Volume 2, Issue 1, February 02, 2017

TC-CCPS Newsletter

IEEE

TC-CPS

Cyber-Physical Systems

Human-centric Graphic Rendering Optimization for Virtual Reality

Xiang Chen, George Mason University

1 Challenges for the Virtual Reality

While smartphones have been the driving force behind the electronics industry for almost a decade, The growth rate of smartphone shipments is anticipated to drop below even that of the personal computer (PC) within the next few years, implying market saturation. Instead, Virtual Reality (VR), especially those on mobile platforms (e.g., Gear VR etc.), are now emerging as the major new markets in electronics. The allure of these devices is clear: VR devices allow users to enjoy immersive experiences in entirely virtual worlds. Some VR products have already appeared on the market, such as Facebook Oculus, HTC Vive, etc.

As VR technology is extremely promising, it currently faces two main challenges. First, there exists an increasing gap between the graphic and data processing capability required by these platforms and the limited computing resources available on mobile hardware. As shown in Figure 1, the volume of graphic rendering data which must be processed for an optimum VR experience is at least 7× larger than that of traditional platforms. Second, VR systems often involve significant human-machine interactions, and the associated data-intensive algorithms such as computer vision, deep learning, and etc.

As VR is still in its infancy, the field has yet to effectively overcome the above two challenges. Current devices either sacrifice visual perception quality for a reduction in computation load, e.g., in mobile systems such as Gear VR on smartphones; or solicit external computation support at the cost of system mobility, e.g., VR headsets which must be tethered to a desktop for GPU support. Further restricting existing devices are the thermal and battery capacity limitations of modern mobile systems, which have emerged as a major concern for extended VR experiences [1]. Identifying solutions which can allow VR to overcome these challenges, enabling truly mobile, immersive experiences, has become the common commitment of the entire research community.

	PC Gaming	**VR**
Frame Rate	30Hz	**>90Hz**
Delay Tolerance	<100ms	**<25ms**
Resolution	1920x1080	**3024x1680**
Frames to Render	1 Screen	**2 for Each Eye**
Pixels to Render	60MP/s	**450MP/s**

Figure 1: Computation load comparison PC vs. VR.

2 Human-centric Graphic Rendering Optimization

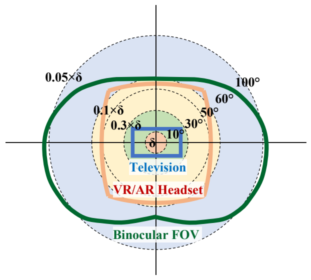
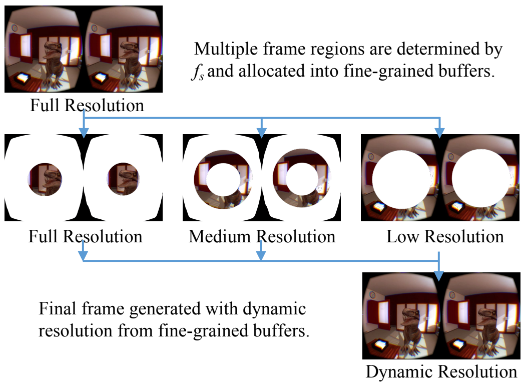


Figure 2: Field of Vision (FOV).

Increasing resolution of a display increases the level of detail it can present, but also increases GPU computational load and power consumption. However, the level of detail observable by the Human Visual Perception (HVP) is not static, and changes based on a number of factors [2]. Therefore, if observable detail decreases below the resolution of the display during use of VR, rendering resolution may be reduced to match in order to minimize graphics rendering workload. We propose a spatial human-centric graphic rendering load scaling technique attacking the major challenges mentioned above. As part of the first challenge, we observe that level of observable detail is affected by an object's location in the HVP Field of Vision (FOV), and that compared to traditional displays, the near-eye display in VR system subtends a much larger portion of the HVP FOV, as shown in Figure 2. This presents an opportunity for scaling GPU workload based on HVP-quality unique to VR.

To address the second challenge and directly relate FOV position to graphics rendering workload, we can approach the problem from the basis of minimum resolvable acuity (MRA) [3], which defines the smallest individual feature in an image that can be optically isolated. The minimum observable feature size defined by MRA, f_s,can be determined as a function of the receptor cell density in the specific region of the retina the feature is projected onto, δ, and the distance of the object from the eye, D:

$$f_s = 2D \times tan(\frac{\delta}{2}) \quad (1)$$



Full Resolution
Multiple frame regions are determined by f_s and allocated into fine-grained buffers.
Full Resolution
Medium Resolution
Low Resolution
Final frame generated with dynamic resolution from fine-grained buffers.
Dynamic Resolution

Figure 3: Dynamic resolution rendering.

