Project Plan
Pushing a Box in a Straight Line

Jose Luis
Claus Meschede
Jan Rudolph

June 15, 2016

1 Motivation & Goals

In our project, we want to make the e-Puck robot learn to push a box along a straight, predefined line. In order to do that, the robot needs to learn about the dynamic behavior of the system and the effects of its movements.

The task can be divided into two separate problems. First of all, the robot has to find the box and move into a position where it can start pushing. Once it is in touch with the box, it can start pushing against it and eventually learn how to react to keep it on the line. Because of the limitations of the e-Puck platform and the project time frame, we want to restrict ourselves to the second problem. This is achieved by using a simplified problem setup with the robot and the box always in contact or at least close to each other. The movement will be executed in discrete steps, with location and orientation changes of the robot small enough to never loose contact with the box. The starting point of the robot will always be on the line, and with the box right in front of it.

2 Markov Decision Process

To derive a learning strategy, we first have to define a Markov Decision Process that is simple enough to make the task feasible, but sufficiently complex to solve the given problem.

2.1 Action Space

As desired outcome of this project, we want the robot to learn in what way the box is reacting on its movements and how it can use this dynamic behavior to keep the box on the line. Therefore, by definition, the robot should always choose a moving direction towards the box and never away from it to solve this problem. Furthermore, it has three different choices to do that. It can try to move in a direction aiming towards the center of

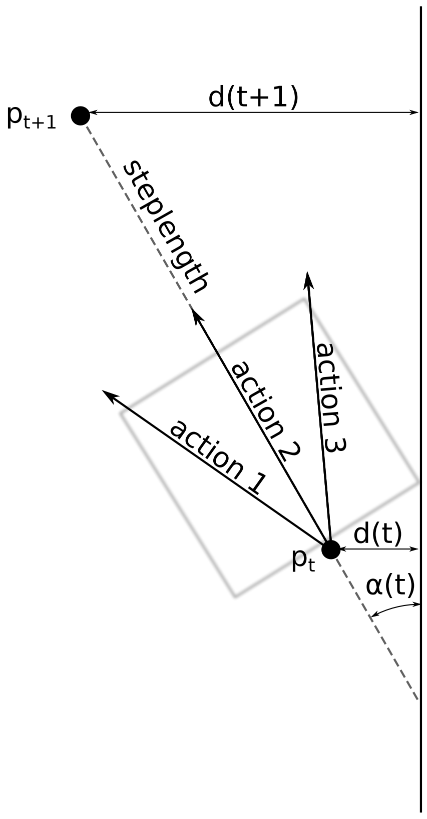

Figure 1: Action Space

the box or it can try aiming left or right from the center of the box. This means that the robot needs to know where the center of the box is and we have to compute some kind of belief state for the box center using the proximity sensors or the on-board camera. For the simulator, we can kickstart our development by cheating, e.g. extracting the absolute position and orientation of the box.

Making these considerations, we can define the action space of the MDP. In the simplest case, the robot has only three different actions to choose from. Go straight, go left or go right. This can be extended by introducing some increments of going left or right (e.g. hard left/right and left/right) or even using a continuous action space. In this project, we will start by using the simplest model first and extend it if needed.

If we look at the movements the robot will perform in each round, the robot will adjust its orientation relative to the box (step 1) according to the decision made by the RL algorithm, followed by a straight movement of predefined length s (step 2). The length can be adjusted, so that the box never looses contact with the box. Afterwards, we will compute the new box position and repeat the two steps again.

