Reinforcement learning and robot navigation

Charles Dufour

Supervisors: Prof. F. Eisenbrand, Jonas Racine

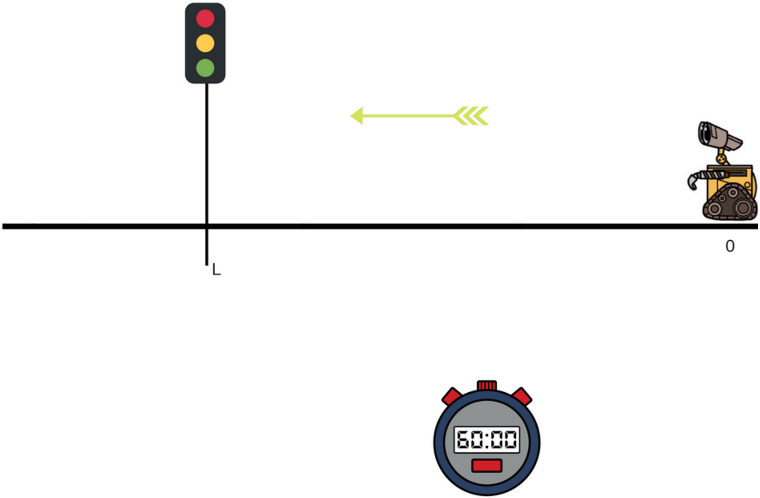
June 20, 2018

Introduction

The problem

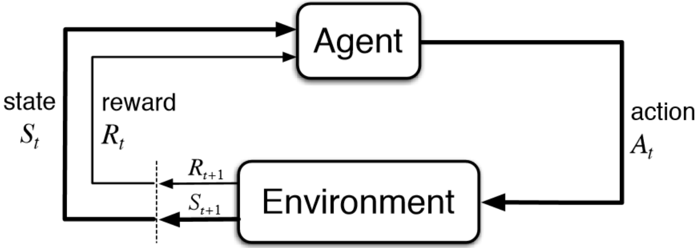
- Framework: a Raspberry Pi 3 robot which can follow lines
- The task: the robot should adapt its speed with respect to traffic lights
- How: using Reinforcement Learning (RL)

The task



Presentation

1. Reminders
2. Modeling
3. Simulation
4. Results



The agent's job is to find a behavior that maximizes the long-run sum of values of the rewards.

Markov Decision Process

Definition: (MDP)

- A set of states $\mathcal{S} = \{s_0, s_1, s_2, \ldots, s_n\}$
- A set of actions $\mathcal{A} = \{a_1, a_2, a_3, \ldots, a_k\}$
- A transition function $T(a, s, s') = \mathbb{P}[s' \mid a, s]$
- A reward function $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
- A discount factor $0 \leq \gamma < 1$

Q-learning

Approximation $Q(s, a)$ (measure "how good is the action")
Action chosen: $\underset{a}{\operatorname{argmax}}\ Q(s, a)$

Modeling

States

States encode the following information:

- distance to the traffic light
- speed
- color of the traffic light
- time spent in current color
- previous action

Modeling

Actions

- slow down
- keep the same speed
- speed up

Possible speeds: $0, 20, 30, \ldots, 70$

Reward function: $\mathcal{R}(s, a)$

- positive reward if robot respects the traffic light
- negative reward for every step "far away" from the traffic light

Positive reward when the task is done and negative reward for every step along the way.

- how to compute the distance from the robot to the traffic light?
- how "fast" is speed $20, 30, \ldots$?
- how to simulate the traffic light?
- what are the exact parameters for the reward function?

