HydraMini: An FPGA-based Affordable Research and Education Platform for Autonomous Driving

Tianze Wu*§, Yifan Wang*§, Weisong Shi†, Joshua Lu‡
*SKL of Computer Architecture, Institute of Computing Technology, CAS, Beijing, China
†Department of Computer Science, Wayne State University, Michigan, USA
‡Xilinx, Shanghai, China
§University of Chinese Academy of Sciences, Beijing, China
{wutianze, wangyifan2014}@ict.ac.cn, weisong@wayne.edu, joshual@xilinx.com

***Abstract*—Autonomous driving has been a hot topic recently, so many industrial and academic groups are putting much engineering and research efforts into this topic. However, it is difficult for most researchers or students to afford a car as a research platform to conduct experiments for autonomous driving. Further, we believe that only when more people have the chance to make contributions will this area be more prosperous. Therefore, in this paper, we present HydraMini, an affordable experimental research and education platform supporting the experiments from hardware systems to vision algorithms, and its high flexibility makes it easily extended and modified. It is equipped with the Xilinx PYNQ-Z2 board as the computing platform, which deploys the Deep Learning Processing Unit (DPU) in FPGA to accelerate the deep learning inference. It also provides useful tools like a simulator for model training and testing in a virtual environment to facilitate the use of HydraMini. Our platform will help researchers and students build and test their own solutions for autonomous driving algorithms and systems easily and efficiently.**

I. Introduction

With the development of AI technology, much work is done by robots instead of humans. Driving is one of these tasks and autonomous driving (AD) has recently become increasingly more popular these years. It is obvious that AD technology will be one of the hottest topics in AI because AD is not one single technology, but rather a highly complex system that consists of many sub-systems [1]. Today, a well-performing AD car usually has multiple sensors like GPS, a LiDAR and a camera to perceive the world around it. The huge amount of sensing data generated is transferred to the computer to do processing; then the location and some other information are calculated to help the car make decisions all by itself. AD is the area where all researchers are able to find their research interests and make contributions.

Since there will continue to be more people trying to learn about AD technology, a platform for AD researching and learning is badly needed. It is well known that in the cloud computing domain, researchers usually want to own machines that have good computation performance. The data center based on virtualization technology [2] and distributed computing [3] today usually has many powerful servers for data processing. In addition, for AD tasks which are usually run on edge side, the importance of good performance is just the same as in cloud computing. The acceleration of the inference process through hardware is becoming increasingly more common and indispensable, it is the cornerstone of the development of AD, and all researchers and students should pay enough attention to it. However, when it comes to edge computing [4], we should care about not only the computing power but also the power consumption and cost. It is a pity that we can rarely find one product that provides both enough computing power and hardware acceleration but remains affordable for most researchers.

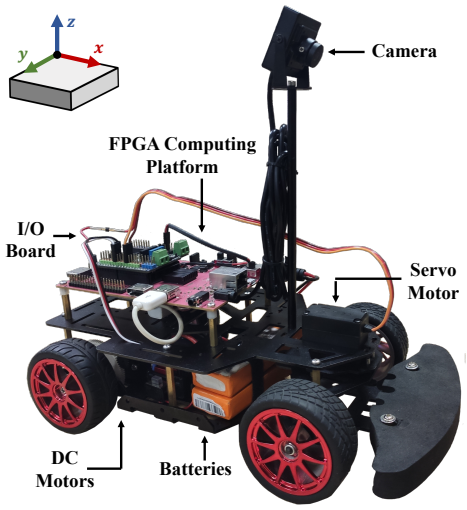


Fig. 1. Overview of HydraMini Platform.

To allow everyone to have opportunity to participate in AD research, we propose HydraMini, an affordable indoor experimental research and education platform for AD. It is concise and powerful; users could both easily learn how to use it and conduct in-depth study. The platform was first designed for the design competition of FPT 2019 [5], for which competitors needed to build an AD car that had the abilities to finish common tasks in AD areas like run along the road, avoid obstacles, understand the meanings of traffic lights and so on. After attending the competition, we found the potential of our design and decided to make the project a common platform for researchers and students. As shown in Fig. 1, HydraMini is a powerful and flexible platform from

TABLE I
PLATFORM COMPARISON.

Features	Donkey Car	F1/10	HydraOne	HydraMini
Computing Board	Raspberry Pi	NVIDIA Jetson TX1/TX2	NVIDIA Jetson TX2	Xilinx PYNQ-Z2
Compputing Architecture	CPU	CPU + GPU	CPU + GPU	CPU + FPGA
Camera	Monocular $\times 1$	Monocular $\times 1$ + Deep $\times 1$	Monocular $\times 2$	Monocular $\times 1$
LiDAR	-	2D LiDAR $\times 1$	3D LiDAR (16 Laser Units) $\times 1$	-
Resource Management Model	-	Publisher/Subscriber (ROS)	Publisher/Subscriber (ROS)	Producer/Consumer
Typical Cost	$\sim$ \$200	$\sim$ \$2,400	$\sim$ \$1,500 (w/o LiDAR)	$\sim$ \$200

hardware to software. It depends on three main components: mechanical component, control system and FPGA accelerator which are the basic of modern AD technology.

