HydraOne: An Indoor Experimental Research and Education Platform for CAVs

Yifan Wang[*†‡], Liangkai Liu[*], Xingzhou Zhang[*†‡], Weisong Shi[*]
[*]*Wayne State University*
[†]*SKL of Computer Architecture, Institute of Computing Technology, CAS*
[‡]*University of Chinese Academy of Sciences*
{*wangyifan2014, zhangxingzhou*}*@ict.ac.cn*, {*liangkai, weisong*}*@wayne.edu*

Abstract

Connected and autonomous vehicles (CAVs) is currently a hot topic and a major focus in the field of edge computing, and it has created numerous pivotal and challenging research problems. In this paper, we present HydraOne, an indoor experimental research and education platform for edge computing in the CAVs scenario. HydraOne is a hardware-software co-design platform built from scratch based on our experience with the requirements of edge computing research problems. We present the design and implementation details and discuss three key characteristics of HydraOne: design modularization, resource extensibility and openness, as well as function isolation. These will help researchers and students fully understand the platform and take advantage of it to conduct research experiments. We also provide three case studies deployed on HydraOne to demonstrate the capabilities of our research platform.

1 Introduction

There are numerous academic studies and industrial works of edge computing that have emerged in the past few years, crossing various domains [21, 23]. Many researchers and developers have focused more attention on critical edge applications [11, 27], framework and middleware for edge computing [32], security and consistency on edge [15], and IoT-edge-cloud interactions [24] *etc.*. However, few studies have discussed how to develop a research platform for edge computing in a specific application scenario, which is more important for sharing and spreading research achievements in the edge computing field.

In the cloud computing domain, the main function of the research platform is data processing. Researchers focus more on computing performance when building the research platform for cloud computing. The development of virtualization technology [16] and distributed computing [4] allowed researchers to build their private cloud computing platform via some open-source framework [22]. In the Internet of things (IoT) domain, the research platform is used to collect and transmit sensor data, so researchers are more concerned about the peripheral interface resources and wireless communication capability of the research platform. Many classic IoT platforms for research and education have been released, such as, Arduino [2] and Telos [18]. The edge computing domain has the characteristics of both cloud computing and the IoT domain. The research platform for edge computing should have enough computational capabilities, and it could collect the data from the data producers in specific computing scenarios and communicate with other entities in the network.

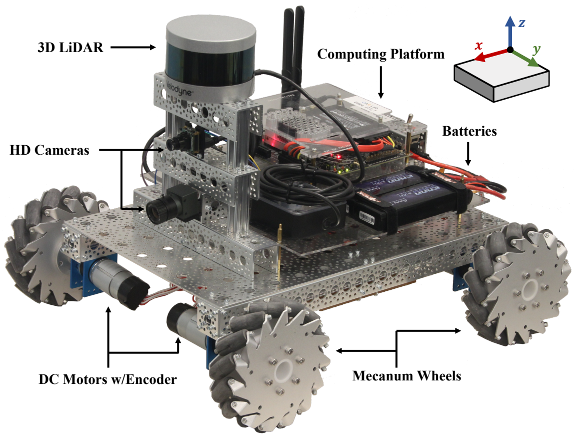


Figure 1: Overview of HydraOne Platform.

Connected and autonomous vehicles (CAVs) are already changing our vision about future vehicles and transportation. A recent report shows that each connected and autonomous vehicle will generate about 4,000GB of data per day [9]. The majority of vehicular data would be processed in the vehicle due to network bandwidth restrictions under which CAVs become a typical edge computing system. A large number of research opportunities still remain in the field of edge computing in CAVs. However, it is challenging for researchers to deploy applications and systems designed for CAVs in real-world environments due to the lack of a research platform for CAVs. To address this challenge, we propose HydraOne, an indoor experimental research and education platform for edge computing in the CAVs scenario. As shown in Figure 1, HydraOne is a full-stack research and education platform from hardware to software, including mechanical components and vision sensors, as well as a computing and communication system. All the resources on HydraOne are managed by the Robot Operating System (ROS) [19]. HydraOne has three key characteristics: design modularization, resource extensibility and openness, as well as function isolation, which allows users to conduct various research and education experiments on CAVs with HydraOne.

