini flower stem in Fig 4.20b

Finally, the picking task can be performed by simply directing the robot end-effector through the stem centroid and grasp the flower. Finally, after the cutting of

the flower stem, the robot put the picked flower in the box and return to its home position either to pick another flower or to continue to the next algorithm location.

4.4 Motion planning and control

As it is clear from the previous section 4.3, the motion planning of the robotic arm need to be coordinated with the visual recognition system. In fact, although the two blocks are generally considered separately in literature, they are strictly related, because, without the visual recognition, the robot arm is blind and cannot perform its task. The robot has to perform different movements depending to the goal point and the orientation of the end-effector. The feedback control loop uses only joint position information, indeed the technique presented is not a visual servoring methods but it uses visual information only to estimate the flower position and orientation to perform the movement. This method could result in a more damaging movement because the robot is not considering the obstacle in its path, but this risk is reduced because the robot works in a ideal environment where the plants are positioned in such a way to give a free workspace and a simple picking position to the robot. Moreover, the zucchini plants are not rigid, but they are very flexible and this reduce the plant's parts breaking in case of collision.

4.4.1 Trajectory planning

The information extracted with the vision system provides an error reference useful for the trajectory planning. Starting with that data, the robot creates a path $x_d(s)$, where s is the path curvilinear abscissa, and then creates a timing low $s(t)$ depending on the distance and the robot speed constraints.

Path planning

Starting with the information found in section 4.3.4, (for example error state vector 4.51) the path planner in the robot central unit builds a path of the form $f(s)$ using as dependent variable the curvilinear abscissa s. In this way, the path planning and the timing low for the trajectory can be essentially decoupled and can be changed almost independently to each other. Since the path between the end-effector position and the goal position is considered obstacle free and the distance to be traveled is not so much, the path takes a linear behavior which means a point-to-point segment. The position and orientation paths will be considered separately. Starting with a general error state vector, i.e. a 6×1 vector of the form $\bar{x}_\epsilon = [x_\epsilon \ y_\epsilon \ z_\epsilon \ \theta_\epsilon \ \phi_\epsilon \ \psi_\epsilon]^T$, The position path $p(s)$, i.e. the path for the first 3 rows of the error state vector, is

computed as:

$$p(s) = p_i + s\frac{p_\epsilon}{\|p_\epsilon\|} \tag{4.52}$$

Where p_i is the initial position of the path, generally equal to the current position of the end-effector, and p_ϵ are the first three components of $\bar{x}_\epsilon$. With 4.52 a linear path, dependent on the variable s, has been created, starting from the current position of the end-effector and finishing in the goal point. As described in section 4.4.2, the closed loop reference is the desired joint velocity, therefore a $\dot{p}(s)$ velocity reference is needed for this scheme. That velocity can be computed deriving directly $p(s)$ with respect to s. Indeed, the velocity reference path is:

$$\dot{p}(s) = \dot{s}\frac{p_\epsilon}{\|p_\epsilon\|} \tag{4.53}$$

For the orientation a similar approach can be used, the only attention should be given to how the end-effector angles and the goal ones have been expressed. In this work, the choice is to use the Euler ZYZ angles, see appendix A.1, therefore both the current state vector and the desired goal vector orientations are expressed in this form. If this angle structure is maintained, it is possible to build an orientation path reference $\Phi(s)$ of the form:

$$\Phi(s) = \Phi_i + s\frac{\Phi_\epsilon}{\|\Phi_\epsilon\|} \tag{4.54}$$

Where Φ_i is the current orientation of the end-effector expressed in ZYZ Euler angles and Φ_ϵ is the orientation error state vector, i.e. the last three components of $\bar{x}_\epsilon$. In the same way of 4.53, the "angular velocity" $\dot{\Phi}(s)$ is computed as:

$$\dot{\Phi}(s) = \dot{s}\frac{\Phi_\epsilon}{\|\Phi_\epsilon\|} \tag{4.55}$$

In this way, the path planning is completed and a path of the form:

$$\dot{x}_d(s) = \begin{bmatrix} p(s) \\ \Phi(s) \end{bmatrix} \tag{4.56}$$

has been found.

Timing law

Once that the path has been planned, the last necessary operation to conclude the trajectory planning is the timing law derivation. This law allows a smooth movement from the initial to the final point, maintaining the robot joint velocities below its speed constraints. Here, a generic approach will be used, but once that the

robot velocity specifications or the ones of its joints are known, this approach can be modeled depending on them. Let's start by looking at the curvilinear abscissa function of time $s(t)$. Generally, it is represented as a single timing law $s(t)$ but in reality it is a vector composed by two different timing laws, one for each component of 4.56. To distinguish them, two different curvilinear abscissa will be here introduced; the position curvilinear abscissa $s_p(t)$ which is the one connected to path 4.52 and the curvilinear abscissa $s_\Phi(t)$ which is the one expressed in 4.54. These two variable vary in the two different range: $s_p(t) \in [0, \|p_\epsilon\|]$ and $s_\Phi(t) \in [0, \|\Phi_\epsilon\|]$. In particular, the left extreme values are linked to $t = 0$ while the final ones are linked to $t = T$, where T represent the ending time of the trajectory. The curvilinear abscissa $s(t)$ could be also normalized to obtain a variable $\hat{s} \in [0, 1]$ but this is not strictly needed and generally it is done only to make the computations clearer. The relation between this two variables is

$$s(t) = \frac{\hat{s}(t)}{l} \tag{4.57}$$

where l represent the path length either in meters for position or radiant for orientation. To make things as simple as possible, a single general case is proposed because the application for position or orientation is straightforward.
There exists different types of timing laws such as polynomials, linear-parabolic and others but, for simplicity, a trapezoidal timing law for the velocity references has been chosen, see Fig. 4.22. The aim is to find the variable $\hat{s}(t)$ for a generic path $p(s)$ which meets the robot velocity and acceleration constraints, i.e. $\dot{p}(t) < V_{max}$ and $\ddot{p}(t) < A_{max}$, and accomplishes the task. To make it possible, some data should be available, i.e. the robot should know in advance:

- the length l of the path
- the maximal robot velocity V_{max}
- the maximal robot acceleration A_{max}

It is important to stress that the robot maximal velocity and acceleration are dependent on the maximal velocities and accelerations of the joints, but generally an upper bound of these values is always possible to be estimated. Anyway, These information are needed to compute the time values T_s and T reported in Fig. 4.22

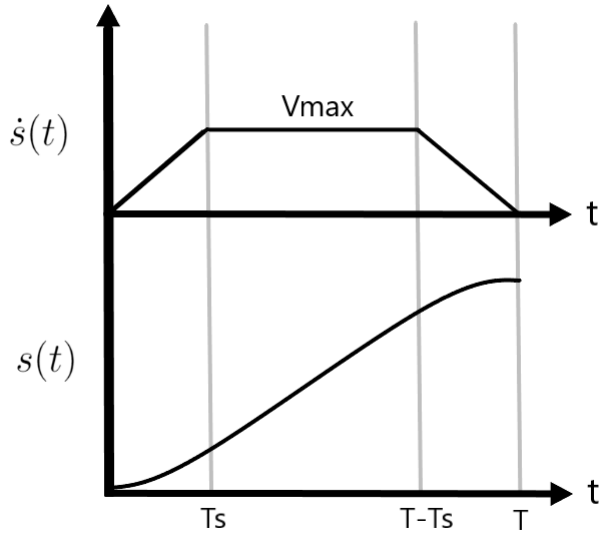


Figure 4.22. Trapezoidal behavior for a generic timing law

These time values are computed with:

$$T_s = \frac{V_{max}}{A_{max}} \tag{4.58}$$

$$T = \frac{l\ A_{max} + V_{max}^2}{A_{max} V_{max}} \tag{4.59}$$

In this way using the time values T_s and T the timing law for the curvilinear abscissa is:

$$s(t) = \begin{cases} A_{max} \frac{t^2}{2} & t \in [0, T_s] \\ V_{max}\ t - \frac{v_{max}^2}{2\ A_{max}} & t \in (T_s, T - T_s) \\ -\frac{A_{max}(t-T)^2}{2} + V_{max}\ T - \frac{V_{max}^2}{A_{max}} & t \in [T - T_s, T] \end{cases} \tag{4.60}$$

Finally, by deriving the timing law in 4.61 the velocity reference timing law $\dot{s}(t)$ is obtained.

$$\dot{s}(t) = \begin{cases} A_{max}\ t & t \in [0, T_s] \\ V_{max} & t \in (T_s, T - T_s) \\ -A_{max}\ (t - T) & t \in [T - T_s, T] \end{cases} \tag{4.61}$$

Now, substituting the position and velocity timing law here found in the previous path, a trajectory is obtained and it can be used as a reference for the kinematic control presented in section 4.4.2