We have built several study cases based on HydraMini in which researchers are able to try an AD car in the traditional way, which uses computer vision methods to follow the road and do some pattern recognition, or they can apply an end-to-end AI model to generate control commands directly. Also we accelerated the inference process of YOLOv3 [6] and made it fast enough to work in a limited edge device. What's more, if a research project is based on ROS [7], which is widely used in robots all over the world, the migration of the project to HydraMini is easy, and with the support of functions of our platform, it will be possible to come up with more exciting ideas. In addition, we provide tools for users, one of which is a simulator using Unity3d [8] based on SdSandbox [9] that allows users to test their design more conveniently. All the resources on HydraMini are managed by PYNQ-Z2 [10], a system which is based on Ubuntu 18.04.

Use of the computing power in PYNQ-Z2 includes both Zynq7020 [11] FPGA, which is used as a specialized hardware accelerator, and an ARM core ARM Cortex-A9, which is familiar to most developers. We believe that in the future hardware acceleration will become an imperative for modern cars to provide edge computing power, so it is necessary to embed FPGA, which is flexible for researching in our platform. Users will find it easy to do research on multi-areas of AD and develop their own algorithms and systems based on HydraMini.

The remainder of this paper is organized as follows. In Section II, we discuss the related work. Section III introduces the design and implementation details of HydraMini. In Section IV we summarize three key characteristics of our platform. We provide several case studies to show the large potential of HydraMini in Section V. Finally, we conclude our work in Section VI.

II. RELATED WORK

In this section, we summarize the related work and several other similar platforms. Compared to other products, our platform costs less and provides FPGA support; It is also easy for users to customize their own components based on it. The basis of our tool kit is simple and affordable, but it has great potential. **Table 1** shows the comparison between these platforms.

Research Platform for Autonomous Devices. Wang et al. presented HydraOne [12], which is an indoor robot-based platform that has sufficient resources and components to conduct related experiments. It has three key characteristics: design modularization, resource extensibility and openness, as well as function isolation, which allows users to conduct various research and educational experiments. Wei et al. presented the CMU autonomous driving research platform, which is based on a Cadillac SRX [13]. This work focuses on vehicle engineering problems, including the actuation, power, and sensor systems on the vehicle. OKelly et al present F1/10 [14]: an open-source, affordable, and high-performance 1/10 scale autonomous vehicle testbed. The F1/10 testbed carries a full suite of sensors, perception, planning, control, and networking software stacks that are similar to full scale solutions.

Hardware Acceleration Technology used in AI. GPU [15] is now widely used by researchers to accelerate the training and inference process of AI. The training library is cuDNN [16] while the inference library is TensorRT [17]. cuDNN is a GPU-accelerated library of primitives for deep neural networks. It provides highly tuned implementations of routines arising frequently in DNN applications. TensorRT makes a pre-trained neural network run quickly and efficiently on a GPU. The Tensor Processing Unit (TPU) [18] was announced in May 2016 at Google I/O when the company said that the TPU had already been used inside their data centers for over a year. The chip has been specifically designed for Google's TensorFlow framework, a symbolic math library which is used for machine learning applications such as neural networks. The Xilinx Deep Learning Processor Unit (DPU) [19] is a programmable engine optimized for convolutional neural networks. The unit includes a high performance scheduler module, a hybrid computing array module, an instruction fetch unit module, and a global memory pool module. The DPU uses a specialized instruction set, which allows for the efficient implementation of many convolutional neural network. With our platform, users are able to easily try DPU or FPGA to help their algorithms run better.

III. DESIGN AND IMPLEMENTATION

In this section, we introduce the design and implementation details of HydraMini. The structure of the system is shown in Fig. 2. Based on the middleware, users are able to add more kinds of sensors or create more workers to handle the data while the middleware provides APIs for users to take

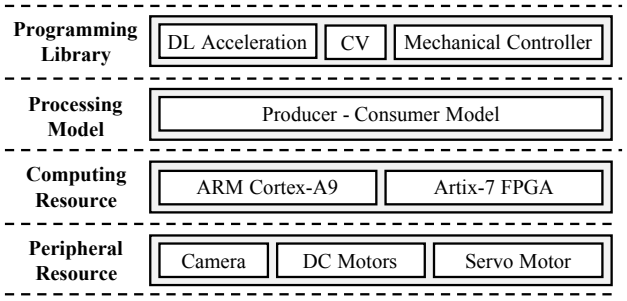


Fig. 2. HydraMini Framework Overview.