While HydraOne is an indoor robot-based platform, it has sufficient resources and components to conduct CAVs experiments. The computing platform on HydraOne collects and processes the sensor data in real-time; then it outputs the control message to the chassis to control the moving speed and direction of the platform, which enables the autonomous driving capability of HydraOne. Users can develop the algorithms of sensor fusion, perception, and decision-making on the platform. HydraOne is equipped with a Wi-Fi module to communicate with the cloud and the edge server, which allows users to conduct the experiments of Vehicle-to-everything (V2X) on HydraOne. In addition to autonomous driving and communication, the research problems supported by HydraOne include but are not limited to the operating system designed for CAVs, safe and trusted execution environments on CAVs, and privacy preservation model and tools for vehicular data.

2 Related Work

In this section, we summarize the related work from the perspective of two research platforms: for autonomous devices and edge computing.

Research Platform for Autonomous Devices. Wei *et al.* presented the CMU autonomous driving research platform, which is based on a Cadillac SRX [29]. This work focuses on vehicle engineering problems, including the actuation, power, and sensor systems on the vehicle. Tomic *et al.* presented an autonomous UAV research platform for urban search and rescue [26]. They introduced the hardware infrastructure of their platform and provided a set of algorithm libraries to help developers complete the urban search and rescue task. Pheeno [30] and r-one [14] both are research and education platform for multi-robot manipulation. These studies designed a low-cost robot platform with a small number of sensors and a low-power communication module, which can help researchers to deploy the experiment of versatile swarm robotic.

Research Platform for Edge Computing. ParaDrop [12] is an edge computing platform designed for multi-tenant on wireless gateways. It uses Linux Container (LXC) to manage the resource on the wireless gateway, which allows researchers to implement their edge computing applications on the gateway. Φ-stack [31] is a full-stack research platform for the smart web of things. Φ-stack contains a novel smart IoT processor(ΦPU) and a RESTful-based software framework (ΦOS and ΦDK), which natively supports the web and intelligence. Researchers can use Φ-stack to deploy the intelligence workloads via RESTful API on low-power smart IoT devices.

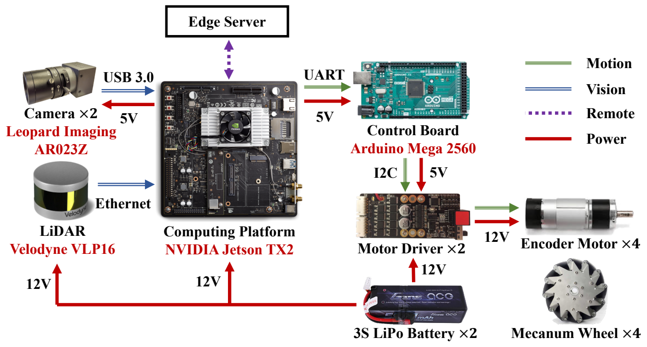


Figure 2: HydraOne Hardware Design

It should be noted that there some similar platform already exist in the community, such as MIT RACECAR [8], DJI RoboMaster Robots [5], and Audi Autonomous Model Cars [1], but they only can be acquired by participating in specific competitions. HydraOne provides an open platform for researchers and students to be able to build their own CAVs experimental platform according to this paper.

3 Design and Implementation

In this section, we introduce the design and implementation details of HydraOne and present three key characteristics of the platform.

3.1 Hardware Design

Design Overview. As shown in Figure 2, HydraOne is equipped with a NVIDIA® Jetson^{TM}TX2 embedded module [17] as the core computing platform. Jetson TX2 collects the data from multiple sensors and feeds the data to several computing tasks in real-time. An Arduino board receives the control message output from the computing tasks via serial port (UART) then sends the control signals to the motor drivers to control the movement of HydraOne. Jetson TX2 is equipped with a Wi-Fi module which allows HydraOne to communicate with an edge server or other entities in the network. The whole HydraOne platform is powered by two 3S LiPo Batteries.