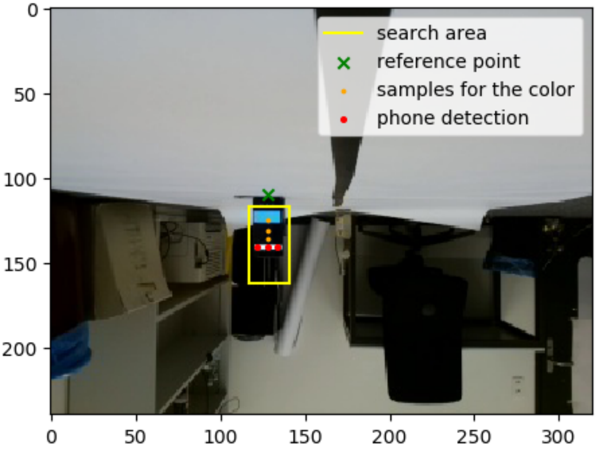


Figure: View from the robot camera

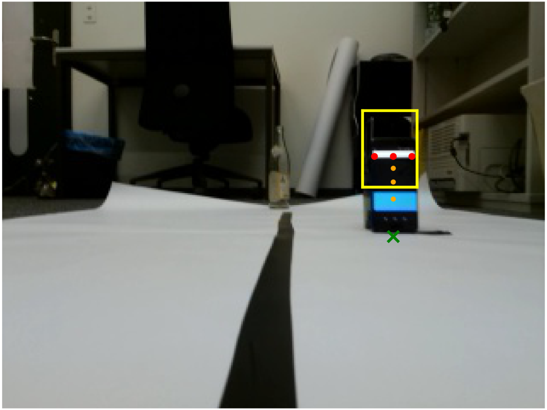
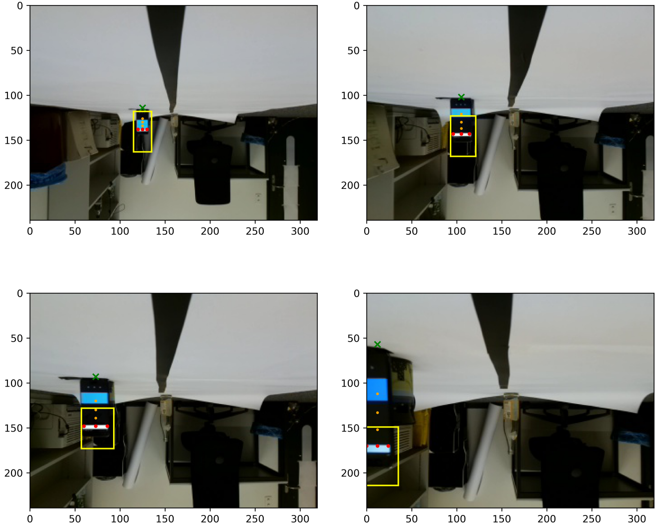


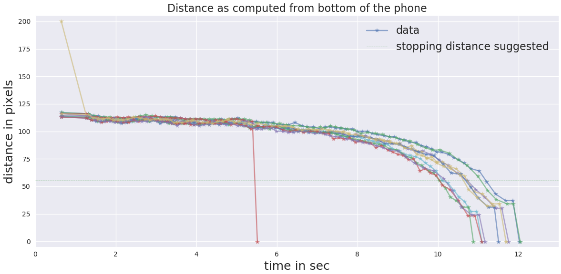
Figure: View from the robot camera

Simulation

measuring distance from the robot to the traffic light



How fast is fast? Graphs for speed 30 and 70



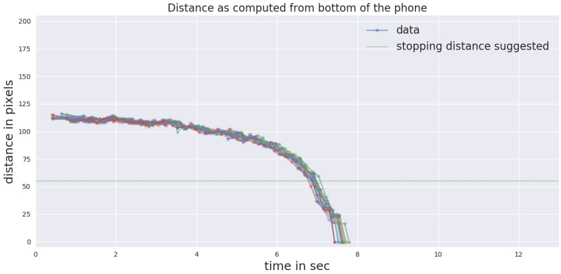


Table: Example of raw data obtained

position	outcomes		
80	75	74	73
81			
82	76	77	

Table: Example of raw data obtained

position	outcomes		
80	75	74	73
81			
82	76	77	

Mean of data from the surrounding points: 75

Table: Example of raw data obtained

position	outcomes		
80	75	74	73
81			
82	76	77	

Mean of data from the surrounding points: 75

Table: After imputation

position	outcomes		
80	75	74	73
81	75	74	76
82	76	77	

Table: Example of data

position	outcomes				
80	75	74	74	73	75

Table: Example of data

position	outcomes				
80	75	74	74	73	75

Table: Probabilities of transitions obtained from the data

from	to	probabilities
80	75	$0.4 = 2/5$
	74	$0.4 = 2/5$
	73	$0.2 = 1/5$

In the simulation, the time is defined in terms of number of interactions.

In the simulation, the time is defined in terms of number of interactions.

We simulated the traffic light in the following way:

- We measured the number of pictures processed per second by the robot (~ 8 im/s)

In the simulation, the time is defined in terms of number of interactions.

We simulated the traffic light in the following way:

- We measured the number of pictures processed per second by the robot (~ 8 im/s)
- Learning for a specific traffic light

In the simulation, the time is defined in terms of number of interactions.

