## Executive Summary

1.2 Partners involved
1.3 Relation with other work packages and tasks
1.4 Links to videos, flyers, ...
1.5 Dissemination / IPR policy

## LettuceThink

2.1 Sensors
2.1.1 Cameras
2.1.2 End-effector position
2.1.3 Artificial Whiskers
2.2 Actuators
2.2.1 X-carve stepper motors
2.2.2 Camera actuator
2.3 Experimental set-up
2.3.1 Adaptive learning system in support to 3D plant reconstruction
2.3.2 Learning of terrain features using artificial whiskers
2.4 Evaluation criteria

## Aerial robot

3.1 Evaluation criteria

## References
1 Executive Summary

Work Package 4 addresses the development of adaptive learning algorithms for agricultural robots, their experimental evaluation and adaptation to the LettuceThink micro-farming platform (developed by SONY CSL) and to the aerial drone (developed by IAAC).

This Deliverable provides a report on the experimental set-up of the adaptive learning system that will be developed throughout the ROMI project. This document is divided into two parts: section 2 is related to the LettuceThink platform; section 3 is related to the aerial drone. We define sensors and actuators to be used within this project, as well as the experimental set-up and a preliminary architecture for the adaptive learning system.

1.2 Partners involved

Leader: UBER
Participants: CNRS, Sony, Inria

1.3 Relation with other work packages and tasks

Relation to WP5: WP4 provides a novel, adaptive approach to information gathering and camera placement that will be used in combination with the techniques developed in WP5.

1.4 Links to videos, flyers, ...

TODO

1.5 Dissemination / IPR policy

TODO.

2. LettuceThink

Learning algorithms will be developed for the control modules of the LettuceThink robot, with the aim of generating intelligent behaviours that move the camera around the plants and maximise the usefulness of the information obtained by the image sensors.

The LettuceThink platform, developed by SONY CSL, consists of a metal frame with an x-carve CNC machine mounted in it. The x-carve is used to provide 3-axes movements to a depth camera mounted at the tip of the vertical z-axis (from now on, called the end-effector camera).

During a workshop carried out in Berlin in February 2018, a reproduction of the original platform has been built at UBER for local experiments, in collaboration with partners from SONY. The platform built in Berlin is fixed onto a desk. Pots with growing plants are placed at the basis of the frame, as illustrated in the picture below.
The following image (picture shot by Sony CSL) shows the x-carve being mounted on the metal frame of the LettuceThink. The original set-up included two Raspberry Pi embedded computers, and an x-carve CNC controller. The main Raspberry Pi is used to interface external computers with the CNC controller and the end-effector depth camera. At the top of the frame, a second Raspberry Pi equipped with an RGB camera was mounted. The camera was facing top-down to provide a wider overview of the x-carve and the plants below it.

This deliverable describes the hardware and software modifications that have been performed by UBER on the platform, as well as the experiments planned for the next months on adaptive behaviours and learning.
During this first year of the project, UBER defined a preliminary architecture for the learning system (described in section 2.3), and identified the sensors (section 2.1) and actuators (section 2.2) that will be necessary for the successful implementation of the learning experiments.

The lists of sensors and actuators, as well as the general architecture, may eventually be updated or improved in the next deliverables, should new features or issues raise in the next months.

2.1 Sensors

2.1.1 Cameras

One of the main tasks of the ROMI project is to generate 3D reconstructions of plants. UBER’s task is to provide support for this process - which is implemented by other ROMI partners - through generating robot movements that optimise the information obtained by the camera sensors, and thus enabling a better reconstruction of the plants.

The process of 3D reconstruction is usually based on an estimation of point clouds representing the object. The typical process consists of shooting pictures around the target object using depth or regular RGB cameras, and then of post-processing these images in order to generate the point cloud.

Therefore, the main sensors for this scope are depth/RGB cameras. During the first year of the project, we tested two cameras: Sony Depthsense and Intel Realsense D435 depth cameras. We chose the Intel camera, since it has better performances when working outdoor compared to the Sony camera. Since the learning algorithm is still not integrated with the 3D reconstruction tool developed by the ROMI partners, it is still not decided whether RGB images or depth images will be ultimately the data to be used for the point cloud generation. We are using at the moment the Intel camera, also since it can detect smaller minimum distances with the depth sensor, compared to the Sony camera. This feature is essential when the end-effector camera is moved too close to the plant.