4.4.2 Kinematic control

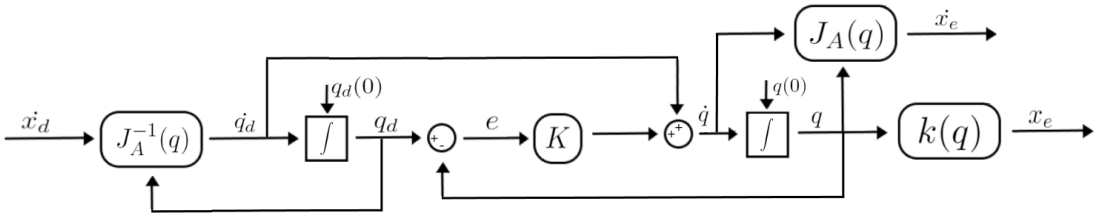


Figure 4.23. Block scheme for the joint kinematic control strategy

As can be seen from Fig. 4.23, a simple kinematic control strategy has been used to perform the robot movements. This control strategy can be applied because most of the industrial robots already have some pre-installed low-level controller for the dynamic control of the robot, i.e. controller for the dynamic relations of the robot (velocity and torque). Thanks to this simplification, the users can control the robot directly using its kinematic and differential kinematic structures considering the low-level control loop of the robot as a simple integrator (in Fig. 4.23 the right integrator).

This control loop has a very simple strategy, it is based on two principal components: a feedforward component $\dot{q_d}$ and a feedback component q. The first one is derived directly from the trajectory reference thanks to the relation in 4.36 and is characterized by the ideal reference that the end-effector has to track. While the second component is extracted by the absolute encoders present on the robot motors and it is necessary for the recovering of the possible initial and transient errors.

Also the initialization of the algorithm is important, indeed the $q_d(0)$ an $q(0)$ values should be previously initialized to have a small initial error to reduce the transient time.

Chapter 5

Real prototypes

In this chapter different solutions for existing picking robot will be presented, see section 5.1, and also some original prototype ideas will be proposed, see section 5.2. The aim of this chapter is to give an idea of the current state-of-the-art for the picking robot currently available and present some new solutions which could complement the existing ones.

5.1 Existing robots for picking tasks

There exists different solution and prototypes for autonomous picking robot and some of them can be bought on the market. Anyway, the robot components and its task performances are not always compatible with the specific task to perform. The robot choice is always tricky and many variables must be considered.
For sake of completeness, the most known robot will be here introduced with a small presentation. One of the most famous autonomous picking robot is the *Sweeper robot* [74] which is the result of an European project developed from different universities belonging to Netherlands, Belgium, Israel and Sweden. This robot can move inside a greenhouse over some rails and is able to recognize, reconstruct and pick sweet peppers.
Iron Ox [24] is an American hydroponic greenhouse which is fully automated, from the seeding to the harvesting of the plants. This is the first start-up which has been able to achieve this automation level marking the history for the agriculture future. This company has been already selling its products in California since 2018 allowing for a non-expensive and fresh products for the American population.
Rubion [54] is a strawberry picking robot developed by the Octinion company. It is an interesting robot able to pick strawberries without bruising them, resulting in a fresh crop. This robot is autonomous and can detect and monitoring the harvest thanks to its sophisticated quality monitoring.

Farmbot [23] is a very interesting open-source Cartesian robot which is able to perform every single operation autonomously. It can seed the plants, waters and monitoring them, removes weeds and perform other operation. This is possible thanks to the exchangeable end-effector. There is also a useful application to monitor the behaviors and to program the robot, but it is not suitable for very large crop areas which is our case of interest. Anyway, this robot is really interesting for the self-sufficiency and deserve to be cited.
Agrobot [4] is a company which creates harvesting and agricultural robot. Their *E-Series* is composed by some big autonomous mobile configurable robot which are able to detect and crop strawberries in different ways. Due to its size and the different arms mounted on it, this robot is able to harvest lots of fruit in a small period, but the size not always can be afforded into a narrow environment like the greenhouses ones.
Root AI [5] is a very interesting company which creates autonomous harvesting robots. A new solution of this company is the *Virgo* robot which is a tomato picking robot characterized by a real-time detection algorithm, a gentle touch of the fruits, an intelligent motion for the picking and a fast picking process. This agency is building different and innovative solutions for the agricultural future.
The *Green Robot Machinery* company is creating an autonomous robot for the cotton harvesting [27] which is able to identify, approach and pick each cotton tuft. This agency is currently developing also other solutions for horticulture crops.
FFrobotics [25]company is also creating harvesting robot for fruit picking, especially for apples or citrus and other tree's fruits. Their solution is interesting because the machine gathers the fruit and store it temporarily inside a box attached to the end-effector and then storage them in another bigger compartment which separates them.
Meteomotion [51] is another interesting solution for tomatoes picking. it is composed by a mobile platform, two robotic arms, two 3D cameras and an autonomous boxing machine. This robot is built exactly for greenhouses picking tasks, is a multi-purpose robot, i.e. not only for picking task, and, such as *Sweeper robot*, it has received European funding for the development. It has been applied only on tomatoes plantations, but it could probably applied to different ones.
The last solution presented is the *Dogtooth* [43] ones. This company is creating interesting harvesting robots for soft fruit picking although the company is at its origins. Some solutions for strawberries picking have been presented but different works are still in progress.
The here-presented companies start from the same ideas, i.e. create an harvesting robot, and then spread over different solutions for the different crops. Some of them

could be also suitable for the zucchini flower picking task presented in this work; only some modification should be performed in the recognition and picking tasks, but the robot structure and its control are already quite stable. Anyway, a new solution, the one presented through the thesis, is presented in 5.2 which could be created with the composition of existing solutions and other original ones i.e 3D printed and assembled.

5.2 Prototype

In this section a prototype for the autonomous zucchini flower harvesting robot is presented. The aim is not the construction of the whole real robot, but it is the presentation different solutions which, ones assembled, could create the robot. For the real construction of the robot, one of the presented solutions have to be chosen for each component and applied with the other ones taking into account the differences of the parameters and the structures. Although these solution are valid for a simulation of the project, in section **??** another solutions built from zero will be used to simulate the whole process presented in the thesis and to complete the work. The aim of the presentation of these multiple solutions presentations is to give at the reader different modalities to achieve the same goal. In this way, each user could choose the robot parts which best fits with their necessities.

5.2.1 Mobile Platform

The first robot part considered is the mobile platform; this carrier should be characterized by three main features:

- The possibility of smooth behaviours with a low position error
- A good stability to not capsize when the robotic arm is moving and/or stretching.
- The right support capacity to sustain the weight of the robotic arm, the collecting boxes and all the other components (batteries, central units, ecc)

Once these three characteristics are met, the mobile platform can be chosen.
The first platform presented is the *4WD Mecanum* Wheel Mobile Robotic Platform produced by King Kong Robot[60], see Fig. 5.1. This platform is a four Mecanum wheel mobile platform driven by four separated 12V DC motors with optical encoders and it is comprehensive of a 12V battery and charger, four ultrasonic sensors (one for each side), an IO expansion and an Arduino ATmega328. Its dimensions are

really contained (400mm x 307mm x 123mm), but it is characterized by a good load capacity (10kg) which is enough for the arm and the light-weght zucchini flowers.

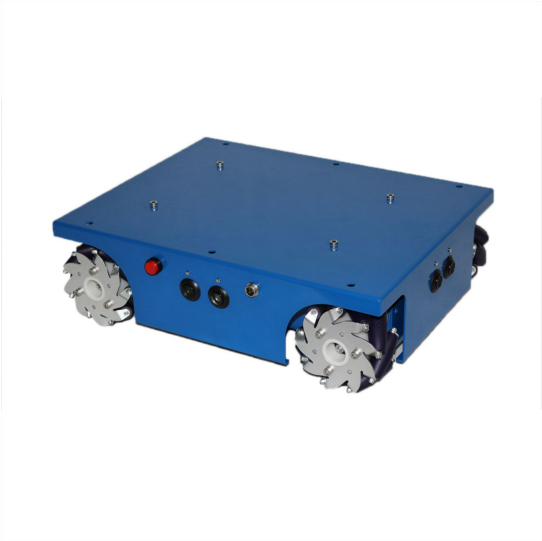


Figure 5.1. *4WD Mecanum Wheel Mobile Robotic Platform* produced by King Kong Robot[60]

Another small solution could be the *Light duty Mecanum wheel mobile platform OMR10* presented by the Omni Mechanical Technology company [14], see Fig. 5.2. This four Mecanum wheel robot can work for 8 consecutive hours with a full charged battery. The charging time is estimated approximately to be 4 hours. It can be commanded through wireless communication, it can support 20kg of weight and its dimensions (480mm x 450mm x 134mm) allow to perform the zucchini flower picking task.