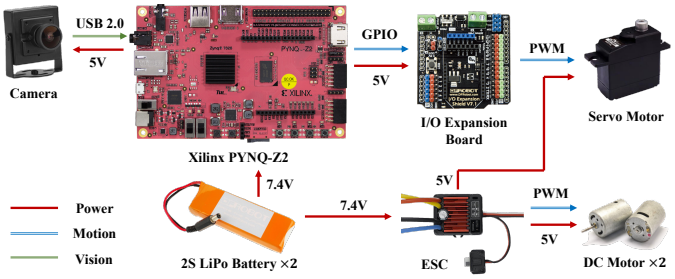


Fig. 3. HydraMini Hardware Design.

advantage of computing resource and control the car. It is easy for users to add or remove component to or from the system.

A. Hardware Design

As shown in Fig. 3, HydraMini is equipped with a Xilinx PYNQ-Z2 board which acts as the main controller; PYNQ [10] is an open-source project from Xilinx that makes it easy to design embedded systems with Xilinx Zynq Systems on Chips (SoCs). Users create high performance embedded applications with parallel hardware execution, high frame-rate video processing, hardware accelerated algorithms, real-time signal processing, high bandwidth I/O and low latency control. Software developers are able to take advantage of the capabilities of Zynq and programmable hardware without having to use ASIC-style design tools to design hardware. System architects have an easy software interface and framework for rapid prototyping and development of their Zynq design. It is suitable for use in our platform because of the convenience and high performance. The board collects the data from multiple sensors and feeds the data to several computing tasks in real-time. An I/O expansion board receives the control message output from the computing tasks and then sends the control signals to the motor drivers to control the movement of HydraMini. The entire HydraMini platform is powered by two batteries, and to provide steady voltage for a DC motor and servo motor, a brushed electronic speed controller is used. The basic sensor is a monocular camera ($1280 \times 720@30$); we also provide a case study in Section V using a LeiShen LS01D LiDAR [20]. It is easy for users to add more sensors to the platform.

Programmable Logic (PL) The programmable logic in PYNQ-Z2 is equivalent to Artix-7 FPGA [21]. The following components are embedded in it:

- 13,300 logic slices, each with 4 6-input look-up tables (LUTs) and 8 flip-flops (FF);
- 630 KB of fast block RAM;
- 4 clock management tiles, each with a phase locked loop (PLL) and mixed-mode clock manager (MMCM);
- 220 DSP slices;
- On-chip analog-to-digital converter (XADC).

Processing System (PS). The Cortex-A9 [22] processor is embedded in PYNQ-Z2. It is a performance and power optimized multi-core processor that features a dual-issue, partially out-of-order pipeline and a flexible system architecture with configurable caches and system coherency using an ACP port. The Cortex-A9 processor achieves a better than 50% performance over the Cortex-A8 processor in a single-core configuration. It has an L1 cache subsystem that provides full virtual memory capabilities. The Cortex-A9 processor implements the ARMv7-A architecture and runs 32-bit ARM instructions, 16-bit and 32-bit Thumb instructions, and 8-bit Java bytecodes in the Jazelle state.

B. Software Design

Framework Overview. The operating system on PYNQ-Z2 is based on Ubuntu 18.04, so libraries are easily installed. The system on PYNQ-Z2 also provides many jupyter notebook documents about how to fully utilize the hardware resources in PYNQ-Z2. To make the control process more efficient and easier to extend, we implement a producer-consumer model [23] which is a classic design pattern in a multi-process synchronization environment. The whole control system depends on three main components: mechanical controller, AI model inference and computer vision analysis.

Producer-Consumer model. Many indoor AD driving platforms like HydraOne [12] and F1/10 [14] use ROS [7] to manage the hardware and software resources today because ROS provides the car with an easy way to communicate with the server and many existing applications. However, to balance the platform's cost and performance, we focused mainly on core functions in the AD, so the communication between the car and server is not so important. Using ROS brings unnecessary overhead to CPU. Thus, a more streamlined and efficient method is used as the base of the control system. Refer to Fig. 4 for an overview of this model.

The sensors play the role of a producer while the decision makers like the AI model and computer vision methods are the consumers. Each kind of data is stored in a queue in the memory. Different producer processes add the data they get to the specified queue, and the data is handled by consumers who cares about this kind of data. The length of each queue is decided by users according to actual demand, so if the producer find the queue is full, it replaces the last element with the latest data.