Moreover, a second Intel RealSense D435 depth camera (from now on, called the top camera) is mounted at the top of the LettuceThink metal frame, to provide a wider overview of the moving x-carve and of the plants. The Intel D435 is substituting the RPi camera mounted on the embedded Raspberry PI, as in the original LettuceThink setup.

The following screenshots show an Intel Realsense Depth Camera D435.

The following figures show a sample screenshot grabbed from the top camera (left image) and the point cloud generated with the depth sensor (right image), where RGB information is overlapped on top of each point. This set-up includes: a real plant (Aloe Vera, characterised by slow growth), visible in the top left part of the image; a fake plastic plant, visible on the right part of the image; some more objects, distributed on the table.
2.1.2 End-effector position

As described above, the LettuceThink platform is equipped with an x-carve CNC machine. The position of the end-effector camera needs to be determined, in order to generate proper adaptive behaviours, as it will be described in section 2.3. The actuators that are currently used in the x-carve machine (stepper motors, as described in section 2.2) are not equipped with rotary encoders. Nonetheless, the 3D position of the end-effector can be retrieved from the x-carve controller. This can be done by sending grbl status requests through the serial port of the Raspberry Pi to the x-carve controller. Grbl\(^1\) is a free, open source, high performance software for controlling the motion of machines and, in particular, of the x-carve. Grbl runs on the Arduino mounted in the x-carve controller. It is written in optimized C utilizing the features of the Arduino’s Atmega328p chips to achieve precise timing and asynchronous operation. While commands are sent to the stepper motors through G-Code (see section 2.2.1), requests about the status of the machine - which include information about the positions of the motors - can be sent from the companion PC (i.e. the main Raspberry Pi of the LettuceThink). These requests, consisting of messages sent through the serial port, are interpreted by the grbl software running on the Arduino board of the x-carve controller. In particular, in order to get the position of the x-carve motors, a status report is asked through a grbl WPos (working position) request, which provides the offset of each motor from the initial position, estimated by counting the steps performed from the beginning of the operation. To ensure obtaining correct and consistent positions, it is required that the x-carve is sent to the home position before further operations.

We developed a driver for controlling the x-carve actuators and for getting their positions using the Robot Operating System (ROS). ROS is a middleware (i.e. collection of software frameworks for robot software development)\(^2\) which is widely used by robotics researchers. It relies on a distributed set of nodes that can be run on the same or different machines and boards, which communicate through messages. ROS facilitates the development and the deploy of software for robots. The drivers we developed are open and freely available at the following github repository: [https://github.com/romi/ros_ws](https://github.com/romi/ros_ws). The repository also includes a ROS driver that wraps RGB and depth images recorded from the Intel D435 into ROS messages, for further use within the ROS middleware.

2.1.3 Artificial Whiskers

We plan to equip the LettuceThink platform also with additional sensory modalities, in order to enable adaptive interaction with plants and terrains which could not be otherwise possible. In particular, we are

---

1. [https://github.com/grbl/grbl/wiki](https://github.com/grbl/grbl/wiki)
developing bio-inspired tactile sensors - i.e. artificial whiskers - that we intend to use for detecting features of plants and of the soil, while the x-carve is operating.

The following figure shows a current version of the artificial whiskers and of the micro-controller board that is used for recording signals from them.

The artificial whiskers are prototyped with an Ultimaker II 3D Printer available at UBER, using flexible PLA material. A whisker, which has the shape of a small cylinder with base diameter of 2mm and length of 5cm, is glued on top of a 15mm piezo-electric disk. When the whisker is touched, a vibration is propagated through the piezo-electric disk, which under mechanical stress, generates an electric charge. This electric charge is converted to an appropriate voltage range through a circuit board developed at our lab. We use the ARM based Nucleo-64 board for interfacing with the raspberry pie (or any computer). We also developed the drivers to integrate the whiskers into the ROS environment.

Section 2.3 will show the whiskers mounted at the tip of the x-carve end-effector, for an experiment on terrain feature detection that we are currently carrying out.

