Poppy the Watchdog

- Applied Reinforcement Learning SS 2016: Project Proposal

Introduction and Overview

Detection and tracking are important problems in autonomous systems. Some of the problems in the area can be formulated as control problems. However, using machine learning methods, one can achieve more complex or dynamic behaviour. For example, one might consider a guard, that will point its weapons only towards hostile intruders, but welcome allies with a friendly greeting.

In this project, we will try to make a Poppy robot (https://www.poppy-project.org/) emulate such a guard. Using reinforcement learning techniques such as Q-learning, we will try to find policies for the robot to control its motors such that it can track and follow a target detected by its cameras. We will use image processing to simplify the input treatment into a set of states, and use the V-rep simulator to train on a virtual robot before implementing it in reality. We will then proceed with additional scenarios to extend the robot's behaviour into for variety of related tasks.

Goals

Primary goal:

- finding a policy that allows the Poppy robot to locate a target by only moving its head, to fix the object in the center of its vision.

Secondary goals:

- sending out different signals depending on the type of cards
- tracking a moving target
- pointing at a target with an arm

Task description and breakdown

Computer Vision Task

In order to simplify the target recognition process, our initial setup will use a coloured card as the target, displayed on a white or colourless background. For this simple location task, we will research different methods, for example edge detection, or more simply, taking the the position of the card as the average of the coloured pixels in the image. See Fig. 1 for reference.
To try this out in reality, we will take reference images using Poppy's cameras and evaluate how well the different approaches perform for locating the target, both offline and online, using real camera data from Poppy.

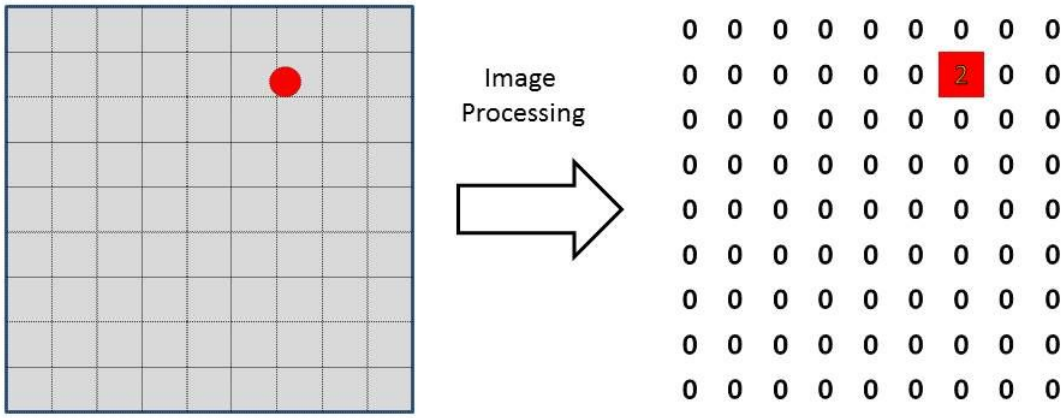

Fig. 1: Example of detection process. For the camera picture on the left-hand side, we make a count of how coloured each pixel is. On the right-hand side, we see the resulting matrix, in this case how much “red” there is in each pixel. We then use these counts to calculate the position of our target, for example the average position weighted by “redness”.

Reinforcement Learning task

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. For the rewards, we will set a zero-reward for all the states except for the one corresponding to the target in the middle (positive or negative reward depending on if we are solving a minimisation or maximisation problem). We will then try out different discount factors allow for promoting the central states.

In order to speed up training, we will 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. The interface to this simulator is similar to that of Poppy, thus we will develop most of the learning framework to fit both interfaces.

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

Practical Implementation

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.