It is easy to add more producers or consumers and new kinds of data. If one process wants to handle or provide different

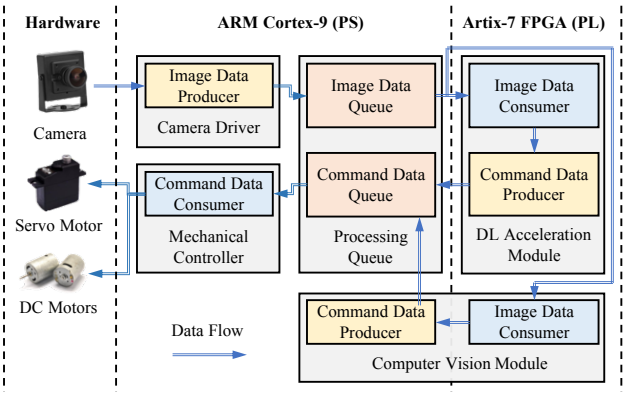


Fig. 4. Producer-Consumer Model.

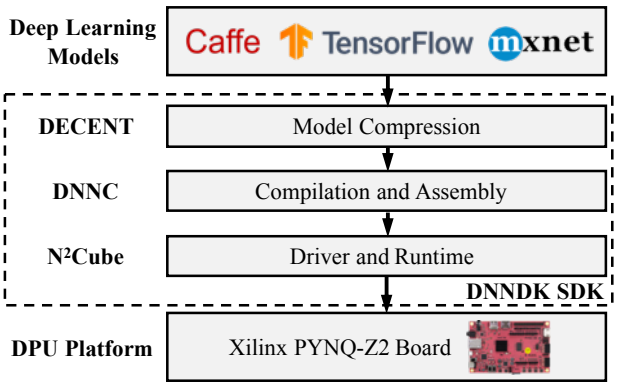


Fig. 5. DNNDK Framework.

kinds of data, it only needs to read or write the related queue. The synchronism of the system is maintained by locks, so each queue will have a lock.

The consumers who usually act as controllers have a shared clock. They will check if their commands to send are outdated according to the clock, and this clock ensures that the car won't receive outdated commands. Also, there exists a token which indicates who is in control at this moment. Users are able to define their own strategy for the transformation of property.

Mechanical Controller. The HydraMini platform has one motor for providing power and one servo motor for direction control. This design has been widely used in real cars. We provide basic control APIs for users. The rotation speed of the motor and the angle of deflection of the servo motor are set directly. Also, higher level methods like 'Speed Up' are provided.

In addition to driving on its own, the car is controlled by using a keyboard. We invoke OpenCV [24] library to read the keyboard signals and then call the mechanical API. Users are easily able to define their own button layout. This ability is mostly used to generate training data.

AI Inference. AI technology is an important part of AD as it handles many tasks like object identification, lane keeping and so on. In our platform the AI inference process is packed as a consumer thread; it reads data from the produced data queue and uses it as the input of the AI network. Then the model produces control commands directly or just provides information for the controller thread to make decisions. The training process of the model is not run in HydraMini but in the server, so users should provide HydraMini with pre-trained model.

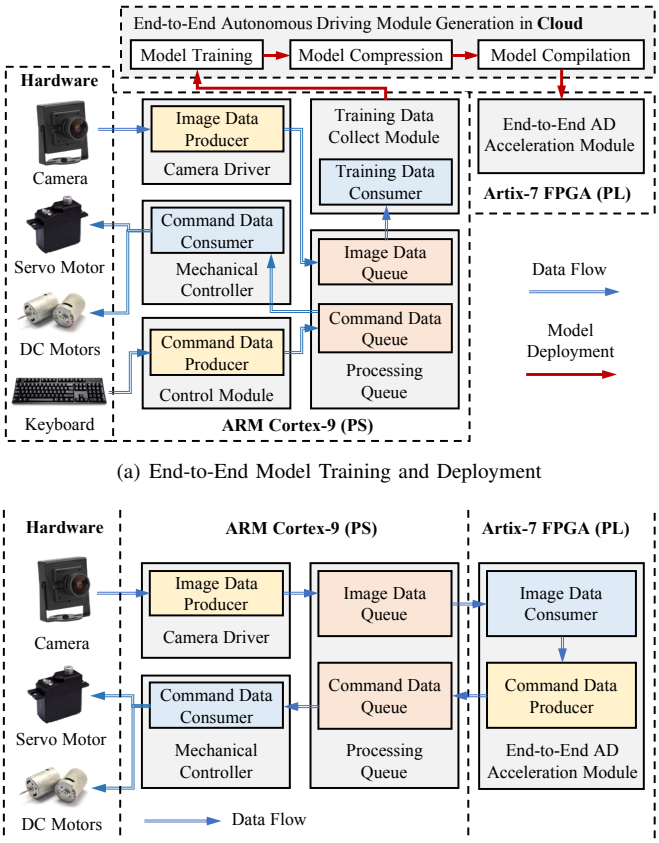
Furthermore, with the power of DPU [19], which is one accelerator in FPGA, the process of AI reference will be accelerated. The AI inference thread is copied, and all the threads run concurrently in DPU, which means higher inference performance.

