The MiniCity: A 1/10th Scale Evaluation Platform for Testing Autonomous Urban Perception and Planning

Noam Buckman[1,†], Alex Hansen[1,†], Sertac Karaman[2] and Daniela Rus[1]

I. Introduction

A major impediment to the adoption of autonomous vehicles (AVs) is the need to fully evaluate and test the full autonomous vehicle hardware and software stack in realistic traffic scenarios. This is especially challenging for perception tasks, such as object detection and localization, which impact various components of the full AV stack and depend heavily on sensor configuration. Recent work [1], [2] highlight the need for evaluating perception algorithms in the context of both the whole autonomous system and specific downstream tasks, such as obstacle avoidance, which is difficult to accomplish using existing datasets or simulators. In this work, we propose the MiniCity, a miniature autonomous vehicle platform for evaluating perception algorithms in a city-wide, multi-vehicle scale. In the MiniCity, 1/10th scale vehicles are equipped with full-scale hardware – Lidar, stereo cameras, and IMUs – and a full autonomy software stack – allowing researches to evaluate their perception algorithm in isolation and the impact to the vehicle's quality of autonomous driving.

Current tools for evaluating autonomous vehicle software and hardware consists of datasets, simulation, or full-scale vehicles. The high-cost and inherent safety risk of full-scale vehicles mean that most full-scale testing is limited to closed course testing or tasks with limited interactions with other vehicles. Increasingly, researchers are relying on datasets or simulators for benchmarking the performance of their algorithms. Datasets [3], [4], [5], [6], [7], [8] provide high-fidelity sensor recordings and are thus a popular choice for evaluating perception tasks such as object pose estimation and lane detection; however, they can not evaluate the impact on downstream modules such as trajectory planning or collision avoidance. Simulators [9], [10], [11], [12] can allow for evaluating the full AV stack; however, they fall victim to the sensor sim-to-real gap and incur high computational cost for simulating multi-vehicle interactions. Miniature robot platforms [13], [14], [15], [16], [17] provide a middle ground in evaluation platforms, providing researchers a lower cost option that can enable real hardware testing while measuring an algorithm's impact on both the individual task (object detection) and the impact on the rest of the autonomy stack (such as collision avoidance).

(a) MiniCity (b) RACECAR

Fig. 1: The MiniCity.

The MiniCity bridges the gap between real-world deployment and simulated testing. The MiniCity's 1/10th scale urban setting consists of small-scale houses, roads, traffic lights, and fully autonomous vehicles, enabling researchers to test within a city setting without the dangers of real world testing. In this work, we highlight the MiniCity's ability to evaluate a vehicle's perception software and hardware, by deploying baseline perception tasks such as object detection and localization in interactive scenarios with multiple autonomous vehicles.

II. MiniCity Platform Description

A. Physical Layout

The MiniCity is a 1/10th scale evaluation platform consisting of scaled houses, roads, and traffic infrastructure, multiple intersections for interactive scenarios, and external motion capture for evaluating the vehicle performance. The MiniCity's roads are made from durable 2ft wide x 1/4" thick rubber gym mats with gaphers masking tape used for lane lines. Doll houses and synthetic grass are placed along the road to add realistic scenery and occlusions. The MiniCity's photorealisim allows us to deploy perception algorithms in environments that appear similar to deployment. The overall size of the MiniCity can expand to multiple intersections, with an overall length of 40 ft, or as short as 16 ft with a single intersection. The small size of the MiniCity relative to a full-scale city allows for experiments with various topological and environmental changes. All physical assets can be re-arranged for different road structures, weather settings, and scenery types.

