POLITECNICO DI TORINO
Master's Degree in Mechatronic Engineering

Title: Visual based local motion planner with Deep Reinforcement Learning
Candidate: Mauro Martini [s260771]
Supervisor: Prof. Marcello Chiaberge

1 Introduction

1.1 Objectives of the thesis

This thesis aims to develop an autonomous navigation system for indoor scenarios with a Deep Reinforcement Learning (DRL) approach. In particular, the work focuses on the realization of a local motion planner to control a mobile robot using depth images of the scene.
The idea of the work is born at the PIC4SeR (PoliTo Interdepartmental Centre for Service Robotics), as part of a broader project focused on service robotics. The thesis embraces the vision of the centre, which is to develop high-tech solutions for application fields such as precision agriculture or smart cities. A versatile autonomous navigation system represents a competitive advantage for a wide range of tasks in all these contexts.

1.2 Project framework

Autonomous navigation is a challenging task in the research area of robotics, which has been tackled with numerous contributions and different approaches. Among them, learning methods have been investigated in recent years due to the successful spreading of Deep Learning (DL). In a Reinforcement Learning (RL) framework an agent learns by experience, i.e. through the interaction with the environment where it is placed, avoiding the need of a huge dataset for the training process.
Deep Deterministic Policy Gradient (DDPG) is the specific Deep Reinforcement Learning (DRL) algorithm applied to train an agent in a simulated environment using ROS (Robot Operating System). A Convolutional Neural Network (CNN) is used to directly select suitable actions for the robot, expressed in terms of linear and angular velocity (CNN output). Input information is composed of robot's orientation and distance from the goal, in addition to images captured by a depth camera. Depth images are single channel grey-scale images which provide distance information. A great focus is also devoted to minimize the computational cost of the model, looking forward to a future hardware implementation. From a research point of view, it is interesting to evaluate the performance of the navigation system when using depth images, compared to a different solution based on LiDAR sensor. On the one hand, a camera offers a rich depth information. On the other hand a simple 2D LiDAR is able to cover a wider field of view. The two implementations are trained and tested in virtual environments with obstacles and the results are compared and discussed.

2 Implementation in virtual environment

TurtleBot3 is the standard robotic platform used for the work, which allows to carry out the whole development in simulation thanks to virtual models. The *waffle* results to be a suitable turtlebot model since it is already provided with an Intel RealSense R200 depth camera. Gazebo is used for modeling virtual indoor scenes where the simulated interaction between the robot and the environment takes place. TurtleBot3 provides standard ROS packages to control the robot actuators at high-level, directly acting on the velocity. Simulations are structured in episodes in which the robot has to reach a specific target point. Each episode is composed of a maximum of 500 temporal steps and it terminates with three possible outcomes: *Goal*, *Collision* or *Timeout*. The system working principle can be summarized as follow (see also Figure 1). At a generic time instant, an artificial neural network selects an action a_t

for the robot, i.e. a velocity command, according to the state s_t and it is executed in the virtual world. The environment samples a score r_{t+1}, then it processes the sensor data received and it sends back a new state s_{t+1}.

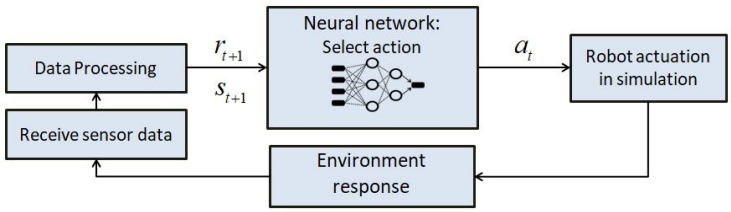


Figure 1: Schematic representation of the working principle of the system.

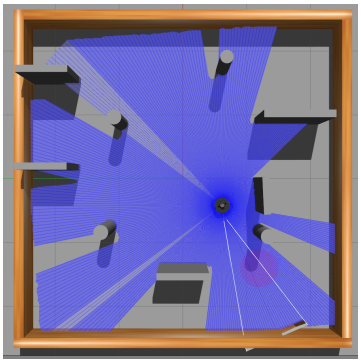
For the visual based navigation, the neural network receives as **input** the following data:

- **Goal information**: the pose of the robot is known thanks to odometry sensor data. They are processed to compute the actual distance from the goal and the robot orientation with respect to the goal (heading angle).
- **Depth image**: the RealSense camera of the waffle turtlebot captures raw depth images with a resolution of 240x320, which is reduced to 60x80 due to computational constraints. For each pixel a maximum distance value of $5m$ is accepted. Then, all the distance values are normalized with respect to this cutoff value ($5m$) to facilitate the learning process of the neural network.