To make good use of DPU and to make the optimization process easy, we provide scripts to do a complete set of optimized tool chains provided by DNNDK, including compression, compilation and runtime. Refer to Fig. 5 to see the framework of DNNDK. First, the Deep Compression Tool DECENT [19], employs coarse-grained pruning, trained quantization and weight sharing to make the inference process in edge meet the low latency and high throughput requirement with very small accuracy degradation. Second the DNNC (Deep Neural Network Compiler) [19], which is the dedicated proprietary compiler designed for the DPU, will map the neural network algorithm to the DPU instructions to achieve maxim utilization of DPU resources by balancing the computing workload and memory access. Third, the users use the Cube of Neutral Networks (N2Cube) [19], the DPU runtime engine to load DNNDK applications and manage resource allocation and DPU scheduling. Its core components include a DPU driver, DPU loader, tracer, and programming APIs for application development.

After using DPU, the performance of the end-to-end model described in the case study is up to $7,000$ FPS, which is fast enough to satisfy the requirement of low-latency in AD. We also tested the YOLOv3 [6] model in DPU, and it achieves 3 fps when detecting 80 categories of things.

Computer Vision Analysis. OpenCV [24] is a widely used library for computer vision analysis. Due to the support for OpenCV in PYNQ-Z2, existing computer vision algorithms are easily invoked, and their own algorithms can be implemented. We have a case study in Section V which shows how to use traditional computer vision methods to manage AD tasks. The control process of these methods is just the same as that of an AI model; the thread running computer vision algorithms will read data from the producers and output commands or information. More threads can be created to increase throughput.

However, these computer vision algorithms may be very time consuming running in ARM Cortex-A9 in Xilinx PYNQ-Z2. To reduce computation complexity, we do several pre-processes like cropping and down-sampling. Time consuming tasks such as Gaussian filter, Canny edge detection, and Hough transform can be moved to FPGA using Xilinx xfopencv library [25]; the BP neural network can be implemented in FPGA using Xilinx Vivado HLS. However, when implementing accelerators in FPGA, users should take the board's



(a) End-to-End Model Training and Deployment

(b) End-to-End Model Inference Data Flow

Fig. 6. Autonomous Driving using End-to-End Model.

resource into consideration and choose the heaviest computational task to implement while not going beyond the resource limit. If there remains enough PL space for users' algorithms, they can be put on the PL side, or just on the PS side.

IV. Key Characteristics

The HydraMini platform has three key characteristics: high flexibility and extensibility, full stack, and easy of use. These three characteristics help users understand the AD technology stack and the platform.

High Flexibility and Extensibility. The idea of our system design is inspired by ROS, in which new functions are easily added by adding ROS nodes. Nodes get and send messages easily through the publish-subscribe mechanism. We have made the ROS system more lightweight and forthright. The thread is similar to a node while the producerconsumer model is similar to the publish-subscribe mechanism. Thus it is easy to add more hardware devices as sensors or threads as handlers. What's more, due to the fact that the base of our system is Ubuntu18.04, it is easy to redefine the whole software framework as needed; an example of using ROS is included in the case studies in Section V.

Full Stack. The full stack here means our product provides almost everything you need to learn about AD technologies from algorithms to hardware. It is important for researchers or students to understand how the car runs and why sophisticated algorithms are able to run in resource limited edge platform. Users could use different physical constructions, different operating systems, different software frameworks, different algorithms and different hardware accelerations, etc. That means researchers are able to do whatever they want based on our platform, and they could use existing modules and understand the operation mode of the whole system. Considering the difficulty users may have in collecting training data and testing AI models in the real world, we provide a simulator modified from SdSandbox [9] which is first used in Donkey Car [26]. The simulator is used to do tests or more. Its usage will be introduced in case studies.

Ease of Use. One of our most important goals is to make our platform easy to use. To achieve this, most libraries of the platform are familiar to users like OpenCV and TensorFlow [27], as they are open-sourced and widely used. Most importantly, users only need to know Linux, C++, Python, AI and a little about DPU usage. The software framework is simple and tidy; we only keep necessary functions and make it extendable. The hardware is simple, too, so if a user doesn't want to add more devices, one camera and two motors are all that is needed. Users do not even have to prepare a real car to do experiments, so hardware is not a concern. Furthermore, we provide many documents for users, so they can easily modify any part of the platform.