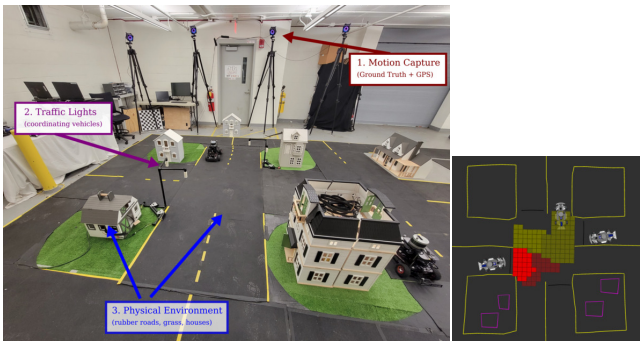
A recently published extended journal version of this work is accessible at https://www.mdpi.com/1424-8220/22/18/6793

This work was supported by the Toyota Research Institute (TRI). This article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

† equal contribution

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA [nbuckman, hansena, rus] at mit.edu

[2]Laboratory of Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA sertac@mit.edu



(a) (b)

Fig. 2: MiniCity Infrastructure. **a)** The Minicity setup with traffic lights, houses, and optitrack. **b** Intelligent traffic manager

B. Ground Truth Position and GPS-Spoofing from Motion Capture

Ground truth localization is both necessary for evaluating localization and perception algorithms, and by providing simulated GPS signal to mimic outdoor environments. The MiniCity, as seen in Fig. 2, includes a system of 10 Optitrack PrimeX 41 motion capture cameras deployed on portable tripods. The flexible setup allows for easily moving to new indoor and outdoor spaces. The Optitrack's MOTIVE software tracks 8-12 passive reflective markers that are rigidly attached to each RACECAR, and publishes pose and orientation at 120Hz. Additionally, the motion capture system publishes a spoofed GPS signal to mimic GPS signals found in the real world using Robot Operating System's (ROS) standard NavSatFix GPS message type. The GPS-spoofing module ingests the millimeter precise pose estimate of the vehicles and publishes a noisy position measurement with various types of noise, such as Gaussian, white, and brown noise distributions.

C. Mapping the MiniCity

We provide maps of the MiniCity that are used for evaluating vehicle performance, lane line violations and traffic rules, and for use by the vehicle's onboard planner. One advantage of the MiniCity's scale is that high-definition mapping is less burdensome than real-world high-definition mapping of a full city. We map the 2D road geometry, lane lines, and building outlines in the OpenStreetMap (OSM) format, a popular open-source map format used for full-scale autonomous vehicle application. The Lanelet2 API [18] provides semantic labeling for each road segment with information such as road direction, lane lines, traffic regulatory elements (traffic lights, speed limits), and overall road route structure. The Lanelet2 API also builds a routing tree of the map's road segments which is used by the car's planner to navigate around the MiniCity.

D. Scaled Traffic Lights and Houses

Intersections and traffic signalling are unique features of city-wide driving. The MiniCity consists of multiple four-way intersections and roundabout, which enables testing of perception algorithms in realistic traffic scenery and in complex scenarios such as a vehicle taking an unprotected left turn around occluded vehicles. The physical traffic lights (Fig. 2a) consist of a to-scale plastic pipe structure, 3D printed enclosures, and pre-fabricated red-yellow-green LED board. A Raspberry Pi 4 controls the LEDs and communicates with the rest of the MiniCity software stack via ROS. The traffic lights can operate as traditional unsignalized (flashing red) and signalized (red-yellow-green) lights, or as intelligent traffic lights, such the socially-compliant autonomous intersection manager [19] shown deployed in the MiniCity in Fig. 2b. The MiniCity also consists of fake grass and doll houses to mimic background scenery during city driving. The houses also provide challenging perception scenarios such as occluded vehicles and obstructed pseudo-GPS. Figure 3 shows a few example views from the RACECAR's onboard cameras that show how the MiniCity mimics full-scale driving scenes.

Fig. 3: Views from the RACECAR driving in the MiniCity.