Sensors. HydraOne is equipped with two HD cameras and one 3D LiDAR which form the basic vision system of HydraOne. The sensors' models are listed as below:

- 2×Leopard Imaging AR023Z 1080p HD color camera, $1920 \times 1080@30$, rolling shutter;
- 1×Velodyne VLP-16 rotating 3D laser scanner, 16 channels, collecting ~ 3 million points/second, field of view: 360° horizontal, 30° vertical, range: 100m.

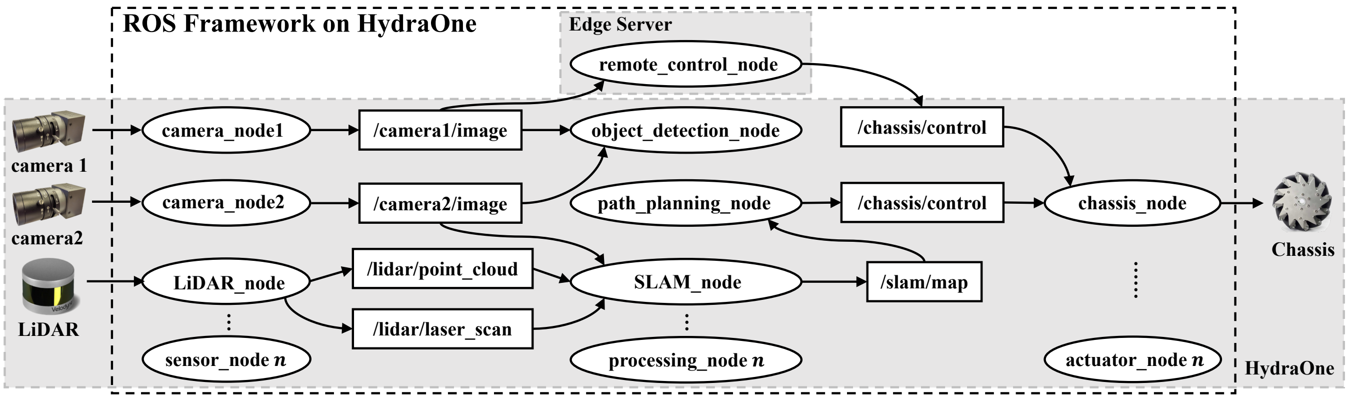


Figure 3: HydraOne Software Framework

Computing Platform. The computing platform on HydraOne processes complicated computing tasks, such as computer vision and machine learning algorithms, to support the various applications of CAVs. The traditional microprocessor cannot meet the computing power requirement of the CAVs scenario, so we should deploy a single board computer on the HydraOne platform. NVIDIA® Jetson™TX2 is a power-efficient embedded AI computing platform which has dual-core Denver and quad-core ARM® Cortex-A57 CPU equipped with 8GB DDR4 memory and 32GB eMMC. The GPU on Jetson TX2 is powered by NVIDIA Pascal™architecture with 256 CUDA cores. The CPU-GPU architecture on Jetson TX2 can accelerate the deep learning workloads which have become an integral part of the CAVs scenario [10, 28]. Several studies have deployed the Jetson TX2 on autonomous devices [8, 25]. Therefore, Jetson TX2 is currently the most suitable choice for a computing platform on HydraOne.

Chassis. The HydraOne platform has a four-wheel drive chassis which is equipped with four DC motors with encoders and two encoder motor drivers. The Proportional-Integral-Derivative (PID) control algorithm is integrated into the motor drivers to achieve precision control for each motor speed. The chassis has four Mecanum wheels (a kind of Omni-wheel) so that the HydraOne can achieve omnidirectional movement. The control message format output from computing platform is *geometry_msgs/Twist* (this will be introduced in the next subsection); the Arduino board is in charge of converting the message to motor speed value and sending the speed value to the motor drivers via I2C bus.