V. Case Studies

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

A. Autonomous driving using end-to-end model

This case study shows our platform can be used for AD with an end-to-end model. Several recent studies have replaced the classic chain of perception, planning, and control with a neural network that directly maps sensor input to control output [28]–[30], a methodology known as end-to-end driving. New approaches based on reinforcement learning are being actively developed [31]. An end-to-end AI model is easy to use since it outputs control commands directly. The model contains mainly CNN and Dense layers [32], and it maps

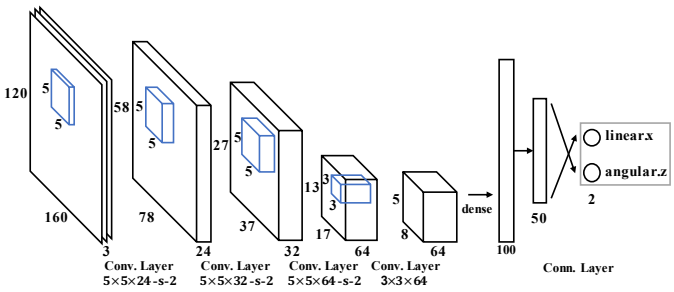


Fig. 7. Model Structure.

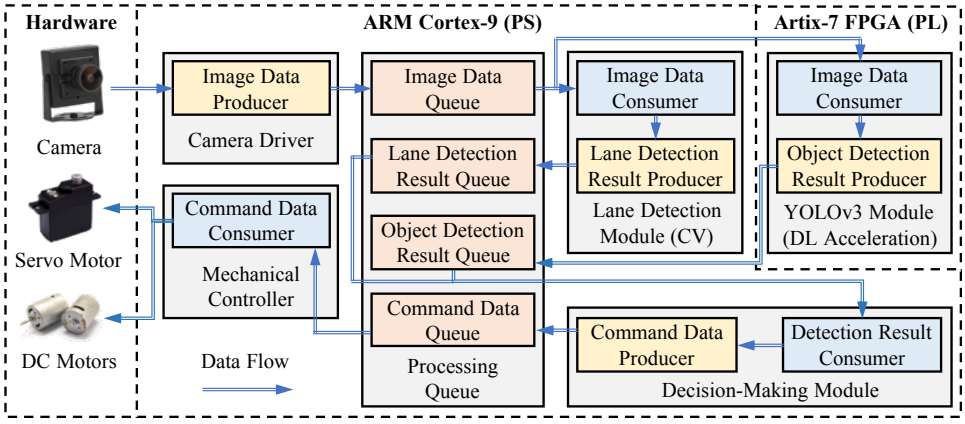


Fig. 8. Autonomous Driving using Traditional Methods.

camera input to control output. The structure of the model is shown in Fig. 7.

CNN layers extract features from the images taken by the car's front camera. Several fully connected layers follow the CNN layers; they finally extract the command information needed for auto-driving. The activation function we use is Relu. The last layer is a Softmax layer [32] for classification or a Dense layer for regression. Although the current model is not perfect, it is convenient to make changes and do optimizations to the model. Fig. 6 shows the whole process.

First, the user controls the car using the keyboard or whatever they like and saves the data from the car's sensors as training data. Alternatively, the user can just get the data from the Internet. Second, after pre-processing the images gotten as input are put into the AI model, and the labels such as keyboard signals are the output. The model is trained using TensorFlow until a satisfactory model is achieved. Third, DNNDK [19] is used provided by Xilinx, to do the compression and compilation, and then the copy generates files to the car. Finally, the car is able to move by itself. It will be controlled by AI models.

B. Autonomous driving using traditional methods

In the above case study we used an end-to-end model, but this case shows another way of doing AD. This time computer vision methods are used to detect the road line and a classic YOLOv3 model is used to find objects.

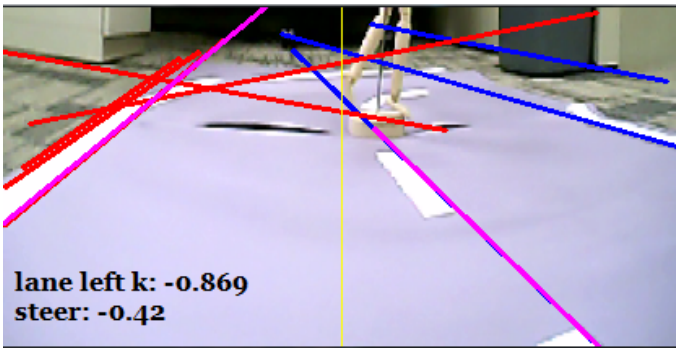


Fig. 9. Lane Detection.

To detect road lane, we first use a 5×5 Gaussian filter to remove the noise of the image. Then Canny edge detection [33] is applied to detect the edges of the image. Hough transform [34] is employed to detect the lines of the image. Based on the orientation and relative position of the detected lines, the side line and stop line of the road are identified. To improve the robustness and accuracy of the algorithm, more techniques have been added to the algorithm, such as K-Means clustering [35] for Hough lines, Kalman/Gabor filtering for sampled data, alternative IPM (inverse perspective mapping) lane detection method [36].

