UNIVERSIDADE FEDERAL DE MINAS GERAIS

COMPUTER SCIENCE DEPARTMENT
POC I

FINAL REPORT

Command Correction for Teleoperation using Reinforcment Learning

Student:
Felipe Cadar Chamone <cadar@dcc.ufmg.br>

Advisor:
Erickson Rangel do Nascimento <erickson@dcc.ufmg.br>

August 2019

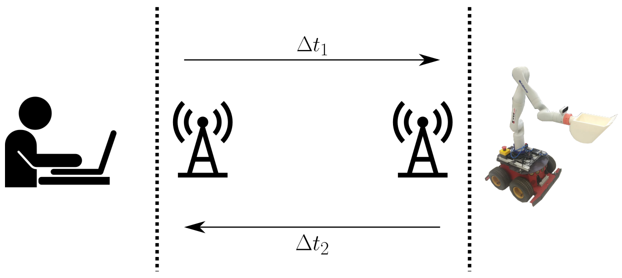

Figure 1: Round trip time delay example in a master-slave teleoperation. Δt_1 and Δt_2 represent the time delay in the communication channel.

1 Introduction

In 2001 *Fong and Thorpe* [1] defined Teleoperation as the action "to operate a vehicle or a system over a distance". It has the potential to improve a large variety of tasks. An operator can control a bulldozer without actually being in the mine and a pilot can fly a drone to explore hostile areas without risking their life, both receiving live information about the environment their devices are inserted on. Nevertheless, with great power comes great responsibility. When teleoperating, for example, a heavy machine, the operator has to know the state of the remote environment as if he were there, and the commands it sends to the machine must be executed instantly, that's when our greatest challenge emerges, the time delay in communication.

Generally, the teleoperation consists of a master and slave systems. The master can send commands through the communication channel to be executed by the slave, and the slave sends feedback to the master act on. When the communication channel is not perfect, every message takes a time t to be delivered. So, every time, the master is responding to feedback that is late by t_2, and his response will be delivered delayed by $t_1 + t_2$, as shown in Figure 1.

Usually, a human operator can perceive and adapt to a constant delay in communication, either by decreasing the velocity of the movements or executing tiny actions and waiting for the outcome. These strategies also decrease productivity and the feeling of immersion. To deal with these problems, we propose to include an intelligent agent in the remote device to adjust the commands of the operator given a delay, the state when the command was executed and the present state of the remote environment.

2 Related Work

Time-delayed teleoperation has been a challenge for many years and several approaches were developed to tackle this problem. There are mainly two strategies, predict the observation or predict the command. In the first, the remote device acquires an observation y in time t, let's call it $y(t)$, and sends it to the operator. When the operator receives it, the observation is delayed by τ, $y(t - \tau)$. Now, the teleoperation framework predicts the observation $\hat{y}(t + \tau)$, so the operator reacts to

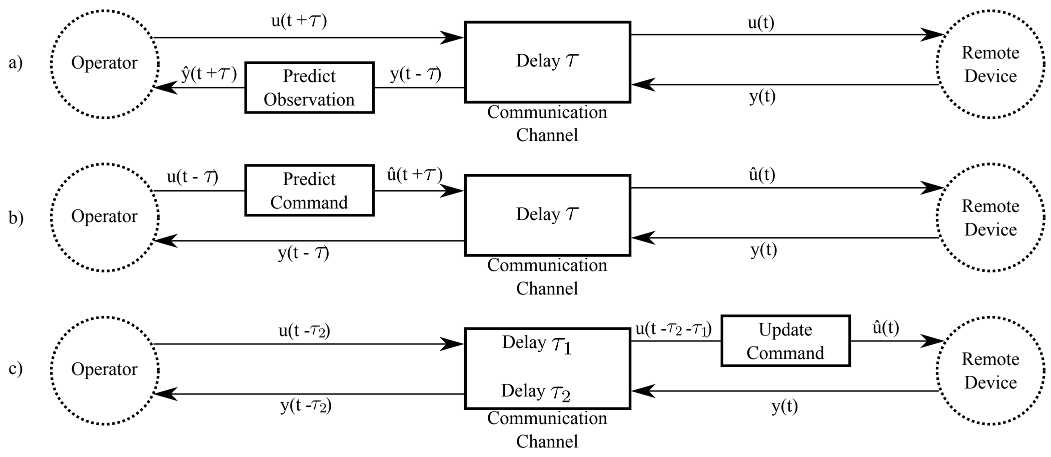

Figure 2: a)

something that is going to happen when his command $u(t + \tau)$ reaches the remote device after τ, $u(t)$. This process is shown in figure 2 a). In the '90s *Bejczy et al.* [2]. used this strategy to teleoperate a robotic arm using visual feedback. They predict the position the arm would be after the delay and displayed it on top of the delayed image. One of the problems with this strategy is that the delay has to be known. When using the internet as the communication channel, the routing may change the delay constantly. More recent works still use similar methods to deal with the delay. *Alonzo et al.* [3] used sensors to virtualize the entire environment in a photorealistic way to display it to the operator.

