Sim-to-Real Adaptation for Mobile Robots Based on LiDAR Odometry

Anonymous Author(s)
Affiliation
Address
e-mail

Abstract: In this paper, we propose a method for the simulation-to-reality problem in the domain of wheeled mobile robotics. While most applications rely on a low-level velocity controller, both in simulation and on the real robot, it may be desirable to directly control other action spaces, e.g., raw control signals. Especially, this problem is of relevance for harsh outdoor environments and rough terrains. To tackle this, we propose a two-step domain adaptation technique that includes: 1) mapping of source action space to the velocity space to apply the corresponding real-world velocity in simulation, thus adapting the dynamics of the robot, and 2) state correction that compensates imperfections of the simulation controllers and thus helps to adapt sequence of simulated observations. We provide quantitative results that show relevance of our approach, showing the benefits of combining these two adaptations.

Keywords: mobile robotics, sim2real, domain adaptation

1 Introduction

The simulation-to-reality (sim-to-real, sim2real) gap are all the imperfections of the simulators that result in different outcomes when compared with the real world. This includes both the gap of the simulated physics and the visual gap of the rendered scenes. The sim2real gap is one of the main bottlenecks in the area that requires calibration or training a model in simulation before any real robot deployment, which is commonly reinforcement learning (RL). Thus, solving this problem is crucial for real-world applications of trained models. In this work, we focus only in the physical part of the sim2real problem, considering the visual part of sim2real gap as separate area.

In past decades, a variety of methods were proposed in order to tackle the sim2real gap. Many efforts were applied in the development of the new high-fidelity simulators which is an important part of the gap reduction process, but still it is almost impossible to fully simulate the real world due to its complexity and stochasticity. Thus, various domain randomization and domain adaptation methods were proposed to enhance available simulators. Those methods were applied for sim2real transfer of RL policies for different types of the robots such as manipulators [1], quadrupedal robots [2], bipedal robots [3], etc.

However, there is a lack of modern research dedicated to the sim2real problem in the wheeled mobile robotics domain. One of the reasons is that for common tasks with uniform surface, especially in the indoor environments, standard controllers and navigation algorithms [4] are enough to achieve good real-world performance without specific adaptation. But for the more harsh environments, it becomes harder to model such motion with high fidelity [5].

In our work, we propose a view on the sim2real gap problem for the area of the wheeled mobile robotics domain. We quantify experimentally the size of the gap in the proposed perspective, and propose method that allows to reduce this gap based on domain adaptation and supervised learning, as long as a way how one can collect data for this adaptation. The proposed technique can be considered as a step towards the reduction of the gap in existing research, and encourage the community to test and develop novel methods of sim2real transfer in the mobile robotics domain. We also show how mapping different action spaces may be applied which also has valuable practical potential.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.
2 Related work

**General Sim-to-Real Transfer:** As it was mentioned in the introductory part, a variety of methods were proposed. The precise modeling, development of high-fidelity simulators and their calibration, which are known as system identification, play an important part and significantly improve the performance of the agents in the real world [2, 6] but still it is not enough for robust performance in the real world. Usually, good results are achieved in combination with other techniques. The *domain randomization* [1] is a powerful technique that is widely used in this problem. The key idea is to randomize various of the world and robot parameters during the training, assuming that real world is a sample of the proposed distribution, and this can help to make policy robust to the changing conditions in the real world. This approach shows good practical results [1, 6, 7] and can be also enhanced with *curriculum learning* technique which assumes incremental complexity of the environment [7, 8].

Another widely used approach is the *domain adaptation*. Usually, domain adaptation is used to reduce discrepancy between simulation observations and actions and real-world ones, and collecting real-world data is required to train the adaptation model. Several works that propose action [3, 9, 10] and state [11] transformation models applied in simulation were introduced. Approaches that consider on-robot policy adaptation [12, 13] also were introduced. Several methods also considered domain randomization using distributions learned from real-world data [14, 15] and that can be considered as a combination of domain randomization and domain adaptation.

Various additional approaches were also introduced [16, 17, 18], including methods based on meta learning [19, 20], knowledge distillation [7, 21] and other techniques.

**Sim-to-Real Transfer in Mobile Robotics:** Most of the proposed methods were not applied in the wheeled robotics domain. There are works that achieved successful policies deployment to the real world without any adaptation (not including the visual part) [22, 23]. Chaffre et al. [8] showed the efficiency of the curriculum learning with incremental environment complexity for mobile robot navigation task. One of the first works [24] that consider sim2real problem for mobile robots in 3D rough terrain environments, and its proposed technique is based on robot motion disturbance and pose estimation noising which are the specific forms of domain randomization.

Our proposed method is mostly based on works [3, 11]. Still, evaluation of various existing sim2real adaptation methods can be an important topic for the future research.

3 Proposed method

Let \( \hat{\tau} = \{x_0, ..., x_N\} \) be an estimation of the robot’s trajectory, obtained with some state estimation method, where \( x_i = [x, y, \theta]^T \) is the robot’s pose at the time index \( i \). Since in our experiments we consider locally planar ground surfaces, we define robot’s poses in 2D, where \( x, y \) are the position coordinates and \( \theta \) is the orientation. Our approach could also use 3D poses without loss of generality.

The state variable can be extended by adding the corresponding action \( a_i \) and the velocity \( v_i = [v, \omega]^T \) is the result of the action execution and includes linear velocity \( v \) and angular velocity \( \omega \):

\[
\tau = \left\{ [x, v, a]_0^T, ..., [x, v, a]_N^T \right\}.
\]

Here action \( a_i \) is not obligatory to be a velocity command - it can have any form defined by target RL setting (e.g. in our case it was a pulse-width modulation level value, see next section for details).