The car runs following the lane of the road, Fig. 9 shows the final lanes extracted from all the detected lines. The yellow line marks the middle of the picture, and the other lines are all lines we find in the photos. Among all the lanes we detected, we choose one line for both the left side and right side, and they are painted purple. The red ones and blue ones are the remainder. With this information, the car adjusts its direction and speed to the road. For example, in the picture the left lane chosen as the road line has a slope of -0.869 which is smaller than the slope datum, and then the controller calculated a steer value -0.42, which means to turn left.

YOLOv3 is used to find objects like people or cars to help the car make decisions like braking to avoid an obstacle. The inference process will be accelerated by DNNDK to meet more stringent real-time requirements; also, the full YOLOv3 model can be replaced with a tiny YOLOv3 model to achieve faster inference speed.

Fig. 8 shows the control process of this case; the taken images will be processed by both computer vision threads and YOLOv3 threads, but this time they won't produce commands directly; instead, the information they get will be used to make decisions. Finally, the commands will be generated by the decision maker.

C. Simulator

The simulator is a tool which helps users test their designs more efficiently. One of the interfaces of the simulator is

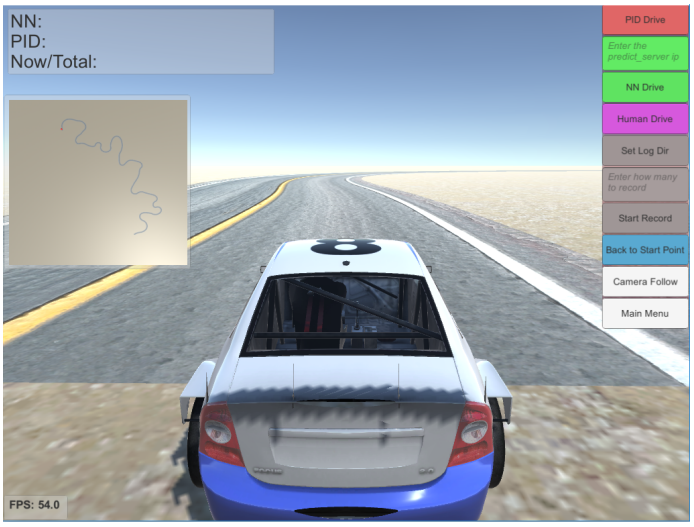


Fig. 10. Simulator Interface.

shown in Fig. 10. The simulator is based on SdSandbox [9], which is a simulator project for Donkey Car [26]. As shown in the picture, this simulator is used to collect training data and test models.

When testing, a server should be built to receive the sensor data from the client of the simulator and generate control messages according to this data. We have already built one example; the server will get images taken by the simulator and handle them using the AI model trained before; then the model outputs control commands that will be sent to the simulator. It is also convenient for users to modify the source code of the simulator to define their own data format or control methods. However, users should be familiar with C# and Unity3d if they would like to do so, and we provide a coding tutorial manual to help.

D. LiDAR in ROS

Due to the high impact of ROS, we make our platform able to support ROS projects by installing ROS in Pynq directly. In this case, we show that it is convenient to build ROS projects based on the hardware of HydraMini. Developers who don't want to use our software product can use ROS instead. The version we use is ROS Melodic, which is mainly used in Ubuntu18.04.

This time we show how to read LiDAR data and control the car using ROS. With LiDAR data, the car is able to handle obstacle avoidance tasks and SLAM tasks whose basical technology is LiDAR data processing. The LiDAR we use is LeiShen LS01D [20], which has already provided one ROS node for users to gain and publish LiDAR data. We read the laser point cloud data by subscribing the published topic. Also, it will be easier to visualize it if RViz [37] is used; Fig. 11 is one example.

The control node of the car is a transplantation of the code from the existing controller. It is easy to send control commands by publishing the commands to the corresponding

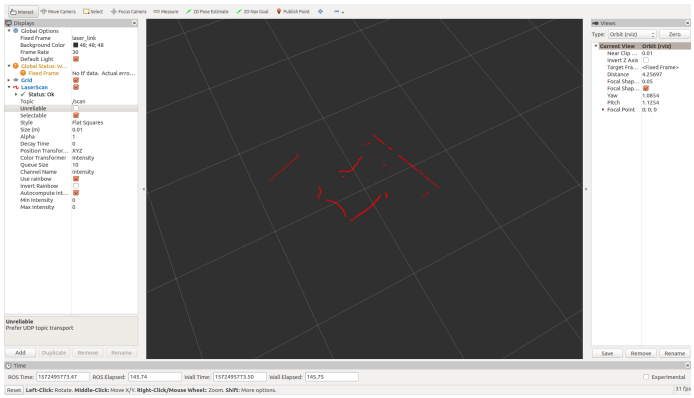


Fig. 11. LiDAR Data.

topic. Thus, it is easy for users to build their projects based on ROS in our platform. Since ROS is widely used, it is important that our platform helps users who want to try ROS.