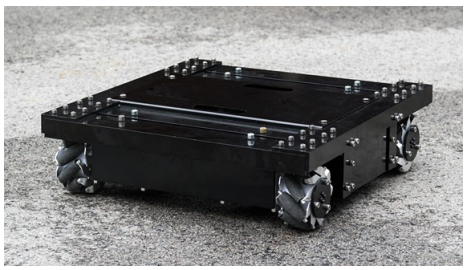


Figure 5.2. *Light duty Mecanum wheel mobile platform OMR10* presented by the Omni Mechanical Technology company [14]

The small dimensions of these two presented robots allows to perform zucchini flower picking tasks for small greenhouses or at most medium ones otherwise a multiple robots cooperation solution should be used to work in a big greenhouse and each robot can either have an area assigned to work in, or they can cooperate in a

distributed fashion to schedule the tasks and work area.
There exists also bigger platforms with different characteristics, but they are more costly and in general they are too big to be used in a greenhouse, e.g. *Mecanum wheel mobile lift platform AGV–OMR06* of Omni Mechanical Technology company [15].
The best choice for the platform should be to build a Mecanum wheel robot autonomously, choosing each component separately using an aluminum structure to create a robotic carrier with the right dimensions and characteristics for a hydroponic greenhouse. Obviously, this could be costly and time consuming but the best industries which has adopted this solutions are the ones which are producing lots of profits.
Anyway, some solutions which can be adopted in this task exists and one which fit with this project is the *Mobile Robot MPO-500* produced by Neobotix[50], see Fig. 5.3. The platform dimensions (986mm x 662mm x 409mm) are suitable for most of the fruit and vegetables greenhouse productions, it is able to move freely for 7 hours or 3 km with a single charge, it has different localization sensors and different optional features, e.g. the automatic charging station, are available. Moreover, some laser scanners and other features can be added, and they could be useful in autonomous greenhouses.

Figure 5.3. *Mobile Robot MPO-500* produced by Neobotix[50]

5.2.2 Robotic arm

Regarding the robotic arm, different solutions are present on the market, but only few of them will be presented because they cover the needed specifics for the arm.
The robotic arm considered is the *Gen3* robotic arm produced by **Kinova**[38], see Fig. 5.4a which is a good 6 DOF robotic arm with different characteristics and accessories. This manipulator arm is a lightweight robot whit a payload of $2kg$ for a completely stretched position or of $4kg$ for a middle-stretched position. The zucchini flower weight is negligible compared to the payload of the robot, but this

arm could be used also for different topics. This robot has a reach of $902mm$ from its basis which is a good enough distance for our task. It has a maximum speed of $50\frac{cm}{s}$ and it is compatible with most of the operative systems and programming languages. There exists also a light configuration of the robot, see Fig. 5.4b, which is more lightweight, has a minor reach region ($706mm$) and can lift objects which are lighter, i.e. $0.5kg$. Moreover, the *Gen3* configuration provide a depth camera as the one presented in section 4.3.1 which can be positioned exactly on the last joint of the robot close to the end-effector, i.e. it is an eye-in-hand camera.

(a) *Gen3* robotic arm produced by **Kinova**[38]

(b) *Gen3Light* robotic arm produced by **Kinova**[38]

Figure 5.4. Kinova 6 DOF Robotic arms

5.2.3 Vision camera

The vision camera, as explained in 4.3.1, is an RGB-D camera. This kind of cameras should be lightweight and high resolution to perform perfectly the assigned task. The Kinova company chosen for the robotic arm, see section 5.2.2, provides an RGB-D camera which can be mounted on the robot end-effector and the output of the camera can be read directly using the connectors of the robotic arm. This solution is very useful for the project because otherwise another solution less elegant should be used e.g. connect a Kinect camera [86] or similar to the end-effector and, if it is not possible to pass the cables inside the robot structure, them should be tied on the external shield of the robot, resulting in a inelegant and dangerous situation.

(a) Kinova RGB-D camera mounted on the end effector of the *Gen3* robot

Figure 5.5. Kinova RGB-D camera mounted on the end effector of the *Gen3* robot

5.2.4 Others

There are also other components which should be chosen to build the real prototype of the robot:

- the battery
- the box used to collect the flowers
- the control units
- the electronics
- the charging station and charging mode

All these components are equally important for the whole project and without them a real prototype could not be invented, but in this work, they are not considered due to the simulative nature of the project.

Chapter 6

Simulation

To verify and demonstrate the functionalities and the effective feasibility of this project, a simulation, which use most of the main concepts presented, has been developed. For the simulation, two main different tools have been used: Matlab [84] and Coppelia Sim [2]. The first one has been used as the main controller of robot, in fact, all the controllers and machine learning algorithms have been developed in Matlab, see section 6.1. Instead, Coppelia Sim, also known as Vrep, see section 6.2, represent the real simulation environment where the robot prototype and the greenhouse have been developed and simulated. Another program used is Thinkercad [85], which is an online CAD used to develop some more peculiar 3D model for the simulation like the flower model used. The other models, the robotic arm and the mobile platform, have been developed directly on Vrep with the software tools.
The communication between the two software, Matlab and Vrep, has been performed thanks to the *Remote Api* files developed by Coppelia Robotics [2]. These files allow to open a synchronous or asynchronous server communication between the two software and different type of operative mode for the data communication and different pre-built functions [3] which are very useful for the programming can be used.

6.1 Matlab

As previously introduced, Matlab is the main core of the simulation. With Matlab, most of the computations and controllers of the simulation has been performed. Indeed, the main algorithm presented in chapter 2 has been implemented on this tool. The algorithm parts developed with Matlab are the following:

- mobile arm kinematics computation
  Both kinematics and differential kinematics are computed with Matlab by

using the Denavit-Hartemberg convention, see sections 4.1.1 and 4.1.3, and derivations.

- Mobile arm controllers
  Two kind of controllers have been used for the simulation: a PID controller performed over the joint position when the final joint position is already known a priori e.g. for the home position or the flower release position, and a Kinematic controller4.4.2 for the regulation of the robot joints during the picking task where only the final position and orientation is known. The kinematic controller has been defined in the Cartesian space and has been built in such a way to regulate the final position of the end-effector over the center of the flower stem and its orientation in such a way that the gripper pliers are parallel to the stem.

- Flower recognition
  This task is the first visual task performed by the robot and is needed to identify if on the current plant visited there are some ready flower to pick, see appendix B. In this way the robot decides whether approach or not the plant using only RGB image information.

- Approaching movement
  The approaching movement is needed to move the robot closer to the plant or to return the robot over the main trajectory. The main trajectories followed by the mobile platform are generally centered over some catwalks positioned on the soil. In this way the possibility of collision between the robot and the environment is minimized and the robot do not cross the catwalk.

- Flower segmentation
  This is the second Visual task of the algorithm, see appendix C. It is a completion of the previous visual algorithm; indeed, this algorithm is an extension of the previous one and it is able to recognize the flower stems, divide them on the image and assign to each of them a border and a centroid. The centroids, which are the (x, y) position on the image plane of the flowers center, are then listed and returned to the main algorithm. The number of the centroid also represents the number of flowers present on the plant; in factm, this number is used to iterate the picking task for each flower.

- Flower localization
  Thanks to the centroids list and using the depth camera information the robot can identify the flower position with respect to the robot and using some homogenous transformation it is able to find the position of the flower both

with respect to the world reference frame and end-effector reference frame. This algorithm has been proven to work with real applications, but some problem arises when it is used on simulations for different aspects. First, a simulated camera is different from a real one, it does not have any real sensor or a focal length and therefore the localization formulas presented in section 4.3.4 cannot be used. Moreover, the sensors used on Vrep are proved to be very noisy, indeed most of the times the computed depth from the camera to the centroids was wrong. For these reasons and other technical ones, the position and orientation information has been extracted directly from the simulator which provides some pre-defined functions for the localization of the scene objects.

- Gripper control The gripper control has been developed both on Matlab and the simulation tool. Matlab has the role of command to the simulation whether the gripper must be closed or not. The command is coded with a Boolean signal which is exchanged with Vrep that performs the real controller on the gripper using a Threaded Childscript.

- Platform movement along trajectories This task, like the previous one, is only commanded by Matlab but the real controller has been developed with a Threathed Childscript in Vrep. Matlab decides through which plant the robot must move and command the start of different trajectories developed on Vrep which are then followed by the platform controller.

- Others There are others many smaller tasks performed on Matlab e.g. the manipulation of the flower dynamic, but their purposes regard more the appearance of the simulation than its real realization. For this reason, they Will be not reported on this theses.