We plan to integrate additional sensors within the x-carve end-effector, including temperature and humidity sensors, as well as ultrasound sensors. The aim is to integrate multi-modal information into the learning process for the better adaptive behaviours.

2.2 Actuators

2.2.1 X-carve stepper motors

The original x-carve CNC machine (see figure below) is equipped with 3 stepper motors for moving the carving bit over the x, y and z axes.
X-carve comes with NEMA 23 stepper motors, displayed in the figure below.

The original range of the vertical axis of the x-carve has been extended by SONY CSL.

As mentioned in section 2.1.2, we developed a ROS driver for controlling the motors of the x-carve, available in a github repository. The driver allows sending target positions to the x-carve, together with the desired speed, as well as sending homing commands. Commands are sent as ROS messages that are published by the ROS node implementing the behaviours strategy of the robot. The ROS driver awaits ROS command messages, puts them into a buffer and sends them to the x-carve controller as G-Code commands.

2.2.2 Camera actuator

We plan to equip the end-effector of the x-carve (the tip of the vertical z-axis) with a small arm, in order to provide more degrees of freedom to the end-effector camera. For this purpose, we plan to use Dynamixel motors, which can be easily integrated and used with the micro-controller board mentioned in section 2.1.3. The following figure shows the motors we will use for the prototype of the camera actuator (still not fully integrated within the LettuceThink platform). Each Dynamixel motor has an integrated encoder and a microcontroller, which controls the motor speed and position. Serial commands can be sent and status data can be read to and from each motor through the ARM board or raspberry pi.

2.3 Experimental set-up

The previous sections have described the set of sensors and actuators to be used in the adaptive learning system. The main aim of WP4, as described in the Description of Work of the ROMI proposal, is to implement adaptive strategies for controlling the movement of the image sensors as they go around and
in-between plants, in two main platforms: the LettuceThink micro-farming robot developed by SONY CSL and the aerial drone developed by IAAC.

This section describes two experiments that will be carried out with the LettuceThink platform in the following months of this project.

2.3.1 Adaptive learning system in support to 3D plant reconstruction

We aim at developing an adaptive learning system that moves the end-effector camera of the LettuceThink in an intelligent way. The intelligent behaviour tries to optimise the information obtained with the image sensors for further post-processing, that is 3D plant reconstruction. During the first year of this project, we designed and partially developed an architecture for this scope.

The aim of the architecture is to generate adaptive behaviours for the camera based on artificial curiosity and goal-directed exploration. Exploration behaviours are an essential means in early stages of development in humans for discovering body capabilities. Through exploring and interacting with their surroundings, infants learn about their own body and action possibilities. These processes have been demonstrated to be promising tools for enabling adaptivity in robots, as well (Schillaci et al., 2016). In fact, robot behaviours based on pre-programmed models of their embodiment often fail, as they lack of the capability to react to unexpected circumstances. Developmental robotics - which tries to enable principles of infant development into artificial systems - offers, instead, new and promising tools for adaptive learning in robots.

The architecture proposed here addresses the challenge of collecting useful images for processes of 3D plant reconstruction. Instead of using predefined trajectories of the end-effector camera around the plant to reconstruct, our architecture aims at directing the movements of the camera towards positions and orientations that produces interesting views of the plant. The system considers a point of view as interesting, if the image it expects to see from that specific camera position and orientation does not match the image that it is actually obtained from the sensor. This mismatch, or prediction error, is used as a sort of surprise signal: the higher the error, the higher the surprise and the need to know more about that. This thus triggers further exploration around that specific point of view (movements of the camera around that specific position and orientation). During this further exploration, the system acquires new knowledge, refines its internal model and thus becomes more skilled in producing better estimations.

A typical challenge in 3D object reconstruction is the presence of occluded parts of the object, which generates imprecise models. Our task is to detect these spots, and to direct the camera towards locations and orientations that uncover the hidden information, thus aiming for a better estimation of the 3D reconstruction of the object.

We thus define an adaptive learning mechanism based on artificial curiosity and exploration behaviour, building up on previous work (Schmerling et al., 2015). The adaptive learning system tries to look at parts of the plant (here defined as goals) and assigns interest factors to them. An interest factor is a function of a prediction error - that is, a function of a dissimilarity measurement between what the system expects to perceive and what it is actually perceived.