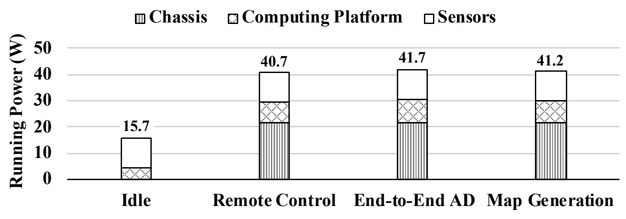


Figure 4: Running Power Breakdown on HydraOne.

Power System. The electronic components and chassis have an independent power supply. Each is powered by one 3S LiPo Battery. We present the running power breakdown of HydraOne in Figure 4. The three applications will be introduced in Section 4. The results show that the whole platform consumes 41.2w on average when running workloads, and sensors, computing platform, and chassis consume 11.2w (27.2%), 8.1w (19.7%), 21.9w (52.2%), respectively.

3.2 Software Framework

Framework Overview. The operating system on Jetson Tx2 is Ubuntu 16.04, so users can easily install the open-source vision and machine learning libraries, like OpenCV, TensorFlow, and PCL *etc.*. To manage the hardware and software resources on HydraOne and provide a clear and easy-to-use development model to researchers, we deploy the Robot Operating System (ROS) [19] on HydraOne. We choose ROS Kinetic Kame distribution, which is the most compatible version to Ubuntu 16.04 to date. The ROS framework on HydraOne is illustrated in Figure 3. All resources and computing tasks on HydraOne can be abstracted as ROS nodes, and they use the publisher-subscriber pattern to share data and results to implement one or several CAVs applications collectively. The communication between HydraOne and the edge server is also implemented by ROS. HydraOne runs the ROS master node, and the edge server should configure the IP address and port of the master node in its environments.

Node Management. The ROS nodes on HydraOne are divided into three categories according to their function: sensor node, processing node, and actuator node. The sensor nodes are data producers that collect the data from hardware sensors and publish them in real-time. It must be noted that one sensor node could publish multiple types of data, and all data producers can be considered as sensor nodes, such as the motor speed monitor node and the battery status monitor node *etc.*. The processing nodes are the instantiation of edge computing on HydraOne, so all CAVs computing workloads should be implemented in the processing nodes. Some processing nodes will publish the middle results to other nodes, and some will publish the control message to actuator nodes. The actuator nodes are in charge of controlling the hardware actuator (*e.g.,* chassis motors) to make correct and prompt responses according to the control message. The actuator can receive the message from multiple processing nodes, so users should manage the control priority when more than one processing node is publishing the control message to the same actuator node.

Message Flow. An ROS message is essentially an implementation of inter-process communication (IPC). ROS nodes can pass the sensor data, processing results, and control message via the ROS message to others. The execution process of the CAVs application can be regarded as message passing and processing among the ROS nodes, which can be abstracted as message flow. Some properties of message flow in our framework are summarized as follows:

- An application of CAVs deployed on HydraOne can be abstracted as one or several message flows
- Message flow consists of messages and nodes, messages connect nodes, and nodes pass or transform messages;
- Ordinarily, the message flow starts from sensor messages and end with control messages to actuator nodes.

Development Model. The development model of HydraOne is based on the node management method and message flow abstraction we mentioned above. We provide sensor nodes of the vision sensors on HydraOne and the actuator node of the HydraOne chassis. The sensor data ROS format is *sensor_msgs/Image* (cameras), *sensor_msgs/PointCloud2* (LiDAR-3D), and *sensor_msgs/LaserScan* (LiDAR-2D). The format of the control message to the chassis node is *geometry_msgs/Twist*, which contains two 3-tuple vectors indicating the linear and angular speed in x, y, z axes, respectively. The users can focus on developing the CAVs applications and algorithms on processing nodes, just subscribe the data from the sensor nodes to feed into their CAVs tasks and output the control message to control the movement of HydraOne. The development model is clear and concise, which allows researchers to test and evaluate their CAVs applications and algorithms in real-world environments easily and quickly.

3.3 Experimental Enablers

The HydraOne platform has three key characteristics: design modularization, resource extensibility and openness, and function isolation. Understanding the three key characteristics of HydraOne will help users take full advantage of the platform to conduct research and education experiments of edge computing on CAVs.

