Functionalized Soft robotic gripper for delicate produce harvesting powered by imitation learning-based control

D1.2. SOFTGRIP SOFTWARE AND HARDWARE ARCHITECTURE

Deliverable Number	2
Work package Number and Title	WP1 Technical and Functional Requirements Analysis
Lead Beneficiary	THL
Version – Status	Final
Due Date	30 June 2021
Deliverable Type[1]	Report
Dissemination Level[2]	PU
Internal Reviewer	SSSA
Filename	D1.2_SoftGrip_Architecture.docx

[1] **Please indicate the type of the deliverable using one of the following codes**:
R = Document, report
DEM = Demonstrator, p ot, prototype
DEC = Webs tes, patent f ngs, v deos, etc.
ETHICS = Eth cs requ rement
ORDP = Open Research Data P ot
DATA = data sets, m crodata, etc.

[2] **Please indicate the dissemination level using one of the following codes**:
PU = Pub c
CO = Conf dent a , on y for members of the consort um (nc ud ng the Comm ss on Serv ces)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101017054

DOCUMENT INFO

Authors

Surname	First name	Organization	E-mail
Authors			
Vartholomeos	Panagiotis	THL	panagiotis.vartholomeos@twi.gr
Koutsoumpa	Christina	THL	christina.koutsoumpa@twi.gr
Dometios	Athanasios	ICCS	athdom@mail.ntua.gr
Giacomo	Picardi	SSSA	g.picardi@santannapisa.it
Vasios	Konstantinos	UESS	k.vasios@essex.ai
Editor			
Koutsoumpa	Christina	THL	christina.koutsoumpa@twi.gr

Internal Reviewers

Surname	First name	Organization	E-mail
Cianchetti	Matteo	SSSA	matteo.cianchetti@santannapisa.it
Retsinas	George	ICCS	georgeretsi@gmail.com
Marougkas	Isidoros	ICCS	ismarougkas@gmail.com

Document History

Date	Version	Editor	Change	Status
16/06/21	1.0	THL	Contribution, integration and first complete version	Sent to internal reviewers
24/06/21	1.1	SSSA	First review, specification table added, minor modifications suggested	Under review
28/06/21	1.2	THL	Minor modifications addressed	Sent to consortium for review
30/06/21	Final	THL	Minor edits	Submitted

TABLE OF CONTENTS

1 INTRODUCTION

1.1 SYSTEM OVERVIEW

The SoftGrip system overview is depicted in Figure 1.1. The robot comprises two devices: (1) a rigid robot mounted over the shelf that moves in x-y-z axis and (2) the soft gripper which is attached on the end-effector of the rigid robot and faces the mushroom cultivation. The rigid robot with the gripper will be installed over the cultivation and will be able to reach any position in the workspace. A central computer hosts the SoftGrip supervision module, which generates the sequence of grasping tasks and supervises their execution. The input to the supervision system is the estimation of mushroom size, position and orientation. This estimation is generated by the vision module, which processes the information captured by low-cost environment cameras installed on the shelves. The supervisor generates a sequence of grasping tasks, which is fed to the grasp planner module, which in turn computes the trajectories of the robot and the grasping primitives of the soft-gripper. Then the low-level closed-loop controllers generate the actuation commands, which are fed to the drivers of the robotic devices in order to execute the grasping primitives. The commands are adjusted based on feedback signals generated by the sensors embedded into the soft arm and the encoders of the Cartesian robot actuators.

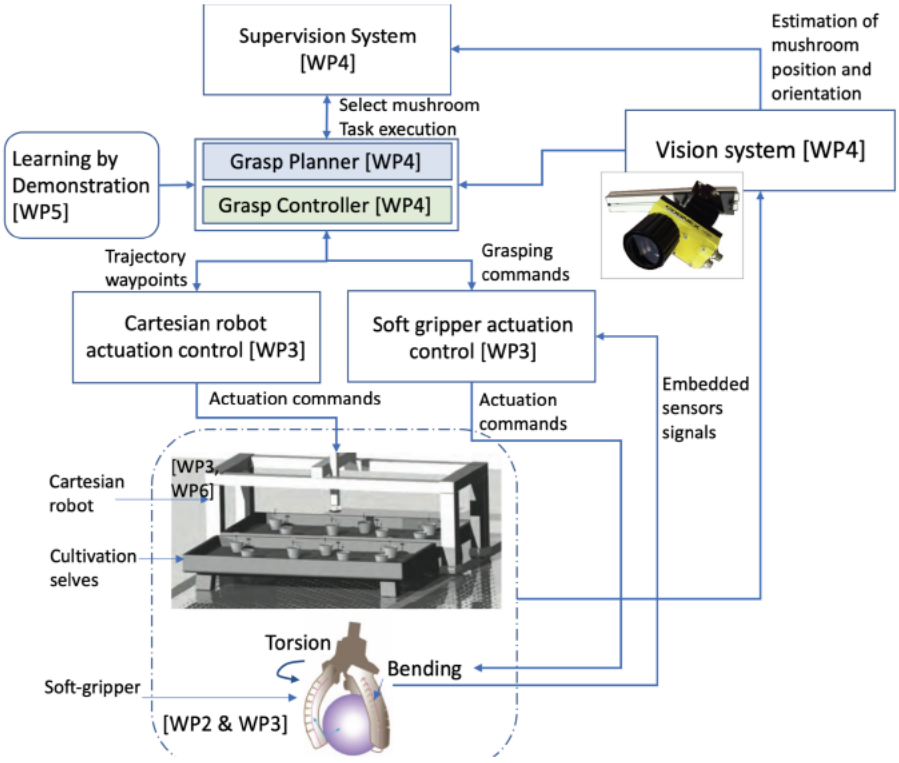

Figure 1.1. System overview.

The system overview presented in Figure 1.1 forms the basis for building the system architecture in the next sections. Figure 1.1 serves also as a context diagram, which presents a high-level picture of the systems' boundaries and its adjacent external entities, and it makes clear the context in which the analysis will take place.

1.2 METHODOLOGY

The architecture will be derived based on a hierarchical approach (see Figure 1.2). The top level is the conceptual architecture, which is what has been presented in the system overview in Figure 1.1 above. The immediate lower level is the functional architecture, where concepts are mapped into major functional modules. Next lower level is a decomposition of the functional modules into implementation components and their

interfaces. Since SoftGrip is a robotic system, it is both hardware and software intensive, and therefore the lower level architecture will be divided into hardware and software architectures.

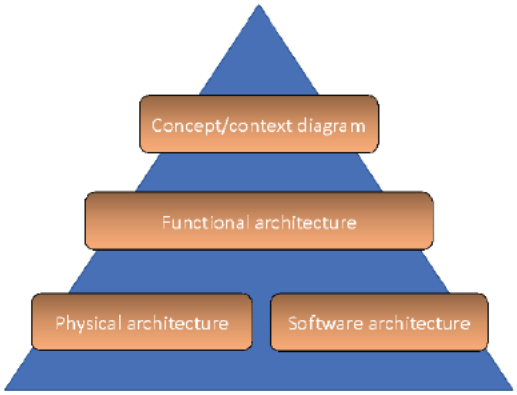

Figure 1.2. Hierarchical architecture design.

An architectural element (either software or hardware) refers to the pieces from which systems are built, and is defined as a fundamental piece from which a system can be considered to be constructed. The architectural elements that will be described in the next paragraphs possess the following key attributes:

- A clearly defined set of functionalities (or services)
- A clearly defined boundary
- A set of clearly defined interfaces, which define the functionalities (or services) that the element provides to the other architectural elements

Hence, the following sections provide answers to the following architectural questions.

- What are the main functional elements of the SoftGrip architecture?
- How will these elements interact with one another and with the outside world?
- What information will be managed, stored and presented?
- What physical hardware and software elements will be required to support these functional and information elements?
- What operational features and capabilities will be provided by the elements of SoftGrip?
- What development, test and support (during pilot trials) will be provided?

The above questions cannot be answered by a single model. It is not possible to capture the functional features and quality properties of a complex system in a single comprehensible model that is understandable. The architecture should be presented in a way that is manageable and comprehensible. The approach that will be used, is to partition the architectural design into a number of separate but interrelated *views*, each of which describes a separate aspect of the architecture. Collectively the views describe the whole system.

1.3 INPUTS TO THE ARCHITECTURE DEFINITION

The required inputs for deriving architecture are the scope, the context and the stakeholder requirements. These inputs will be consolidated based on information from the system requirements and stakeholder analysis that were defined in deliverable D1.1 [1]. The process of architecture definition is depicted in Figure 1.3.

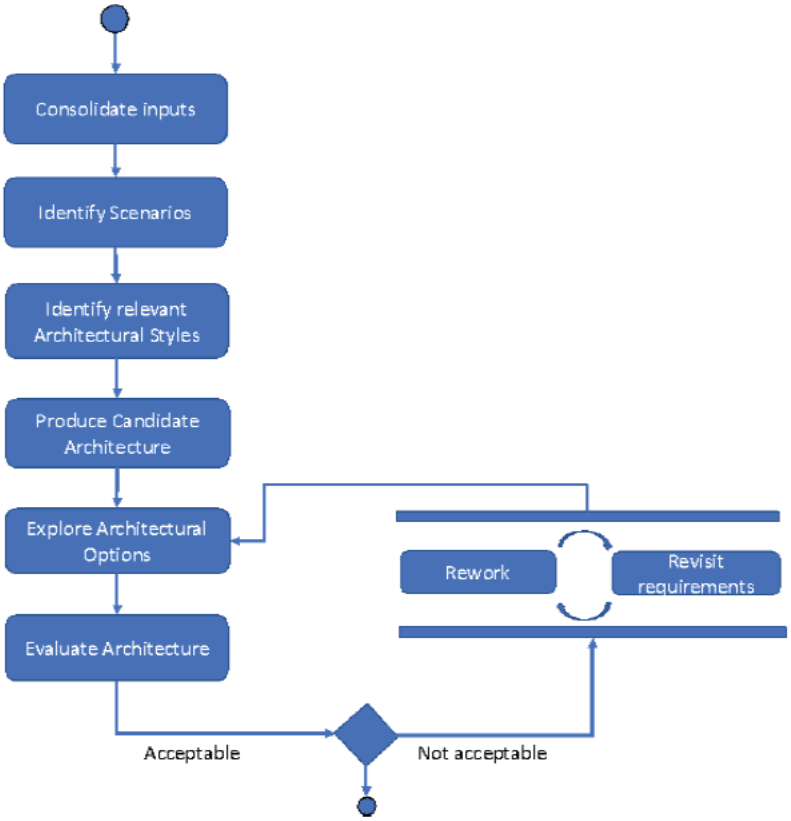

Figure 1.3. Architecture definition process.

1.4 DEFINITION OF ENGINEERING SPECIFICATIONS

The engineering specification of the SoftGrip system have been derived, as anticipated in D1.1 through the Quality Function Deployment (QFD) methodology. The QFD consists in translating the requirements of the stakeholders into measurable specifications that will be used in the following stages to benchmark the design choices. The table in which stakeholders' requirements are translated into engineering specification is reported in Figure 1.4 and is called House of Quality (HoQ). With respect to the HoQ presented in D1.1, here we have refined the definitions of the specifications and set target values for each of them. In particular, for each specification we have set a lowest threshold, which must not be exceeded; a target to aim at in absolute terms; and a project target that takes into account the intrinsic limitations that a research project has compared to an industrial product. We have used a colour code, reported in the table, to differentiate target values calculated as informed guess, based on available information; target values that still need to be estimated because of the lack of data in this phase of the project; and target values that are not binding, e.g. those which do not pertain to functional specifications and can be relaxed in the scope of the project.

Below the HoQ, a list of all specifications is reported along with a short description and an indication on how the target values were computed.

x = 1
[] = 0

	WHAT	WHO: Owner of mushroom growing plant	WHO: Manager/Supervisor of mushroom growing plant	WHO: Harvester (working alongside robots)	WHO: Quality Contrller of Packhouse	WHO: Regulatory agencies	WHO: Retailer / Consumer of mushrooms	Harvesting rate	Harvesting cycle time	Ratio of damaged mushrooms on the total	Ratio of undected mushrooms	Number of grasping attempts per mushroom	Harvesting success over total attempts
								↑	↓	↓	↑	↓	↑
								n/min kg/hour	s	%	%	n	%
Mushroom picking performance	Fast collection	x	x	x				o	o	o	o	o	o
	Do not damage mushrooms	x	x	x	x	x	x	o		o		o	o
	Mushroom deterioration detection	x	x	x									
	Select correct mushrooms	x	x	x							o		
Working environment	Fits in the allocated space	x	x	x									
	Ability to harvest multiple size	x	x	x									
	Ability to place mushroom on a conveyor	x	x	x									
	Silent		x	x									
	Resistant to environmental conditions	x	x	x									
	Transportable between tunnels and shelves	x	x	x									
	User friendly (simplicity of human intervention and monitoring)		x	x									
	Safety of human operators		x	x									
Manufacturing and integration	Durability of the system	x	x	x									
	Simplicity of installation/deinstallation		x	x									
	Low cost	x											
	Environmentally friendly	x				x							
	Safe food interaction (protection from chemical damage)				x	x							
	Simplicity of maintenance	x	x	x									
Informed guess		Lowest threshold						1	60	10%		3	85%
To be estimated		Target						3	20	0%		1	100%
Non binding		Project target						1	60	5%		3	90%