We simulated the traffic light in the following way:

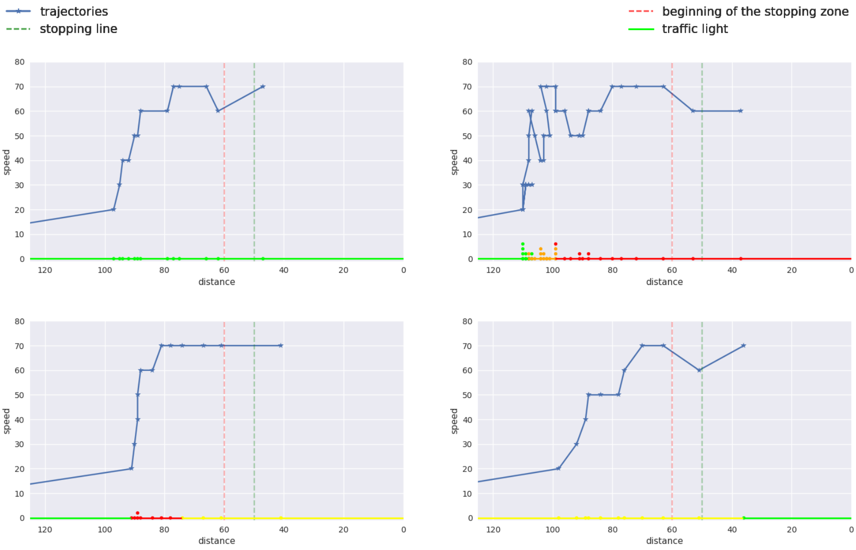
- We measured the number of pictures processed per second by the robot ($\sim$ 8 im/s)
- Learning for a specific traffic light
- Traffic light simulated to behave the same as in real life in terms of *time*

Issues with previous naive thinking

- The distance measurements are approximative, hence we cannot have a fixed distance to stop, so we use a *stopping zone*.
- Reward functions are not as simple as they seem. We had to test a lot of them, small changes have influence ...

Simulation

Reward function, naive approach 1 million episodes



Solutions

- add great negative or positive reward for respecting or not red light ($\sim \pm 300$)

Solutions

- add great negative or positive reward for respecting or not red light ($\sim \pm 300$)
- try to bait the robot to learn what we want

Solutions

- add great negative or positive reward for respecting or not red light ($\sim \pm 300$)
- try to bait the robot to learn what we want
- and try a lot of times different reward functions (maybe 20) with a lot of iterations ($\sim$ 10 millions, about half a day)

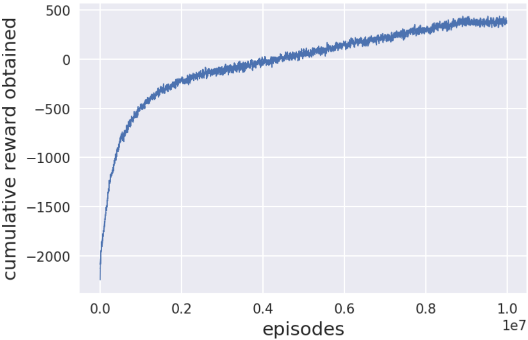
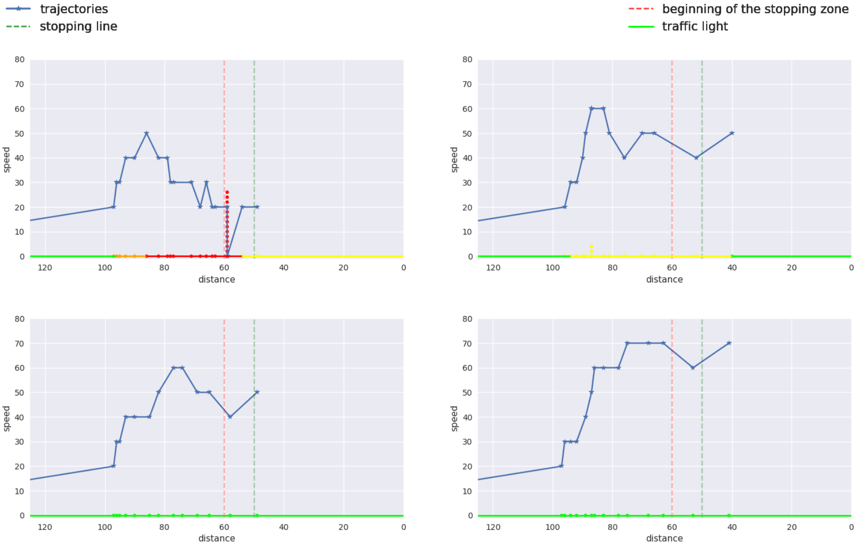


Figure: Q-learning curve (10 millions iterations)

Examples of trajectories obtained



What could be done next?

- we tried to learn with 50 millions iterations ($\geq$ 60 hours), but the results were not impressively better than with 10 millions iterations
- do grid search in order to find a better reward function
- generalize the learning to any traffic light
- Kalman filters to improve measurements
- ...

Thank you for your attention
Demo time !