μCity: A Miniatured Autonomous Vehicle Testbed

Joshua Tabor, Shenghong Dai, Varun Sreenivasan, Suman Banerjee
University of Wisconsin-Madison

ABSTRACT

This work presents μCity, an environment to design, develop, deploy, and test autonomous vehicles (AVs) within a laboratory setting using miniaturized vehicles, as well as corresponding road and traffic infrastructure. μCity uses a controlled indoor environment in which all artifacts (vehicles, traffic signals, etc.) are centrally coordinated and managed. It provides experimenters with a workflow through which they can attempt new design of individual components of AVs, without requiring these designs to be immediately deployed on public roads, thereby eliminating some safety considerations. We present a preliminary study to demonstrate feasibility of μCity, and identify numerous challenges to make such platforms realistic, scalable, and replicable by many researchers worldwide. We believe μCity can democratize research in AV systems in the future.

Figure 1: An image of a μCity AV

CCS CONCEPTS

• **Computer systems organization** → **Robotic autonomy**; • **Software and its engineering** → *Software system structures.*

ACM Reference Format:
Joshua Tabor, Shenghong Dai, Varun Sreenivasan, Suman Banerjee. 2022. μCity: A Miniatured Autonomous Vehicle Testbed. In *17th ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch'22), October 21, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3556548.3559631

1 INTRODUCTION

The many benefits of autonomous vehicles (AVs) in reducing traffic congestion, accidents and air pollution [3], have spurred research, testing, and development of these systems by many hi-tech and vehicular technology companies (Google/Waymo, Ford, Tesla, Cruis, Uber, Baidu, to name a few). The costs associated with creating, deploying, as well as evaluating the performance and safety properties of full-scale AVs are significant. For example, per Waymo's former CEO, just the cost of sensors and on-board computing platforms required on an AV exceeds $130,000 [5]. When AVs are deployed on public roads, they need to satisfy local and federal regulations, which of course is a necessity, but introduces significant barriers. Inconsistent certification frameworks among administrative domains make AV authorization difficult. Finally, AVs, like any other software or hardware system, are not fail-proof, especially in the developmental stages, and when a system fails, it can cause loss of life, injury, and significant damage to property. In any failure scenarios of these nature in the real world, it can be nearly impossible to put a vehicle in the exact situation to reproduce system failures without taking additional serious risks.

In all systems research, there are alternatives that trade off lower cost with fidelity in experimentation, namely simulation, emulation, controlled experimentation, and real-word field deployments. Research in AV systems adopt some of these opportunities. At one extreme, we have simulation platforms, such as CARLA [2]. It is an example of an open urban driving simulator that has been proposed to support AV development, training and testing. Other examples include Baidu Apollo, Airsim [10] and LGSVL [9]. Of course, while simulators are cheap and easy to instantiate, they provide the least fidelity in representing the real world accurately. At the other extreme, is field deployments on public streets being undertaken by companies, such as Waymo, Tesla, Uber, Baidu, Cruis, and many others.

As an alternative to testing on public streets and to using simple simulations, the University of Michigan has developed a controlled experimentation facility for AV testing in form of a closed test track, called MCity [1]. Among other things, Mcity has realistic traffic lights, roundabouts, straightaways (for freeway simulation), fake store fronts, and a variety of road surfaces all for testing full-sized AVs in a lower-risk approximation of the real world. This facility allows them to study autonomous driving in the real world and provides a closed system for reproducing failures, all without risk to the general public. While this system is certainly an excellent infrastructure for AV testing, its cost limits ease of replication.

μCity— miniaturized controlled experimentation enhanced by simulations: This work, μCity, presents a complementary environment to design, develop, and test AV systems. μCity is a miniaturized experimentation infrastructure in which scaled-down version of vehicles ply in a scaled-down environment, replete with roadways, traffic lights, obstacles, and most such related artifacts that exist in the real world. The cost of creating μCity is a fraction of that of a real-scale environment, and can be a potential pathway that democratizes experimentation with AVs in a semi-realistic setting. Our evolving μCity-setup includes wirelessly connected