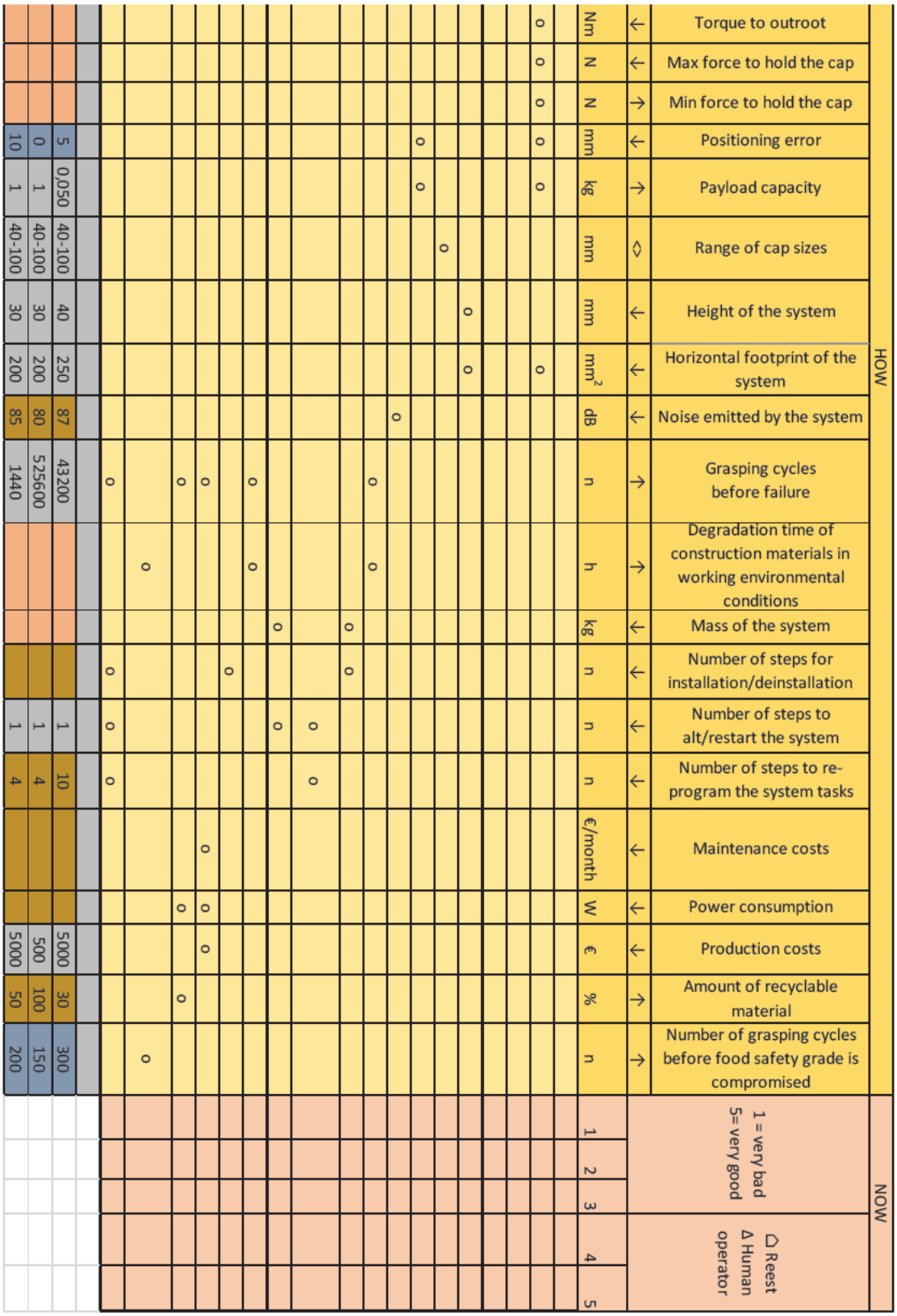

Figure 1.4. House of Quality of the SoftGrip system reporting the engineering specifications and their target values to benchmark the design in the following stages of the project.

Harvesting rate: the number of mushrooms harvested by the system during non-stop operation over time. The target value for this specification is calculated based on standard operations in a grow room with human operators as follows. Measured in units/min. A higher rate is desired [↑].

Target calculation: the worst-case scenario is considered, i.e. when there are more mushrooms to be picked (4th day of first flush → 7.7 kg/m^3 → 277 kg/shelf → 15400 mushrooms/shelf).

- size of the shelf (Ss) = 1.2 x 30 m^2 = 36 m^2,
- Average mushroom weight →18g
- Number of robots per shelf → 5
- Robot working hours → 24
- Harvesting rate → 15400/(24*60)/5 → 2.14 units/min

Harvesting cycle time: time required for a full harvesting cycle from mushrooms selection to placing. Measured in seconds [s]. A lower time is desired [↓].

Targets are computed as the reciprocal of Harvesting rate.

Ratio of damaged mushrooms on the total: the number of blemished/bruised mushroom over the total number of successfully harvested mushrooms in a given amount of time. Measured as the percentage [%] of blemished/total. A smaller number of blemished mushrooms is desired [↓].

Target calculation: informed guess based on average percentage of value packs produced by a plant.

Ratio of undetected mushrooms: Ratio of harvestable mushrooms detected by the system vs total number of harvestable mushrooms as estimated by a human harvester. Measured as a percentage. A higher number of correct choices is desired [↑].

Target calculation: To be estimated.

Number of grasping attempts per mushroom: Number of attempts required before the mushroom cap is firmly hold by the gripper. Fine re-positioning may be required in case of failure. Measure as number of attempts [n]. A lower number of attempts is desired [↓].

Target calculation: informed guess based on estimated actuation times and competitors for which data is available [2].

Harvesting successes over total attempts: Ratio between the number of non-damaged outrooted mushrooms over the total number of attempts. Multiple grasping attempts for the same mushroom count as one for this specification. Measured as the percentage [%] of non-damaged outrooted mushrooms/total number of attempts. A higher rate of harvesting successes is desired [↑].

Target calculation: informed guess based on Ratio of damaged mushrooms and a safety margin to account for mushrooms that cannot be outrooted or exhibit excessive damages.

Torque to outroot: average twisting torque exerted by the system on the mushrooms to successfully outroot them without causing any damage. Measured in Newton-metre [Nm]. A smaller torque to outroot is desired [↓].

Target calculation: to be estimated, missing data due to COVID situation.

Max force to hold the cap: lowest shear force applied by the system to the mushroom cap in contact points without causing any damage. Measured in Newton [N]. A smaller max force to hold the cap is desired [↓].

Target calculation: to be estimated, missing data due to COVID situation.

Min force to hold the cap: lowest shear force applied by the system to the mushroom cap in contact points to hold it firmly. Measured in Newton [N]. A smaller min force to hold the cap is desired [↓].

Target calculation: to be estimated, missing data due to COVID situation.

Position error: difference between reached and commanded end effector position. Measured in millimetres [mm]. A lower positioning error is desired [↓].

Target calculation: informed guess based on experience on electric actuation. This value will be refined based on the ability of the soft gripper to compensate for positioning errors of the cartesian robot.

Payload capacity: Maximum weight of mushroom that the robot can handle. Measured in kilograms [kg]. A higher payload capacity is desired [↑].

Target calculation: based on typical weight of mushrooms. Given the low weight of mushrooms this specification is not expected to be hard to meet.

Range of cap sizes: Range of mushrooms' cap sizes that can be harvested. Measured in millimetres [mm]. The system must be able to harvest all sizes within the range [<>].

Target calculation: based on mushroom cap sizes that the system must harvest.

Height of the system: Height of the system above the cultivation. The system should be low enough to allow the gripper to move above the cultivation without hitting.

Target calculation: based on the vertical spacing between two shelves.

Horizontal footprint of the system: Horizontal footprint of the smallest bounding box of the soft gripper. Measured in squared meters [m^2]. A smaller footprint of the system is desired [↓].

Target calculation: based on the largest mushroom cap to be harvested.

Noise emitted by the system: Ratio between the energy of the noise emitted by the SoftGrip system in standard operative conditions (N1) over the background noise in the plant measured when the SoftGrip system is off (N0). Measured as 10log10(N1/N0) [dB]. A lower noise is desired [↓].

Target calculation: based on the Directive 2003/10/EC – noise of the European Agency for Safety and Health at work. We consider this a non-binding specification because it is non functional with respect to the harvesting task.

Grasping cycles before failure: Average number of mushrooms grasping attempts (successful or not) before any component of the SoftGrip system fails. Measured in number of attempts [n]. A higher number of grasping cycles before failure is desired [↑].

Target calculation: informed guess based on the expected working life of the system, considering a harvesting cycle time of 1 minute (Project target), considering that the system will be active 24/7 (most likely an overestimation) and considering one year as a reasonable target, one month for the lowest threshold and one day for the project target.

Degradation time of construction materials in working environmental conditions: Amount of time before construction materials, exposed to the environmental conditions of mushrooms growing room, begin to lose their functional properties. Measured in hours [h]. A longer degradation time is desired [↑].

Target calculation: to be estimated based on number of grasping cycles before failure and reasonable specification of materials available.

Mass of the system: Mass of the overall system. Measured in kilograms [kg]. A smaller mass of the system is desired [↓].

Target calculation: to be estimated based on the load that can be withstood by existing cultivation shelves.

Number of steps for installation/deinstallation: Number of elementary steps (e.g. screw/unscrew a screw, connect/disconnect a pipe, etc…) to install/uninstall the SoftGrip system. Measured in number of steps [n]. A smaller number of steps for installation/uninstallation is desired [↓].

Target calculation: To be estimated based. This is a non-binding specification because it does not pertain to the functionality of the system.

Number of steps to alt/restart the system: Number of elementary steps required by a human supervisor to alt and/or restart the system in case of emergency or need for maintenance. Measured in number of elementary steps (e.g. press the emergency button, unlock the protective barrier, etc…). A lower number of steps is desired [↓].

Target calculation: An emergency/restart routine could be activated by simply pressing one button.

Number of steps to re-program the system tasks: Number of elementary steps required by a human supervisor to set the SoftGrip system tasks. Measured in number of elementary steps (e.g. pause the system, open the user interface panel, select a pre-programmed task, etc…). A lower number of steps is desired [↓].

Target calculation: This is a non-binding specification because *ad hoc* software routines can be designed to specifically simplify the re-programming. e.g. stop system open user panel select task restart system.

Maintenance costs: Average costs to maintain the SoftGrip system (e.g. cleaning, substitution of components, supervising personnel, etc…). Measured in euro [€/month]. Smaller maintenance costs are desired [↓].

Target calculation: This is non-binding and it is still to be estimated based on the economical convenience of adopting the robotic solution. Factors in play will be the necessity to substitute/repair malfunctioning units and maintenance costs related to restoring the food safety grade (e.g. through cleaning, substituting protective covers, etc…)

Power consumption: Average power consumption of the overall system during typical operations. Measured in watts [W]. A smaller power consumption is desired [↓].

Target calculation: This is non-binding and it is still to be estimated based on the economical convenience of adopting the robotic solution.

Production costs: Total costs to produce the SoftGrip system. Measured in euro [€]. Smaller production costs are desired [↓].

Target calculation: based on the overall convenience of employing a robotic solution in the existing working environment. It depends on the number of units per plant.

Amount of recyclable material: Ratio between the mass of recyclable materials in the construction of the SoftGrip system over the total mass of the system. Measured as the percentage [%] of recyclable mass/total mass. A greater amount of recyclable material is desirable [↑].

Target calculation: This is a non-binding specification with respect to the system functionalities. It will mostly impact maintenance costs.

Number of grasping cycles before food safety is compromised: The number of grasping attempts before the SoftGrip system, especially the part directly in contact with the mushroom, loses the food safety required grade. Measured in number of cycles [n]. A higher number of grasping cycles before food safety is compromised is desired [↓].

Target calculation: estimated guess based on the number of times human harvesters change gloves per day. They change them 5-10 times per day, corresponding to around 150 mushrooms harvested.

2 FUNCTIONAL ARCHITECTURE OVERVIEW

The functional architecture comprises architectural models that identify system function and their interactions. Generally, more than one architecture can satisfy the requirements presented in D1.1 [1]. Usually, each architecture and its set of associated allocated requirements have different costs, schedules, performance, and risk implications. In the following paragraphs, we present a first version of the architecture, which serves as a baseline for further improvement in the course of the project. The high-level architecture is depicted in Figure 2.1.

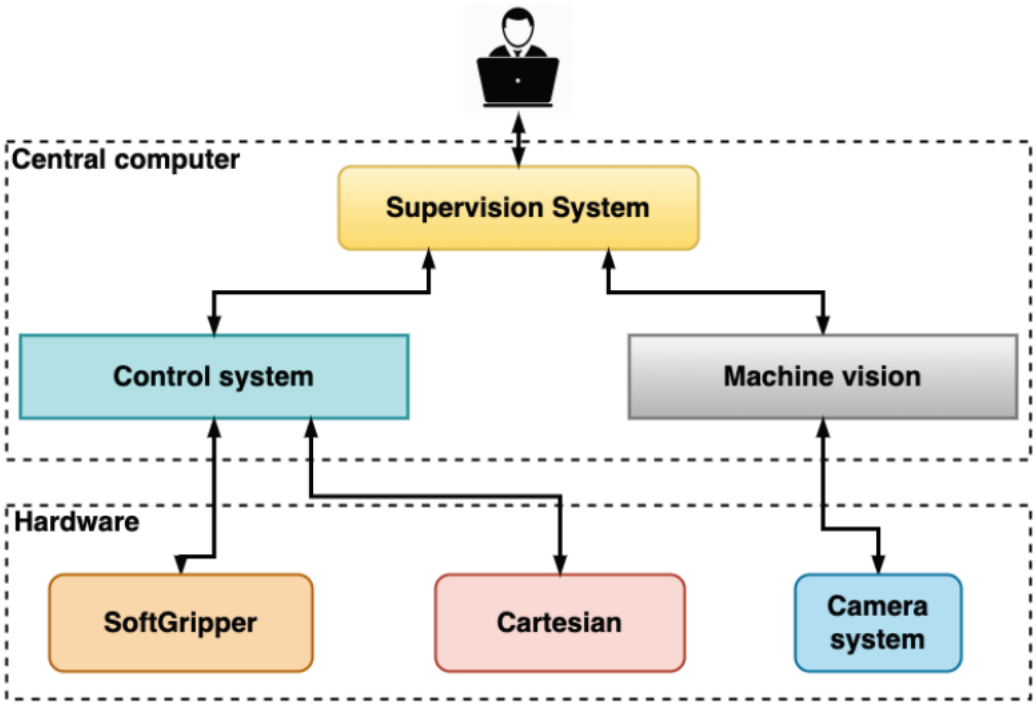

Figure 2.1. High-level functional architecture of the SoftGrip system.

2.1 SUPERVISION UNIT

The operational flow of SoftGrip will be based on the robotic task sequence of the harvesting activities. It also includes the initialization, the termination and the error handling actions of the SoftGrip system. The robotic task sequence for the harvesting process will be derived by interviewing mushroom harvesters. The robotic operation for approaching, grasping, out-rooting and placing in a tray will be broken down into tasks and subtasks whose execution and error handling (e.g. grasping fails, outrooting fails, mushroom drops from the gripper while transfer to the tray, etc.) will be supervised and tracked by the Supervision Unit show in the following Figure 2.2. To this end, the entire SoftGrip process will be modelled as a sequence of states and is supervised by a finite state machine, which will be developed and modelled as a directed graph. Each node of the graph will correspond to a state, and each directed edge will correspond to an event that triggered a change of state and optimally some associated action (for example grasping failure, repeat the tasks or terminate the process, etc.).

2.2 CONTROL SYSTEM

2.2.1 BLOCKS OF CONTROL ARCHITECTURE

The control system shown in Figure 2.2 comprises three main control blocks: (1) The motion controller of the rigid robot; (2) the grasping controller activated for the fine grasp of the mushroom; and (3) the out-rooting controller activated for performing the tilt and twist actions that outroot the mushroom. The motion controller of the rigid robot is responsible for the position control of the robot so that the gripper moves close to the mushroom. The positioning comprises two steps: the coarse positioning for approaching the mushroom and then the fine-tuning just above the mushroom. The grasping controller is responsible for planning and controlling the grasping process. This involves the pre-shaping of the gripper fingers for acquiring a finger opening and a curve suitable for grasping the mushroom. The grasping controller is responsible for closing the grip and grasping the cap of the mushroom and runs in combination with the rigid robot controller in order to provide the necessary degrees of freedom during grasping. The outrooting controller is responsible for the tilt and twist motions that break the roots of the mushroom. It is combined with the rigid controller so that the soft gripper executes the combined motion in 3D space required for successful outrooting.