The proposed architecture is composed by three main building blocks:

- an interest model, which stores a set of goals. For each goal, the model keeps track of how much is interesting to explore it;
- a controller - also called inverse model - which generates a motor command for moving the camera in order to reach a desired goal;
- a predictor - also called forward model - which anticipates what would be the visual input obtained from the camera if executing an intended motor command.
The following text provides an example of the steps needed for executing a curiosity-based exploration behaviour:\footnote{For the moment, we consider a goal as a screenshot of the plant taken from a specific point of view, and a motor command as a target \(x,y,z\) position of the end-effector of the x-carve. Reaching a goal would mean moving the end-effector camera, so that the visual input recorded from camera matches the goal image. Preliminary tests consider using only the \(x,y,z\) motors of the x-carve. Future experiments will integrate the additional joints of the camera actuator (previously described in section 2.2.2).}

1. The interest model selects the most interesting goal to explore
2. The goal is fed into the inverse model, which generates a motor command \((x,y,z\) target positions) to be sent to the x-carve
3. A copy of the motor command generated in step 2 is fed into the forward model, which estimates what would be the image obtained by the camera sensor after the movement
4. The prediction made in step 3 is compared to the actual observation - i.e. the image obtained by the camera sensor after the execution of the motor command generated in step 2. A prediction error is calculated as the mismatch between prediction and observation. This is used to update the interest factor of the current goal: if the error is high (or if there is a big change between the previous error and the current one), the interest factor is increased. In this case, the system's expectations are not met, thus it is asked to gather more information about that specific goal (i.e. explore more around that point of view)
5. Update the inverse model and the forward model with the new sensory and motor data
6. Go back to step 1

The following diagram illustrates the steps described above.
Throughout the exploration mechanism, different learning processes are run, where the inverse and forward models are updated in real-time. The models will be implemented as artificial neural networks, and will be randomly initialised. As from our previous experience with similar architectures (Schmerling et al., 2015), this would initially generate a random behaviour, along which sensory and motor information are collected to refine the models. Over time, the movements will be more precise and thus better capable of reaching specific goals.

We are currently implementing and testing parts of the architecture. In particular, we have been carrying out experiments on predictive models, although on a different robotic platform (humanoid robot Softbank Nao). We implemented a forward model for generating raw images, given specific motor commands, using deep convolutional network. The following figure shows an illustration of the neural network that has been implemented and tested on the Nao robot. Inputs to the network are motor commands sent to the left arm of the robot. Output of the network is the image obtained from the camera of the robot (the illustration shows a simulated picture of the arm of the Nao).

This study is reported in the following publication and acknowledges being supported by the ROMI project:

We are currently carrying out training tests with data collected with the LettuceThink platform on a similar neural network.

As to the current state of the architecture, as mentioned before, goals are represented as images of the plant. This raises an issue about how to measure dissimilarity between predicted and observed images. Images are, in fact, high-dimensional sensory information and defining a proper measurement is challenging. For the moment, we opted for using an unsupervised learning mechanism to generate low-dimensional representations of these images. Such low-dimensional vectors can be easily compared, for instance using simple Euclidean distance between them. Therefore, we implemented a deep convolutional autoencoder (illustrated in the next figure).

![Convolutional Autoencoder](image)

A convolutional autoencoder is a deep neural network that is trained to reproduce the same image passed as input. It is typically used for compressing high-dimensional data, such as a raw image, into a low dimensional code, and then for uncompressing such a code into something that matches as much as possible the original image.

We recently tested similar networks in the context of learning features about robot ego-noise (auditory noise of the robot motors). Results are reported in the following publication (which also acknowledges ROMI in funding the study):

Pico, A., Schillaci, G. and Hafner, V.V. (2018), Predictive Models for Robot Ego-Noise Learning and Imitation, Proceedings of the 8th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob)

We are currently carrying out training tests with data recorded with the LettuceThink platform. The following images show a result of a compression/uncompression process.
In the top row, a set of images obtained from the end-effector camera is shown. Each image is passed as input to the autoencoder, and it is then compressed and re-generated by the network. The bottom row of shows the reconstructed images.