Design Modularization. The idea of modular design is inspired by LEGO$^{®}$ robot [6]; all the hardware modules are connected via standard interfaces, so users can easily test, replace, and upgrade each module. The ROS node and message is the implementation of the modular design of the software framework on HydraOne. Every node has a limited function and is connected via standard interfaces (messages). The design modularization will help users learn every module on HydraOne on both the hardware and software level and fully understand the development model of the platform.

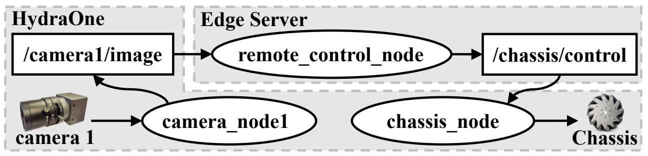


Figure 5: Message Flow of Remote Control.

Resource Extensibility and Openness. Based on the modular design, the HydraOne platform is resource extensible. The structural components of HydraOne allow users to easily mount new hardware resources on the platform, and users need to provide the driver node of each resource to publish or subscribe to data resources. All the platform resources are open. In addition to the development model we provide, users can access, analyze, and customize any resources on HydraOne, even replace the whole software framework. The resource openness allows users to explore some system and architecture research problems on HydraOne, such as, designing an operating system for CAVs.

Function Isolation. Function isolation is reflected in two aspects: within the framework and as the framework is shared with other libraries. The sensor nodes and actuator nodes are responsible for managing hardware resources, and each node only manages one hardware module. The processing nodes do not access the hardware directly to prevent exclusive access to hardware resources. The processing nodes will call other function libraries to complete the computing tasks. Some libraries provide their own process manage function, like *session.run()* in TensorFlow. We recommend that users use ROS to manage each node (process) to isolate the ROS function with other libraries, and only use the standard interface to call them. The function isolation will make it easier for users to program and debug on the HydraOne Platform.

4 Case Studies

In this section, we use three case studies deployed on HydraOne to show the capabilities of our research platform.

4.1 Remote Control

Currently, autonomous driving systems are based on computer vision and machine learning algorithms, which cause failure in some unrecognized situations [7]. However, as it is extremely hard for training datasets to cover all circumstances

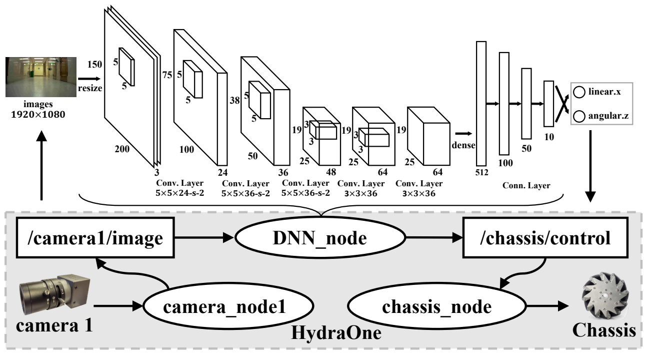


Figure 6: Message Flow of End-to-end Autonomous Driving.

of the real environment, autonomous driving system failure is unavoidable. Therefore, the remote control is an essential application to guarantee the safety of autonomous driving vehicles.

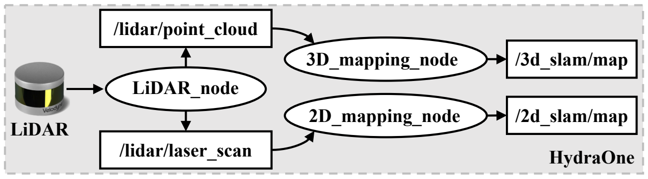
The message flow of remote control is shown in Figure 5. The remote control node is a processing node launched on the edge server. It subscribes the *sensor_msgs/Image* message, which is published by camera node, and displays the images to let the operator know the HydraOne status. The operator sends the *geometry_msgs/Twist* message via keyboard or other controllers according to the HydraOne status. The chassis node subscribes the control message to adjust the running speed and direction of HydraOne.