There are also some Matlab cross-functions which have been developed on Vrep, but they are used directly on Matlab. This has been done because of the Remote Api lack of functionalities in Matlab with respect to Vrep. In practice, some functions have been developed in Lua over the Vrep tool but are then used in Matlab thanks to a Remote API function which allows to run Vrep functions directly on Matlab. In this way the Remote Api can be enhanced with the Vrep once and its function holes can be filled.

6.2 Vrep

Vrep is the real simulator where the simulation has been performed. Before reporting the functionalities and the development performed on this tool it is important to

introduce the built environment.

6.2.1 Simulation Environment

Using the pre-built models which Vrep offers, a small simulated hydroponic greenhouse environment has been developed, see Fig. 6.1. This environment is composed by two main areas: the real greenhouse, which is the real useful part for the simulation, and the packaging area which has been introduced only to improve the aesthetic appearance of the simulation adding a pictorial touch to it. These two areas have been divided with some transparent panels to visually separate the real simulation area, i.e. the greenhouse, from the unused one, i.e. the packaging area.

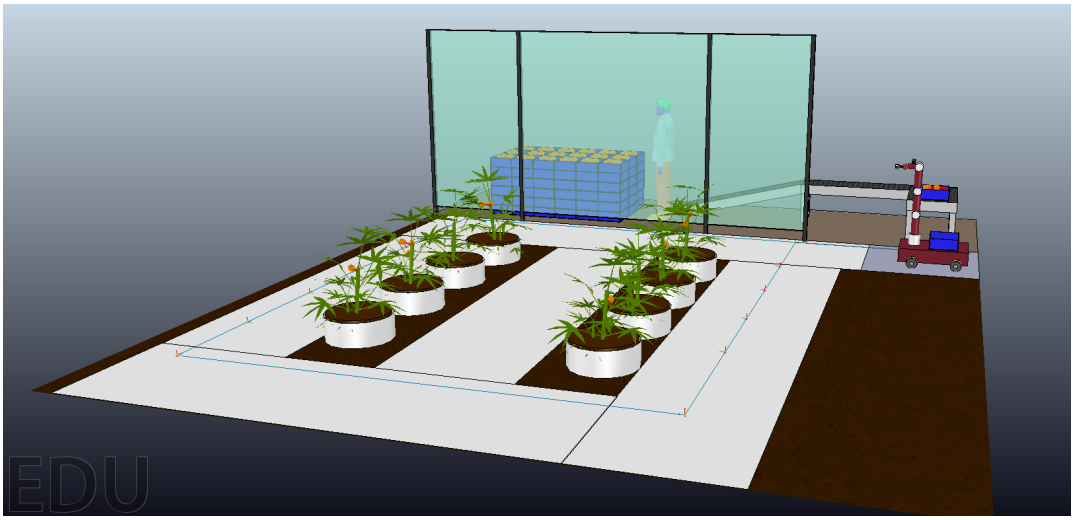


Figure 6.1. Representation of the environment used in the simulation

Starting from the second one it is composed by three synchronized conveyors belt upon which there is a flower-filled box which follows the belts till the final position near a person which are controlling the other flower boxes before the shipping.
Instead, the greenhouse is composed by the robot, the catwalks over which the robot can move, the paths that the robot mobile platform can follow and the plants over which the flower that the robot must pick are positioned. The flower 3D model, see Fig. 6.2, has been created with *Thinkercad* [85] a simple online CAD software developed by Autodesk which allows to create more complicated shapes with respect to Vrep which offers only primitive shapes.

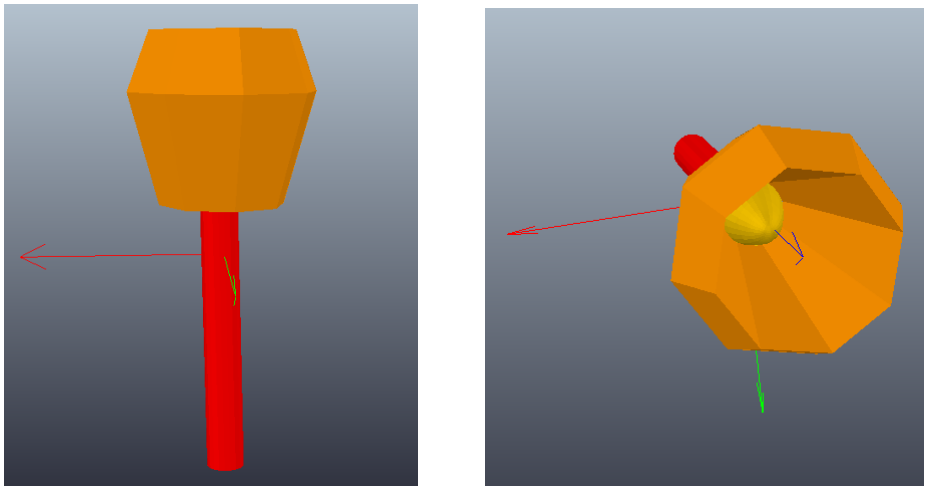
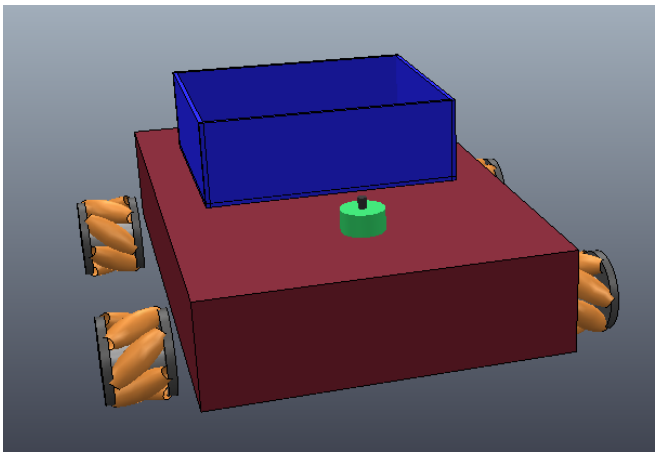


Figure 6.2. Flower 3D CAD model

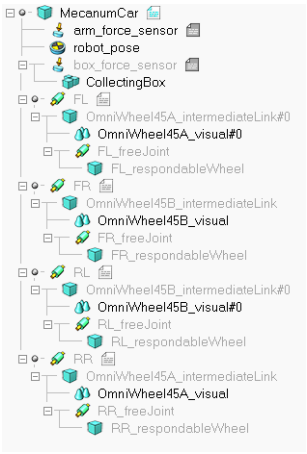
6.2.2 Robot assembling

The robot used in this simulation has been realized directly on Vrep using different primitive shapes, force sensors for the connections and joints. Using these pre-built objects two Vrep model has been created: the mobile platform model and the Robotic Arm one.

Mobile Platform



(a) Mecanun car model image



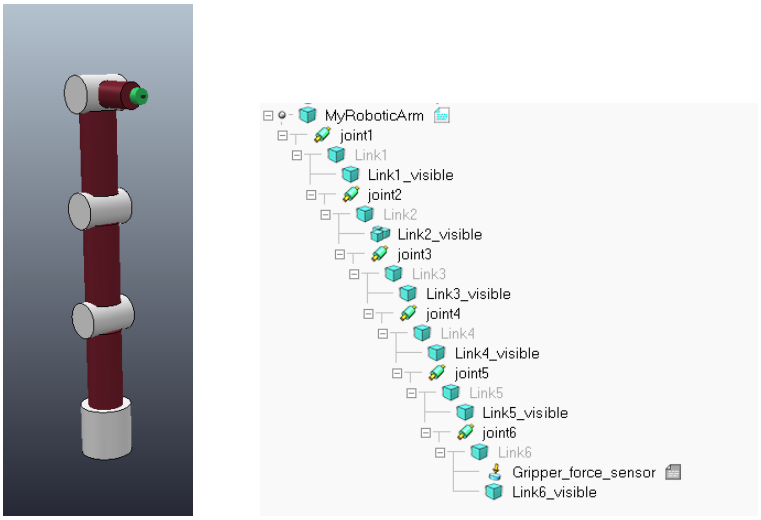
(b) Tree structure of the Mecanum car model

Figure 6.3. Mecanum Car Model

Fig. 6.3a represents the Mecanum car model developed in Vrep. It is composed by (see Fig. 6.3b) a rectangular shape representing the case of the robot, four Mecanum

wheels, a collection box and a force sensor where the robotic arm will be attached. The model is simple and the aesthetic aspect has been considered only in part. The distance between the wheels has a crucial importance in this model because in this way the slipping behaviors of the wheels are minimize, indeed the robot can turn around the center of the robot without slipping. except for this foresight, the model has been developed as presented in section 3.1.1.