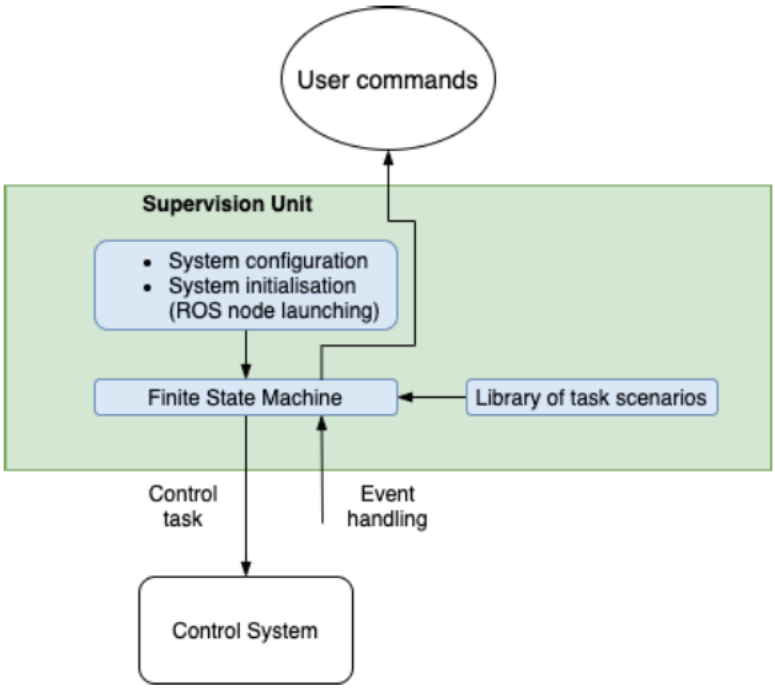

Figure 2.2. Supervision Unit block

All 3 modules of the control system are executed on the central control station and communicate with the rest of the system components through the hardware-interface software module (also running in the Central computer). The latter hosts software libraries for implementing the communication protocols between the central computer control software and the external control components (e.g. microcontrollers, drivers, etc.).

The architecture of the control system will be designed according to a centralised management model of control and will be layered. The central control system, running on the central station PC, is the high level of the architecture and manages the execution of all soft real-time processes associated with task supervision, machine vision, planning and high level control. The centralised system controller decides the start and the termination of processes depending on the system state variables. It checks whether other processes have produced information to be processed or to pass information to them for processing. The controller will loop continuously, polling the nodes and processes for events or state changes (e.g. grasping is achieved, outrooting is achieved, etc.). The low-level control layer that is hard real-time controls the operation of the sensors and actuators and runs on the embedded system. Further details of the control architecture are discussed in Section 4. Software Architecture.

2.2.2 ARCHITECTURE IMPLEMENTATION

The Robotic Operating System (ROS) middleware lends itself very well for implementing the centralised management model of control due to its publish-subscribe protocol. A central node that hosts a finite state machine will initiate, monitor and terminate the rest of the nodes (processes) concurrently, as shown in Figure 2.3.

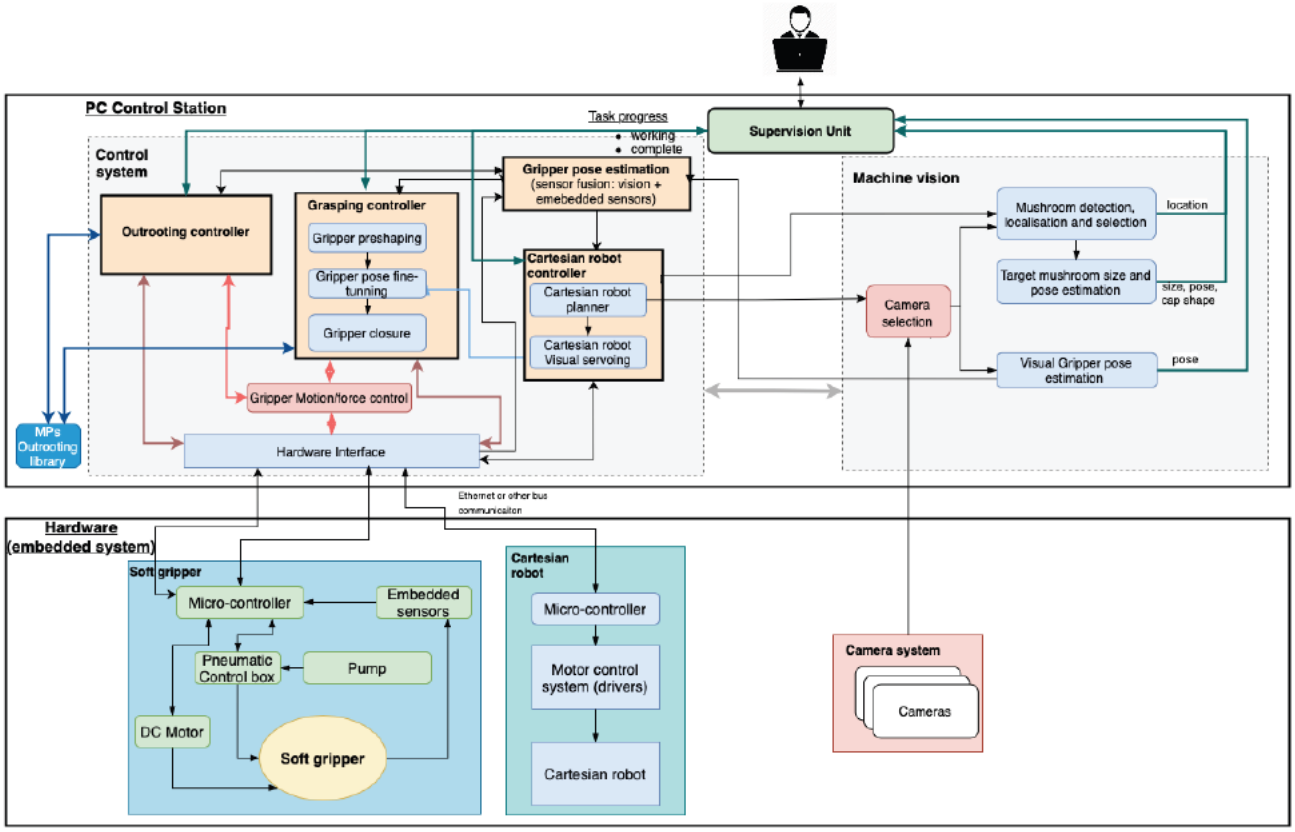

Figure 2.3. Analytical functional architecture of the SoftGrip system.

2.3 MACHINE VISION SYSTEM

The machine vision system comprises the camera system and the machine/computer vision software system. The camera system captures RGB and Depth images, out of which a suitable subset of images is selected according to the input from the Cartesian robot controller block with the aid of the camera hardware interface. The subset of images pictures the gripper and the target mushroom at each time providing the necessary information to the algorithmic blocks of mushroom recognition and gripper visual pose estimation. The visual information is fused with the sensing and rigid robot motor information to estimate the relative poses of the target mushroom and the gripper, which serves as feedback to the control system. The camera system and machine/computer vision software system are further described in the sections 3.3 and 4.1.

3 PHYSICAL/HARDWARE ARCHITECTURE

The hardware architecture comprises the main robotic part and the camera system.

The main robotic part includes two robotic devices: (1) a rigid robot mounted over the shelves that moves in x-y-z axis and (2) the soft gripper which is attached on the end-effector of the rigid robot through the wrist and faces the mushroom cultivation. The rigid robot with the gripper will be installed over the cultivation and will be able to reach any position in the workspace. The architecture includes also the PC, the drivers, the power sources and the sensors used for the operation as well as the closed loop control of the two robotic devices.

The camera system comprises a set of cameras, the related interface components and possibly lamps.

3.1 RIGID ROBOT

The rigid robot will be designed to offer the required degrees of freedom for achieving the approach strategy towards grasping the mushroom cap. To this end the design will offer at least the 3D translational degrees of freedom (dof) and if required we will offer two additional rotational dof (pitch and yaw) to provide the angle of attack. However, it should be mentioned that our goal is to deliver a soft gripper that incorporates rotational dofs sufficient to implement the angle of attack. The rigid robot rotational dofs are a backup plan in case the soft gripper rotational dofs are not sufficient. Hence, in total, the rigid robot will offer at most 5 dofs, as shown in the following figure. The robot will be installed on the shelf roof and its workspace will cover the entire growing area.

The actual design (be it a cartesian robot or a gantry robot) will be decided in WP3, Task 3.4. At this stage the architecture of the hardware components and of their interconnections are schematically presented in Figure 3.1.

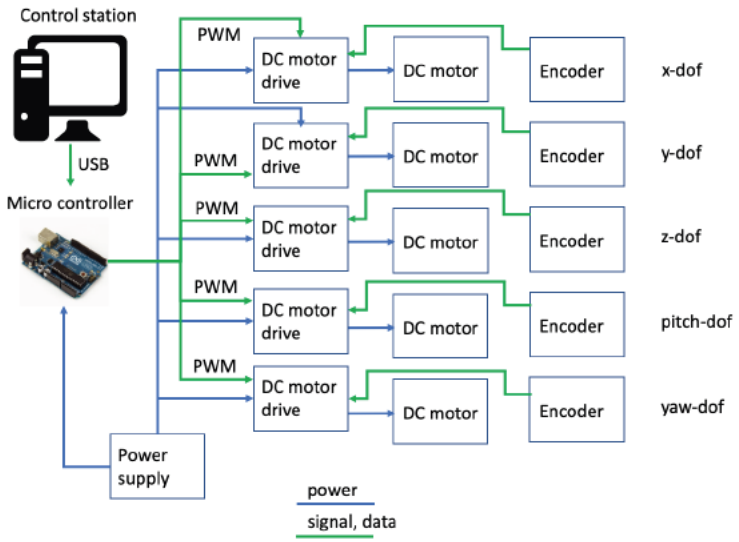

Figure 3.1. Components of the cartesian robot and their interconnection.

3.2 SOFT ROBOT

The soft gripper comprises two subsystems, namely the fingers, responsible for grasping the mushroom caps, and the wrist, responsible for performing the outrooting primitive. In addition to that, the soft gripper architecture includes the actuation, pneumatic source, power supply and micro-controller for the low-level control of the device. At the time of writing, there are still two options for the actuation of fingers (pneumatic or cables driven

by a servomotor), but the general architecture is not affected by this decision. The soft gripper architecture is reported in Figure 3.2, and in the remainder of this section, a brief description of each block is provided.

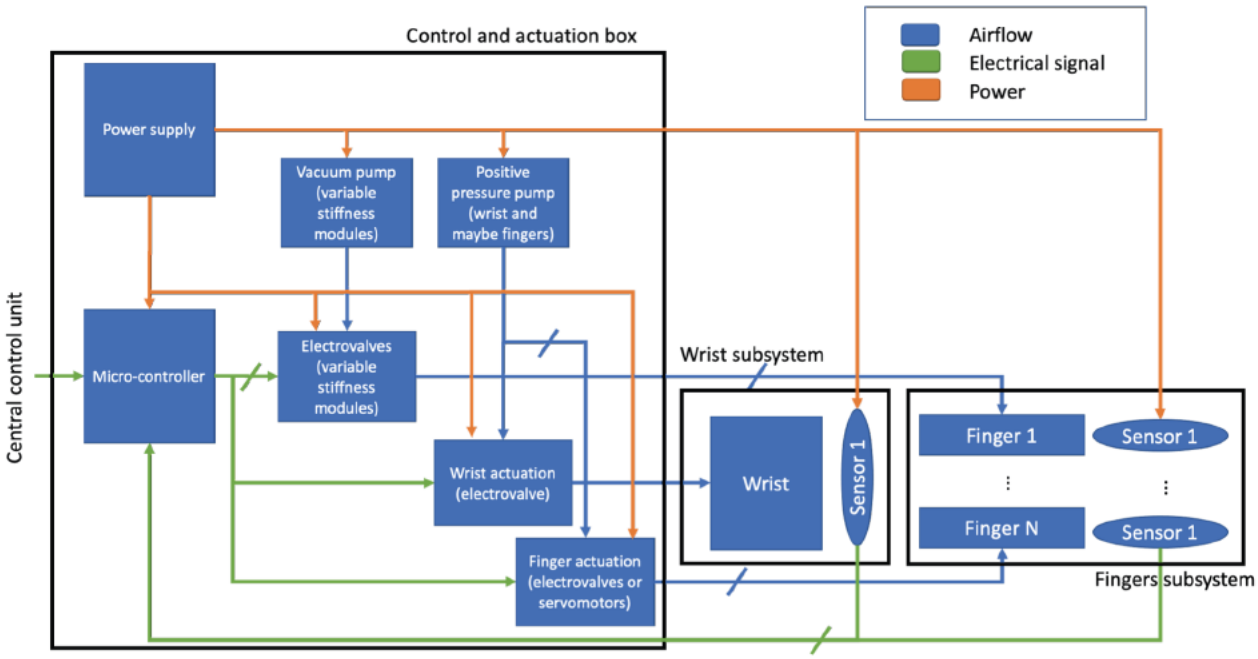

Figure 3.2 Architecture of soft-gripper system.

Power supply: The power supply is responsible for powering the micro-controller, the pneumatic sources for positive pressure and vacuum, the electro-valves and (depending on the final actuation choice) the servomotors, as well as the sensors on the wrist and the fingers. In this block AC/DC and DC/DC converters to regulate the voltages required are included.

Vacuum pump: The vacuum pump provides negative pressure to variable stiffness modules integrated in the fingers. It will be powered by the Power Supply and directly connected to the electro-valves responsible to activate/deactivate the aforementioned modules. The variable stiffness modules are not shown in a dedicated block as they are part of the Finger's design.

Positive pressure pump: The positive pressure pump provides positive pressure to the pneumatic actuator in the wrist and (decision pending) fingers.\ It will be powered by the Power Supply and directly connected to the electro-valves responsible to activate/deactivate the aforementioned actuators. The soft actuators for the wrist and fingers are not shown in a dedicated block as they are part of the Finger's design.

Micro-controller: The micro-controller is responsible for the low-level control of the soft gripper device. It will receive set-points from the central control unit and data from the sensors on the wrist and fingers, and it will regulate the electro-valves and (decision pending) servomotors.

Electro-valves (variable stiffness modules): A set of electro-valves are responsible for activating/deactivating the variable stiffness modules. They will be powered by the Power supply, connected to Vacuum Pump and driven by the Micro-controller. The number of electro-valves depends on the number of independent variable stiffness modules which in turn depends on the final finger design.