While most measurements of MRA focus solely on f_s for features in the center of FOV, Figure 2 illustrates that the value of δ varies greatly across the major regions of human FOV covered by modern VR systems [4]. From Equation (1), user-observable level of detail in these areas is greatly reduced compared to the center of the FOV. Therefore, in order to reduce GPU rendering load, the rendering resolution in these areas may be reduced to match the user-observable level of detail without impacting HVP-quality. The final challenge culminates with the proposed spatial, human-centric graphic load scaling technique, which is based on the changes in δ across the FOV, and with which we cast fine-grained resolution scaling on the near-eye display to match the calculated f_s in that region. This is accomplished via the technique shown in Figure 3, where the GPU allocates multiple pixel buffers to a scene, each at a resolution corresponding to a calculated f_s in a specific region of the FOV. When rendering a frame, the GPU renders to each buffer only the part of the scene required at that resolution. Once this is completed, the buffers may then be combined back into a single frame and forwarded to the display. Hence, GPU rendering workload is reduced without impacting HVP-quality.

In Figure 4, the average power consumption for a GPU is shown when rendering a scene at three different display resolutions, full, medium, and low. Each of these resolutions is selected as it corresponds to the δ value in the regions previously specified in Figure 2. As can be seen, as resolution decreases, so does power consumption. However, if the lower resolutions were utilized for the entire display, the user would surely notice. The final bar in the figure shows power consumption when rendering with our fine-grained resolution scaling technique, in addition to the relative portion of the frame covered by each resolution level and contribution of each to the overall rendering power envelope. It can be seen that our method still utilizes far less power than full resolution, but would go unnoticed by the user. These results indicate that GPU rendering workload, and therefore computational requirements and power consumption, can be greatly improved by adjusting it based on a deeper understanding of the HVP.

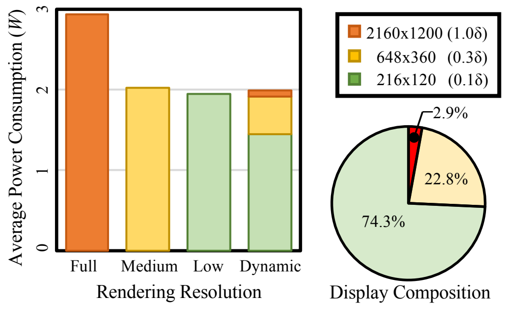


Figure 4: Spatial resolution scaling performance

References

[1] C. Rinaldi, "How Long Can You Use a Device with the Gear VR Before It Overheats?" *https://www.androidpit.com/*

[2] S. Anstis, "Picturing Peripheral Acuity," *Perception*, Vol. 27, No. 7, pp. 817-825, 1998.

[3] K. Nixon, X. Chen, and Y. Chen, "Scope - Quality Retaining Display Rendering Workload Scaling based on User-Smartphone Distance," in *Proceedings of the International Conference on Computer Aided Design* (ICCAD), 1:1–1:6, 2016.

[4] J. Sardegna, S. Shelly, A. R. Rutzen, S. M. Scott, "The Encyclopedia of Blindness and Vision Impairment," *Infobase Publishing*, pp. 253, 2014.

Cloud Resource Scheduling, Monitoring, and Configuration Management in the Software Defined Networking Era

Khaled Alwasel[1], Ayman Noor[1], Yinhao Li[1], Ellis Solaiman[1], Saurabh Kumar Garg[2], Prem Prakash Jayaraman[3], Rajiv Ranjan[1]
[1]School of Computing Science, Newcastle University, United Kingdom
[2]Information and Communication Technology, University of Tasmania
[3]Faculty of Science, Technology and Engineering, Swinburne University of Technology

In recent years, Cloud Computing has emerged as a major technology for delivering on-demand IT (Information Technology) services to consumers across the globe [1]. It provides IT resources based on a pay-as-you model and offers a rich set of services and tools. The Cloud Computing stack consists of three layers – Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a service (SaaS). IaaS offers hardware resources (computation, network, and storage) as virtualized cloud services. PaaS provides software services (appliances) and programming tools, whereas SaaS provides ready to use application stacks.