Robotic Arm



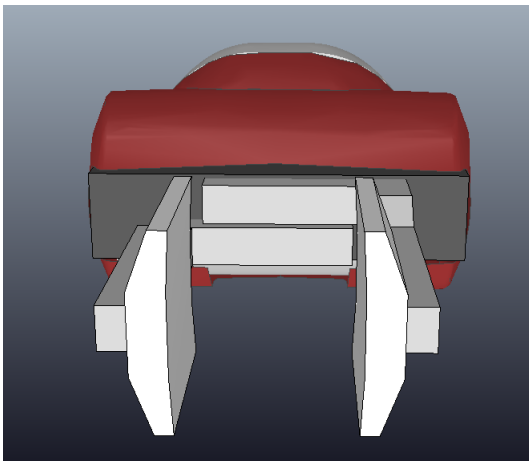
(a) Robotic Arm model image

(b) Tree structure of the Robotic Arm model

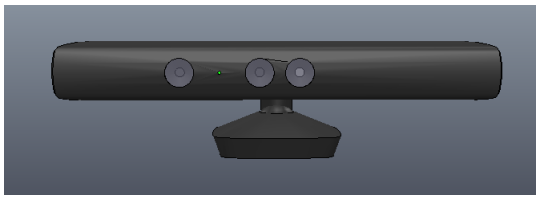
Figure 6.4. Robotic Arm model

Also the Robotic Arm model, see Fig. 6.4a, has been developed following the descriptions presented in section 4.1. In fact, the joint initial position and orientation has been decided to fit perfectly with the presented kinematics. To prove that the simulated robotic arm is represented exactly from the kinematics developed, different simulations have been performed using the forward kinematics. These simulations have proved that the robotic arm is well structured because the norm of the 3D position error was always under the $0.5cm$.
The model, see Fig. 6.4b, is composed by different cylindrical visual parts, six joints, different connector shapes and a force sensor for the end-effector connection. The kinematic structure of this robot is not realizable because some joints centers are superimposed and is is not always possible in the reality, but using some foresight, different companies have developed 6Dof robotics arm very similar to this one by performing only slight changes on the forward kinematics here presented.
To the robotic arm model have been then added also a gripper, namely the *Baxter*

Gripper which is a pre-built gripper model offered in Vrep by Rethink Robotics, see Fig. 6.5a, and the *Kinect Camera*, see Fig. 6.5b, which represents the RGB-D camera, offered by Lyall Randell and slightly modified by Nicola Piccinelli.



(a) *Baxter Gripper* - Robotic Arm end-effector



(b) *Kinect Camera* - Robotic Arm RGB-D camera

Figure 6.5. Other accessories of the Robotic Arms

6.2.3 functionalities

Except for the sumlation environment, the main functionalities of Vrep are essentially three:

- the trajectory creation for the robotic platform
- the robotic platform trajectory following controller
- the gripper controller

The first one, the trajectory creation, has been performed using the path model of Vrep. These paths can be modeled and rearranged to fit the necessity of the programmer. The trajectories are created by assigning a speed profile to a dummy object which is constrained to follow the path to which it is assigned, using a given speed profile. These trajectories are coordinated from Matlab which by sending some signals to Vrep can start some specific trajectories depending on the task that

the robot has to perform.
The second functionality is the trajectory following of the robot. The mobile platform controller has been constructed similarly to the one presented in section 3.3.3, but the position and orientation regulation has been separated. This has been done because the robot moves on parallel or perpendicular path constrained from the structure of the greenhouse and then the robot do not need to regulate also the angle during the trajectory, also because the platform is a omni-directional car and a slight error in orientation does not reflect over the error in position. Thus, the controller used is only a position controller which follows the dummy trajectory explained in the previous point. When the current trajectory is completed the robot autonomously extract the orientation of the next path to be followed and regulate its angle according to that. This controller is valid also for non-perpendicular path and also for curvilinear path, but in the latter case, the position and the orientation of the robot are not performed simultaneously and this could result in a bad looking movement e.g. the tangent velocity of the robot is not parallel with the Mecanum car direction. Despite this, for this simulation, this controller works pretty well and it is able to follow different speed profiles trajectories without any problem. The last functionality is the gripper controller which is a simple PID controller which regulates the joint speed of the gripper pliers. In this way, by sending a Boolean signal from Matlab which represent the open/close signal, the controller can either open or close the pliers.
There are also other functionalities implemented in Vrep such as the coordination between the conveyors belt used to transport a flower-filled collecting box from the robot docking station to the storage station and other functions which are used directly from Matlab to simplify some computations.

6.3 Simulation results

The simulation works well and generally the robot is able to collect all the flowers present on the scene even if sometimes happens that the robot does not gather appropriately the flower. The robot takes around 4 minutes to complete the simulation controlling all the plants present in the environment and returning to its docking station. The robot takes less than 25 seconds to pick a flowers and in some cases even less. Also, the Machine learning flower recognition algorithms are very fast. furthermore, in this simulation the robot actions have been separated to highlight each of them, but some movements could be performed simultaneously e.g. the approaching movement could be performed at the same time of the robotic arm motions. In this way the simulation time can be even more decreased, but the

dynamics behaviors must be considered to ensure that the robot does not flip during these movements.

Chapter 7

Conclusions

In conclusion, in this project, different solution to the same problem have been analyzed, i.e. zucchini flower picking in a hydroponic greenhouses. The different robot versions presented are all applicable in a real agriculture 4.0 scenario where a farmer wants to use a robot to pick some fruits from some plants. Also, some real implementation of one solution has been presented and they are all applicable in the reality by assembling the different structures chosen. To prove the efficiency and the real applicability of this project a simulation has been performed by building a possible robot and using the algorithms developed during the thesis.

7.1 future improvements

Regarding the future improvements, many modifications can be applied. My idea of autonomous harvesting robot or even better autonomous greenhouse is to build an environment self-sufficient in almost all the tasks. The human labor should be reduced consistently, especially for arduous tasks. Under this view an autonomous greenhouse should cover different aspects and the robots working in them should perform different tasks, even more than gathering fruits. Ones this idea has been introduced, some of the possible future improvements can be analyzed.
One of the simple tasks that the picking robot analyzed do not have is the charging/discharging of the boxes containing the harvested fruit. This simple task generally is performed by a human being which takes the boxes and move them on some packaging machine. This task could be performed either directly with the autonomous robot or using other solutions.
Other improvements could be applied at the localization and obstacle avoidance algorithm, not only to improve the mobile platform movements in the greenhouse, but also for a movement optimization in the case of cooperative robots. Indeed, the communication between different robot cooperating into the greenhouse is essential

both for costs and profits maximization. These kind of distributed robot systems could allow to build a greenhouse completely autonomous [24] which is able to put on the market low costs fresh fruits and vegetables.
An important thing which is not always considered, is the possibility that the robot works into a varied environment, i.e. an environment where different types of plants are present. Nowadays, It is known that the companion planting, i.e. plant different plants in the same environment, allows the agricultural companies to save money for pesticides and insecticides resulting in a more biological, fresh and strong plantation. If the robot is able to work with a various environment, classify the different fruits and plants and is able to recognize and prevents diseases, insects' infestations and fungal infections, the harvest could be maximized under every aspects: freshness, variegation, quality and last but not least in savings.
Another interesting improvements could be the introduction of a multi-purpose end-effector [23] to perform different tasks with the same robot. An exchangeable end-effector could be useful because the robot could be able to perform the different tasks which a normal human should perform, e.g. pruning, picking, monitoring and others. In this way, a single robot can perform the different tasks needed and the company can save space, money resulting another time in a more consistent environment.
Also, the image segmentation can be improved with the use of neural network. With this machine learning technique, it is possible to classify many different things: fruits, stems, branches, disease, insects and so on. In this way, the robot could recognize different plants parts, different fruits, different problems, and it could be able not only to report them to the farmer but also to intervene. There exists already some solution for this problem, but lots of improvements are still needed.
Regarding the robot regulation, it can be performed with more sophisticated and efficient methods like the visual control technique which allows the robot not only to perfectly follow a trajectory in the space, but also, to perform an online obstacle avoidance which can be essential in some cases.
A dynamic model of the manipulator arm can be derived to use it in the control loop. In this way, a low-level dynamic controller can be computed and the robot movements should result in a smoother behavior and more efficient tracking or regulation. The improvement here presented are only few of the ones that the future is facing us but the engineer specializations and the continuous increasing interests in this field will allow us to be ready to deal with them in the right way.