MobiArch'22, October 21, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9518-2/22/10…$15.00
https://doi.org/10.1145/3556548.3559631

miniature vehicles (1 foot long) with AV functions, 3d-printed traffic lights, roadways, monitored and centrally managed through a control station, all of which fit into a modest-sized laboratory room. The control station also incorporates safety rules that shutdown vehicles at risk of imminent "accidents." If such accidents do happen, the damage is minimal, and there is no risk to human lives under any circumstances. In this paper, we highlight some of the key developments undertaken so far in creating μCity, and the major challenges ahead. Our goal is to publicly release the entire design of μCity, so that others can replicate this infrastructure in an easy manner, thereby democratizing AV research from the hands of a select few (certain hi-tech companies) to a broader research community to a certain extent. This paper also presents an initial design of μCity, along with an experimental feasibility study of some key aspects. We present a brief comparison of μCity against other alternative approaches in Table 1.

	Simulation	Mcity	Field Test	μCity
Realism	Low	Highest	Highest	High
Repeatability	Highest	Medium	Low	High
Safety	Highest	High	Low	Highest
Flexibility	Highest	Medium	Low	Highest
Usability	Medium	Low	Low	Highest
Cost	Low	Highest	High	Medium

Table 1: Comparison of AV evaluation approaches.

Finally, we observe that μCity is motivated by prior activities, e.g., F1TENTH [7], which creates vehicles that are one-tenth the scale of a Formula 1 racing vehicles, and designed as autonomous radio-controlled (RC) cars equipped with sensors. The focus of F1TENTH is to support races between these scaled-down models and to drive pedagogical activities through such efforts. While F1TENTH meets some of the goals of a miniaturized, yet realistic AV laboratory, there are many other additional challenges that need to be solved, and that is a focus of μCity.

Contributions: The main contributions of this effort are to design μCity, a miniaturized environment for controlled experimentation with AVs. The vehicles and their environments are scaled-down versions of real-life systems. We have undertaken a preliminary feasibility study of μCity, and also highlighted some of the challenges in realizing such a system that would be easy to use, experiment with, and replicate. We expect to open-source μCity for broad public use.

2 μCITY FRAMEWORK

To design the miniaturized laboratory-scale infrastructure, μCity, we needed to implement some specific components. We describe them in turn.

2.1 Control Infrastructure

The proposed laboratory setup provides the physical environment in which the μCity AVs will operate. To accomplish any useful AV exploration, a rich environment is required for the AVs to navigate. This environment should include changing traffic signals, lane and road markings, traffic signs, dynamic obstacles such as "pedestrians,"

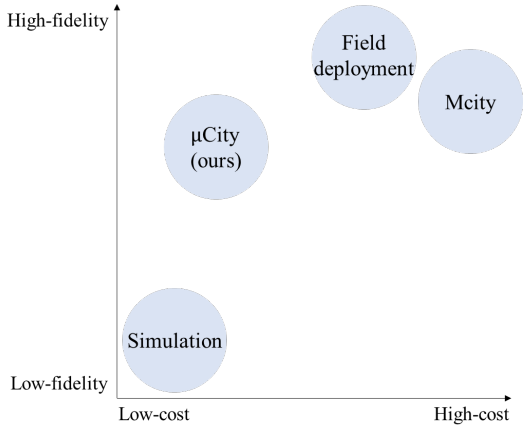

Figure 2: Comparison between multiple AV testing methods.

changing weather conditions, and should be rapidly configurable. The environment should also provide adequate tracking and recording of all AV activities. Our overall control infrastructure needs to manage all these components. We partition the control functions into two parts.

2.1.1 Control for Mobile Components. Each moving object within the laboratory environment should be tracked with a high-fidelity motion capture system, such as the HTC Vive or OptiTrack motion capture system that offers accuracy in the order of at least centimeters. This tracking enables many interesting capabilities within the proposed system. With accurate tracking, AVs can more readily follow precise, predefined trajectories. This enables fully repeatable experiments in which automated (as opposed to autonomous) vehicles run precise trajectories. It allows different versions of the AV to be tested in similar conditions. This level of repeatability is difficult to achieve on public roads or in other situations where precise tracking and control is not available.

The high-fidelity tracking also allows for supervision of all the vehicles operating in the system. If a vehicle is about to collide with another vehicle or obstacle, a safety system monitoring the tracking information could shut the cars down and cancel the experiment, limiting or preventing damage to the hardware. The experiment could also be shut down if certain repeatability requirements fail to be met. For instance if a vehicle is programmed to follow a precise trajectory and that vehicle falls more than 10 cm behind, the experiment can be cancelled and restarted.