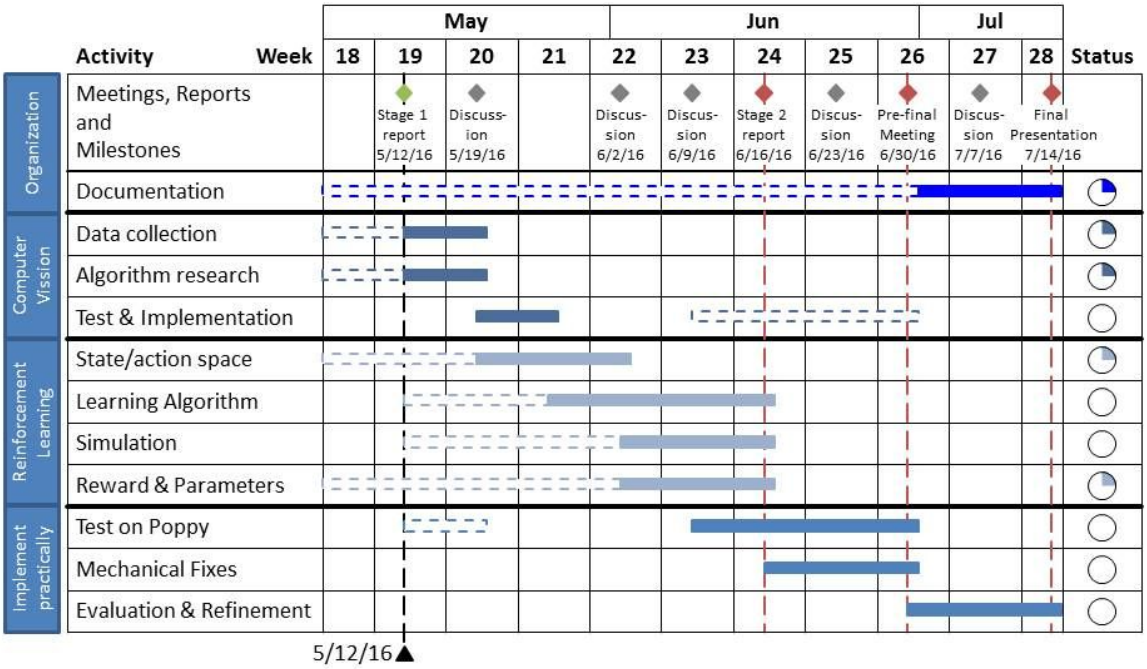

Fig. 2: Gantt-chart, continually updated during the project to track the progress of different tasks. Documentation will be done throughout, with a focused period towards the end of the project.

For an overview of the schedule, we have created a Gantt-chart to track our progress, which can be seen in Fig. 2. A more detailed with task distribution over the team can be seen in the Appendix.

Appendix: Task Distribution

Computer Vision Task

Step / Subtasks	Who	Deadline
COLLECTION OF REFERENCE IMAGES		
▪ Write a python script that makes poppy perform predefined rotational movements taking images of the environment	Z	19/05
▪ Take reference images using the aforementioned algorithm	Z/B	19/05
ALGORITHM RESEARCH		
▪ Search for color-segmentation solutions and the python APIs (naiv-channel difference, k-means-segmentation etc.)	B	19/05
▪ Search for a center detection algorithm	E	19/05
TESTING AND IMPLEMENTAION		
▪ Find possibilities to implement these algorithms in python using APIs and libraries or write the code directly.	E/B	26/05
▪ Test the algorithms offline using reference images.	Z/E	26/05
▪ 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

Reinforcement Learning Task

Step / Subtasks	Who	Deadline
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
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
▪ Create a function prototype for a function that returns an action according to a policy	E	16/06
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
▪ Compare the different learning algorithms based on stability, robustness and speed in the simulator	E/Z/B	16/06
REWARD DESIGN		
▪ If necessary do an additional tuning of the rewards or the discount factor using the previously created funcions.	E	16/06

Practical Implementation

Step / Subtasks	Who	Deadline
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
TRANSFER OF ALGORITHMS ON POPPY		
▪ Identify possible mechanical issues and create fixes (e.g. lock the poppy torso in place)	B	30/06
PERFORMANCE EVALUATION		
▪ Evaluate the performance and possibly improve the algorithm using simulation and real poppy robot.	E/Z/B	14/07