Electro-valves (wrist): A set of electro-valves are responsible for activating/deactivating the pneumatic actuators in the wrist. They will be powered by the Power supply, connected to Positive Pressure Pump and driven by the Micro-controller. The number of electro-valves depends on the number of independent pneumatic actuators in the final wrist design.

Finger actuation: Actuators for the fingers' subsystem. At the time of writing the actuation technology of the fingers has not been decided, so that this block can be either a set of electro-valves, connected similarly to the Electro-valves for the wrist, or a servomotor to pull cables passing through the fingers.

Wrist: A pneumatic actuator is responsible for implementing the out-rooting primitive, i.e. a combination of twisting and elongation/tilting. Depending on the final wrist design, the twisting and elongation/tilting degrees of freedom will be coupled or independent. It is connected to the Electro-valves (wrist) and to the Wrist sensor. The wrist will be physically connected to the fingers by a rigid palm (not shown in the architecture).

Wrist sensor: The wrist sensor quantifies the outrooting forces exerted by the wrist. It will be powered by the Power Supply and feed the data to the micro-controller.

Finger: A pneumatic or tendon driven actuator is responsible for grasping the mushroom to be harvested. It is connected to the Finger actuation and to the finger sensor. The finger will be physically connected to the wrist by a rigid palm (not shown in the architecture). The number of fingers depends on the final design.

Finger sensor: Sensor to quantify the forces/pressures exerted by the fingers on the mushroom cap. It will be powered by the Power Supply and feed the data to the micro-controller.

3.3 CAMERA SYSTEM

The vision system comprises a set of cameras that acquire RGB and Depth images which carry the necessary information for the visual estimation of a mushroom's size, position and orientation as well as the gripper's pose. The spatial configuration of the cameras ensures multiple views and coverage of the necessary field of view of the workplace i.e. mushroom cultivation and robot.

3.3.1 CAMERA SELECTION

The camera specifications requirements have been determined according to the use case functional requirements, the mushroom cultivation characteristics, the tunnel environmental conditions and the Dutch Shelving specifications described in the deliverable D1.1 *SoftGrip use case description, functional requirements, and standards* [1] as well as the cost of the commercially available cameras. The identified significant camera specifications for the SoftGrip Machine vision system are listed below.

A. Camera specifications requirements

1. Minimum depth distance
2. Field of view (FoV)
3. Spatial RGB and Depth resolution
4. Resistance to outdoor conditions i.e. humidity
5. Low-light performance
6. Frame rate

A set of indicative specification values that fulfil the SoftGrip system requirements are the following. Minimum depth distance ~35cm, FOV: V ± 45o/ ± 30o or servo + camera, RGBD resolution HD, resistant to relative humidity up to 91%, low-light camera → long shutter speed, relatively high frame rate.

The final values of the specifications along with the selected cameras will be presented in the deliverable D4.1 *SoftGrip machine vision system.*

B. Candidate camera solutions

A family of RealSense candidate cameras have been examined, such as the L515, D455, D435i/D435, D415 and SR305. Candidate cameras that fulfil the above indicative specification values in the RealSense family of cameras are the L515 and the D435 cameras, since the ideal range for the depth sensing is within the specifications (i.e. minimum depth distance ~35 cm).

3.3.2 CAMERA SPATIAL CONFIGURATION

Taking into consideration the candidate camera specifications, the Dutch Shelving characteristics as well as the data requirements of the candidate computer vision algorithms, possible spatial configurations of the

cameras have been explored. Each configuration describes the number of cameras, their positions and angles with respect to the cultivation bench. It ensures full coverage of the entire field of view of the workplace, multiple views for the target mushrooms and adequate quality of the acquired images i.e. minimum distortion. Multiple camera configurations are likely to fulfil these requirements and in order to assess each of them (e.g. eliminate blind spots and give an insight into the number of the cameras needed to cover the whole area of interest) we have developed a camera simulation tool, which will be presented in WP4 in further detail.

A configuration of static cameras has been considered in the current version of architecture. Two candidate configurations, one with pairs of opposite cameras (Figure 3.3) and one with alternating cameras (Figure 3.4), have been simulated and are presented below. Since the mushroom cultivation shelf is quite long i.e. 30 m length, the camera configuration has been designed for a shorter workcell (~5 m) which follows the Dutch Shelving requirements in terms of width (1.2 m) and distance from the proximal shelves (0.6 m). The workcell will repeat along the length dimension in order to cover the entire Dutch Shelf.

1. Opposite-pair cameras simulation

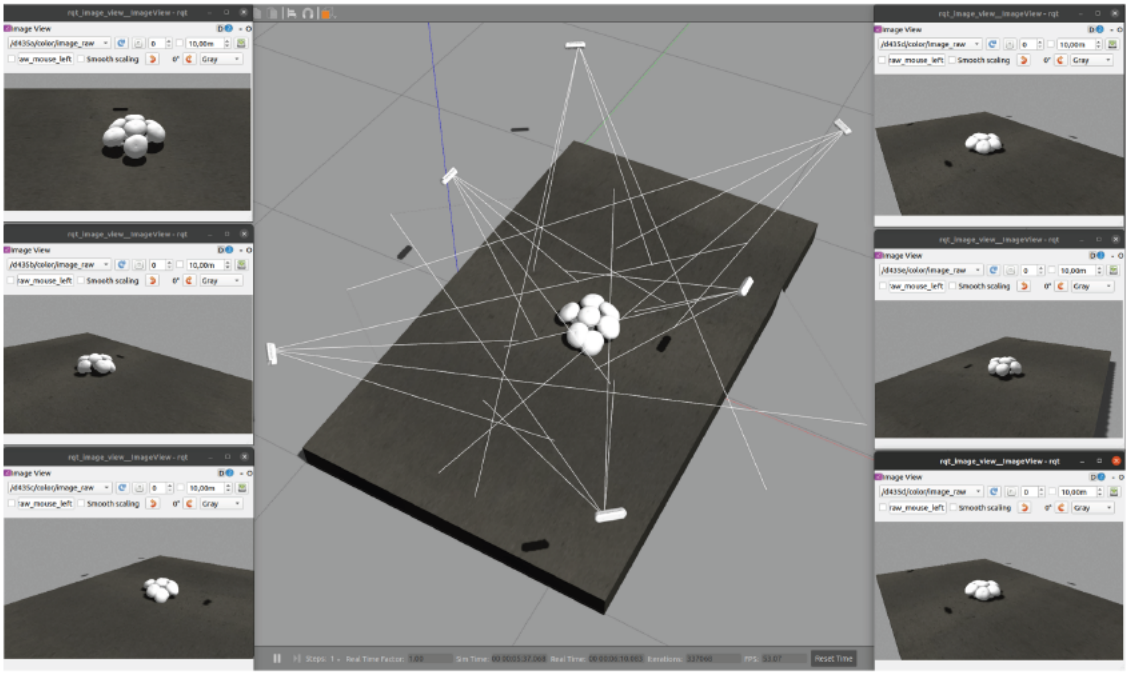

Figure 3.3. Machine vision system simulation for a workcell with opposite-pair camera configuration and the corresponding camera views. Six realsense D435i cameras in opposite pairs are added in a Gazebo simulation, together with a white mushroom on a draft version of a shelf. The six smaller windows alongside the figure present the FoV for each camera.

2. Alternating cameras simulation

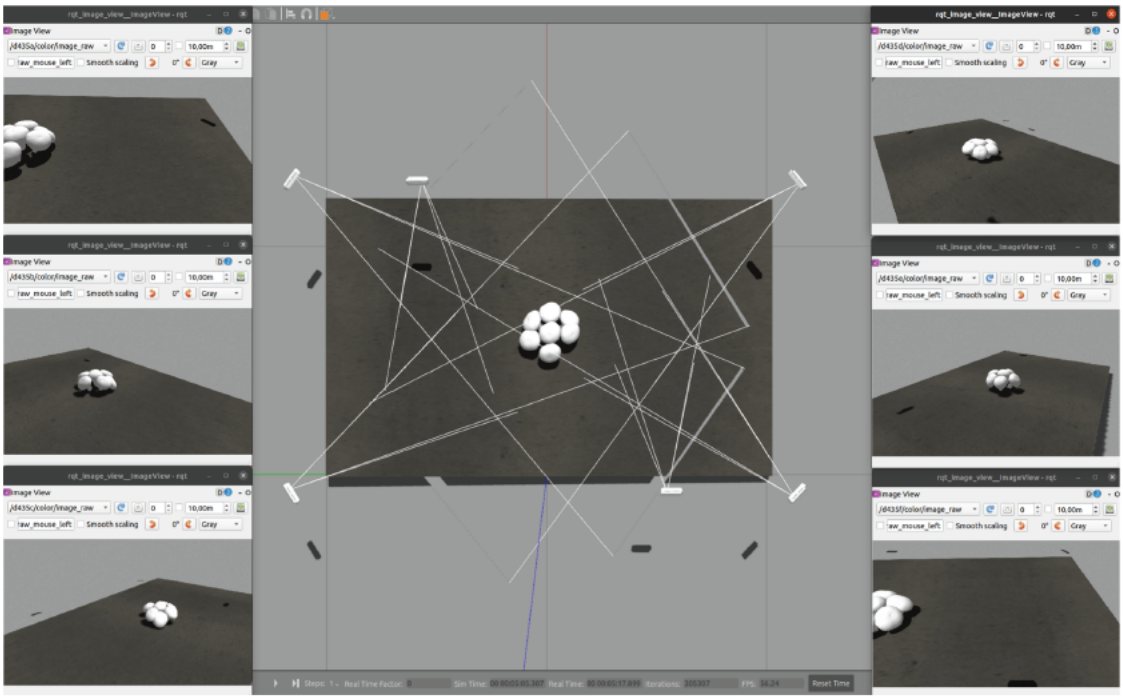

Figure 3.4. Machine vision system simulation for a workcell with alternating camera configuration and the corresponding camera views. Six realsense D435i cameras, one in each corner and a series of alternating cameras along the long sides of the shelf have been added in a Gazebo simulation, together with a white mushroom on a draft version of a shelf. The six smaller windows alongside the figure present the FoV for each camera.

For the current version of the architecture, the opposite-pair camera configuration has been selected, because this provides the highest number of mushroom cultivation views. As simulations and computer vision algorithm exploration progress, the camera configuration design will be updated and most likely simplified. As a next step, the simulations will be refined to take into account further parameters of the realistic cultivation such as the height of the compost, lightning, mushrooms variations and configurations etc.

Possible future additions to the machine vision system are 1) (static) servo-cameras to cover larger FoV 2) (mobile) camera attached to an overhead gantry 3) (mobile) camera attached to the gripper.

3.3.3 CONNECTION INTERFACES

In this paragraph, different camera interfaces and configuration will be investigated. This is a coupled problem since there are constraints such as the maximum cable lengths, interface bandwidth and processing capabilities of the PC used that have to be respected for a robust solution. Except for the obvious centralized choice where all cameras are connected to the main PC, there are other configurations that are worth examining.

Candidate solutions are the following.

1. **Distributed:** Modular approach, every camera is connected on a single board computer (SBC). Camera frames are processed locally and are added to the ROS network.
2. **Centralized:** All cameras are connected to a high-end computer, e.g.: Intel NUC, and all the processing needed is performed there. The maximum number of cameras to be connected on the NUC requires further analysis.
3. **Daisy-chain:** Synchronize one master device with multiple subordinate devices. This is an Azure Kinect feature, but can be implemented by developing the appropriate software for the daisy-chain configuration.

3.3.3.1 DISTRIBUTED ARCHITECTURE

Figure 3.5 presents the topology of 3 RealSense cameras that are connected to a client PC over the network (left) and a similar topology for the ZED cameras (right). The same configuration can be easily extended to multiple cameras connected over the network. The main advantage of this configuration is that it is modular and easily expandable. If one module works according to the specifications, then the solution can be expanded to larger or smaller shelves with many cameras.

Figure 3.5. The RealSense cameras are connected via USB 3.x to a local SBC which will perform all the processing needed to transfer the camera frames over the network to the main PC.

3.3.3.2 CENTRALIZED ARCHITECTURE

A classic approach is the centralized one where the cameras are connected to high-end PCs via USB 3.x. USB 3.0 is the third major version of the Universal Serial Bus (USB) standard for interfacing computers and electronic devices. Among other improvements, USB 3.0 adds the new transfer rate referred to as SuperSpeed

USB (SS) that can transfer data at up to 5 Gbit/s (625 MB/s), which is about 10 times faster than the USB 2.0 standard. In addition, USB 3.1, released in July 2013, is a recent version of the USB (Universal Serial Bus) standard for connecting computers and electronic devices. It is capable of data transfer speeds up to 10Gbps, and while it can use the USB-C connector type, it can also use a variety of other connector types. To achieve USB 3.1 transfer speeds, the USB host connection, cables, and device must all support USB 3.1. USB 3.1 is also known as USB 3.1 Gen 2 (10Gbps).

Unlike previous standards, the USB 3.x standard does not specify a maximum cable length, requiring only that all cables meet an electrical specification: for copper cabling with AWG 26 wires, the maximum practical length is 3 meters.

Candidate camera interfaces

All candidate cameras presented below support USB 3.0 or USB 3.1 which is enough to transfer point clouds and high resolution images to the high-end computer that will perform the gripper and mushroom pose estimation. Specifically, the cameras investigated are the following:

1. RealSense D455: USB-C 3.1 Gen 1 [3]
2. RealSense D435 USB-C 3.1 Gen 1 [4]
3. ZED 2: USB 3.0 port with 1.2m integrated cable [5]
4. ZED mini: USB 3.0 Type-C port [6]
5. Azure Kinect: USB3.1 Gen1 [7]

Bandwidth and CPU

When connecting multiple cameras to a single USB 3.0 hub, the first limitation to consider is the bandwidth limitation of the host system. The USB3.0 “SuperSpeed” interface of the cameras in principle supports 5Gbps. However, accounting for the encoding overhead, the raw data throughput is 4 Gbit/s theoretically. According to Intel, it is reasonable to achieve ~3.2 Gbit/s in practise [8]. In addition, a device that requires more than ~30% of the bus bandwidth needs to be considered carefully and the end users will only expect one device to be in use at a time [9] [7]. Therefore, in general, care should be taken to stay well below 0.3x4Gbps=1200Mbps to ensure robust continuous streaming. This assumes that no other devices share the same USB port controller. Also, note that while many computers will have multiple USB connectors, this is sometimes achieved by using an internal USB hub as opposed to having multiple independent controllers. As a result, the number of independent USB 3.x controllers on a PC is crucial for the task and should be selected based on the number of cameras that will be used.