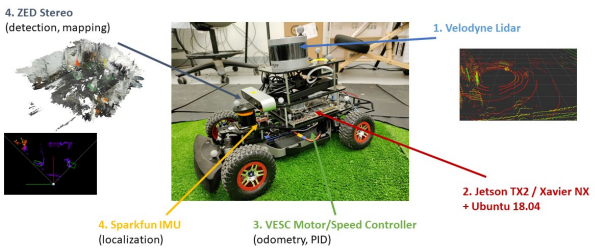


Fig. 4: RACECAR Hardware Platform

E. RACECAR Hardware

The MiniCity consists of 1/10th scale autonomous vehicles based on the RACECAR [17] platform, as shown in Fig. 4. We provide configurations for multiple types of sensors and compute which allow for comparing various hardware configurations. For compute, the RACECARs use either an Nvidia Jetson TX2 or the newer Jetson Xavier NX, the latter consisting of a NVIDIA Volta GPU, 6-core ARM CPU, and 8GB RAM. The sensor suite is composed of a VLP-16 Velodyne Lidar, a Hokoyu 2d Lidar, a 9DoF Sparkfun IMU, a ZED stereo camera, and an Enertion FOCBOX speed controller that supplies wheel encoder odometry. The enhanced computation and sensing from previous versions of RACECAR and other educational platforms means we can deploy full-scale algorithms on the miniature vehicles.

The platform's code uses ROS Melodic for interprocess communication and external vehicle-to-all (V2X) communication. We compartmentalize each component of the pipeline into its own ROS Node to easily allow swapping algorithms and comparing component performance. For GPU intensive processes, such as object detection and lane detection, we implement the algorithm in NVIDIA's Linux4Tegra Docker container and publish ROS topics over the host vehicle's networking.

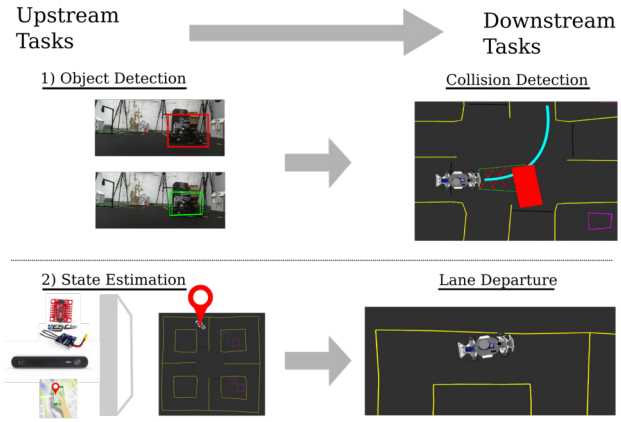
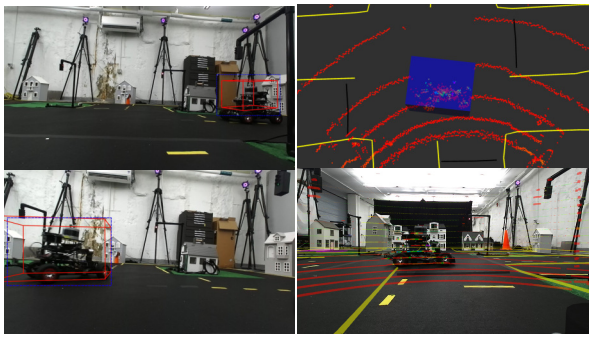


Fig. 5: Upstream and Downstream Perception Tasks

III. Evaluating Perception Effects on Motion Planning

A. Upstream and Downstream Tasks

Perception tasks typically are located at the very earliest, or upstream, stages of any autonomous vehicle stack. For example, the vehicle's ability to estimate its own location in a map, directly affects the vehicle's ability to generate trajectories and control the vehicle on the road. Similarly, the output of an object detector – pose estimates and bounding boxes of ado vehicles – directly impact an autonomous vehicle's ability to avoid obstacles and drive safely. A main contribution of the MiniCity is the ability to safely test both the upstream and downstream performance of perception and motion planning algorithms.



(a) StereoRCNN Detections (b) PIXOR Detections

Fig. 6: Onboard Comparison of object detectors in the presence of ado vehicles.

B. Object Detection Effects on Collision Avoidance

Each RACECAR is equipped with both a Velodyne VLP-16 Lidar and Zed stereo camera, allowing us to test multiple classes of perception algorithms. We deploy two state-of-the-art object detectors using one or both of these sensors. Stereo-RCNN [20] feeds a pair of stereo images to a regions with convolutional neural network (RCNN) to predict keypoints, regions of interest (ROI), and object classes and finally 3D bounding boxes for each vehicle. Given that Stereo-RCNN's classifier is typically pre-trained on Imagenet or similar datasets that lack pictures of RACECARS, we retrain the network using images of RACECARS. We also deploy PIXOR [21] which first creates a birds-eye-view feature map to input into a convolutional neural network (CNN) that computes a pixel-level estimate of the objects pose and orientation.

