Applied Reinforcement Learning, Sommersemester 2016
Ben Pfirrmann, Erik Wannerberg, Zhiwei Han

Poppy the Watchdog

- Applied Reinforcement Learning SS 2016: Intermediate Report (Stage 2)

Purpose

The main purpose of this report is to briefly revisit the project idea and describe the progress in each area respectively. We will highlight the changes with respect to the initial proposal, our first-hand experience working with the poppy robot and the VREP simulator, and how difficulties have been handled alongside with implementation.

Recap: Project Idea

The main idea of this project is to emulate the behavior of a guard that will point its weapons only towards hostile intruders, and welcome allies with a friendly greeting. This will be done using a Poppy robot (https://www.poppy-project.org/). The implementation is based on reinforcement learning techniques, using computer vision methods for object detection.

Task Overview and Progress

Computer Vision Task

Initial Idea

In order to simplify the target recognition process, our initial setup was to use a coloured card as the target, displayed on a uniform differently coloured background, and research what algorithms could be used to track this target.

Progress and Challenges

Among the methods we found, we chose and developed a solution that is based on an image segmentation algorithm to detect coins (http://blog.christianperone.com/2014/06/simple-and-effective-coin-segmentation-using-python-and-opencv/). For testing purposes we took reference images with Poppy's camera and evaluated the results. So far, both the algorithm and as the image acquisition itself work using a separate laptop.

However, we currently experience difficulties using the shipped ODROID board. One major challenge is a spuriously appearing green-coloured segment along the image border, which may result from a malfunctioning driver. We try to address this issue by manually starting the installed UVC video driver and to run the detection afterwards. This approach was not yet successful, and we may need to adapt our general implementation to adjust to this. For example, we could try to use a regular computer instead of the ODROID board for control. Another issue is a seemingly random distribution of detected locations without focusing on the main coloured object, which may be due to bad lighting conditions or noise (see Fig. 1).

Aside from real camera images, we have also created a pseudo-Computer Vision framework in python for simulating poppy in the V-REP Simulator (http://www.coppeliarobotics.com). Here, we place an object (e.g. a cube) in front of poppy and compute the angle between the object's position and Poppy's head-orientation. Then we emulate a camera based on an ideal camera

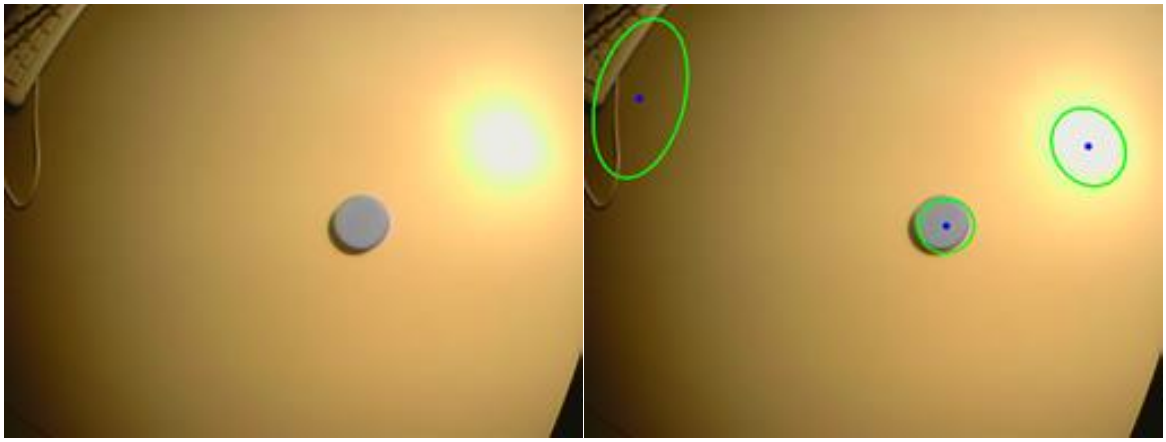

Fig. 1: Example images for object recognition. **Left:** Original image. **Right:** Image with objects recognized, marked with green ellipses with blue dots in the center. Note that not only the intended blue circular cork is recognized by the algorithm, but also a shadow in the upper-left corner and a reflection to the upper-right.

model and projecting the angle on a plane perpendicular to the head orientation. This we can discretize, and obtain states similar to that in the real scenario.

Reinforcement Learning task

Initial idea

In order to transform the task into a Reinforcement Learning problem, we needed to set it in a Markov Decision Process (MDP) with states, actions and rewards. For the states, the basic problem would use the discretized position of the target in the camera image. For the actions, we would use a discrete movement of the head positioning, such as a left/right turn of a fixed angle or a fixed amount of time, in two dimensions. The number of discretized states and actions was to be decided based on required accuracy and computational resources available.

In order to speed up training, we were to use the V-REP robot simulator, which already has models for Poppy, to train on simulated data before using an actual robot.

To find an optimal policy and value functions, we would try different learning algorithms, and test their performance against each other.

Progress and Challenges

So far, as the state space is very generic, it has not required any changes. We have experimented with fewer discretized positions, such as through ignoring horizontal position, in order to obtain a simplest possible model for a first implementation. For future needs we may then choose to increase the complexity again.

With V-REP, we can now simulate a poppy model, where we have already embedded our pseudo-CV-algorithm. A simple template with relevant initialization procedure has been created that allows for quick start-up. For testing the actual motor commands, we have also implemented a pre-programmed state-action policy.