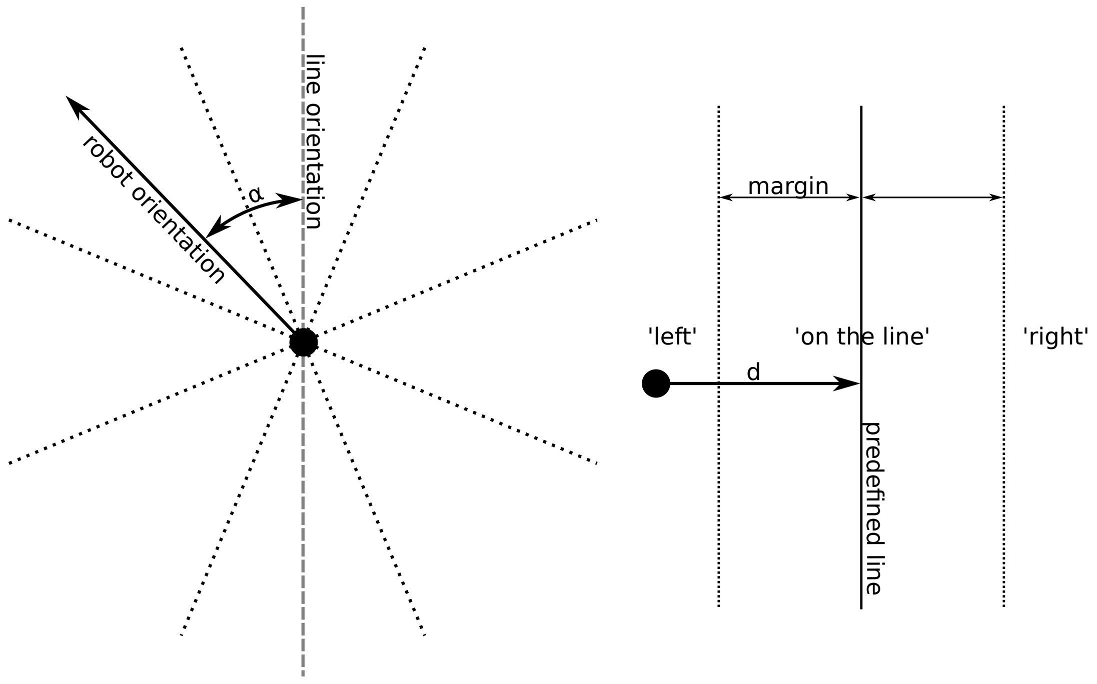

Figure 2: Discrete State Space

2.2 State Space

To keep the box on a line, the robot somehow needs to know where this line is. Making informed decisions requires a state space that contains the minimum amount of information needed. Our problem has at least two different situations where the robot should learn to behave differently. On the line itself it should try to keep the box on the line and besides it should try to push the box back on the line. Additionally the robot needs to know if its actions will keep it on the line or if its actions will lead back to the line eventually. We also have to make a distinction between the right and the left side of the line, as the optimal behaviour will be different for each of those situations.

Therefore, we propose to keep track of the robot movements computing a distance d indicating the robot-line distance and an orientation α that gives the angle between the line robot-box and the predefined pushing-line. After step 1 of each round, the angular orientation of the robot with regard to the line will be updated. After step 2, we will update the line distance. To keep the MDP state space small, we then discretize these values. The continuous distance could be divided into 3 states indicating 'left', 'right' or 'on the line' and the orientation in n discrete orientations, e.g. 8 orientations would give a state space size of 24 states.

2.3 Reward Function

Designing the reward function can then be achieved by considering the desired outcome again. Pushing the box perfectly on the line means d = 'on the line' and α = 'line-orientation' in terms of our discrete state space. Therefore, we can simply define a function that only gives a reward in this particular state. If it turns out, that this is not working, because the reward state is not encountered often enough, we can add a smaller, additional reward for minimizing the distance d.

3 Subgoals & Timeline

For the complete project, a total time of 9 weeks is available, starting after the deadline for the Stage 1 report. We defined a number of smaller, reachable sub-goals, and allocate 2 weeks of the total time as safety margin. We aim to complete goals 1 and 2 in time for the Stage 2 report due after 5 weeks, having then already solved the problem inside the simulator. The sub-goals are:

1. Manual Control of the robot in the simulator, with observation of the state space
   (∼2 weeks in which we lay the foundation for all further work, and which has to be completed before we can start with the next sub-goal.)
   a) Implementation of the discrete movements (turning, going straight) and orientation/distance tracking
   b) Implementation of the box center tracking using the proximity sensors, camera or the stereo microphone. At first, to get the simulation running, we can cheat by using the known box position in our simulation.
2. Learning inside the Simulator
   (∼1 week in which we ignore all problems of the real, imperfect world. We can work on this in parallel with sub-goal 3.)
   a) Implementation of the RL Framework and algorithm
   b) Test, adapt, tweak parameters, repeat
3. Observation of the state space with manual control on the real robot
   (∼2 weeks)
   a) Trial and implementation of orientation and distance tracking on the real robot
   b) Trial and implementation of the box center tracking in the real world
4. Learning in the Real World
   (∼1 week)
   a) Press start and hope for the best
5. Documentation and preparation of the presentation
   (∼1 week)