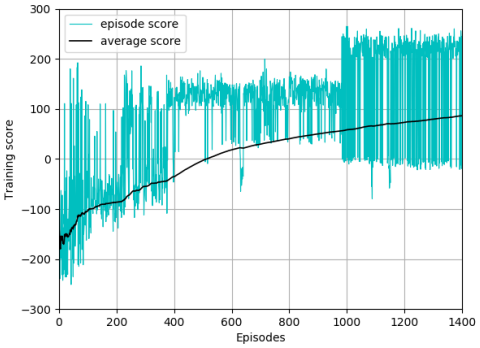
The **output** is a velocity command composed of a linear velocity in the range $[0, 0.2]m/s$ and an angular velocity in the interval $[-1, 1]rad/s$. A LiDAR based navigation has been also developed to carry out a comparison between the two perception systems. LiDAR measurements are simply filtered from 359 to 36 all-around points, selecting the minimum distance values for each angular range of 10°.

3 Training

The training process is certainly the core of the work. Reinforcement Learning enables multiple advantages to tackle the navigation task. In fact, the agent is trained without the need of a dataset and results to be a versatile motion planner for mapless navigation. The Deep Deterministic Policy Gradient is the actor-critic algorithm chosen to train the agent. This particular architecture exploits two different neural networks. The **actor** network aims to approximate a deterministic policy to select actions, and it is the one controlling the robot. The **critic** network is responsible for evaluating the actions chosen by the actor, by approximating the action-value function $Q(s, a)$.



(a) *Final configuration of the training stage with obstacles in Gazebo simulated environment.*



(b) *Score function of the learning process. Around episode* 400 *convergence is reached. Around* 1000 *obstacles are increased and goal reward is set to* +200.

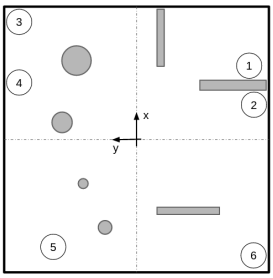
Figure 2: Training: virtual environment and learning curve obtained with the final reward function.

The design of the actor Convolutional Neural Network is a central challenge of the thesis. The final model presents a stack of convolutional layers to extract features from depth images. Features are concatenated with goal data and forwarded to fully-connected layers. Two distinct output neurons, having a *sigmoid* and a *tanh* activation function, respectively map the linear and angular velocity command. For the implementation with LiDAR, the neural network is composed of some fully-connected layers. A key element of the training process with reinforcement learning is the reward. It is a numerical score assigned at each time step to the agent to evaluate its behaviour. The reward function assign $+100$ when the goal is reached (after episode 1000 it is increased to $+200$ to compensate the higher amount of obstacles in the stage), and -10 in case of collision with an obstacle. The reward is needed to compute the target values to train the critic. ADAM optimizer makes use of the critic gradients to update the weights of the networks. Figure 2 shows the final training stage in Gazebo and the learning curve obtained with the mentioned reward function.

4 Test and results

	Goal pose	Result	Total time	Total path length	Final goal distance
Goal 1	$[1.2, -1.8]m$	Goal	$11.378s$	$2.076m$	/
Goal 2	$[0.2, -2.0]m$	Goal	$11.194s$	$1.947m$	/
Goal 3	$[2.0, 2.0]m$	Goal	$17.041s$	$2.761m$	/
Goal 4	$[0.8, 0.2]m$	Collision	$8.181s$	$1.389m$	$0.862m$
Goal 5	$[-1.9, 1.2]m$	Goal	$13.743s$	$2.226m$	/
Goal 6	$[-2.0, -2.0]m$	Goal	$22.948s$	$2.801m$	/

(a) *Test results of the camera-driven robot. The final outcome for each testing episode is the principal metrics for obstacle avoidance. Total time and path length give a indication of the optimality of the navigation.*



(b) *Schematic representation of the 6 goals given in the test stage.*

Figure 3: Testing stage: target locations and obstacles scheme on the right, results on the left.

The agent is tested in a virtual environment with obstacles of different shapes (see scheme in Figure 3b). The ability of the robot to navigate efficiently in an unknown scenario is evaluated with some basic metrics: outcome of the testing episode, total time in seconds, total path length in meters. The test is performed by giving 6 target points to be reached in different areas of the world. Each of them offers a peculiar challenge for the obstacle avoidance task. The robot is always spawned at the centre of the scene. If it collides with an obstacle the test is considered failed. Results of a testing session of the visual based navigation system are shown in Figure 3a. The same test is carried out with the LiDAR based implementation. Both the agents have shown the ability to reach target points in the presence of static obstacles. The peculiar advantages and limitations of the local motion planner when based on LiDAR points and depth images emerge in the tests. In both cases, obstacle avoidance can be improved working on several aspects. The following considerations can be done for a qualitative comparison:

- The camera-driven agent collides when trying to reach Goal 4. This is probably related to the fact that the lateral columns enter in the field of view of the camera too late to let it counteract. The LiDAR based agent collides with the narrow columns in the path towards Goal 5. Narrow objects are a clear issue for LiDAR points.
- A comparable optimality of navigation is related to a similar training process, although the visual based agent shows a smoother behaviour and reduced times to reach the goal.
- The visual based navigation can be improved enriching the input data of the neural network with successive depth images of the scene, as well as looking for a more effective reward function.
- An hybrid solution camera-LiDAR may be a promising configuration.

Despite the difficulty of a task such as autonomous navigation with a single depth camera, the project can be considered a precious source of experience and it paves the way for further developments. A natural next step would be the implementation on the physical robot platform.