This safety system gives users the confidence to run a large number of tests without human intervention, so large-scale tests on AV algorithms and hardware can be more easily performed. The described safety system is also important if the system is to be used for teaching, during which new students are more likely to give unsafe commands to the vehicles.

Finally, there are other mobile components, e.g., dynamic obstacles like bicycles or pedestrians, that are part of the environment. These units can be 3D printed and can undergo controlled motion to emulate different human behaviors that an experimenter may want to evaluate.

Figure 3: A 3D printed stoplight, including LEDs and a wifi connected MCU.

2.1.2 Control for Static Components. μCity incorporates a number of static entities, especially miniature traffic patterns that can be created in a laboratory environment — intersections, small segments of freeway, traffic circles, and parking lots can all be simulated at a small scale. With more area for setup, even small cities can be constructed.

Common traffic control infrastructure such as traffic lights, stop signs, and speed limit signs are created using rapid-prototyping technology like 3D printing. Infrastructure with active components (like traffic lights, street lights, and railroad crossing gates) are powered by a small battery and one of various wifi-connected microcontrollers (MCUs). These MCUs connect the active infrastructure to a central computer over an in-lab WiFi network. This allows the behavior of the active infrastructure to be easily configured and synchronized.

In addition to controlling the infrastructure, it is also essential to create traffic scenarios based on precise maps. These maps can be implemented in simulation and in the lab setting, allowing for both simulated testing and open-road testing in μCity. In the simulation, the 3D models of infrastructure components can be used, making the simulation as close to the real-world as possible. The lab's tracking system can also aid re-creation of simulated traffic scenarios in the real world. Instead of precisely surveying each location for a stoplight manually, a mobile app can be used to display the location of a tracker on the traffic scenario map. By moving the tracker around, the precise location of the infrastructure piece can be rapidly determined and this information can be utilized to appropriately place items, allowing new scenarios to be setup rapidly.

In addition to rapidly re-creating different traffic scenarios, different environmental conditions can also be simulated in the laboratory through creative and interesting methods. For instance, a fog machine can be used to create foggy driving conditions or the laboratory lighting can be changed to simulate different times of day. This allows rapid AV performance testing in different lighting conditions compared to other techniques.

2.2 Orchestration Framework and Experimenter Workflow

All of the active components in μCity are designed to be controlled by a common interface through which all orchestration is done. This orchestration framework runs on a network connected to servers in the laboratory and is the main interface through which experimenters interact with the system. The experimenter specifies every detail about the planned experiment: a map detailing the physical placement of all the objects in the lab, which perception, planning and control software modules they wish each vehicle to use, paths for the vehicles to follow (if applicable), timing for any traffic lights or other active traffic infrastructure, and conditions for the test to be cancelled. Each part of the AV system is designed to communicate with the orchestration framework.

During the experiment, the orchestration software should display the locations of the objects within the system, as well as any debug information related to them such as projected paths for the objects, the states of stop lights, and any other debug information the user chooses to be shown. It also should have an emergency stop button to cancel the test.

2.2.1 Modularity to support easy experimentation. Some key components that make up tasks performed by any AV system are: perception of the environment, planning, and control. Each of these components is interdependent. Parking is a good example of this interdependence. In order to park in a parking spot, the vehicle first needs to be able to recognize spots designated for parking. After recognizing spots to park in, the vehicle then needs to choose a spot and calculate a path to park in that spot. Different paths can be calculated using different cost functions. Perhaps the vehicle plans to take the fastest, safest, or most energy-efficient route to the parking spot. Regardless of the cost function, the car also needs to be able to follow the planned route.

We see such components as discrete modules and the software architecture reflects this. An experimenter in μCity is able to modify each module individually, either in simulation or in the real world. Each module has standardized inputs and outputs. In particular, this allows perception models based on simulated data and real data to be used interchangeably without changes to code in the other modules. From a software perspective, there is no difference to running the modules in simulation or in real life except for where the sensor data comes from and where the controls are directed. This architecture allows the experimenter to only modify and experiment with components of their interest without needing to worry about the rest of the infrastructure.