Attaching multiple cameras will also require processing power from the host system for reading the USB ports, streaming the high-bandwidth data, and doing some amount of real-time post processing, rendering, and analysis. When streaming high resolution color and depth, the processing requirements quickly add up, and will impose another important limitation that needs to be considered. As a result, care should be taken to select a computer platform that supports the intended workload (e.g. high-end NUC, etc).

3.3.3.3 DAISY-CHAIN CONFIGURATION

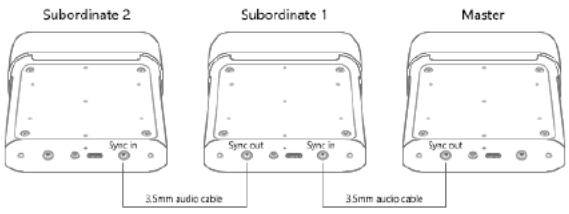

Figure 3.6. Daisy chain configuration of the Microsoft Kinect Azure. The same topology is possible with other camera types but require additional development since according to our research only the Kinect Azure offers such a feature.

In the Daisy Chain network, one camera is connected to the next without any intervening devices, thus the message is sent from one camera to the next and so on. A Daisy Chain can be linear or ring. The daisy chain network is one of the simplest network topologies. Concerning the star topology, all nodes are individually connected to a central connection point, like a hub or a switch. In both configurations, the master device provides the triggering signal for the subordinate devices. Azure Kinect is the only one out of the five options presented in the previous paragraph that supports both features out of the box. If any of the rest of the options is chosen, it would require extra effort for the software development part.

3.3.4 SELECTION OF SOLUTION

There are many reasons to use multiple camera devices, as explained below:

- Fill in occlusions. For example, in the following diagram (Figure 3.7), the left-side camera sees the gray pixel "P2." However, the white foreground object blocks the right-side camera IR beam. The right-side camera has no data for "P2." Additional synchronized devices can provide the occluded data.

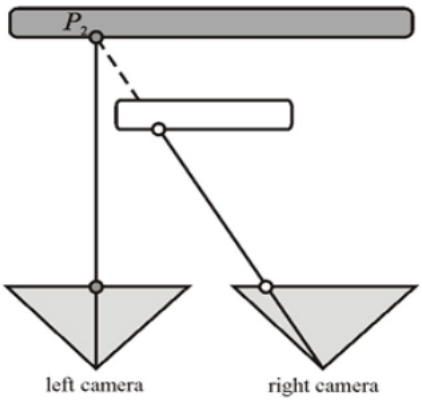

Figure 3.7. Star configuration of the Microsoft Kinect Azure. The same topology is possible with other camera types but require additional development since according to our research only the Kinect Azure offers such a feature.

- Scan objects in three dimensions.
- Increase the effective frame rate to a value that is greater than 30 frames per second (FPS).
- Capture multiple 4K color images of the same scene.
- Increase camera coverage within the space.

Since SoftGrip is a research project, development time should be taken into great consideration and the solution should be expandable with robust performance. The *distributed* architecture is a trade-off between expandability and development time. It can be adapted easily to multiple Dutch shelves, utilizing the inherent ability of ROS to enable data exchange over a network of nodes. Last, the size of each module (camera and SBC) is not the optimal one, but still it is not prohibitive for this task.

3.4 PHYSICAL COMMUNICATION INTERFACES

This paragraph presents the main options for a physical communication bus responsible for transferring data between the central computer and the hardware components of SoftGrip or among the embedded system microcontrollers, sensors and actuators. The protocols for the communication between the central PC and the embedded system will be implemented within the hardware interface module (shown in Figure 2.2) The main criteria for selecting the communication bus are listed below:

- Physical robustness

- Noise immunity
- Data rate
- Ease of design
- Open software
- Size
- Cost

Typical communication bus options are:

- Embedded buses, SPI, I2C
- USB 2.0
- USB 3.0 (and onwards)
- Ethernet 10BASE/100BASE
- Ethernet 1000BASE
- Ethernet 10G
- CAN-FD
- RS485/422
- RF (Wi-Fi/Bluetooth)

Table 1 evaluates the bus options according to the list of criteria presented above.

Table 1. Physical communication interfaces.

BUS	Robustness	Noise immunity	Max Data rate	Ease of Design	Open Software	Size	Cost
I2C/SPI	Depends on design	Poor	20Mbit/s	Easy	Good	Depends on the design	Low
USB 2.0	Good	Good	480Mbits/s	OK	OK	Small	Moderate
USB 3.0	OK	OK	10Gbit/s	Hard	Hard	Moderate	High
Ethernet 10/100 base	Good	Good	100Mbit/s	OK	OK	Large	Moderate
Ethernet 1000 base	Good	Good	10Gbit/s	Hard	Hard	Large	Moderate
Ethernet 10GE	Good	Good	10Gbit/s	Hard	Hard	Huge	High
CAN-FD	Depends on design	Excellent	8Mbit/s	Easy	OK	Depends on design	Low
RS485/422	Depends on Design	Excellent	10Mbit/s	Easy	Good	Depends on design	Low
RF	Good	Poor	54Mbit/s	Hard	Good	Small	Moderate

Preliminary discussion on physical communication bus selection: Typical data rates for motor control usually require a refresh rate around 500Hz and involve 16 bytes packet size. This means a data rate estimation around 8kB/s for motor control of the rigid robot or of motors driving tendons (if tendons are used). Furthermore, typical requirements for a central control station or main embedded computer for control and monitoring are a refresh rate at 10Hz and a packet size of 256 Byte. This results in a data transfer rate of around 3 kB/s. Low-level MCU with sensors such as those that will be used in the SoftGrip architecture, require a refresh rate around 10Hz transmit 32Byte per package and therefore require around 6kB/s.

A rough estimate of the SoftGrip embedded system indicates that it will be equipped with 5-6 motor controllers or servovalves controllers, one central computer for monitoring and control and supervision, around 5-6 MCU with sensors. The camera system is not considered here, since this will be equipped with its own physical communication bus, as discussed in a previous paragraph. For this indicative case the maximum theoretical data rate required is given by: 6x8kB/s+6x0.32kB/s+3kB/s = 53kB/s = 423,36 kBits/s. Hence, a reasonable option would be the CAN bus or at most the Ethernet 10/100 base.

3.5 HARDWARE 3D OVERVIEW

The following Gazebo simulation demonstrates part of the hardware architecture including the machine vision system (cameras), the gantry for the rigid robot and a cluster of mushrooms on a workcell of a shelf. The first three images (Figure 3.8, 3.9 and 3.10) present the hardware without the cameras to give a clear view in the candidate configuration. In the last two images (Figure 3.11 and 3.12), six cameras are added. The corners of the Dutch shelves are candidate locations for the cameras. The same goes for the moving part of the Gantry as presented in the last two figures. The final position of the cameras will be chosen after a detailed analysis, taking into account all possible constraints, e.g. field of view.

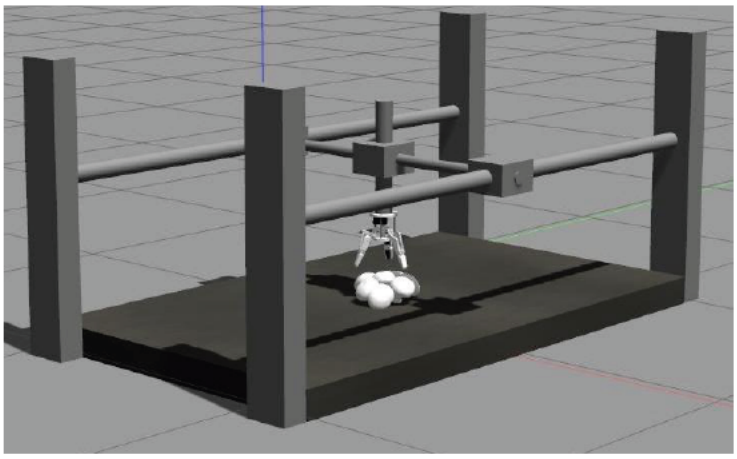

Figure 3.8. View from an angle of the gantry robot and the grippers

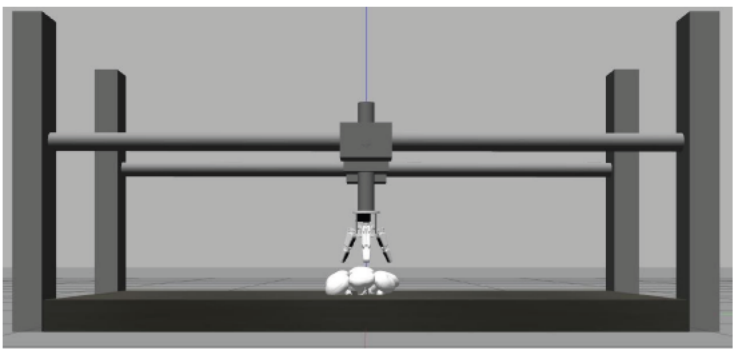

Figure 3.9. Side view of the gantry robot and the gripper.

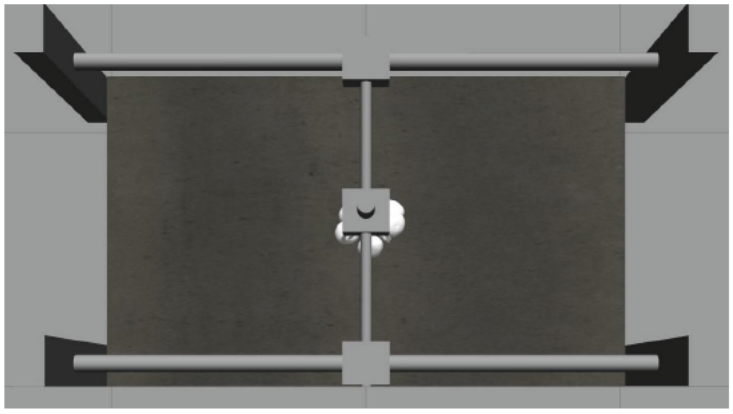

Figure 3.10. Top view of the gantry robot and the gripper.

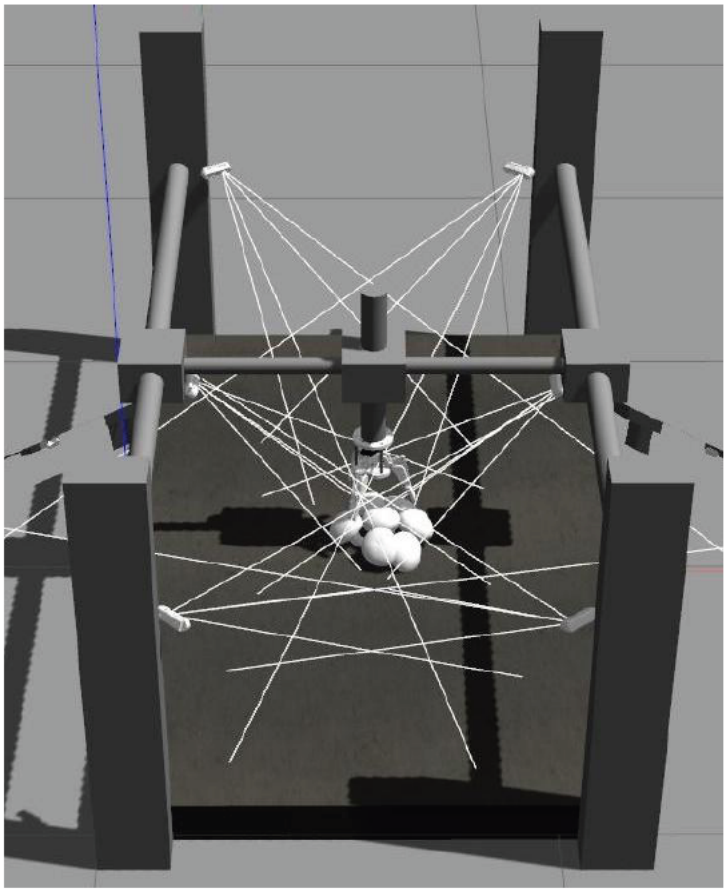

Figure 3.11. View of the gantry robot and the gripper with six cameras.

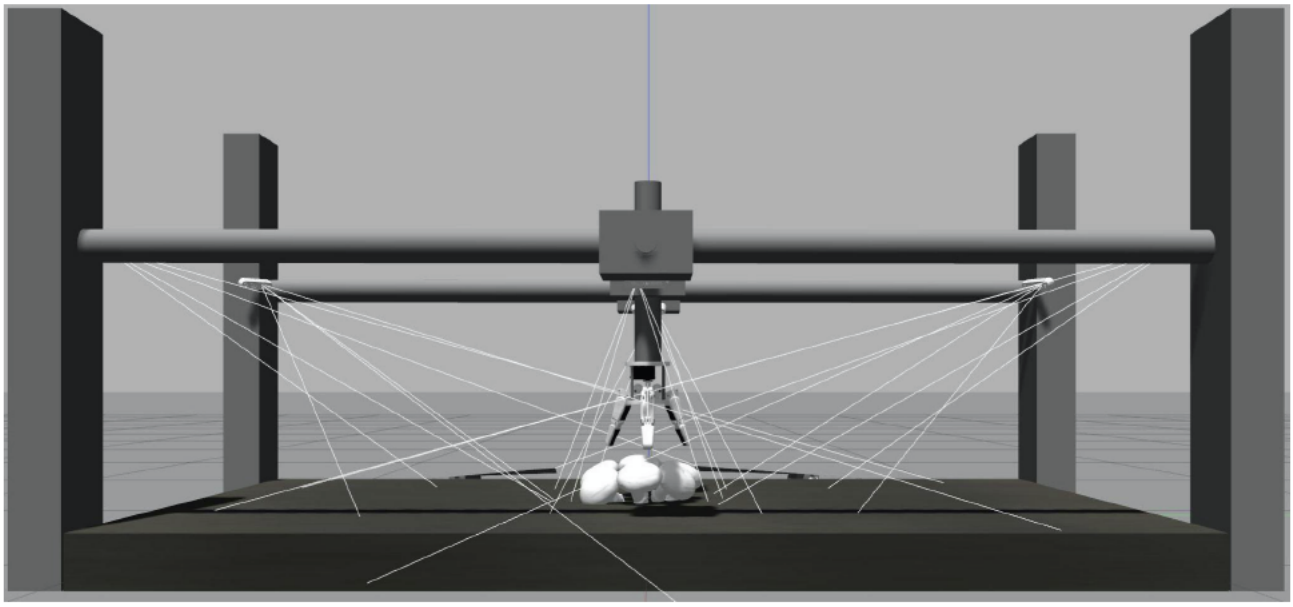

Figure 3.12. Side view of the gantry robot and the gripper with six cameras.

4 SOFTWARE ARCHITECTURE

The architecture of a software-intensive system is the structure of the system, which comprises software elements, the externally visible properties of those elements, and the relationships among them [10].