Appendix A

Euler Angles

The following appendix is based on [68] and [70].
A rotation matrix which belong from the $SO(3)$ group, i.e. Special Orthonormal group for $\mathbb{R}^3$ space, is composed by nine entries. A minimal representation for a tri-dimensional space is composed by three parameters, in general for an $SO(n)$ space the minimal representation is composed by $\frac{n(n-1)}{2}$. Therefore, the rotation matrix is a redundant representation, indeed the additional six parameters of the rotation matrix are needed to maintain the orthogonality of the matrix.
The minimal representation for the rotations in the $\mathbb{R}^3$ space is composed by three angles $\Phi = [\phi\ \theta\ \psi]^T$. each of these angles represents a simple rotation around one of the three axis. In fact, by composing three elementary rotation around three consequently non-parallel axis, it is possible to obtain any general rotation matrix in the three dimensional space.
These angles represent a group of rotational angels called Euler Angles and there exists twelve different possibilities for the choice of these angles. One of the most used is the ZYZ Euler Angles rotation which is introduced in the next section A.1.

A.1 ZYZ angles

As just said, the ZYZ angles is one of the most common group of the Euler angles used. This rotation is composed by three elementary rotations around the current reference frame:

- a rotation of ϕ around the z axis of the base reference frame
- a rotation of θ around the y' axis of the new reference frame obtained from the first rotation
- a rotation of ψ around the z" axis of the last reference frame obtained with the second rotation

As it is clear from the above list, the angles are expressed with respect to the current reference frame and not a fixed one. This means that, after that a rotation of an angle around an axis has been performed, then the next rotation is around the axis of the new reference frame found.
In practice, a rotation matrix of this form has been created:

$$R(\Phi) = R_z(\phi)R_{y'}(\theta)R_{z"}(\psi) = \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi - c_\phi s_\psi & s_\phi c_\theta s_\psi - c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \tag{A.1}$$

But one of the most important uses for this angles is the inverse solution, which allows to find a group of three angle starting from a given rotation matrix.

$$R = \begin{bmatrix} r_{aa} & r_{ab} & r_{ac} \\ r_{ba} & r_{bb} & r_{bc} \\ r_{ca} & r_{cb} & r_{cc} \end{bmatrix} \tag{A.2}$$

Comparing A.2 with A.1 and assuming that r_{ac} and r_{bc} are different from zero it is possible to define the following inverse relations

$$\phi = Atan2(r_{bc}, r_{ac}) \tag{A.3}$$

$$\theta = Atan2(\sqrt{r_{ac}^2 + r_{bc}^2}, r_{cc}) \tag{A.4}$$

$$\psi = Atan2(r_{cb}, -r_{ca}) \tag{A.5}$$

This solution is valid for any values of $\theta \in (0, \pi)$; indeed, another solution for the interval $(-\pi, 0)$ is given by

$$\phi = Atan2(-r_{bc}, -r_{ac}) \tag{A.6}$$

$$\theta = Atan2(-\sqrt{r_{ac}^2 + r_{bc}^2}, r_{cc}) \tag{A.7}$$

$$\psi = Atan2(-r_{cb}, r_{ca}) \tag{A.8}$$

Note that if $\theta = \{0, \pi\}$ then the rotation ϕ and ψ is constrained around the same axis and do not cover all the three dimensional space. This case is called a singularity for the Euler Angles.

Appendix B

K-mean algorithm

The K-mean algorithm is a technique used to find some clusters with similar characteristics into a given dataset. This algorithm is the non-probabilistic limitation of another algorithm, known as Expectation-Maximization (E-M) algorithm, which instead works with mixtures of Gaussian distribution. A full treatment of this algorithm can be found in [10] while here a small and simple explanation will be reported.
This algorithm is an unsupervised learning algorithm. These kinds of algorithms are characterized by a dataset of information in input which is not correlated with a target dataset. Indeed, differently from the supervised learning where the input dataset is composed by two subsets, one for the instances and one for their classes or function output, the unsupervised learning works with a singular set of instances which are initially considered equally. This is the reason why these algorithms are generally used to find some clusters of instances which have similar behaviors or information, to obtain a first rough classification of the dataset. For our purposes, the various information of the different clusters are not needed but the separation of them is necessary in order to perfectly separate the flowers from the image.
Returning to the algorithm it works in a really simple but smart way:

1. The algorithm is initialized with the dataset information, an image in our case, and the number K of wanted clusters. Moreover, K randomly centroids, in our case RGB color means, are initialized.

2. Using the current centroids, the algorithm assigns to them each nearest instance creating in this way K clusters near that instance. In our case the instance are the pixel colors and if their similar or their values are close to the centroid ones then it assigns that pixel to the centroid.
   Then once that the assignment is completed, recompute the centroids of the new clusters by taking their mean color values.

3. now, the algorithm takes each instance in the clusters and it recomputes the distance from the new found centroids. If there are some instances which are nearest to a centroid which is not the one of their belonging cluster, then move that instance in the cluster associated to that centroid. Then, once all the new assignments are finished recompute all the centroids.

4. Repeat step 3. until convergence is met, which means that no new assignment are performed between the clusters.

Although this algorithm is simple, it works well for our necessity. The convergence can be assured if for each iteration the mean distance of the instances from their centroid is decreased and if the dataset has only finite possible partitions.
Some contraries are that the number K of clusters has to be decided in the initialization before that the iterative part of the algorithm starts and, as can be seen in section 4.3.2, this could create some problems. This algorithm is also sensitive to initial conditions if the data of the image are not enough. The problem is that depending on the initialization, the algorithm could encounter some local minimum which stops the algorithm in an unwanted situation. This problem can be crossed by doing different attempts and comparing the solution to assure their equality. The algorithm is no robust to outliers, which means that if there are in the image some instances which are really far from every centroids, they modify the mean in a malicious way. Think for example to a red object into our picture, it could move the yellow/orange centroid far to the one which we are looking for.

B.1 K-mean image segmentation algorithm

In this section, the algorithm used to achieve the image segmentation is reported. The aim of this algorithm is to divide an image into clusters and extract the zucchini flowers cluster from them. This is useful to know if some zucchini flowers are present or not on a specific plant. It is important to stress that on Matlab the XYZ space is called Lab space where L, which represents Y, is the brightness, a which is X represents the blue wavelengths and b that is Z constitutes the other wavelengths [46].

Listing B.1. K-mean image segmentation algorithm

```
% clear all the variables and figures
clc; clear all; close all;
% Load the image for the segmantation
zucc = imread('zucc.jpg'); imshow(zucc);

% move the image from the RGB space to the CIE XYZ (Lab) one and extract the
% color plane
```

```
xyz_zucc = rgb2lab(zucc);
xz = xyz_zucc(:,:,2:3);
xz = im2single(xz);

% number of different colors to be extracted. It characterize the number of
% the final clusters
n_Clusters = 3;

% k-mean segmantation
pixel_labels = imsegkmeans(xz,n_Clusters,'NumAttempt',3);

% shows the clusters labelled with different grays colors
figure
imshow(pixel_labels,[])

% use different masks to extract form the original image the clusters found.
mask1 = pixel_labels==1;
cluster1 = zucc .* uint8(mask1);
figure
imshow(cluster1)

mask2 = pixel_labels==2;
cluster2 = zucc .* uint8(mask2);
figure
imshow(cluster2)

mask3 = pixel_labels==3;
cluster3 = zucc .* uint8(mask3);
figure
imshow(cluster3)

clusters={cluster1 cluster2 cluster3};
masks={mask1 mask2 mask3};

% find the flower cluster
for i=1:n_Clusters
    cluster=clusters{i};
    mask=masks{i};

    redChannel = cluster(:, :, 1);
    greenChannel = cluster(:, :, 2);
    blueChannel = cluster(:, :, 3);

    meanR = mean(redChannel(mask));
    meanG = mean(greenChannel(mask));
    meanB = mean(blueChannel(mask));

    fprintf('meanR: %3.f    meanG:%3.f    meanB: %3.f\n',meanR,meanG,meanB)
    if (meanR>210 && meanG>160 && meanG<240 && meanB<80)
       flowerCluster=cluster;
    end
end
```

Appendix C

Flower separation algorithm

In this appendix, the flower separation algorithm is presented, see algorithm C [47]. The algorithm starts with the loading of the image which represents the flower cluster. This image is then converted in a gray scale image and then binarized to have a black and white image bw. The bw image is then filtered to attenuate some noises present in it by deleting all that group of pixels which are under some range. After this, two filling operation are present. The first one, rows 23-24, fill the holes which can be contained in a circular shape of a chosen radius, while the second one, row 29, fill all the holes contained in a closed boundary. The first one is useful when the flowers are behind some objects or the image contain some noise which results in a not well represented flower, while the second one is used to obtain a flat surface representing the flowers with no holes inside of it. The rest of the algorithm divides the flat separated surfaces and labels them with different numbers, row 35, and then compute the centroids of the separated flowers and store them in an array, rows 48-49. The other instructions are needed to print the images on screen to have a representation of the algorithm functionalities.