Another strategy is to predict the command. Again, the remote device acquires an observation y in time t, $y(t)$, and sends it to the operator. When the operator receives it, the observation is delayed by τ, $y(t-\tau)$. The operator sends a command u reacting to $y(t - \tau)$, $u(t - \tau)$. Now the framework predicts what would be the command after the delay, generating $\hat{u}(t + \tau)$. After the delay, the command $\hat{u}(t)$ arrives at the remote device. This scenario is shown in figure 2 b). In 2010, *Smith and Jensfelt* [4] used this approach to predict the movement of a robotic arm. In this strategy, the delay also needs to be known previously, which makes it difficult to be applied using the internet as the communication channel. Other works, like *Jayaprashanth et. al* [5], also utilized this strategy combined with control tactics to improve the efficiency of the teleoperation.

We propose to introduce a third approach, placing an agent in the remote machine to update the delayed command. This would remove the need to previously know the delay time and would still apply the right command to the current observation, as shown in Figure 1 c).It is common to use machine learning in controlling the device when we want it to be autonomous at some level. However, we don't want it to be autonomous, we want to help the human operator to achieve his goals in the teleoperation with lower-level commands, and is difficult to do so because there is not much available data for training. That is why we choose to use reinforcement learning. If we could simulate a teleoperation task, the agent would learn to adjust commands thro each iteration, hopefully, getting better and better.

3 Methodology

3.1 Simulation

The goal of our methodology is to train a reinforcement learning agent to adjust the delayed commands sent by the operator. We first have to be able to simulate our teleoperation in a way that we can control the communication delay. Inspired by the experiments described by *Smith and Jensfel* [4], we will simulate the task of drawing a line inside a track. Using this task we can calculate the rewards based on how well the agent adjusts the input command to avoid the sides of the track and increases the productivity of the job.

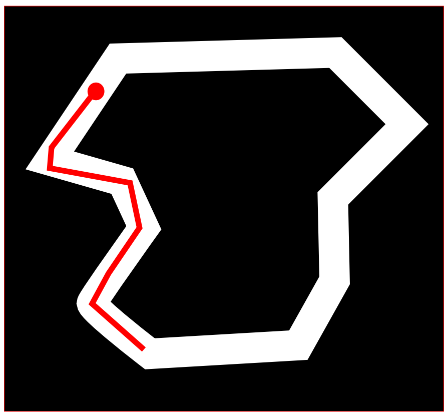

Figure 3: Task of drawing a line inside a track in the simulator.

The simulator consists of two modules, the environment, and communication. The first one is responsible for actually simulating the task. Given a command and a state, it calculates the next state and reward. The communication module controls the delay between the agent and the environment.

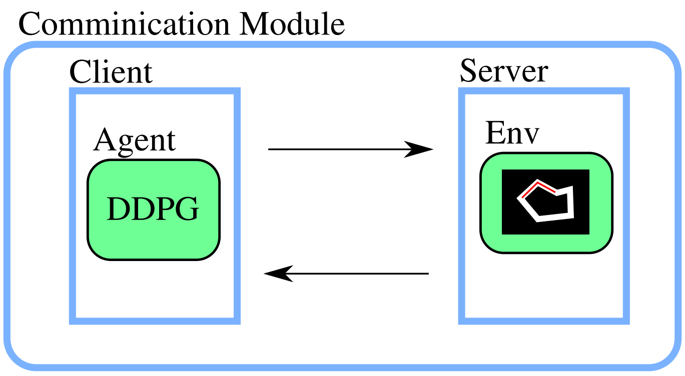

Figure 4: Simulator modules.

The first module was implemented using the OpenAI's environment conventions, which means that it is compatible with most reinforcement learning libraries.

The second module consists of a server and a client. The server constantly runs the environment and streams information about the state of the agent. The client captures these information packages and responds with actions to be executed by the agent in the server. Both sides have control over the delay of the packages they sent.

Since the modules are independent, we can use the environment as a standard simulator and use any other environment that follows the same conventions in the simulator server.

4 Results

To validate the simulator, we train a reinforcement learning agent using Deep Deterministic Policy Gradient (DDPG) [6] to complete the track without bumping to the edges. Note that the trained agent does not represent the goal of this work, but we can analyze its behavior to validate the simulator.

We trained the agent without delay and then evaluate the mean reward with delays of 100, 200 and 300 milliseconds. With this experiment we expected to see that the mean reward decreases as the time delay increases. This behavior can be verified at the graph in figure 5.