4.1 COMPUTER VISION

The computer vision software supports the node of the machine vision system in Figure 2.2., which performs two tasks: 1) mushroom recognition and 2) gripper pose estimation. The interfaces of the node are described below:

- Input:
  - RGBD images (from multiple views) in the proximity of the gripper
- Outputs:
  - target mushroom's location, size, pose and cap shape
  - gripper's pose

The input to the machine vision system is a subset of RGB and Depth images coming from the camera system described in Section 3.3. The subset of images is selected each time in the proximity of the robot and includes images that capture 1) part of the mushroom cultivation or 2) the gripper. These images are used for 1) the recognition of the target mushroom, 2) the recognition of the gripper (Figure 4.1). Potentially images that capture both the gripper and the mushroom cultivation will be acquired, if we proceed with the additional task of refining the relative positions and orientations i.e. poses between the mushroom and the gripper using visual feedback.

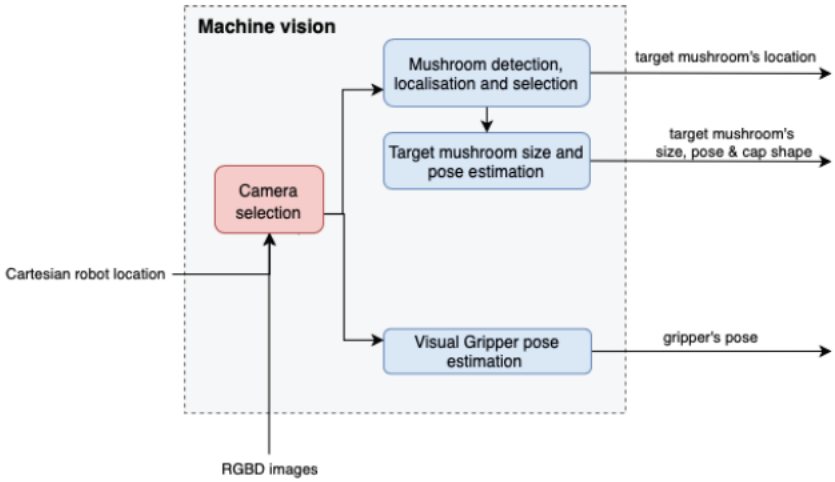

Figure 4.1. Machine vision system.

4.1.1 MUSHROOM RECOGNITION

The mushroom related tasks corresponding to the upper part of the machine vision system diagram can be further analysed into a pipeline of image processing tasks. The diagram in Figure 4.2 describes the image processing pipeline for mushroom recognition and feature extraction that is designed for the current version of SoftGrip architecture.

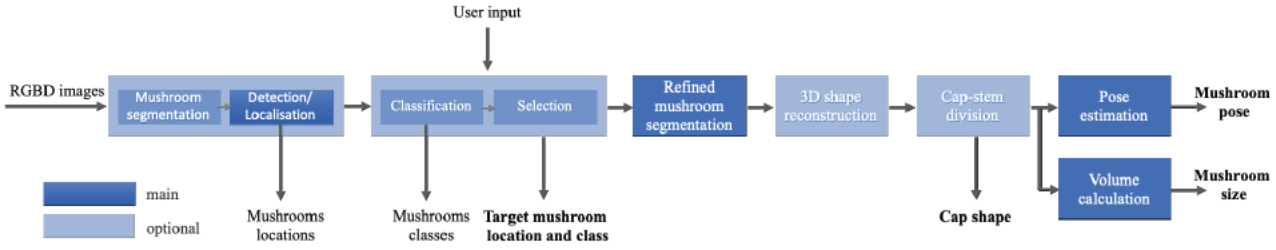

Figure 4.2. Image processing pipeline for mushroom recognition and feature extraction. The bold blocks represent the necessary image processing tasks, whereas the faded blocks represent optional image processing tasks.

The incoming RGBD images first undergo a ***rough segmentation*** to facilitate the ***detection*** and ***localisation*** of candidate mushrooms on each image. Then the candidate mushrooms are potentially ***classified*** into classes according to their suitability for harvesting as described in *D1.1* [1] to facilitate the following ***selection*** of the target mushroom. The target mushroom undergoes ***refined segmentation*** to extract its mask, in multiple viewpoint images. This information is combined to ***3D reconstruct*** the mushroom volume, which is then ***divided*** into the ***cap and stem*** volume with a 3D segmentation or recognition approach. The cap ***volume*** is used to calculate the cap's ***size***, while the cap and stem volumes are used to ***estimate*** the position and orientation i.e. ***pose*** of the target mushroom.

The designed image processing pipeline includes the maximum number of main image processing steps that may be required for the mushroom recognition task *T4.1*. This pipeline may be modified in the course of the project, possibly to include fewer steps, according to the following factors: 1) the form of testing cultivation i.e. level of similarity to a realistic mushroom cultivation, 2) the level of automation vs user input in the selection of the target mushroom, 3) the algorithms employed in the preceding image processing and 4) the algorithms' sensitivity to camera configuration and data availability i.e. the dataset size and level of annotation (particularly in deep learning techniques). The mushroom recognition pipeline will be further analysed and implemented in WP4 Task 4.1.

4.1.2 GRIPPER POSE ESTIMATION

The lower part of the machine vision system diagram refers to the visual estimation of the pose of the gripper. The visual information provided by the RGBD images is fused with the sensing information i.e. pressure, position, acceleration and potentially the motor encoders' information in order to provide an accurate estimate of the gripper's pose with respect to the shelf as well as the target mushroom. It is worth noting that the visual information about the gripper serves *complementary* to its sensing information, aiming to address the high pose uncertainty of the gripper's soft parts and, this way, to provide a more accurate feedback about the gripper's pose to the control system.

The visual information can potentially be a feed of information about the following phases of the gripper's motion.

1. **The gripper is initialised.** The sensing system verifies the open position of the gripper's fingers. The vision system possibly provides information for the position and orientation of the gripper with respect to the shelf and the mushroom.
2. **The gripper is approaching the target mushroom.** The visual information is important for the servoing of the gripper towards the target mushroom. The sensing feedback verifies the open position of the fingers and the absence of load to the soft gripper.
3. **The gripper is grasping the mushroom.** The sensing system provides information about the pressure exerted on the mushroom and the curvature of the fingers of the gripper. The visual information can potentially assist with the refinement of the alignment of the gripper's pose i.e. position and orientation to the mushroom's pose. It can also provide information about the opening of the

fingers with respect to the mushroom cap size. The visual and mostly the sensing feedback flag the optimal closure of the gripper fingers around the mushroom.

4. **The gripper is outrooting the mushroom.** The gripper tilts and twists gently the mushroom according to the sensing feedback. The visual feedback can potentially steer the pulling of the mushroom in a suitable direction (optional).
5. **The gripper is depositioning the mushroom.** The gripper and the grasped uprooted mushroom are visually servoided towards the collection container (optional).

Possible further additions to the machine vision system for gripper pose estimation is a camera attached to the soft gripper i.e. wrist, in-hand camera or an in-hand depth sensor.

The diagram reported in Figure 4.3 describes the image processing pipeline for the visual pose estimation of the gripper designed for the current version of SoftGrip architecture.

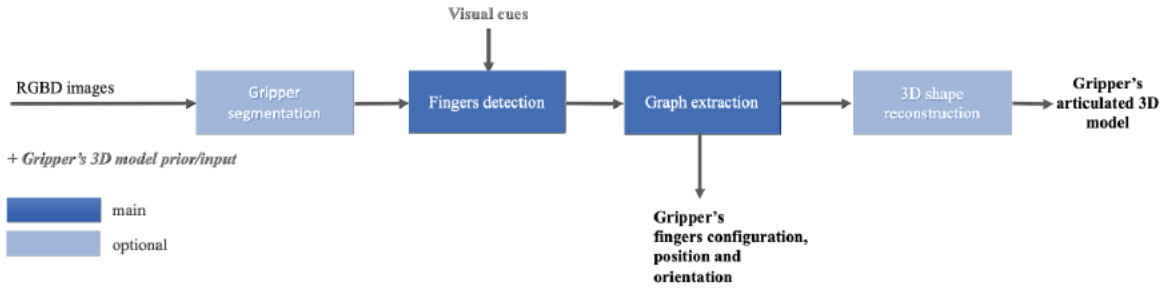

Figure 4.3. Image processing pipeline for gripper visual pose estimation. The bold blocks represent the necessary image processing tasks, whereas the faded blocks represent optional image processing tasks. Similar convention is followed for the input/output.

The input of the gripper visual pose estimation block is a set of RGBD images of the gripper and possibly the target mushroom, acquired from multiple viewpoints. *Visual cues* such as markers on the gripper's tendons are likely to be incorporated and considered as input to the step of the gripper's ***fingers detection***. This information is further processed to extract the ***gripper's graph*** model, which provides the fingers configuration i.e. fingers' position, orientation and interconnection. For a model-based gripper's pose estimation, the ***gripper's 3D model*** may also be incorporated to the graph extraction step, either as prior knowledge about the gripper at resting mode (inherent in the graph extraction block) or as temporal input describing the gripper's shape at the previous state (or time point). The optional step of ***3D shape reconstruction*** of the gripper provides an articulated 3D model of the gripper.

4.2 SKILL ACQUISITION

4.2.1 DATA COLLECTION

The cornerstone of the Skill Acquisition framework will be the datasets collected by expert demonstrations of the mushroom picking task. The following data streams are planned to be obtained:

- Videos from multiple cameras capturing the entire process followed by the expert picker.
- Force profiles measured on the pickers' fingertips by tactile sensors placed on a glove-like device.
- (Optional) Finger tracking measurements from sensors integrated in the glove-like device.

The data above will be post-processed to obtain information on the location and pose of the mushroom being picked, the configuration of the expert's fingers and the force interactions during the outrooting phase of the mushroom picking process.

This information will be leveraged to achieve Learning by Demonstration, which will lay the groundwork for Imitation Learning approaches that will underpin the Skill Acquisition in the outrooting controller. Both of the envisaged architectures share a common structure illustrated in Figure 4.4. Although the schematic pertains to the training phase of the system, the last stage (Policy Refinement on Real Tasks) may extend into the deployment phase facilitating a lifelong learning paradigm.

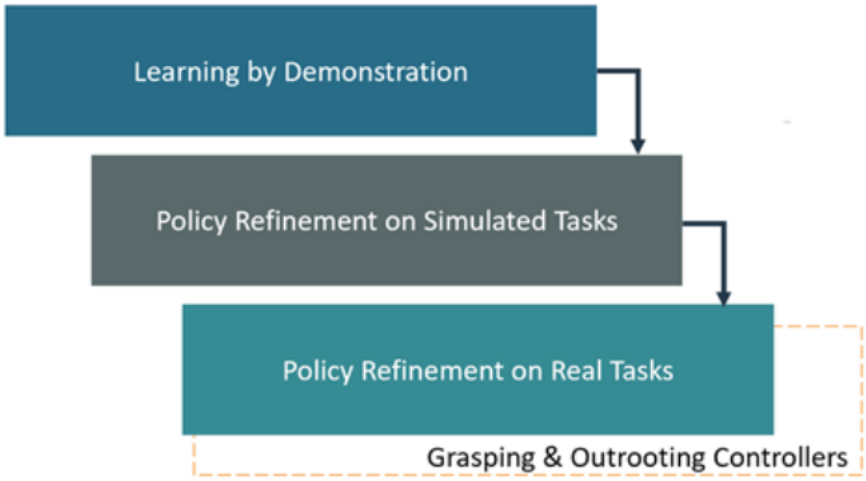

Figure 4.4. Skill Transfer framework overview

The individual tiers shown above will be further explained in the following subsections.

4.2.2 APPROACH 1: MOVEMENT PRIMITIVES EXTRACTION

The architecture of the Imitation Learning approach incorporating Movement Primitives is shown in Figure 4.5.

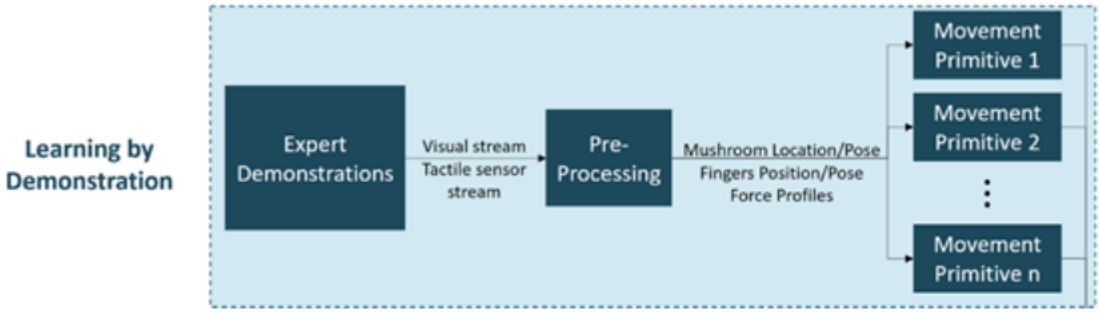

Figure 4.5. Movement Primitive-based demonstration data encoding

The core principle of the approach illustrated above is that Learning-by-Demonstration is accomplished by extracting a vocabulary of movement primitives which can be regarded as the building blocks of the control policy. The Movement Primitives formalise nonlinear dynamic equations that can be flexibly adjusted to construct arbitrarily complex motor behaviours without the need for manual parameter tuning.

Input:

- Expert Demonstrations

Output:

- Library of Movement/Interaction Primitives

4.2.3 APPROACH 2: DIRECT IMITATION LEARNING

Although the above problem formulation is expected to work sufficiently well for capturing the mushroom robot skill, as a next step we will attempt a Direct Imitation Learning approach. This will allow for improved generalisation not only within mushroom picking but across similar tasks of picking other soft produce. This new approach is illustrated in Figure 4.6.

Here, the extraction of Movement Primitives is replaced by a combination of Representation Learning and Supervised Learning that operate on the expert demonstrations. A pre-processing step for extracting meaningful features could be included but the architecture allows for learning directly from the original data streams. This is facilitated by the fact that Imitation Learning is based on the embeddings learned rather than a set of manually extracted features.

The embeddings mapping is also introduced before the Reinforcement Learning modules of the policy refinement stages. In addition to this, there is significant flexibility in terms of the action space formulation; actions may directly represent joint control commands.

A significant addition in this approach is the Multi-task/Meta Learning module, which will span across simulated and real tasks. This will enable structured learning from simpler versions all the way to the fully complex mushroom picking task, allowing for greater extensibility to other contact-rich interaction-based skills.