In order to ensure bespoke application performance (e.g., minimize response time, maximize throughput); each layer of the Cloud stack requires optimum resource Scheduling, Monitoring, and Configuration-management (SMC). Numerous studies have addressed SMC by proposing various frameworks, models, and algorithms. Most of these studies however have assumed *traditional, non-programmable* Cloud DataCentre (CDC) networks. In this article, we discuss how resource scheduling, monitoring, and configuration-management becomes a challenging task when considering Software Defined Networking (SDN), and Network Function Virtualisation (NFV) performance parameters as part of overall cloud application scheduling process.

The recent technology revolution of so-called **Software Defined Networking (SDN)** [2] has given full control and programmability of CDC network. **Network Function Virtualization (NFV)**, on other hand, is a recent networking concept often presented with SDN, delivering network services using software processes hosted on CDC-based virtual machines (VMs), instead of proprietary dedicated hardware.

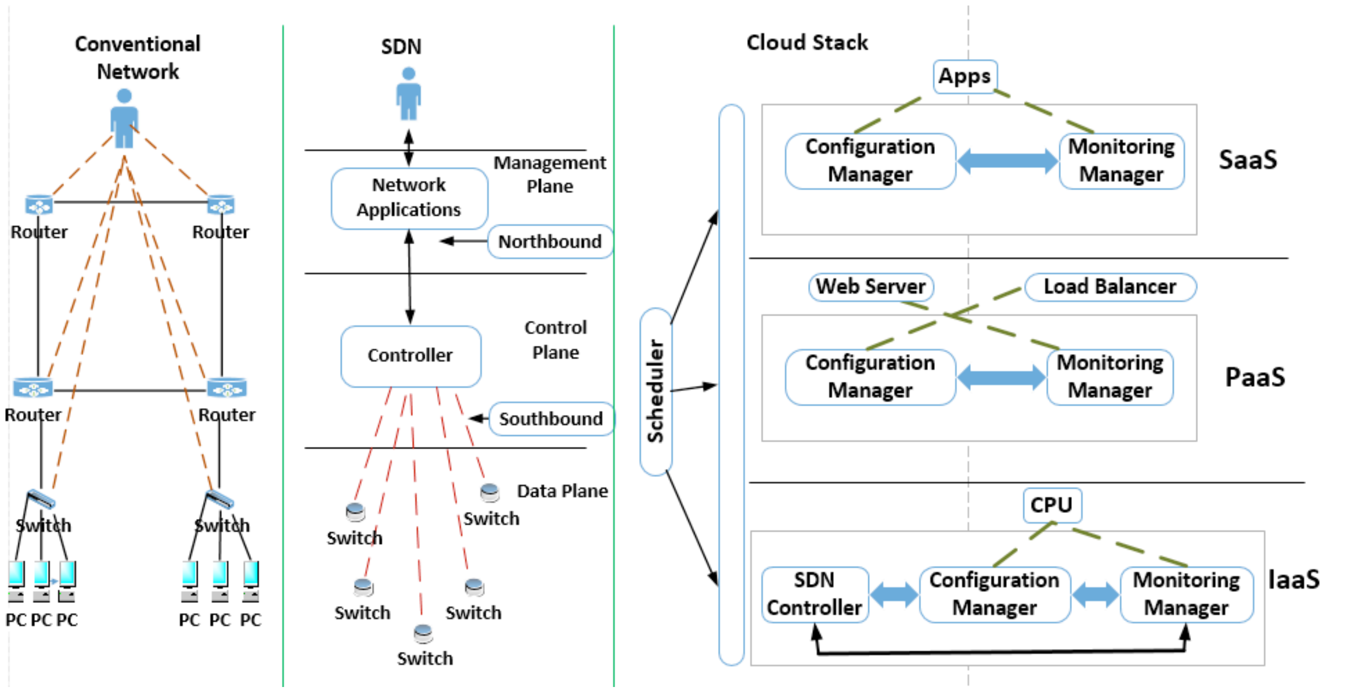


Figure 1: Conventional Network vs SDN network along with Cloud Scheduler, Monitor, Configuring Manager Architecture.