The motivation behind using this network is to produce a representation of goals that is easier to use in the curiosity-based exploration behaviour. The following image illustrates that high dimensional goals (images) can be represented, using autoencoders, as points in a lower dimensional space (for the sake of simplicity, a 3D space is shown).

As mentioned before, this simplifies also the calculation of the dissimilarity between predicted and observed images, as it ends up in just a calculation of the Euclidean distance between two points in such a low-dimensional space.

Another challenge is how to determine goals. We are currently carrying out experiments on unsupervised learning of goals. This additional learning process consists of a preliminary collection of a dataset of images through a random exploration behaviour of the end-effector camera. These images are compressed into low-dimensional codes (in the order of 16 dimensions) and then used for training a Self-Organising Map (SOM). A SOM is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretised representation of the input space of the training samples. It is therefore a method to do further dimensionality reduction.

Using SOMs allows to determine, in an unsupervised fashion, a fixed number of goals. The following figure shows an illustration of a self-organising map, where goals (cluster centroids) are positioned on top of the original dataset. Here, each red dot represents an encoded image, i.e. the compressed low-dimensional code of each image of the original dataset.

---

This mechanism has been implemented using the MiniSOM Python library\(^5\), which we adopted in previous work (Kajic et al., 2014).

The architecture is still not fully functional, as experiments and tests are currently being carried out. Nonetheless, all the main building blocks have been identified. Further work is still needed to refine the adaptive behaviour and to integrate it with the 3D reconstruction process developed by the other ROMI partners. Moreover, as mentioned before, we are investigating different goal representations that better fit with the tasks of 3D plant reconstruction.

### 2.3.2 Learning of terrain features using artificial whiskers

A second experiment we want to carry out involves the usage of artificial whiskers for detecting features of the terrain. Artificial whiskers can represent an interesting sensing modality in the context of micro-farming robots, as they allow to gently touch plants without damaging them, as well touching soil and terrain, without modifying too much the shape of their surface.

We equipped the end-effector of the x-carve with the artificial whiskers described in section 2.1.3, as shown in the next figure.

We are currently recording data while the end-effector is moving back and forth on top of a rectangular pot. We are testing that with two types of terrain: soil and stones. A preliminary experiment will consist of training a recurrent neural network (RNN), such as a Long Short-Term Memory (LSTM) RNN, with the collected data and of using the trained network for classification purposes.

### 2.4 Evaluation criteria

The evaluation criteria for the performance of the adaptive learning system is ultimately depending on the improvement of the 3D reconstruction of plants, compared to the original approach of using pre-defined

\(^5\) [https://github.com/JustGlowing/minisom](https://github.com/JustGlowing/minisom)
trajectories around the plant to reconstruct. This process is however planned to be terminated at the end of the project. Therefore, for the moment, we plan to analyse the performances of each of the trained models described in section 2.3.1 and 2.3.2. In particular, the trend of the prediction errors of the forward models, as well as the precision of the inverse model in generating commands to reach specific goals, will be taken as main measurement of the performance of the system.

3. Aerial robot

As to the current state, there has been not much development on the adaptive learning system for the aerial drone. In fact, IAAC is currently developing a cable-bot drone as a promising alternative to flying drones in the context of the ROMI project. We plan to set up this robot platform in Berlin, as soon as details are available. Conceptually, the architecture described in section 2.3.1 is applicable to any robotic platform that is equipped with any sensory and motor modality.

The plan is to have camera sensors in the aerial drone, similar to those used in the LettuceThink, as well as 3D movement capabilities. Therefore, the experimental set-up will not be much different to the one described in section 2, beside being wider in size. We aim thus at developing curiosity-based exploration behaviours also for the aerial drone. In October 2018, during a project meeting in Paris, we identified a possible scenario with ROMI partners, where a cable-bot is exploring a field through this intelligent behaviour and signalling spots to reach to a mobile LettuceThink platform for weeding purposes.

3.1 Evaluation criteria

Evaluation criteria will be better defined as soon as the aerial drone is finalised and functional, as well as the components of the scenario are integrated. As for the moment, the same evaluation criteria defined for the single components of the adaptive learning system architecture of the LettuceThink can be applied here.

References