Also the frequency of the data sampling may be higher than frequency of the action-taking, it but should be multiple of the last one.

We propose to solve sim2real adaptation problem in two parts: the *velocity adaptation* and *trajectory adaptation*. The velocity adaptation is expressed with the action transformer \( \psi \) - a function approximator that transforms an action \( a \) in the action domain to the velocity \( v \) in the velocity domain, which is then applied in simulation with some standard velocity controller (e.g. differential drive
This action transformer can have a form of recurrent neural network like LSTM [25] or GRU [26]:

\[ v_i = \psi (a_i, h_\psi), \]  

(2)

where \( h_\psi \) is the current hidden state of the network. One can also extend the input of the action transformer with any additional arguments. In order to compensate imperfections of the simulated controllers, the additional state correction based on neural augmented simulation (NAS) [11] is applied. Given current pose \( x_i \), applied velocity \( v_i \), and obtained new pose in simulation \( x^s_{i+1} \), the pose correction is estimated using recurrent function approximator - the pose transformer \( \phi \):

\[ \hat{x}^t_{i+1} = x^s_{i+1} + \phi (x_i, v_i, h_\phi), \]  

(3)

where \( \hat{x}^t_{i+1} \) is the estimation of the corresponding pose in the real world environment. This estimation is manually set as next in simulation: \( x_{i+1} := \hat{x}^t_{i+1} \). As in the previous case, the input of this pose transformer can be extended. The motivation behind such a two-step approach is, on the one hand, keep the dynamics of the robot closer to the real world by applying action transformation, and, on the other hand, keep “static” observations closer to the real-world by applying pose correction.

4 Experiments and results

Experimental setup. Our experiments target the custom-build Akula mobile robot (Figure 1), equipped with Velodyne VLP-16 LiDAR. We used the LiDAR odometry and mapping method (LOAM) [27] to collect trajectories. While LiDAR-based odometry methods provide one of the best results in Kitti benchmark [28], strictly this is not the ground-truth data, but this is one of the most precise approaches among the feasible approaches. The simulation environment is based on Gazebo simulator [29]. The sampling frequency of the LOAM is 10 Hz, and the same value was set to the action-taking frequency in simulation with reference to the simulation time.

In our setup, the source action space has a form of pulse-width modulated signal values from the range \([-255, ..., 255]\) used to control left and right DC motors: \( a = [\text{PWM}_{\text{left}}, \text{PWM}_{\text{right}}]^T \). The motivation behind this is to see how the proposed method captures the properties of the outer environment and also the internal dynamics of the robot.

The sequences of the point clouds were collected while manually running trajectories of different length and shape. Using the LOAM method, the dataset of trajectories was extracted offline from the recorded point clouds.

Training and evaluating adaptation models. As a base model, the 2-layer GRU with output multi-layer perceptron was used both for action and pose transformers. Those models were trained separately on training subset of the trajectories using mean squared error as a loss function.
The performance of the trained models was evaluated in simulation using test ground-truth trajectories. As a quality metric, we used the absolute translation root mean square error between ground truth and executed trajectories of the same length $l$:

$$\text{RMSE}_{\text{translation}}(\tau^{\text{gt}}, \tau^{\text{exec}}) = \sqrt{\frac{1}{l} \sum_{i=1}^{l} \left((x_{i}^{\text{gt}} - x_{i}^{\text{exec}})^{2} + (y_{i}^{\text{gt}} - y_{i}^{\text{exec}})^{2}\right)} \quad (4)$$

and the absolute rotation root mean square error:

$$\text{RMSE}_{\text{rotation}}(\tau^{\text{gt}}, \tau^{\text{exec}}) = \sqrt{\frac{1}{l} \sum_{i=1}^{l} (\theta_{i}^{\text{gt}} - \theta_{i}^{\text{exec}})^{2}} \quad (5)$$

We compare four cases: 1) execution of ground truth velocities (GT actions); 2) execution of predicted velocities (Action transformer); 3) execution of ground truth velocities with pose correction (GT actions and pose transformer); 4) execution of predicted velocities with pose correction (Action and pose transformers). The results are shown in Table 1.

We observed that the execution of ground truth velocities and execution of actions from the action transformer provide similar trajectories and both of them can produce trajectories that are significantly different to the ground truth. This shows the importance of the pose transformer. As expected, the application of the pose transformer significantly reduced the discrepancies. Application of both action and pose transformer showed slightly worse results compared to the sole pose transformer, which is expected since the imperfection of the action transformer is a source of the errors on the pose transformer’s inputs. An example of qualitative comparison is given at Figure 2.

**5 Conclusions**

In our work, we have analysed the sim2real problem in the mobile wheeled robots area. We have defined a sim2real gap form for this case and proposed a solution which shows potential in solving the defined problem. The proposed pipeline serves two purposes: on the one hand it targets to adapt robot’s dynamics to the real world, and on the other hand it allows to compensate imperfection of the action execution in simulated environment, and the importance of the last one has been shown in our experiments. We also have shown how simulation of complex systems, like real DC motors, can be substituted with low-level to high-level action transformer which, as it have been shown in the experiments, can provide positive results. This approach also opens new potential opportunities for policy transfer between different platforms and environments. Finally, we have collected data from a real robotic platform, however, this process is not limited to the proposed approach.

The future work in this direction should include implementation and evaluation of other existing methods, previously applied for different types of the robots, for example, the grounded action transformation methods, and their evaluation by training and executing some real-life policies instead of evaluation on manually defined policies, including experiments in harsh environments. Another direction for future research is the enhancement of the data collection for training sim2real adaptation models. Also, as it was suggested above, one can consider to study the application of the action transformation between different action spaces for policy transfer between environments and robotic platforms.
References