For the learning algorithms themselves, we have progressively advanced through more and more application-focused algorithms. For a start, we created a learning algorithm using only our

model, on a purely mathematical deterministic simulation of the setup. In this simplified model, we assume that each action ('left', 'right', 'none') leads to the next (left, right or same respectively) state, in the discretized horizontal positions. In this approach we use a modified version of TD-Learning and policy iteration, and successfully learn a feasible policy.

Furthermore, we have created a modular code framework in Python to allow programming of learning algorithms that are oblivious of the actual environment that is interacted with. This means that we can easily implement the same algorithm on V-REP, the poppy robot or a deterministic mathematical function, and develop different algorithms side-by-side. In this framework, we now have a working SARSA(0) and a SARSA(λ) implementation, however using a different reward scheme. We will continue to develop and modify this in the coming weeks.

Practical Implementation

Initial Idea:
Since we need to implement the algorithms on an actual robot and make sure that it works correctly, we needed to research the interfaces with Poppy, and to test how the steering of the motors works in reality. After implementation, we would also need to evaluate how the performance compares with that in the simulator. We also allocated some time for fine-tuning of parameters and debugging.

Progress and Challenges
We have not yet been able to test learning algorithms on the real Poppy as we still focus on the simulator. However, we have got some first-hand experience with the hardware setup itself.

For taking images we first assembled camera and head, taking part in building the robot, this in cooperation with another group and university employees.

In this part, one of the major challenges is the shipped ODROID-XU4 board (http://www.hardkernel.com/main/main.php). In order to use this, we were required to install additional software and for example to assemble the memory card with the shipped operating system (Linux).

The hardware limitations also further complicate the development process. Aside from the aforementioned image-acquisition problems, the limited number of USB ports is also of significance. As we would ideally connect a keyboard, a mouse, a Wi-Fi connector, the USB camera and the control-port for the motors simultaneously, having only two ports is simply not enough. However, with USB hub this should be possible.

Additionally, we are sometimes unable to gain remote access to the poppy browser interface over the network (via http://poppy.local) in order to for example access to the jupyter notebook where our algorithms would run.

Organisation

Initial Idea:
We assigned specific tasks to each group member (see Appendix) and set deadlines.

Progress/Challenges:

Overall, we stick with the predefined tasks. However, in some cases the workload was redistributed and based on our findings some tasks were expanded and others reduced, but the main steps remain unchanged. The appendix not only provides a list of steps and tasks but also small comments on particular changes. Moreover, we continually assign low-level tasks that are required depending on our new findings.

A major learning is that we need to adapt our tasks to the construction progress of poppy. One example is the procedure of image acquisition. As the motors were not available in the beginning, we could not take reference images using predefined movements. Instead we needed to take images manually using our own computer. Similar examples have already been described in the sections above.

Although the initially proposed Gantt-Chart has so far been of little use, with most of its information already being covered by the detailed list of tasks, we deem it still useful for presentations. It is likely to be included in the final presentation.

Appendix: Task Distribution

Computer Vision Task

Step / Subtasks	Who	Deadline	Comment
COLLECTION OF REFERENCE IMAGES			
▪ Write a python script that makes poppy perform predefined rotational movements taking images of the environment	Z	19/05	No movements possible, images manually taken with notebook and camera
▪ Take reference images using the aforementioned algorithm	Z/B	19/05	
ALGORITHM RESEARCH			
▪ Search for color-segmentation solutions and the python APIs (naive-channel difference, k-means-segmentation etc.)	B	19/05	Completed
▪ Search for a center detection algorithm	E	19/05	No research required, implementation-inherent
TESTING AND IMPLEMENTAION			
▪ Find possibilities to implement these algorithms in python using APIs and libraries or write the code directly.	E/B	26/05	Done
▪ Test the algorithms offline using reference images.	Z/E	26/05	Done
▪ Test the algorithms online with poppy and verify the accuracy of the result (i.e. are poppy images correctly mapped to states)	E/B	26/05	Green-border difficulty, possible driver issue

Reinforcement Learning Task

Step / Subtasks	Who	Deadline	Comment
STATE AND ACTION SPACE FORMULATION			
▪ Create artificial states that will be used as input data to the simulator (the artificial states can be generated based on the images taken with poppy and based on geometrical reasoning)	B/E	02/06	Camera-Emulator in VREP (Pseudo-CV)
LEARNING ALGORITHM AND IMPLEMENTATION			
▪ Create a python default template (i.e. import relevant modules, define objects and respective default values for actions, states, rewards, value function) to allow for simple coding.	Z	16/06	Done
▪ Create a function prototype for a function that returns an action according to a policy	E	16/06	Included in the template above
SIMULATION (VREP)			
▪ Try several learning algorithms with the predefined setup and make them run properly on VREP using the artificial states as an input sequence.	E/Z/B	16/06	In Progress
▪ Compare the different learning algorithms based on stability, robustness and speed in the simulator	E/Z/B	16/06	tbd
REWARD DESIGN			
▪ If necessary do an additional tuning of the rewards or the discount factor using the previously created funcions.	E	16/06	Will be part of algorithm testing

Practical Implementation

Step / Subtasks	Who	Deadline	Comment
MECHANICS AND CONTROL			
▪ Implement algorithm prototypes (from the Reinforcement learning task) on poppy and identify, if the performance/behaviour largely deviates form the VREP simulation	E/Z/B	30/06	tdb
TRANSFER OF ALGORITHMS ON POPPY			
▪ Identify possible mechanical issues and create fixes (e.g. lock the poppy torso in place)	B	30/06	tbd
PERFORMANCE EVALUATION			
▪ Evaluate the performance and possibly improve the algorithm using simulation and real poppy robot.	E/Z/B	14/07	tbd