Poppy the Watchdog

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

### About this report

The main purpose of this report is to briefly revisit the project idea and to subsequently describe the respective progress. The latter encompasses the major changes with respect to our initial proposal, our first-hand experience working with poppy and VREP and how we handled difficulties alongside with the implementation.

### Recap: Project Idea

The main idea of this project is to consider a guard that will point its weapons only towards hostile intruders, but welcome allies with a friendly greeting. The functionality shall be emulated using a Poppy robot (<https://www.poppy-project.org/>). The implementation is based on reinforcement learning techniques as well as 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 uses a coloured card as the target, displayed on a uniform coloured background.

Progress and Challenges

For this simple location task, we researched different methods, and developed a solution that is based on coin-segmentation. For testing purposes we took reference images with Poppy’s camera and evaluated the results. Both, the algorithm as well as the image acquisition itself, work offline. However, we currently experience difficulties using the shipped ODROID board. One major challenge is a green-coloured segment along the image border. Another one is a seemingly random distribution of detected locations without focusing on the main coloured object. Both 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. However, this approach was not yet successful. Due to these difficulties we may need to adapt our general implementation. For example we could try to use a regular computer instead of the ODROID board for control. In worst case, we may also need to only focus on position detection only, because the colour extraction has not yet been implemented.

Besides real camera images, we also created a pseudo-CV framework in python for the VREP 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 model and projecting the angle on a plane perpendicular to the head orientation. The resulting projection gives a relative object position and can be interpreted as state-data for poppy.

##### Reinforcement Learning task

Initial idea

In order to transform the task into a Reinforcement Learning problem, we need to set it in a Markov Decision Process (MDP) with states, actions and rewards. For the states, the basic problem will use the discretized position of the target in the camera image. For the actions, we will 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 discretised states and actions will have to be decided based on required accuracy and computational resources available.

In order to speed up training, we use the v-rep robot simulator (<http://www.coppeliarobotics.com/>), that already has models for Poppy, to train on simulated data, before using an actual robot.

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

Progress and Challenges

The overall state space does not appear to require any changes as it is very generic. The complexity, however will be further adapted based on our future results. Possible solutions may lead to less discretized positions or to a horizontal position only.

We are already able to run VREP with a poppy model. A simple template with relevant initialization procedure has been created that allows for quick start-up. Here, we already embedded our pseudo-CV-algorithm. Moreover we implemented an application that emulates the learning algorithm (i.e. a tracking algorithm based on the aforementioned pseudo-CV-Algorithm). This allowed for testing the actual motor commands.

So far, we were not able to implement a complete learning algorithm on poppy or VREP, as the other tasks were time-consuming. The algorithms we developed, still need improvement and debugging. However, we already created a python-template consisting of class-prototypes that allows for simple adaption to different learning strategies on both the simulated Poppy and the real one. Besides that we work on different learning algorithms individually.

Moreover, we created one converging algorithm leading to a feasible policy that does a pure mathematical simulation of the setup. However, it is based on a model were we just assume which action (‘left’, ‘right’, ‘none’) leads to which discretized horizontal position. In this approach we use a modified version of TD-Learning and policy iteration.

###### Practical Implementation

Initial Idea:

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

Progress and Challenges

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

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

One of the major challenges is the shipped ODROID board. Here, we were required to install additional software and to assemble e.g. the memory card with the shipped operating system (Linux). Moreover, the hardware limitations further complicate the development-process. Besides the aforementioned image-acquisition problems, the limited number of USB ports is of great significance. Ideally, we could simultaneously connect to a keyboard, a mouse, a Wi-Fi module, the USB camera and the control-port for the motors. However, without a USB hub this is currently not possible and we can only connect to two devices at the same time. In addition to that, we are sometimes unable to remotely access the http://poppy.local-page that gives access to the jupyter notebook.

###### Organisation

Initial Idea:

Originally we assigned specific tasks to each group member (see. Appendix) and we 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. It would be cumbersome to describe all subtle changes in detail, as the main steps remain unchanged. Therefore, 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.

The initially proposed Gantt-Chart was only of little use as it did not add any value. Most of its information was already covered by the detailed list of tasks. However, it would have been useful for weekly presentations and 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 |