Input:

- Expert Demonstrations

Output:

- Optimal policy in terms of control commands

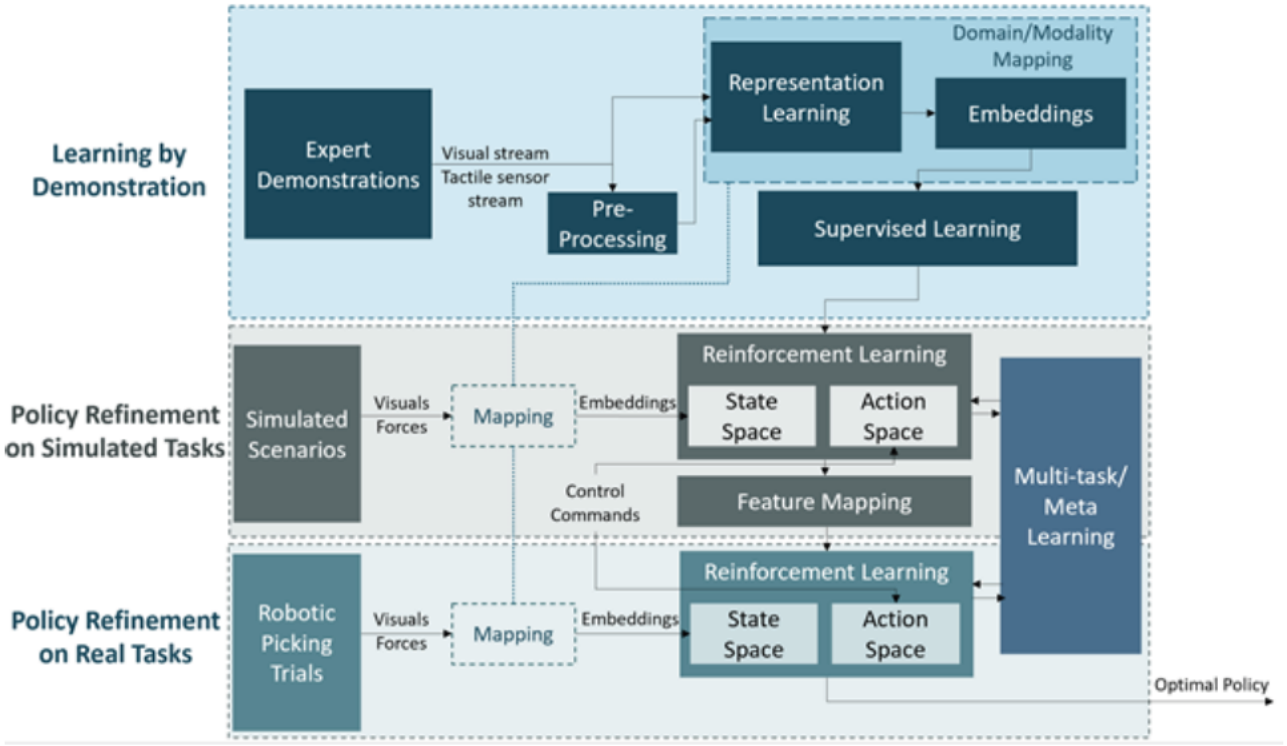

Figure 4.6. Direct Imitation Learning architecture

4.3 CONTROL SYSTEM

4.3.1 TASK PLANNER

Task planner is a ROS node integrated in the software architecture, which is responsible for coordinating the mushroom picking activity and planning the initial and final robotic action, that of the cartesian robot. The interfaces of the node are described in the following:

- Input:
  - Output of Machine vision system including target mushroom's location, size, pose and cap shape and gripper's pose.
- Output:
  - Reference Trajectory for the cartesian robot, to bring the gripper in an optimal initial pose to initiate grasping action.
  - Reference Trajectory for the cartesian robot, to transfer the mushroom to a collection container.

To complete its basic functionality, the Task planner requires the visual information derived both from the mushroom segmentation and the pose estimation of the target mushroom. More specifically, the ROS node implements a simple heuristic to decide which mushroom is the target for picking based on their size labelling and position on the shelf. Following this decision, the pose of the target mushroom is used to determine the optimal relative initial pose of the gripper for a successful grasping. The target pose of the gripper is then used to calculate the reference trajectory of the cartesian robot, which is provided to the low-level motion controller for calculation of the robot actuation commands. This node will then initiate the grasping controller and sequentially the outrooting controller, that constitute the mushroom picking action. A similar approach will provide the second output of the Task planner, which will be activated sequentially to the picking action of the gripper. The planner will calculate in this stage the reference trajectory for the cartesian robot that will start from the picked mushroom position to a goal position above the mushroom collection container.

4.3.2 GRASPING CONTROLLER

The ROS node described as Grasp planner, once initiated by the Task planner, will have the role of calculating the reference trajectory of the gripper's fingers, to achieve a firm and delicate grasp of the target mushroom. The interfaces of the node are summarized below:

- Input:
  - Output of Machine vision system including target and neighbouring mushroom's location, size, pose and cap shape and gripper's pose.
  - Initial Gripper pose/ state model-based estimation.
  - Force sensor values in the tip of each finger.
- Output:
  - Approach #1: Reference Trajectory of each finger, to accomplish a firm grasp of the mushroom.
  - Approach #2: Soft-gripper Actuator-level commands to execute grasping action.

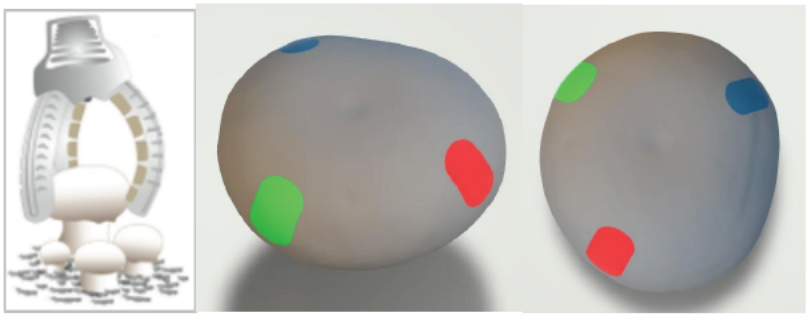

Figure 4.7. The target positions for each finger of the gripper are calculated on the mesh model of the Cup

Grasp planner node will read the initial pose/state estimation of the gripper and will combine this information with the pose and the size of the target mushroom for harvesting, in order to determine the gripper **pre-shaping** form. As a next step, the mesh model of the mushroom's cup is used to calculate in detail the **target placement of each finger** on the cup's surface (Figure 4.7) either analytically or from expert's demonstration data, in order to achieve force closure during in-hand manipulation and a convenient configuration for the implementation of the outrooting actions. Following the calculation of the target finger configuration the **reference trajectory of each finger** will be calculated, starting from the pre-shaping state towards the target configuration on the surface of the cup. This trajectory calculation will take into account the neighbouring mushrooms which will be considered as obstacles to be avoided during the motion of the fingers, to prevent any unwanted damage by the motion of the soft gripper. Having the reference finger trajectories calculated in the task space, research will be conducted on two main approaches as a final step of the Grasp planning algorithm:

1) *Approach #1 (Model-Based Actuation):* The reference trajectory of each finger will be provided as an **input to the Actuation level Model-Based Motion/Force controller**, which will determine the robot actuation commands for the trajectory tracking. The Grasping action will be terminated upon achieving the designated force values at the tip of each finger.
2) *Approach #2 (Model-Free Actuation):* The reference trajectory of each finger in the task space will be encoded with Probabilistic Motion Primitives both in task and actuation space. This encoding will be based on prior targeted demonstrative grasping actions executed by the gripper, which can generate the required mapping between the task and actuation space. The combination of such encoding with modern Reinforcement learning techniques for online motion adaptation in case of execution discrepancies, can **generate directly the desired actuation commands** at each time step. The Grasping action will be terminated upon achieving the designated force values at the tip of each finger.

4.3.3 OUTROOTING CONTROLLER

The ROS node described as Out-rooting Controller, once initiated by the Task planner, will have the role to calculate the reference motion of the gripper and the cartesian robot, to accomplish a blemish free outrooting of the target mushroom, based on the expertise of professional mushroom harvesters. The interfaces of the node are summarized below:

- Input:
  - Output of Machine vision system including target mushroom's location, size, pose and cap shape and gripper's pose.
  - Initial Gripper pose/ state model-based estimation
  - Library of Motion Primitives
  - Force sensor values in the tip of each finger
- Output:
  - Approach #1: Reference Trajectory of each finger, to accomplish outrooting of the mushroom.
  - Approach #2: Soft-gripper Actuator-level commands to execute out-rooting.
  - Approach #3: Task or Actuator-level command

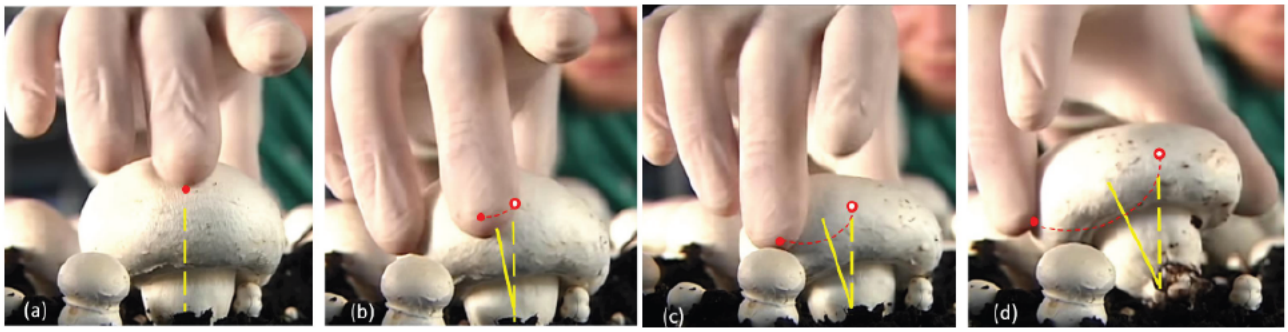

Figure 4.8. Fresh mushroom picking. The image sequence (a)-(d) illustrates a combination of tilting and rotating the mushroom grasped with 2 or 3 fingers, to achieve out-rooting.

Outrooting is a complex action, which involves a combination of mushroom tilt and twist actions, requiring a very delicate in-hand manipulation, as shown in Figure 4.8. Thus, the sequence of outrooting actions will be derived from data encoding techniques (described in Section 4.2), capturing the expertise of human harvesters both in terms of motion and interaction force properties. Once the outrooting action is segmented into phases, are encoded with an **Interaction Primitive**, which encodes both desired motion and force profiles. The full outrooting action will be reproduced as a sequence and composition of Interaction Primitives in the task space, which are **adapted to the target mushroom pose and size** by the outrooting controller, calculating simultaneously the reference motion for the robotic device (both the gripper and the cartesian robot) and the reference forces on the mushroom's cup at each time step. The controller's output can similarly be described with two distinctive approaches:

Approach #1 (Model-Based Actuation): The reference trajectory and desired force profile of each finger will be provided as an **input to the Actuation level Model-Based Motion/Force controller**, which will determine the robot actuation commands for the trajectory and force profile tracking. The Out-rooting action will be terminated upon the completion of the whole sequence of motions, maintaining the designated force values at the tip of each finger.

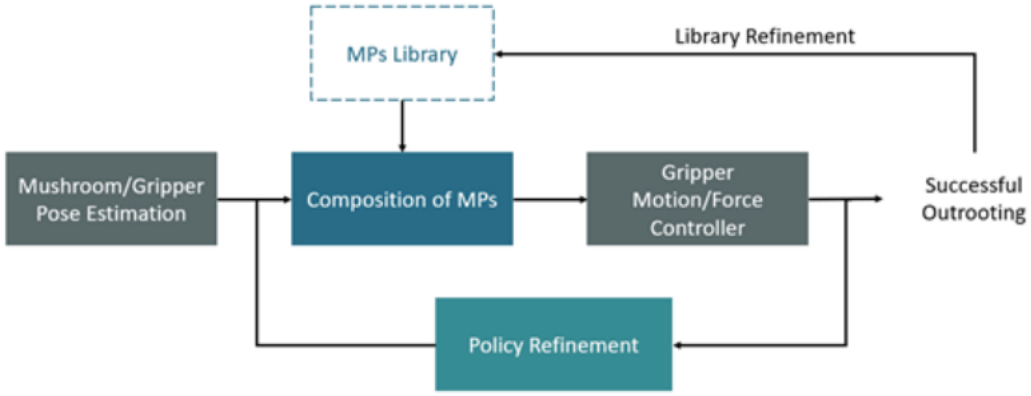

Figure 4.9. Approach #1 outrooting controller architecture

Approach #2 (Primitive-based Actuation): With a Movement/Interaction Primitive library in place, outrooting can be cast as a problem of identifying the optimal composition of the Movement/Interaction Primitive components such that the picking task is achieved.

The resulting representation will be based on prior targeted demonstrative outrooting actions executed by the gripper both in simulated and real settings, which can generate the required mapping between the task and actuation space. The combination of such encoding with modern Reinforcement learning techniques for online motion adaptation is required in order to cope with execution discrepancies. The refined control policy will thus generate the **desired actuation commands for the Soft Gripper** to complete the outrooting task. The

Outrooting action will be terminated upon the completion of the whole sequence of motions, maintaining the designated force values at the tip of each finger.

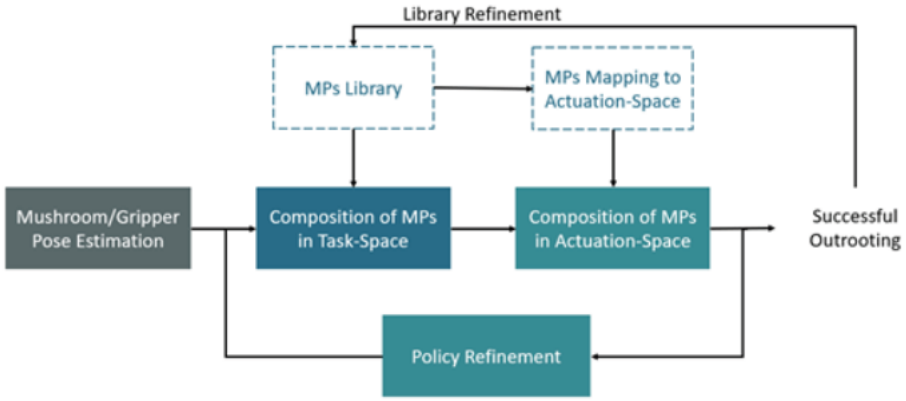

Figure 4.10. Approach #2 outrooting controller architecture

Approach #3 (Actuation based on Imitation learning): In this approach we will attempt to develop imitation learning algorithms adopting a looser data-driven perspective; instead of prescribing a predetermined structure in the form of Movement Primitives, we will investigate the implementation of a Representation Learning module that will produce embeddings of the sensorial streams allowing for Reinforcement Learning to take place over a reward function constructed based on the embedding space. Such an approach would be significantly more complex in terms of designing the learning algorithms to achieve convergence but it would potentially enable a huge advantage in terms of generalisation.