2.2.2 Generating visual ground truth. Real world efforts have been made to gather video and visual data about the environments around roadways, in cities, on highways, like the German Traffic Sign Recognition Benchmark (more than 50000 images) [11], which can be used to test ML models. However, if experimenters want to use them in μCity, they will not readily work due to differences in the environment between the real world and μCity. While machine learning techniques, such as transfer learning, could have potential use here, we still anticipate collecting and labeling visual data for μCity, on which models can be re-trained.

One of the big advantages of having complete control and knowledge of all artifacts in μCity, is that images can be automatically labeled. The vehicles can be programmed to follow well-defined trajectories, and the cameras mounted on them can gather data. All objects in μCity can be 3D-scanned or are 3D printed, which allows us to generate precise 3D models for them. These models can be used in concert with tracking information to calculate the position of objects of interest within images taken by the vehicle.

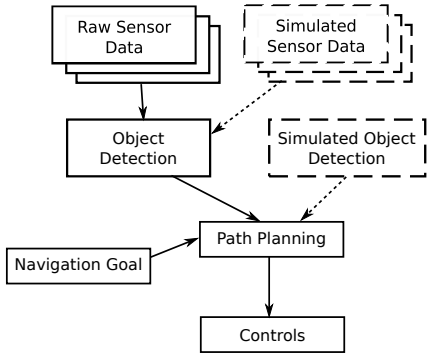

Figure 4: A block diagram with simulated (dashed) and real (solid) sensor data and detections

2.3 Augmenting μCity with simulated vehicles and sensors

With the modular design of μCity, it is possible to integrate simulated objects into the environment. In particular, with knowledge of the locations of every object in μCity, it is relatively simple to create new virtual sensors on AVs that generate or ingest appropriate data on which the AVs can act upon. Data going to these virtual sensors can either be based on the locations of real objects, or the locations of simulated ones. Lidar, radar, and ultrasound sensors can all be simulated accurately given the complete knowledge of the environment which is continuously tracked by our control infrastructure. While it is difficult to simulate cameras in the same way, the result of model inference on cameras can be simulated, which achieves a similar result. It is also possible to create a combination of real and virtual sensors. For instance, consider an experimenter who wants to experiment with radar in different places on their AV. Let us assume they already have four cameras, one pointed in each direction and want to explore their vehicle's performance with radar. Instead of buying or manufacturing small-scale radar systems to fit the laboratory-scale AVs, such sensors can be simulated. The user can even adjust the virtual radar's parameters to experiment with different types of radar. While they are testing these different parameters, they can run actual sensing algorithms on physical cameras while the vehicle creates and follows its own paths.

2.4 Autonomous Vehicle Hardware

The design of the AV hardware itself is also important. In the F1TENTH competition, the vehicles are based on a commercial RC car (a Traxxas Slash 4x4 Platinum Edition) with some custom laser-cut and 3D-printed parts to mount sensors and an embedded computer. This approach of using custom designed parts with a commercial chassis is valuable and what we propose for this system. There are a number of advantages. First, this saves development time. Users can buy a pre-assembled RC car, remove some parts and add the custom parts. This is quicker than assembling a car from scratch. Furthermore, the commercial RC cars have proven designs developed by engineers that have familiarity in designing RC cars of that scale. Designing one from scratch would be time consuming. Secondly, when the vehicle is damaged, the majority of the parts for the vehicle are available online and can be kept in stock. This reduces downtime for damaged vehicles. Thirdly, it is an advantage to have a common platform between groups as this allows the sharing of resources between groups. If one group trains a model to recognize the cars, it is likely that the same model will also recognize cars built by another group. This is also convenient as car-specific parameters (such as turning radius, maximum speed, wheel width, etc.) will be consistent across cars, making sharing software between groups easier.

The size of the vehicle is also important to consider as there is a tension between the size of the vehicles and the amount of computation and sensors they are capable of carrying. With smaller vehicle sizes, an increasing number of streets can fit within a given laboratory space, but the vehicles can fit less sensors and processing on board, which means there is less diversity in the amount of experiments that can be run.