The object detection evaluation begins with deploying each detector on the vehicle while running the full autonomy stack. An ado vehicle navigates the MiniCity autonomously as drone traffic, running its own collision avoidance and control. In addition, a human operator simulates high-risk scenarios such as an ado car speeding through an intersection or stopped at a cross road. The ego vehicle operates autonomously with a mission of picking up and dropping drivers, using either a camera-based StereoRCNN detector or PIXOR detector.

TABLE I: Evaluation of object detectors in the MiniCity.

Method	Detections		Handovers	Collision Avoidance	
	Recall	Precision	per min	Sensitivity	Specificity
Ground Truth	-	-	0.00	0.89	0.98
StereoRCNN [20]	0.061	0.091	2.05	0.16	0.93
PIXOR [21]	0.442	0.559	0.39	0.80	0.73

For a given detector, the MiniCity evaluates the upstream task of accurately detecting and estimating the pose of other RACECARs, by computing the intersection-over-union (IOU) of the 3D bounding boxes projected to the bird-eye-view plane. The onboard detection recall and precision are presented in Table I in the first two columns with $\alpha_{IOU} = 0.05$. For downstream evaluation, we focus on the collision avoidance capabilities of the cars which is directly related to the accuracy of the object detector and pose estimator. We measure the number of human handovers per minute (due to collision errors) and the subsequent sensitivity and specificity of the collision avoidance detector. The sensitivity and specificity is defined using ground truth detections to evaluate the true positive and negative rate of the collision avoidance detector activating, and comparing with the actual activation of the collision avoidance (CA) module.

C. State Estimation Effects on Lane Following

State estimation is done through an Extended or Unscented Kalman Filter, implemented by the open-source robot_localization ROS package [22]. As inputs, the Kalman Filter takes odometry estimates from the onboard stereo ZED camera, the GPS from Optitrack, wheel encoder velocity

TABLE II: Upstream evaluation of localization algorithms

Sensor Configuration	Position Error Mean (m)	Position Error Stdv. (m)	Position Error Change (%)	Angular Error Mean (-)	Angular Error Stdv. (-)	Angular Error Change (%)
All Sensors	0.1465	0.013	-	0.1458	0.016	-
No Zed/GPS	0.1757	0.029	19.94	0.1445	0.012	**-0.91**
No Zed	0.1465	0.013	-0.04	0.1445	0.012	-0.87
No GPS	0.2152	0.086	46.90	0.1445	0.015	-0.89
No IMU	0.1468	0.014	0.18	0.1513	0.021	3.74
No linear IMU	0.1464	0.013	**-0.09**	0.1484	0.018	1.78
IMU + Encoder Only	0.1742	0.027	18.86	0.1459	0.013	0.06

estimates, linear and angular accelerations from the Spakfun IMU. The RACECAR uses the pose estimate to localize within the static OSM map, inferring its current lane for routing and its relationship to the intersection. In addition, the RACECAR's onboard Zed stereo camera continuously generates a 3D map of the environment which can be used alongside the OSM for localization.

TABLE III: Downstream evaluation of state estimation

Sensor Configuration	Frequency of Line Violation (% of run duration)	Severity of Line Violation (% of car body over line)
GPS + IMU + Encoder	10.3	2.7
GPS-Only	23.3	6.3
IMU + Encoder Only	35.4	16.1

We focus on the relative contribution of various sensor modalities on the overall quality of the onboard state estimation of the vehicle's pose. Specifically, we evaluate the effect of each sensor on the vehicle's estimate of its position $\boldsymbol{p} = [x, y, z]^T$ and orientation represented by quaternion $\boldsymbol{q}$ in the MiniCity's reference frame. We use the high quality ground truth pose provided by Optitrack to compare the state estimate to the ground truth pose, for various sensor configurations. Results for position error and angular error are reported in Table II, with Position Error Metric $= ||\boldsymbol{p} - \boldsymbol{p}_{gt}||_2$ and Angular Error Metric [23] $= \left|\left| \log\left(R(\boldsymbol{q})R(\boldsymbol{q}_{gt})^T\right)\right|\right|$, where $R(\boldsymbol{q})$ is the rotation matrix corresponding to rotation $\boldsymbol{q}$, and $\boldsymbol{p}_{gt}$ and $\boldsymbol{q}_{gt}$ are the ground truth position and orientation, respectively. For the upstream evaluation, we re-run the Kalman Filter with different sensor configurations and measure the relative position and angular error as a percentage difference from our baseline with all sensors (Row 1).