4.2 Autonomous Driving

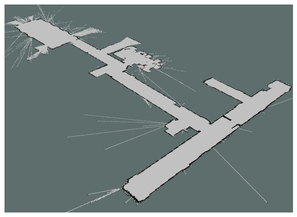
As a CAVs research platform, supporting autonomous driving is one of the core functions. Inspired by some end-to-end deep neural network (DNN) algorithms (*e.g.,* SSD [13] and YOLO [20]) and DAVE-2 [3], an end-to-end system for self-driving cars, we deploy a DNN-based end-to-end autonomous driving application on HydraOne.

The message flow of end-to-end autonomous driving is shown in Figure 6. The DNN node is a processing node launched on HydraOne. It subscribes the *sensor_msgs/Image* message and takes the resized images as the input of the DNN model. The output of the model is the linear speed on x axis and angular speed on z axis, which are fed into *geometry_msgs/Twist* message to control the chassis. The DNN model consists of five convolutional layers and four fully connected layers, and we chose ReLU as the activation function for all layers. Other details of the model, like the convolution kernel and stride size, are illustrated in Figure 6.

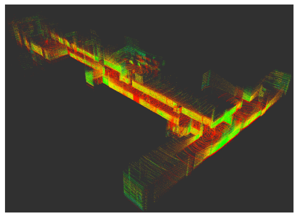
We use a joystick to control HydraOne when collecting the training data. The message data is recorded by rosbag file. The *geometry_msgs/Twist* message is the label of images, and we use the time-stamp to match them. We train the model on a GPU server and download the model to the edge (HydraOne) to process the data. The training and inference process is implemented on the TensorFlow framework.



(a) Message Flow of Map Generation



(b) 2D Map



(c) 3D Map

Figure 7: Map Generation

4.3 Map Generation

The indoor map will help HydraOne have a better understanding of the surrounding environment. Furthermore, a complete indoor map is the base data of some upper-level CAVs applications, such as path planning. We use the LiDAR data to implement an indoor mapping case study on HydraOne.

The message flow of indoor mapping is shown in Figure 7(a). The LiDAR node published 2D laser scan data *sensor_msgs/LaserScan* and 3D point cloud data textitsensor_msgs/PointCloud2. The mapping nodes subscribe to the LiDAR data and publish 2D and 3D map messages, respectively. The demo of the map data is presented in Figure 7(b) and Figure 7(c). The indoor mapping function usually runs in conjunction with the remote control or free space detection nodes to generate the map in new environments.

5 Conclusion

In this paper, we present the design, implementation, characteristics and case studies of HydraOne, the indoor experimental research and education platform for edge computing in the CAVs scenario. HydraOne is modularly designed; the resources (hardware and software) on HydraOne are extensible and all open to users. In addition, all the function modules on HydraOne are isolated. These three characteristics will allow users to take full advantage of the HydraOne to conduct research and education experiments of edge computing on CAVs. The research problems for CAVs supported by HydraOne are not only limited to algorithms and applications for autonomous driving and V2X, but also include full-stack system-related problems, such as hardware platform and architecture, operating system and software middleware, security and privacy *etc.*. We hope this platform will be valuable to researchers and students who are working on edge computing in connected and autonomous vehicles.

References

[1] AUDI AG. Audi Autonomous Driving Cup, 2019. `https://www.audi-autonomous-driving-cup.com`.

[2] Massimo Banzi and Michael Shiloh. *Getting started with Arduino: the open source electronics prototyping platform*. Maker Media, Inc., 2014.

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[4] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[5] DJI. RoboMaster Robotics Competition, 2019. `https://www.robomaster.com`.

[6] LEGO Group. LEGO Mindstorms Robot, 2019. `https://www.lego.com/en-us/mindstorms/build-a-robot`.

[7] Lei Kang, Wei Zhao, Bozhao Qi, and Suman Banerjee. Augmenting self-driving with remote control: Challenges and directions. In *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*, pages 19–24. ACM, 2018.