At the very minimum, each μCity vehicle should be able to carry a camera or lidar, a capable embedded computing system (that is one powerful enough to either stream the camera or lidar data with low latency or process the sensor data locally) and the hardware to support the chosen motion capture system. If algorithms with more sensors need to be implemented on this minimally sized vehicle, the virtual sensors from Section 2.3 can be implemented. Alternatively, a larger vehicle can be created to hold physical versions of the sensors of interest and a powerful enough embedded computer to process or stream data from those sensors.

3 FEASIBILITY STUDY

3.1 Experimental Setup

We implemented a prototype of μCity with basic functionality to determine if the system is feasible. The prototype fits within an area with dimensions of approximately 6 x 5m. It uses 4 Lighthouses from the localization system for the HTC Vive virtual-reality headset. The system also has a single miniature autonomous vehicle. The vehicle is a modified H-King Desert Fox 1/10th scale RC truck. The vehicle uses an Nvidia Jetson Xavier NX embedded computer specialized for machine learning tasks which runs Ubuntu 18 and Robot Operating System (ROS) [12]. It also has a single Intel Realsense 435i, RGB-Depth camera, as well as a Vive Tracker 2.0 that provides the car's location to the rest of the system. Some of the vehicle's components have been replaced with 3D printed versions to provide mounting locations for the Jetson Xavier NX, the Vive Tracker, and the Realsense camera. Figure 1 shows an image of the vehicle. There are also two 3D-printed stoplights (Figure 3), a wifi network, and a server to process localization data and control the experiments.

3.1.1 Object Detection Benchmark. We ran an SSD MobileNet detection model on the Jetson Xavier Nx to evaluate its performance for our application. The SSD MobileNet combines SSD detection [6] with a MobileNet [4] backbone. We created our own custom dataset (886 images 1280 x 720) by taking images of our 3D printed stop lights and generating corresponding bounding box annotations. We utilized this dataset to fine-tune our SSD MobileNet detection model, which was pre-trained by Nvidia on the Pascal VOC dataset (11,530 images).

The SSD-MobileNet model provides accuracy whilst meeting the low latency constraint, which is vital for real-time detection systems. In order for a detection system to be considered real-time, it needs to perform inference at a minimum of 30 frames per

second [8]. We verified that our test framework is indeed real-time by performing a latency test on the SSD-MobileNet model. Over all the images in our dataset, the average inference time was 0.01 s with a standard deviation of 0.0011 s. The minimum inference time for an image was 0.0073 s and the maximum was 0.016 s. Thus, our test system is, on average, able to perform detection at a speed of 100 fps, which is comfortably in the real-time range. Using these performance metrics, we can expect to be able to run object detection on three cameras simultaneously.

3.1.2 Tracking-based Labeling. As a proof-of-concept, an experiment was performed to test the efficacy of automatic labeling with tracking data. The goal was to recreate human-labeled bounding boxes of stoplights using a camera and tracking system. To do this, three values need to be measured accurately: the location of the camera, the location of the object to be labeled (a 3D printed stoplight), and the intrinsic matrix, T_i of the camera taking the images. T_i is a 4x4 matrix that represents how the camera maps 3D points to 2D coordinates on an image. We also need a set of 3D points, $P_l[n]$, in the stoplight's frame that represent the 3D bounds of the object to be labeled. For 3D printed objects, these points can be easily created from the object's STL file. For non-3D printed objects, these points can be measured.

To calculate the bounding box for the object, we first obtain T_c^l, the homogeneous transform between the camera and the light (this is provided by the tracking system). Then, we transform each of the points in the object's 3D bounding box into the camera's frame: $P_c[n] = T_c^l \cdot P_l[n]$. Next, we use the intrinsic matrix to calculate the image space locations of the 3D bounding box locations: $P_i[n] = T_i \cdot P_c[n]$. Lastly, we obtain the bounding box coordinates B_1 and B_2 by finding the minimally sized box that contains all the locations.

We compared the bounding boxes generated using this method with human-labeled images using the intersection over area (IoU) metric: $(BB_h \cap BB_a)/(BB_h \cup BB_a)$ where BB_a is the automatically labeled bounding box, and BB_h is the human labeled one.