Listing C.1. Flower image separation algorithm

```
1
2  %Starting from the flowers cluster extracted from the original
3  %figure using the k-mean algorithm, here a flower separation algorithm is presented
4
5  %load the cluster image
6  cluster = imread('flowercluster.png');
7  imshow(cluster);
8
9  %image binarization
10 grayIm = rgb2gray(cluster);
11 bw = imbinarize(grayIm);
12 figure
13 imshow(bw)
14
15 %noise attenuation
```

```
bw = bwareaopen(bw,20);
figure;
imshow(bw);

%fill the image void spaces by finding circles in the images
%the radius is set to 0 so this function is not used
%it is useful for very noisy images.
fill = strel('disk',0);
bw = imclose(bw,fill);
figure
imshow(bw)

%fill all the closed boundaries
bw = imfill(bw,'holes');
figure;
imshow(bw);

%find the boundaries of the non empty figures and separate the differen surfaces
%(B boundary functions, L labeled image)
[B,L] = bwboundaries(bw,'noholes');

% plot the image with different colors and highlight the boundaries
figure
imshow(label2rgb(L,@jet,[.5 .5 .5]))
hold on
for k = 1:length(B)
  boundary = B{k};
  plot(boundary(:,2),boundary(:,1),'w','LineWidth',1)
end

%extract the centroids from the labeled image and plot on the previous
%image
s = regionprops(L,'centroid');
centroids = cat(1,s.Centroid);
plot(centroids(:,1),centroids(:,2),'r*')

% number of flower present in the image. it is equal to k
Numberflowers = length(centroids);

%print of the original flower with the respective centroids in different figures.
for i = 1:Numberflowers
    clust = L==i;
    cluster = zucc .* uint8(clust);
    figure
    imshow(cluster);
    hold on
    plot(centroids(i,1),centroids(i,2),'r*')
end
```

Bibliography

[1] Norsuryani Zainal Abidin, Nurul Ain Mohamed, Zainah Md Zain, Maziyah Mat Noh, Norhafizah Md Zain, and Dwi Pebrianti. Backstepping control of non-holonomic car-like mobile robot in chained form. In *Proceedings of the 10th National Technical Seminar on Underwater System Technology 2018*, pages 173–180. Springer, 2019.

[2] Coppelia Robotics AG. Coppelia sim - vrep. `https://www.coppeliarobotics.com/`.

[3] Coppelia Robotics AG. Remote api functions for matlab to comunicate with coppelia sim. `https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm`.

[4] Agrobot. Agrobot picking robots. `https://www.agrobot.com/`.

[5] Root AI. Root ai farming robot. `https://root-ai.com/`.

[6] Abdullah Akay and Yusuf Sinan Akgul. 3d reconstruction with mirrors and rgb-d cameras. In *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 3, pages 325–334. IEEE, 2014.

[7] Dario Albani, Joris IJsselmuiden, Ramon Haken, and Vito Trianni. Monitoring and mapping with robot swarms for agricultural applications. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[8] CW Bac, Jochen Hemming, and EJ Van Henten. Robust pixel-based classification of obstacles for robotic harvesting of sweet-pepper. *Computers and electronics in agriculture*, 96:148–162, 2013.

[9] Luca Bascetta, Marco Baur, and Giambattista Gruosso. Robi': A prototype mobile manipulator for agricultural applications. *Electronics*, 6(2):39, 2017.

[10] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[11] Timo Blender, Thiemo Buchner, Benjamin Fernandez, Benno Pichlmaier, and Christian Schlegel. Managing a mobile agricultural robot swarm for a seeding task. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6879–6886. IEEE, 2016.

[12] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 521–528. IEEE, 2000.

[13] Yi-Chich Chiu, Suming Chen, and Jia-Feng Lin. Study of an autonomous fruit picking robot system in greenhouses. *Engineering in agriculture, environment and food*, 6(3):92–98, 2013.

[14] Omni Mechanical Technology Co. Light duty mecanum wheel mobile platform omr10. `http://www.omrobot.com/en/product/mecanum-wheel-robot.html`.

[15] Omni Mechanical Technology Co. Mecanum wheel mobile lift platform agv–omr06. `http://www.omrobot.com/en/product/Mecanum-Wheel-AGV.html`.

[16] Peter I Corke. Visual control of robot manipulators–a review. In *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, pages 1–31. World Scientific, 1993.

[17] Xin Dong, Mehmet C Vuran, and Suat Irmak. Autonomous precision agriculture through integration of wireless underground sensor networks with center pivot irrigation systems. *Ad Hoc Networks*, 11(7):1975–1987, 2013.

[18] Tom Duckett, Simon Pearson, Simon Blackmore, Bruce Grieve, Wen-Hua Chen, Grzegorz Cielniak, Jason Cleaversmith, Jian Dai, Steve Davis, Charles Fox, et al. Agricultural robotics: the future of robotic agriculture. *arXiv preprint arXiv:1806.06762*, 2018.

[19] Halil Durmuş, Ece Olcay Güneş, Mürvet Kırcı, and Burak Berk Üstündağ. The design of general purpose autonomous agricultural mobile-robot:“agrobot”. In *2015 Fourth International Conference on Agro-Geoinformatics (Agro-geoinformatics)*, pages 49–53. IEEE, 2015.

[20] Puwadol Oak Dusadeerungsikul and Shimon Y Nof. A collaborative control protocol for agricultural robot routing with online adaptation. *Computers & Industrial Engineering*, 135:456–466, 2019.

[21] AH Amer Eissa, AA Abdel Khalik, and AA Abdel. Understanding color image processing by machine vision for biological materials. *Structure and Function of Food Engineering*, pages 227–274, 2012.

[22] Gopi Krishna Erabati. 3d object recognition and relative localization using a 3d sensor embedded on a mobile robot, 2020.

[23] Farmbot. Farmbot open-source cartesian autonomous robot. `https://farm.bot/`.

[24] Robot farming startup Iron Ox. Iron ox autonomous farming robots. `https://ironox.com/`.

[25] FFrobotics. Ffrobotics harvesting robots. `https://www.ffrobotics.com/`.

[26] Spyros Fountas, Nikos Mylonas, Ioannis Malounas, Efthymios Rodias, Christoph Hellmann Santos, and Erik Pekkeriet. Agricultural robotics for field operations. *Sensors*, 20(9):2672, 2020.

[27] Kadeghe G Fue, Wesley M Porter, Edward M Barnes, and Glen C Rains. An extensive review of mobile agricultural robotics for field operations: Focus on cotton harvesting. *AgriEngineering*, 2(1):150–174, 2020.

[28] Zhenqing Gao, Yuanxin Yang, Yanping Du, Yuan Zhang, and Zhaohua Wang. Kinematic modeling and trajectory tracking control of a wheeled omni-directional mobile logistics platform. *DEStech Transactions on Engineering and Technology Research*, 2017.

[29] Yuanyue Ge, Ya Xiong, and Pål J From. Symmetry-based 3d shape completion for fruit localisation for harvesting robots. *Biosystems Engineering*, 197:188–202, 2020.

[30] Yuanyue Ge, Ya Xiong, Gabriel Lins Tenorio, and Pål Johan From. Fruit localization and environment perception for strawberry harvesting robots. *IEEE Access*, 7:147642–147652, 2019.

[31] R González, F Rodríguez, J Sánchez-Hermosilla, and JG Donaire. Navigation techniques for mobile robots in greenhouses. *Applied Engineering in Agriculture*, 25(2):153–165, 2009.

[32] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[33] Masood Ul Hassan, Mukhtar Ullah, and Jamshed Iqbal. Towards autonomy in agriculture: Design and prototyping of a robotic vehicle with seed selector. In *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*, pages 37–44. IEEE, 2016.

[34] IEEE. Ieee xplorer web for documentation. `https://ieeexplore.ieee.org/Xplore/home.jsp`.

[35] PVS Jayakrishna, M Suryavamsi Reddy, N Jaswanth Sai, N Susheel, and KP Peeyush. Autonomous seed sowing agricultural robot. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2332–2336. IEEE, 2018.

[36] Panagiotis Katsigiannis, Lazaros Misopolinos, Vasilis Liakopoulos, Thomas K Alexandridis, and George Zalidis. An autonomous multi-sensor uav system for reduced-input precision agriculture applications. In *2016 24th Mediterranean Conference on Control and Automation (MED)*, pages 60–64. IEEE, 2016.

[37] Nazmuzzaman Khan, Gregory Medlock, Scott Graves, and Sohel Anwar. Gps guided autonomous navigation of a small agricultural robot with automated fertilizing system. Technical report, SAE Technical Paper, 2018.

[38] Kinova. *Gen3* robotic arm produced by kinova. `https://www.kinovarobotics.com/en/products/gen3-robot`.