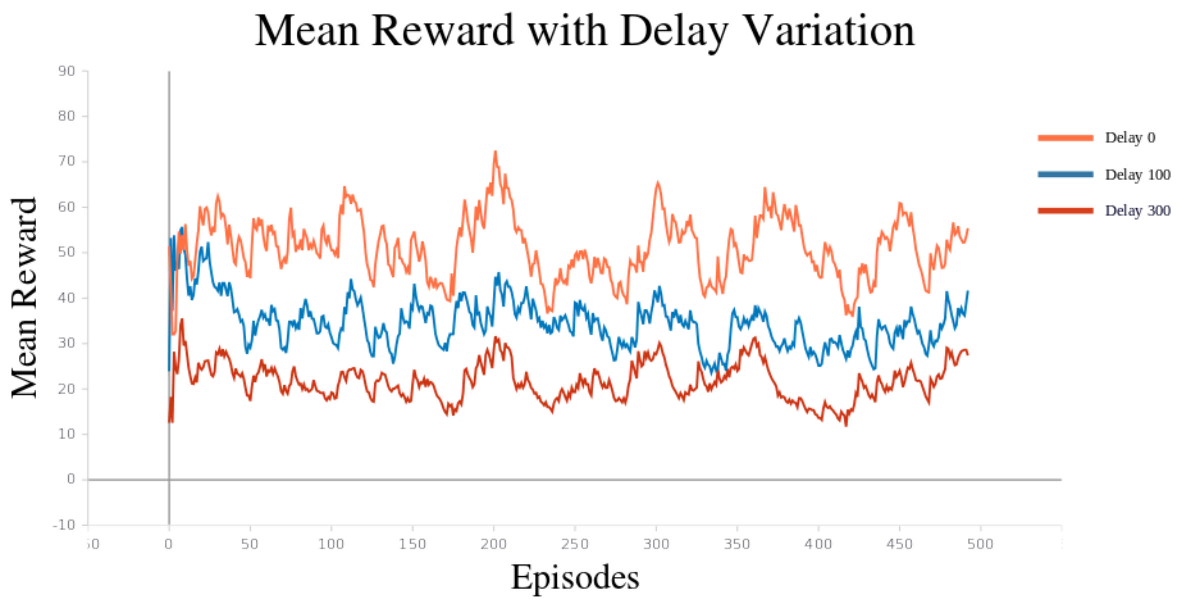

Figure 5: Mean Reward per episode with the communication delay variation.

Another evaluation point is the consistency of the set delay. The figure 6 shows the distribution of the measured delay given a 100ms delay request. We can see that the majority of the packages arrived with the correct time delay.

5 Conclusion

In this work, we successfully implemented a reinforcement learning environment compatible with the OpenAI conventions and a communication protocol to control

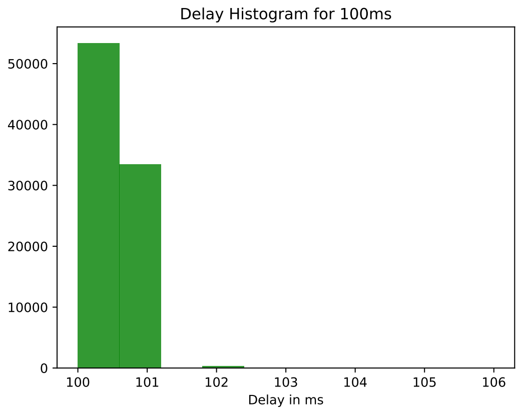

Figure 6: Distribution of measured delay for 100ms request.

time delay that simulates a teleoperation task. This implementation is crucial to our next goal, train an agent to adjust the delayed command sent by the operator and apply it in real-life scenarios.

6 POC II

Our next step in this work is to use the developed simulator to train the main agent, that will learn to adjust the commands sent by the operator.

References

[1] Terrence Fong and Charles Thorpe. *Autonomous Robots*, 11(1):9–18, 2001.

[2] A.K. Bejczy, W.S. Kim, and S.C. Venema. The phantom robot: predictive displays for teleoperation with time delay. In *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press.

[3] Alonzo Kelly, Nicholas Chan, Herman Herman, Daniel Huber, Robert Meyers, Pete Rander, Randy Warner, Jason Ziglar, and Erin Capstick. Real-time photorealistic virtualized reality interface for remote mobile robot control. *The International Journal of Robotics Research*, 30(3):384–404, October 2010.

[4] Christian Smith and Patric Jensfelt. A predictor for operator input for time-delayed teleoperation. *Mechatronics*, 20(7):778–786, October 2010.

[5] Jayaprashanth Jayachandran, Jason Gu, and Ya Jun Pan. Teleoperation of a mobile robot using predictive control approach. In *2006 Canadian Conference on Electrical and Computer Engineering*. IEEE, May 2006.

[6] Timothy Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 09 2015.