A Case Study of the Sim-to-Real Gap When Designing PID and MPC Controllers in Simulation

Harry Zhang[1], Stefan Caldararu[1], Thomas Hansen[1], Shouvik Chatterjee[1], Nevindu Batagoda[1], Ishaan Mahajan[1], Sriram Ashokkumar[1], Aaron Young[2], Luning Fang[1], He Shen[3], Xiangru Xu[1], Dan Negrut[1],

[1] Department of Mechanical Engineering,
University of Wisconsin - Madison,
Madison, Wisconsin, United States
Email: {hzhang699, scaldararu, thansen8, schatterjee9, batagoda, imahajan, ashokkumar2, lfang9, xiangru.xu, negrut}@wisc.edu

[2] Department of Mechanical Engineering,
Massachusetts Institute of Technology,
Boston, Massachusetts, United States
Email: aryoung@mit.edu

[3] Department of Mechanical Engineering,
The California State University - Los Angeles,
Los Angeles, California, United States
Email: he.shen@calstate.edu

EXTENDED ABSTRACT

1 Introduction

This extended abstract reports on early results generated with a research testbed whose purpose is the characterization, measurement, and mitigation of the gap between simulation and reality in robots and off-road autonomous vehicles. Sometimes called the sim2real gap [1], it pertains to the difference in behavior of a robot whose autonomy stack is designed in simulation and then deployed in the real world. In most cases, the behavior in simulation is manifestly superior to the one in the real world. There are many causes for this, e.g.: the difference associated with the process of simulating sensors; the idealization of the dynamics model used to represent the motion of the real robot; shortcomings in capturing the world within which the robot operates; the non-deterministic temporal manner in which the underlying Robot Operating System (ROS) infrastructure operates, which leads to race conditions in producing and using information; and the difference in compute power available on an actual robot and when carrying out simulation. Ultimately, the sim2real gap is the observed inability of a control policy designed in simulation to work well with the real robot over a broad spectrum of operation regimes. Closing the sim2real gap is a case by case undertaking. In this abstract, we show a manifestation of this gap as it played out in conjunction with an in-house developed framework called ART/ATK – Autonomy Research Testbed/Autonomy Toolkit [2, 3], which is used to design and test autonomy stacks.

2 Methods

In this experiment, we used a 1/6th scale vehicle; its digital twin was created in Chrono [4, 5], see Fig. 1. The overall goal of our experiment was to compare the performance of vision-based Model Predictive Control (MPC) and PID control on V-1 (actual vehicle) and dtV-1 (digital twin of V-1). In this exercise, we placed cones in a lab and built a ROS2-based autonomy stack that took readings from a camera sensor attached to the vehicle, identified the red cones and green cones, established the next waypoint between the red–green cones (path planning), and produced a control action that moved the autonomous vehicle to the waypoint. The same autonomy stack; i.e., the same software, was run both on the vehicle and in simulation. Moreover, the same embedded system, an NVIDIA Jetson AGX, was used to run the autonomy stack. In one case, the Jetson card was attached to the physical vehicle. In the other case, the Jetson was "attached" to a workstation running Chrono, which simulated the vehicle dynamics, camera sensor, and the rest of the virtual world the digital twin vehicle operated in.

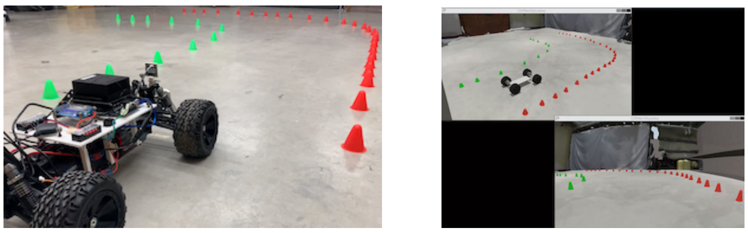
The MPC formulation is based on a finite-horizon optimal control problem [6]. The state vector of the vehicle is $[x, y, \theta, v]^T$, where x and y are the position of the vehicle in global reference frame, θ is the yaw angle, and v is the velocity in longitudinal direction. The control command vector is $[\alpha, \delta]^T$, where α and δ are throttle and steering input, respectively. A bicycle model is used for the vehicle dynamics [6]. The cones are set up, as shown in Figure 1, with green and red color indicating left and right boundaries. The target point at each time step for the MPC controller is produced from a cone-detection algorithm and a planning algorithm based on curve-fitting. At each time step, an optimization problem is solved for the optimal control input using the OSQP package [7].