[39] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[40] Keerthy Kusumam, Tomáš Krajník, Simon Pearson, Tom Duckett, and Grzegorz Cielniak. 3d-vision based detection, localization, and sizing of broccoli heads in the field. *Journal of Field Robotics*, 34(8):1505–1518, 2017.

[41] Shipeng Li, Di Li, Chunhua Zhang, Jiafu Wan, and Mingyou Xie. Rgb-d image processing algorithm for target recognition and pose estimation of visual servo system. *Sensors*, 20(2):430, 2020.

[42] Yunwang Li, Sumei Dai, Yuwei Zheng, Feng Tian, and Xucong Yan. Modeling and kinematics simulation of a mecanum wheel platform in recurdyn. *Journal of Robotics*, 2018, 2018.

[43] Dogtooth Technologies Limited. Dogtooth picking robots. `https://dogtooth.tech/`.

[44] Guichao Lin, Yunchao Tang, Xiangjun Zou, Juntao Xiong, and Jinhui Li. Guava detection and pose estimation using a low-cost rgb-d sensor in the field. *Sensors*, 19(2):428, 2019.

[45] Nathan Lovell and Vladimir Estivill-Castro. Color classification and object recognition for robot soccer under variable illumination. In *Robotic Soccer*. Citeseer, 2007.

[46] MathWorks. Color-based segmentation using k-means clustering. `https://it.mathworks.com/help/images/color-based-segmentation-using-k-means-clustering.html`.

[47] MathWorks. Identification of closed areas. `https://it.mathworks.com/help/images/identifying-round-objects.html`.

[48] Eka Maulana, M Aziz Muslim, and Veri Hendrayawan. Inverse kinematic implementation of four-wheels mecanum drive mobile robot using stepper motors. In *2015 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 51–56. IEEE, 2015.

[49] Neha S Naik, Virendra V Shete, and Shruti R Danve. Precision agriculture robot for seeding function. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 2, pages 1–3. IEEE, 2016.

[50] Neobotix. Mobile robot mpo-500. `https://www.neobotix-robots.com/products/mobile-robots/mobile-robot-mpo-500`.

[51] Netrise. Meteomotion automatic tomatoes picking. `https://metomotion.com/`.

[52] Tien Thanh Nguyen, Erdal Kayacan, Josse De Baedemaeker, and Wouter Saeys. Task and motion planning for apple harvesting robot. *IFAC Proceedings Volumes*, 46(18):247–252, 2013.

[53] Tien Thanh Nguyen, Koenraad Vandevoorde, Erdal Kayacan, Josse De Baerdemaeker, and Wouter Saeys. Apple detection algorithm for robotic harvesting using a rgb-d camera. In *International Conference of Agricultural Engineering, Zurich, Switzerland*, 2014.

[54] Octinion. Rubion strawberries picking robot. `http://octinion.com/products/agricultural-robotics/rubion`.

[55] Søren Marcus Pedersen, Spyros Fountas, Claus G Sørensen, Frits K Van Evert, and B Simon Blackmore. Robotic seeding: economic perspectives. In *Precision Agriculture: Technology and Economic Perspectives*, pages 167–179. Springer, 2017.

[56] Luis Pérez, Íñigo Rodríguez, Nuria Rodríguez, Rubén Usamentiaga, and Daniel F García. Robot guidance using machine vision techniques in industrial environments: A comparative review. *Sensors*, 16(3):335, 2016.

[57] Rodrigo Méndez Perez, Fernando Auat Cheein, and Joan R Rosell-Polo. Flexible system of multiple rgb-d sensors for measuring and classifying fruits in agri-food industry. *Computers and Electronics in Agriculture*, 139:231–242, 2017.

[58] Alessio Plebe and Giorgio Grasso. Localization of spherical fruits for robotic harvesting. *Machine Vision and Applications*, 13(2):70–79, 2001.

[59] ResearchGate. Researchgate for discovering researches. `https://www.researchgate.net/`.

[60] King Kong Robot. 4wd mecanum wheel mobile robotic platform. `https://www.robotshop.com/eu/en/4wd-mecanum-wheel-mobile-robotic-platform.html`.

[61] Omni Robot. Mecanum wheel example. `https://omni-robots.com/products/304-8mm-12-inches-mecanum-wheel/`.

[62] Juan Jesús Roldán, Jaime del Cerro, David Garzón-Ramos, Pablo Garcia-Aunon, Mario Garzón, Jorge de León, and Antonio Barrientos. Robots in agriculture: State of art and practical experiences. *Service Robots*, 2018.

[63] Abdul Salam and Usman Raza. Autonomous irrigation management in decision agriculture. In *Signals in the Soil*, pages 379–398. Springer, 2020.

[64] Alistair J Scarfe, Rory C Flemmer, HH Bakker, and Claire L Flemmer. Development of an autonomous kiwifruit picking robot. In *2009 4th International Conference on Autonomous Robots and Agents*, pages 380–384. IEEE, 2009.

[65] Kareemulla Shaik, Edwin Prajwal, B Sujeshkumar, Mahesh Bonu, and Balapanuri Vamseedhar Reddy. Gps based autonomous agricultural robot. In *2018 International Conference on Design Innovations for 3Cs Compute Communicate Control (ICDI3C)*, pages 100–105. IEEE, 2018.

[66] P Shanmugavadivu and Ashish Kumar. Boundary detection of objects in digital images using bit-planes and threshold modified canny method. In *Mining Intelligence and Knowledge Exploration*, pages 192–200. Springer, 2013.

[67] BS Shivaprasad, MN Ravishankara, and BN Shoba. Design and implementation of seeding and fertilizing agriculture robot. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, 3(6):251–255, 2014.

[68] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[69] K Durga Sowjanya, R Sindhu, M Parijatham, K Srikanth, and P Bhargav. Multipurpose autonomous agricultural robot. In *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, volume 2, pages 696–699. IEEE, 2017.

[70] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[71] Hamid Taheri, Bing Qiao, and Nurallah Ghaeminezhad. Kinematic model of a four mecanum wheeled mobile robot. *International journal of computer applications*, 113(3):6–9, 2015.

[72] Yun-Chao Tang, Chenglin Wang, Lufeng Luo, Xiangjun Zou, et al. Recognition and localization methods for vision-based fruit picking robots: a review. *Frontiers in Plant Science*, 11:510, 2020.

[73] Yingzhong Tian, Shiyu Zhang, Jiaorong Liu, Feixue Chen, Long Li, and Beixin Xia. Research on a new omnidirectional mobile platform with heavy loading and flexible motion. *Advances in Mechanical Engineering*, 9(9):1687814017726683, 2017.

[74] Wageningen University and Greenhouse Horticulture Research. Sweeper robot. `http://www.sweeper-robot.eu7`.

[75] Keyvan Asefpour Vakilian and Jafar Massah. A farmer-assistant robot for nitrogen fertilizing management of greenhouse crops. *Computers and electronics in agriculture*, 139:153–163, 2017.

[76] Eldert J Van Henten, Jochen Hemming, BAJ Van Tuijl, JG Kornet, J Meuleman, J Bontsema, and EA Van Os. An autonomous robot for harvesting cucumbers in greenhouses. *Autonomous robots*, 13(3):241–258, 2002.

[77] Manuel Vázquez-Arellano, Hans W Griepentrog, David Reiser, and Dimitris S Paraforos. 3-d imaging systems for agricultural applications—a review. *Sensors*, 16(5):618, 2016.

[78] Guohua Wang, Yabo Yu, and Qingchun Feng. Design of end-effector for tomato robotic harvesting. *IFAC-PapersOnLine*, 49(16):190–193, 2016.

[79] Kangkan Wang, Guofeng Zhang, and Hujun Bao. Robust 3d reconstruction with an rgb-d camera. *IEEE Transactions on Image Processing*, 23(11):4893–4906, 2014.

[80] Xiaoqin Wang, Y Ahmet Şekercioğlu, and Tom Drummond. A real-time distributed relative pose estimation algorithm for rgb-d camera equipped visual sensor networks. In *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–7. IEEE, 2013.

[81] Wikipedia. Cie xyz trimulus values. `https://en.wikipedia.org/wiki/CIE_1931_color_space`.

[82] Wikipedia. Mecanum wheel. `https://en.wikipedia.org/wiki/Mecanum_wheel`.

[83] Wikipedia. Spherical coordinates of a vector. `https://en.wikipedia.org/wiki/Spherical_coordinate_system`.

[84] Inc. © 1994-2020 The MathWorks. Matlab simulink. `https://www.mathworks.com/products/matlab.html`.

[85] Inc © 2020 Autodesk. *Thinkercad* - online cad. `https://www.tinkercad.com/`.

[86] ©Microsoft. Microsoft kinect development tools. `https://developer.microsoft.com/en-us/windows/kinect/`.