Over 128 images captured from 0.48m to 4.13m away from the traffic light, we achieved an average IoU of 0.686 with a standard deviation of 0.143. It can be seen from figure 5 that the IoU value drops off as the camera moves further from the stop light. We consider an IoU of 0.5 to be acceptable for labeling (although the bounding box's size will need to be increased to ensure the light is within the box), and an IoU of 0.25 to be acceptable for navigation purposes. With a few exceptions, the current setup should be able to label images within about 3.17m and navigate with tracking-derived labels up to at least 4.13m. While these distance are short for full-sized vehicles, at a 1/15 scale, they are equivalent to 47.6m and 62.0m on a full-sized vehicle.

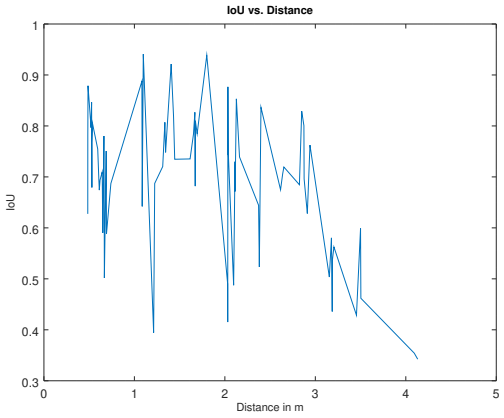

Figure 5: IOU value at different distances

4 SUMMARY AND CHALLENGES AHEAD

We propose μCity, a miniature autonomous vehicle testbed. μCity complements currently existing methods of testing autonomous vehicles. It is low-cost, repeatable, safe, flexible, all while being easy to use and realistic. It allows for a mixture of simulation and realistic scenarios, and expands access to autonomous driving technology. While the system has great promise, there are still some challenges that need to be overcome for it to reach maximum effectiveness.

Next steps: A challenge for miniature AVs is that they have significantly less room than full-size vehicles for sensor and compute hardware. Tesla vehicles, for example, use 8 cameras and 12 ultrasonic sensors to accomplish their self-driving features. It is difficult to find a compute platform small enough to fit in our proposed vehicles that is capable of processing 8 simultaneous video streams.

Another challenge is that of reproducing dynamic obstacles like pedestrians. In our proposed system, pedestrians can be created using small, motorized platforms. While this can be used for certain types of tests, real pedestrians have more nuanced movement that is difficult to recreate at a small scale.

Furthermore, as the experiment in Section 3.1.2 showed, there is still a need to improve the localization-based labeling techniques. While these performed well enough for a 1/15 scale vehicle at an effective 47.6m, a greater range would be beneficial. This might be achieved by either a higher-fidelity localization system, a more accurate mounting methods for the camera, or a better estimation of the camera matrix.

Finally, we believe this can be a community effort, where different contributors can enhance μCity and extend them in different ways, e.g., for easy integration of diverse weather conditions, light conditions, and the like. Similarly, various simulated environments can be instantiated that focus on specific traffic patterns of interest. We look forward to working together with others towards making μCity useful to a broad class of researchers.

5 ACKNOWLEDGEMENTS

All authors were supported in part via the following awards: US National Science Foundation's CNS-2112562, CNS-2107060, CNS-2003129, CNS-1838733, and CNS-1647152 and and the US Department of Commerce's 70NANB21H043.

REFERENCES

[1] 2015. Mcity Grand Opening. *UMTRI Research Review* 46, 3 (Sep 2015), 1–3.
[2] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR, 1–16.
[3] Daniel J Fagnant and Kara Kockelman. 2015. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice* 77 (2015), 167–181.
[4] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[5] Alan Lao. 2021. Self-Driving Cars FAQ: How Far Away Is Far Away? *Motor Trend* (Jul 2021).

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single Shot MultiBox Detector. *Lecture Notes in Computer Science* (2016), 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

[7] Matthew O'Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. 2020. F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning. In *Proceedings of the NeurIPS 2019 Competition and Demonstration Track (Proceedings of Machine Learning Research, Vol. 123)*, Hugo Jair Escalante and Raia Hadsell (Eds.). PMLR, 77–89. https://proceedings.mlr.press/v123/o-kelly20a.html

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640 [cs.CV]

[9] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. 2020. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1–6.

[10] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*. Springer, 621–635.

[11] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 0 (2012), –. https://doi.org/10.1016/j.neunet.2012.02.016

[12] Stanford Artificial Intelligence Laboratory et al. [n.d.]. *Robotic Operating System*. https://www.ros.org