VI. Conclusion

In this paper, we present the design, implementation, characteristics and case studies of HydraMini, an affordable experimental research and education platform for autonomous driving. The platform is highly flexible and extensible, full stack, and easy to use. These three characteristics help users understand AD technology well and take full advantage of HydraMini. Students and researchers could use our platform for full-stack research on algorithms, applications, system, mechanical control and hardware acceleration. We hope this platform helps users enjoy their research and learning process without incurring high costs.

References

[1] S. Liu, L. Li, J. Tang, S. Wu, and J.-L. Gaudiot, "Creating autonomous vehicle systems," *Synthesis Lectures on Computer Science*, vol. 6, no. 1, pp. i–186, 2017.

[2] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE, 2009, pp. 124–131.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[5] T. Wu, W. Liu, and Y. Jin, "An end-to-end solution to autonomous driving based on Xilinx FPGA," EasyChair, Tech. Rep., 2019.

[6] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[8] Unity Technologies, "Unity real-time 3d development platform." 2020, https://unity.com/learn.

[9] T. Kramer, "Sandbox simulator for training a self-driving car." 2020, https://github.com/tawnkramer/sdsandbox.

[10] TUL Corporation, "TUL PYNQ-Z2 Board Based on Xilinx Zynq SoC." 2020, http://www.tul.com.tw/ProductsPYNQ-Z2.html.

[11] Xilinx Inc., "Zynq-7000 SoC product." 2020, https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html.

[12] Y. Wang, L. Liu, X. Zhang, and W. Shi, "HydraOne: An indoor experimental research and education platform for CAVs," in *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[13] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 763–770.

[14] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio *et al.*, "F1/10: An open-source autonomous cyber-physical platform," *arXiv preprint arXiv:1901.08567*, 2019.

[15] E. Nurvitadhi, D. Sheffield, J. Sim, A. Mishra, G. Venkatesh, and D. Marr, "Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC," in *2016 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2016, pp. 77–84.

[16] NVIDIA Corporation, "cuDNN: NVIDIA CUDA Deep Neural Network Library," 2020, https://developer.nvidia.com/cudnn.

[17] ——, "NVIDIA TensorRT: Programmable inference accelerator." 2020, https://developer.nvidia.com/tensorrt.

[18] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 1–12.

[19] Xilinx Inc., "DNNDK: Deep Neural Network Development Kit." 2019, https://www.xilinx.com/products/design-tools/ai-inference/edge-ai-platform.htmldnndk.

[20] LeiShen Intelligent System Co., Ltd., "LeiShen LiDAR LS01D ." 2020, http://en.leishen-lidar.com/product/leida/LS01/index.html.

[21] Xilinx Inc., "Xilinx Artix-7 FPGA." 2020, https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html.

[22] Arm Limited (or its affiliates), "ARM Cortex-A9." 2020, https://developer.arm.com/ip-products/processors/cortex-a/cortex-a9.

[23] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating systems: Three easy pieces*. Arpaci-Dusseau Books LLC, 2018.

[24] OpenCV Team, "OpenCV: Open source computer vision library." 2020, https://opencv.org/.

[25] Xilinx Inc., "Xilinx xfOpenCV Library." 2020, https://github.com/Xilinx/xfopencv.

[26] Donkey Car Team, "Donkey Car." 2020, https://www.donkeycar.com.

[27] Google, "Tensorflow: An end-to-end open source machine learning platform." 2020, https://www.tensorflow.org.

[28] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[29] L. Chi and Y. Mu, "Deep steering: Learning end-to-end driving model from spatial and temporal visual cues," *arXiv preprint arXiv:1708.03798*, 2017.

[30] H. M. Eraqi, M. N. Moustafa, and J. Honer, "End-to-end deep learning for steering autonomous vehicles considering temporal dependencies," *arXiv preprint arXiv:1710.03804*, 2017.

[31] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8248–8254.

[32] Keras Team, "Keras: The python deep learning library." 2019, https://keras.io.

[33] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[34] J. B. Burns, A. R. Hanson, and E. M. Riseman, "Extracting straight lines," *IEEE transactions on pattern analysis and machine intelligence*, no. 4, pp. 425–455, 1986.

[35] S. Z. Selim and M. A. Ismail, "K-means-type algorithms: A generalized convergence theorem and characterization of local optimality," *IEEE Transactions on pattern analysis and machine intelligence*, no. 1, pp. 81–87, 1984.

[36] J. Wang, T. Mei, B. Kong, and H. Wei, "An approach of lane detection based on inverse perspective mapping," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 35–38.

[37] Open Robotics, "RViz: the 3D visualization tool for ROS." 2020, http://wiki.ros.org/rviz.