The PID formulation mainly uses proportional control in both longitudinal (throttle and velocity) motion and lateral (steering) motion. In the autonomy stack, the only difference between MPC & PID is in the controls algorithm; the same perception, path planning, and vehicle dynamics model are implemented for fair comparison.

3 Experiments

Five tests in reality and three in simulation were conducted for each controller using two different reference speed values, 0.5 m/sec and 1.0 m/sec, respectively. An 'L' curve path is set up to test a sharp turning scenario. The run is considered to be a success if the autonomous vehicle is able to traverse the path without colliding with the cones. The reality testing was done in a workspace fitted with a Motion Capture system to record position data and provide vehicle's velocity feedback data. The reality results were then compared to those of the simulation.

As shown in Figure 2, both controllers result in oscillations in trajectories when reference speed is 0.5 m/sec. This oscillation tendency reduces with a larger reference speed, 1 m/sec. However, both MPC and PID scenarios recorded failure tests at the speed of 1m/s, which indicates that in reality higher velocity inputs do not give V-1 enough time to react.



(a) Real vehicle V-1 (b) Digital twin dtV-1.

Figure 1: Setup for studying the sim2real gap, drawing on ART/ATK and Chrono.

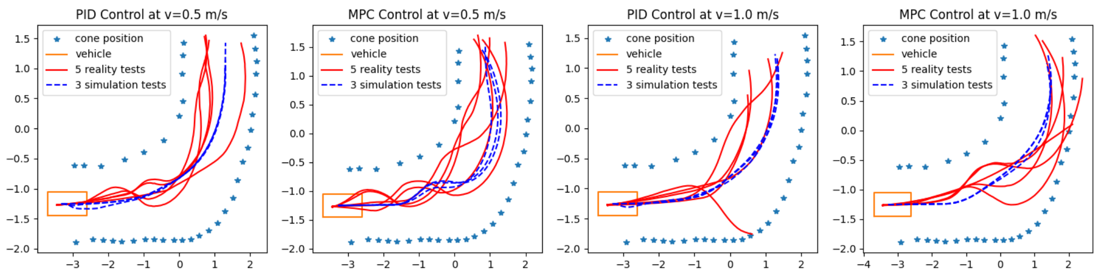


Figure 2: Vehicle trajectories for Simulation and Reality testing. Yellow rectangle shows position of the vehicle at time $T = 0$.

4 Conclusion

In simulation, dtV-1 uses a virtual camera sensor with a higher frame rate that meshes well with the object recognition execution speed. However, on V-1 we posit that the actual camera registration rate is not delivering the expected frequency, which would slow the target point updating frequency. Note that the simulation is essentially always producing the same results, while the V-1 follows different trajectories from run to run. For future work, in addition to investigating and improving the camera-based control scenario, a GPS waypoints-based control policy with Extended Kalman Filter localization scenario will be implemented. This will allow for the validation of the GPS sensor available in Chrono [8], as well as further sim-to-real comparison.

References

[1] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *European Conference on Artificial Life*. Springer, 1995, pp. 704–720.

[2] UW-Madison Simulation Based Engineering Laboratory, "Autonomy Toolkit," http://projects.sbel.org/autonomy-toolkit/, 2022.

[3] A. Elmquist, A. Young, I. Mahajan, K. Fahey, A. Dashora, S. Ashokkumar, S. Caldararu, V. Freire, X. Xu, R. Serban, and D. Negrut, "A software toolkit and hardware platform for investigating and comparing robot autonomy algorithms in simulation and reality," *arXiv preprint arXiv:2206.06537*, 2022.

[4] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *High Performance Computing in Science and Engineering – Lecture Notes in Computer Science*, T. Kozubek, Ed. Springer International Publishing, 2016, pp. 19–49.

[5] Project Chrono Development Team, "PyChrono: A Python wrapper for the Chrono multi-physics library," https://anaconda.org/projectchrono/pychrono, accessed: 2023-01-14.

[6] M. Rokonuzzaman, N. Mohajer, S. Nahavandi, and S. Mohamed, "Review and performance evaluation of path tracking controllers of autonomous vehicles," *IET Intelligent Transport Systems*, vol. 15, no. 5, pp. 646–670, 2021.

[7] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[8] A. Elmquist, R. Serban, and D. Negrut, "A sensor simulation framework for training and testing robots and autonomous vehicles," *Journal of Autonomous Vehicles and Systems*, vol. 1, no. 2, p. 021001, 2021.