[8] Sertac Karaman, Ariel Anders, Michael Boulet, Jane Connor, Kenneth Gregson, Winter Guerra, Owen Guldner, Mubarik Mohamoud, Brian Plancher, Robert Shin, et al. Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT. In *2017 IEEE Integrated STEM Education Conference (ISEC)*, pages 195–203. IEEE, 2017.

[9] Brian Krzanich. Data is the new oil in the future of automated driving, 2016. `https://newsroom.intel.com/editorials/krzanich-the-future-of-automated-driving/`.

[10] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 751–766. ACM, 2018.

[11] Liangkai Liu, Xingzhou Zhang, Mu Qiao, and Weisong Shi. Safeshareride: Edge-based attack detection in ridesharing services. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 17–29. IEEE, 2018.

[12] Peng Liu, Dale Willis, and Suman Banerjee. Paradrop: Enabling lightweight multi-tenancy at the network's extreme edge. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 1–13. IEEE, 2016.

[13] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[14] James McLurkin, Adam McMullen, Nick Robbins, Golnaz Habibi, Aaron Becker, Alvin Chou, Hao Li, Meagan John, Nnena Okeke, Joshua Rykowski, et al. A robot system design for low-cost multi-robot manipulation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 912–918. IEEE, 2014.

[15] Seyed Hossein Mortazavi, Bharath Balasubramanian, Eyal de Lara, and Shankaranarayanan Puzhavakath Narayanan. Toward session consistency for the edge. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[16] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 124–131. IEEE, 2009.

[17] NVIDIA Corporation. NVIDIA Jetson Platform, 2019. `https://www.nvidia.com/en-us/autonomous-machines/embedded-systems`.

[18] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 48. IEEE Press, 2005.

[19] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: An open-source robot operating system. In *ICRA workshop on open source software*, page 5. Kobe, Japan, 2009.

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[21] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[22] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3):38–42, 2012.

[23] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[24] Nisha Talagala, Swaminathan Sundararaman, Vinay Sridhar, Dulcardo Arteaga, Qianmei Luo, Sriram Subramanian, Sindhu Ghanta, Lior Khermosh, and Drew Roselli. ECO: Harmonizing edge and cloud with ML/DL orchestration. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[25] Nils Tijtgat, Wiebe Van Ranst, Toon Goedeme, Bruno Volckaert, and Filip De Turck. Embedded real-time object detection for a UAV warning system. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2110–2118, 2017.

[26] Teodor Tomic, Korbinian Schmid, Philipp Lutz, Andreas Domel, Michael Kassecker, Elmar Mair, Iris Lynne Grixa, Felix Ruess, Michael Suppa, and Darius Burschka. Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE robotics & automation magazine*, 19(3):46–56, 2012.

[27] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 159–173. IEEE, 2018.

[28] Yifan Wang, Shaoshan Liu, Xiaopei Wu, and Weisong Shi. CAVBench: A benchmark suite for connected and autonomous vehicles. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 30–42. IEEE, 2018.

[29] Junqing Wei, Jarrod M Snider, Junsung Kim, John M Dolan, Raj Rajkumar, and Bakhtiar Litkouhi. Towards a viable autonomous driving research platform. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 763–770. IEEE, 2013.

[30] Sean Wilson, Ruben Gameros, Michael Sheely, Matthew Lin, Kathryn Dover, Robert Gevorkyan, Matt Haberland, Andrea Bertozzi, and Spring Berman. Pheeno, a versatile swarm robotic research and education platform. *IEEE Robotics and Automation Letters*, 1(2):884–891, 2016.

[31] Zhiwei Xu, Xiaohui Peng, Lei Zhang, Dong Li, and Ninghui Sun. The ϕ-stack for smart web of things. In *Proceedings of the Workshop on Smart Internet of Things*, page 10. ACM, 2017.

[32] Daniel Zhang, Yue Ma, Chao Zheng, Yang Zhang, X Sharon Hu, and Dong Wang. Cooperative-competitive task allocation in edge computing for delay-sensitive social sensing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 243–259. IEEE, 2018.