In Table III we evaluate the downstream effects of various sensor configurations by evaluating the percentage of time the vehicle crosses a traffic lane line, where a lane line violation is defined as any part of the car crossing a road border or yellow line. In addition, to quantify the severity of the line violations, we report the average percentage of the car body that crosses over the line during a line violation. For downstream evaluation, we compare three localization configurations and repeat each run four times. We find that when utilizing the full sensor suite for localization (IMU, GPS, encoder), the lane violations correspond to only a very small percentage of the body over the line.

REFERENCES

[1] X. Huang, et al., "Tip: Task-informed motion prediction for intelligent vehicles," arXiv preprint arXiv:2110.08750, 10 2021.

[2] A. Buhler, et al., "Driving through ghosts: Behavioral cloning with false positives," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5431–5437, 10 2020.

[3] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 3354–3361, 2012.

[4] H. Caesar, et al., "nuscenes: A multimodal dataset for autonomous driving," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11 621–11 631, 6 2020.

[5] M. F. Chang, et al., "Argoverse: 3d tracking and forecasting with rich maps," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2019-June, pp. 8740–8749, 2019.

[6] P. Sun, et al., "Scalability in perception for autonomous driving: Waymo open dataset," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2443–2451, 2020.

[7] J. Geyer, et al., "A2d2: Audi autonomous driving dataset," arXiv preprint arXiv:2004.06320, 4 2020.

[8] X. Huang, et al., "The apolloscape open dataset for autonomous driving and its application," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, pp. 2702–2719, 2020.

[9] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," pp. 621–635, 2018.

[10] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4cv: A photo-realistic simulator for computer vision applications," International Journal of Computer Vision, vol. 126, pp. 902–919, 9 2018.

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," Conference on Robot Learning, pp. 1–16, 2017.

[12] A. Amini, et al., "Learning robust control policies for end-to-end autonomous driving from data-driven simulation," IEEE Robotics and Automation Letters, vol. 5, pp. 1143–1150, 2020.

[13] M. O'kelly, et al., "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," Proceedings of Machine Learning Research, vol. 123, pp. 77–89, 2020.

[14] B. Balaji, et al., "Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning," Proceedings - IEEE International Conference on Robotics and Automation, pp. 2746–2754, 2020.

[15] B. Goldfain, et al., "Autorally: An open platform for aggressive autonomous driving," IEEE Control Systems, vol. 39, pp. 26–55, 2019.

[16] L. Paull, et al., "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 1497–1504, 5 2017.

[17] S. Karaman, et al., "Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at mit," ISEC 2017 - Proceedings of the 7th IEEE Integrated STEM Education Conference, vol. 00, pp. 195–203, 2017.

[18] F. Poggenhans, et al., "Lanelet2: A high-definition map framework for the future of automated driving," IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, vol. 2018-Novem, pp. 1672–1679, 2018.

[19] N. Buckman, A. Pierson, W. Schwarting, S. Karaman, and D. Rus, "Sharing is caring: Socially-compliant autonomous intersection negotiation," IEEE International Conference on Intelligent Robots and Systems, pp. 6136–6143, 11 2019.

[20] P. Li, X. Chen, and S. Shen, "Stereo r-cnn based 3d object detection for autonomous driving," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2019-June, pp. 7636–7644, 2019.

[21] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 7652–7660, 2018.

[22] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," Advances in Intelligent Systems and Computing, vol. 302, pp. 335–348, 2016.

[23] D. Q. Huynh, "Metrics for 3D rotations: Comparison and analysis," Journal of Mathematical Imaging and Vision, vol. 35, no. 2, pp. 155–164, 2009.