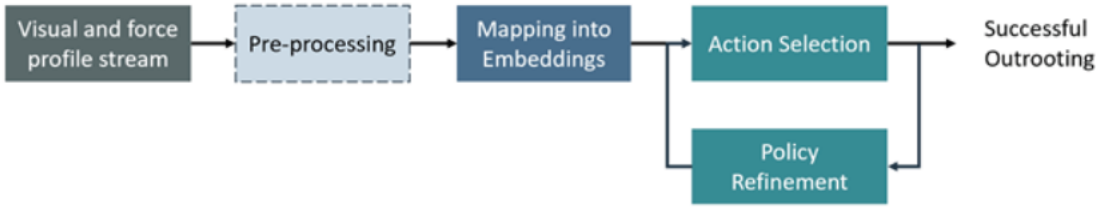

Figure 4.11. Approach #3 outrooting controller architecture

4.3.4 OUTROOTING SKILL TRANSFER WITH SIM2REAL TECHNIQUES

Taking into account the recent major research trends in robotics and more specifically the breakthroughs in robotics control and manipulation [11-18] we treat the topic of Simulation to Reality (Sim2Real) separately, devoting a special section describing our approach and proposed software architecture for Sim2Real in the context of Imitation Learning as described in sections 4.2.2 and 4.2.3 for SoftGrip.

The “Sim2Real” aims to provide a concrete set of definitions, tools, techniques, experimental setups, data management [19,20] for the optimal policy design in systems with non-trivial hard to identify and control dynamics.

Apart from being able to achieve state-of-the-art levels of performance, a set of additional reasons make Sim2Real a very appealing methodology for optimal policy design in the context of SoftGrip:

1. **Bootstrapping and Model-Based Design approach**. Initial hypothesis testing and early-stage software testing and debugging within a Model and Software in the Loop Testing.
2. **Data starvation**. Although a set of demonstration data as well a set of rich and real visual data from the field should be available shortly in the project timeline, the generation of synthetic data should accelerate the development rate and given the research nature of the project a way to quickly evaluate numerous hypotheses and imitation learning algorithms "as-we-go" would be of immense value. Additionally, the synthetic data should be able to teach the imitation learning policy the dominant dynamics of the outrooting task allowing the expert demonstrations for fine-tuning in a meta-learning context.
3. **Accessing hidden world states**. For robust policy design accessing hidden world states and variables is very valuable for learning a task invariant.

We propose a set of software architectures based on current state-of-the-art techniques for Sim2Real in the context of the SoftGrip project. These techniques are not mutually exclusive, they can be combined, and it is already anticipated that a combination and possibly an extension should be able to effectively assist the Imitation Learning Pipeline in the task of robust and optimal mushroom outrooting.

4.3.5 SYNTHETIC DATA GENERATION BASED ON SIMULATION FOR DEFORMABLE OBJECTS WITH FEM

As an initial starting point, we base the synthetic data generation on the pybullet simulator drawing inspiration from the work conducted in the Volumetric Grasping Network [15].

This approach allows for:

1. Rapid prototyping with python for quick hypothesis evaluation
2. Generation of data "as-fast-as-possible" in a non-real time fashion
3. Ability to deploy with ROS the trained policy in real robotic setups for initial experimental evaluation.

An extension based on finite element analysis for simulation of deformable objects is now part of bullet (Figure 4.12) [21] which directly applies to the SoftGrip project.

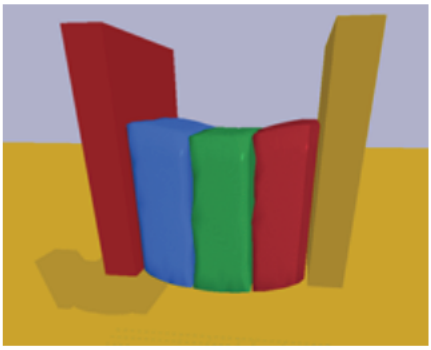

Figure 4.12. Deformable Object Manipulation -- Simulation in Bullet [17]

4.3.6 DOMAIN RANDOMIZATION (DR) FOR SOFTGRIP

Domain Randomization (DR) is one of the predominant techniques for Sim2Real. DR stands behind major breakthroughs in the field [11-12,18]. The major principle is simple, namely, gradually change key scene geometry and dynamics parameters while the agent is training. This should force the agent to learn the necessary generalizations and abstractions, therefore being more robust and ultimately performing better after deployment to the real world.

It should be noted though that DR is relatively a very recent technique (if we assume that is new) and the case for DR is still under investigation from the research community. There have been cases where it was argued that DR was not a key component for a successful deployment to the real application [15, 22, 23].

We propose an adaptation of the automatic domain randomization (ADR) [11] during the training phase of the imitation learning policy (Figure 4.13).

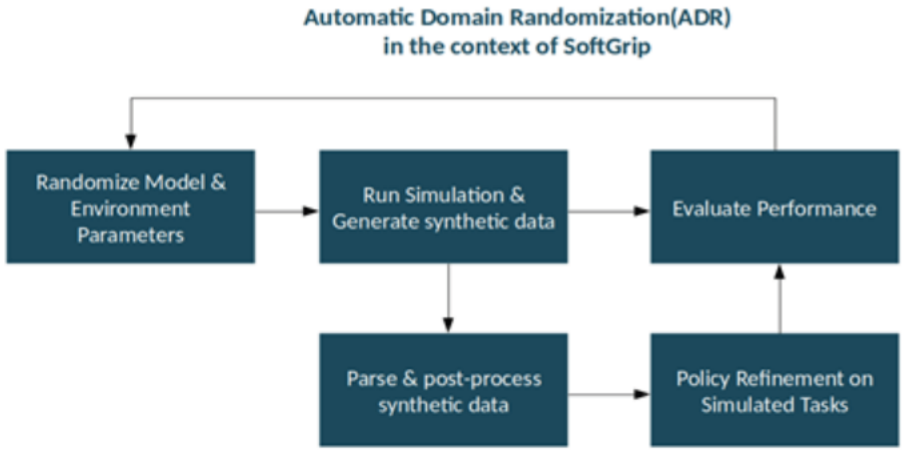

Figure 4.13. Automatic Domain Randomization in the context of SoftGrip

4.3.7 REAL3SIM – MODELING ACTUATION DYNAMICS WITH DEEP REINFORCEMENT LEARNING

We expect the actuation dynamics of the SoftGrip for hardware and software combined to exhibit complex dynamic behavior which is practically intractable to efficiently model and estimate with traditional analytical methods. Although a model of the soft gripper would be available later down the line, a set of residual dynamics would still remain unmodelled. These phenomena we estimate to be particularly important during the uprooting and an a priori knowledge incorporated in the imitation learning pipeline would give a significant, even critical, performance boost to the trained agent as demonstrated in [14].

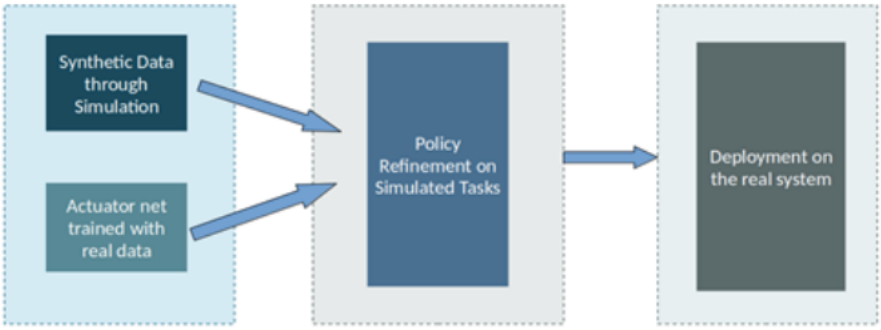

Figure 4.14. Real2Sim Learning SoftGrip actuation dynamics. Incorporation of learned actuator dynamics into the learning pipeline.

4.3.8 DOMAIN ADAPTATION FOR SIM2REAL

From a systemic point of view the discrepancy or "domain shift" between the simulation and real world can be treated as a domain adaptation problem [25]. Additionally, a very important aspect of the imitation learning architecture is the different embodiment/physical correspondence between the hand of the expert that will provide the demonstrations and the target robotic hardware. The domain adaptation architecture should also be able to make the necessary mapping by learning the necessary goal-state-action abstractions for tackling the so-called "correspondence problem".

Essentially the product of the domain adaptation architecture described in the current subsection is the module "feature mapping" as described in sections 4.3.2 and 4.3.3.

We proposed the scheme reported in Figure 4.15 for learning the feature mapping for the imitation learning for SoftGrip inspired from the work conducted in [26].

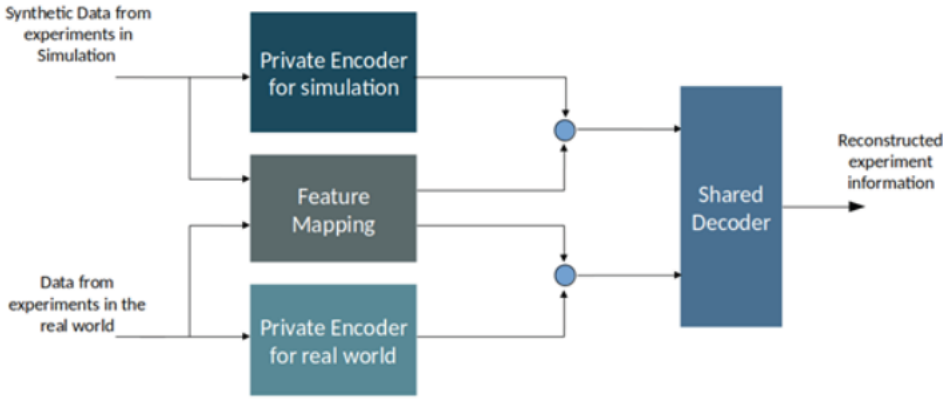

Figure 4.15. Feature Mapping for domain shift handling

4.3.9 MOTION/FORCE CONTROLLER

The low-level Motion/Force Controller is a ROS package, consisting of several nodes which are responsible for calculating the actuation commands both for the rigid robot and the soft robotic gripper.

Rigid Robot Controller

- Input:
  - Reference motion in Task space obtained as an output of Grasping or Outrooting controllers
- Output:
  - Commands for the joints of the Cartesian robot

The motion controller of the rigid robot is a ROS Node responsible for the position control of the robot so that the gripper moves close to the mushroom. The positioning comprises two steps: the coarse positioning for approaching the mushroom and then the fine-tuning just above the mushroom.

Actuation level Model-Based Motion/Force controller

- Input:
  - Reference motion in Task space
  - Reference force profiles in the tip of each finger
- Output:
  - Actuation level commands to the Soft-Gripper

The Model-Based Motion/Force Controller in the actuation level is a ROS node, which is responsible for the execution of the reference motions and force profiles generated from the Grasping and the Outrooting controller. The calculation of commands in actuation level is achieved with the use of accurate modelling of the Soft-Gripper's kinematic structure employing a continuum mechanics model, which allows for fast simulation for the solution of the configuration for given actuation commands. Then the continuum mechanics simulation is formulated as a successive approximation problem where a set of initial actuation commands is provided and after a few iterations converges to the solution of actuation commands that yields the desired soft gripper configuration.

5 CONCLUSIONS

The current deliverable presented the SoftGrip architecture description, which is divided in three parts: first an overview of the functional architecture (section 2), then the hardware architecture (section 3), and finally the software architecture (section 4). The functional architecture was presented at a high level of abstraction and includes all SoftGrip functional modules and their interconnections. The hardware architecture presented the modules and the components of the rigid robot, the soft robot and its subsystems (embedded system, actuation/sensing), the camera system and the central PC that hosts the software components, as well as the physical communication interfaces among components and subsystems. The section of the hardware architecture also presented a 3D CAD illustration of each component integrated on the Dutch shelves. The software architecture presented the modules of the machine vision and the control system including the grasping controller, the outrooting controller and the skill transfer architecture. Finally, the candidate architectures for the implementation of the simulation-to-real algorithms were presented and compared.

This deliverable serves as an introductory work on the design of the system, and more importantly, it offers a documentation of the system and its components so that all partners have a common understanding of the SoftGrip system. This is the first version of the architecture and will be updated as the project progresses.

REFERENCES

[1] Deliverable D1.1 *SoftGrip use case description, functional requirements, and standards*

[2] Huang, Mingsen, et al. "Development of A Robotic Harvesting Mechanism for Button Mushrooms." Transactions of the ASABE (2021): 0

[3] https://gr.mouser.com/new/intel/intel-realsense-depth-camera-d455/

[4] https://www.intelrealsense.com/depth-camera-d435/

[5] https://www.stereolabs.com/zed-2/

[6] https://www.stereolabs.com/zed-mini/

[7] https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification

[8] https://dev.intelrealsense.com/docs/multiple-depth-cameras-configuration

[9] https://dev.intelrealsense.com/docs/multiple-depth-cameras-configuration

[10] Nick Rozanski, Eóin Woods, Software systems architecture: working with stakeholders using viewpoints and perspectives, Addison-Wesley, 2005.

[11] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. (2019). Solving Rubik's Cube with a Robot Hand.

[12] Fereshteh Sadeghi, and Sergey Levine. (2017). CAD2RL: Real Single-Image Flight without a Single Real Image.

[13] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. (2020). Deep Drone Acrobatics.

[14] Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. Science Robotics, 4(26), eaau5872.

[15] Michel Breyer, Jen Jen Chung, Lionel Ott, Roland Siegwart, and Juan Nieto. (2021). Volumetric Grasping Network: Real-time 6 DOF Grasp Detection in Clutter.

[16] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. (2020). TossingBot: Learning to Throw Arbitrary Objects with Residual Physics.

[17] David Hoeller, Lorenz Wellhausen, Farbod Farshidian, and Marco Hutter. (2021). Learning a State Representation and Navigation in Cluttered and Dynamic Environments.

[18] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. (2020). Learning to Manipulate Deformable Objects without Demonstrations.

[19] Höfer, Sebastian, et al. “Perspectives on Sim2Real Transfer for Robotics: A Summary of the R:SS 2020 Workshop.” ArXiv:2012.03806 [Cs], Dec. 2020. arXiv.org, .

[20] S. Höfer et al., "Sim2Real in Robotics and Automation: Applications and Challenges," in IEEE Transactions on Automation Science and Engineering, vol. 18, no. 2, pp. 398-400, April 2021, doi: 10.1109/TASE.2021.3064065.

[21] Bullet 2.89 with FEM deformables, PyBullet deep_mimic and Laikago https://github.com/bulletphysics/bullet3/releases/tag/2.89

[22] Zhaoming Xie, Xingye Da, Michiel van de Panne, Buck Babich, and Animesh Garg. (2021). Dynamics Randomization Revisited:A Case Study for Quadrupedal Locomotion.

[23] J. Dao, Helei Duan, Kevin R. Green, J. Hurst, and Alan Fern (2020). Learning to Walk without Dynamics Randomization.

[24] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. (2017). Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping.

[25] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. (2018). One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning.

[26] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. (2016). Domain Separation Networks.