As shown in Figure 1, SDN consists of three layers: The *Management plane*, the *Control plane*, and the *Data-plane*. The Control plane (decision maker) is abstracted away from networking devices (e.g., switches and routers) to a central logic controller, providing a global-network view, programmability, and finer-grained control. It communicates with the data and management planes via two interfaces: southbound and northbound, respectively. The Data plane receives traffic policies from the control plane (e.g., via OpenFlow protocol), and forwards traffic accordingly. The Management plane consists of software services (e.g., via APIs) used by administrators to control and configure network devices.

1 Scheduling Challenges

Scheduling (a well-known NP-complete problem) is an essential mechanism for cloud computing aiming at managing application performance driven diverse goals including minimizing response times, maximizing resource utilization, and maximizing throughput [3, 4, 5]. It plays an important role at each layer of the Cloud stack where operations are mapped appropriately to intended components based on given Quality of Service (QoS) parameters and scenarios. Numerous studies have contributed to cloud scheduling in terms of workflows and VM, but none of them consider SDN based scheduling along with the application and VM (mult-level scheduling). SDN can provide the cloud scheduler with critical factors (e.g., link availability, link utilization, and link failure at run time), resulting in smarter scheduling decisions. However, integrating inputs from the three layers is a non-trivial task making scheduling decisions a difficult task.

The key challenging issues in SDN based scheduling is that there are too many parameters from each of three layers that need to be considered and integrated. The application parameters (first layer) include response time, processing time, rate of failure, throughput, and priorities. The VMs' parameters (second layer) include total number of VMs, total number of idle VMs, total number of VM reservation requests, and cost. For each VM, the scheduler also need to retrieve CPU capacity and utilization, memory capacity and utilization, and storage capacity and utilization. The SDN's parameters (third layer) include topology (re)configuration based on changes in real time, routing mechanisms based on packets/flows in real time, and link capacity, link failure, link reservation, and bandwidth utilization. The scheduler needs to retrieve simultaneously all the parameters from the three layers of the cloud stack at a particular time instance. The second issue is how the design of decision making model that can integrate these parameters each of which might be giving a conflicting picture. Clearly, designing a scheduler while addressing these issues is undoubtedly a daunting task, requiring further investigation of new schemes, models, and algorithms.

Scheduling related research can be categorized into three areas based on above mentioned the problem: application, VM, and SDN scheduling. To the best of our knowledge, no attempt has been made for designing a scheduler that is able to operate according the parameters generated from the proposed architecture. Nevertheless, there are research work addressing the problem considering the application layer or/and VM layer parameters. For example, Sukhpal et al. [3, 4] proposed techniques to provision resources based on QoS requirements (e.g., CPU utilization) resulting in less execution time and cost. Warneke et al. [5] presented a Nephele framework with the principle of job scheduling and task execution where a job manager divides a given job's process into a number of tasks and then assigns VM(s) accordingly. Still, research on designing a multi-level scheduler that is Application-VM-SDN-Agnostic has not been solved yet.

2 Monitoring challenges

Monitoring can be defined as the process of observing or checking the quality of something over a period of time. Similarly, monitoring in Cloud computing is the process of auditing and managing the operational workflow, and the different processes within a cloud-based IT asset or infrastructure. Cloud monitoring is an important part of cloud security and management [14, 15], and it can be implemented within Cloud infrastructure through automated software providing central access and control over the cloud infrastructure. There are many applications of cloud monitoring, such as:

1. Troubleshooting: Monitoring allows analysis of network protocols and behaviour. It also helps in finding network traffic problems.

2. Security: Monitoring mechanisms help with sampling data to look for network-based attacks.

3. Performance: Monitoring helps in optimizing the performance of the CDC resources and network.

In SDN, there can be many problems that can remain undetected in the network, thus requiring process monitoring process. Monitoring in SDN also helps in checking the utilization of CPU, latency and the total request count. Additional activities can also be checked such as memory usage and data volume. Due to these features, monitoring plays an important in SDN data centres [12]. Network monitoring and visibility has become increasingly challenging since IT infrastructure must support network, server, and storage virtualization, as well as user access to cloud-based applications. Integration of service context and dynamic or real time change are among the hard challenges of the monitoring process.

While end-to-end monitoring is important for all type of Cloud applications (multi-tier web, content delivery network, etc.), it is much more critical in context of big data analytics application where network performance determines (e.g. network throughput and latency) performance of virtual machine hosted analytics software component (e.g., mappers and reducers in context of Apache Hadoop) In big data analytics applications, SDN can be leveraged to program switches in order to provide optimal data flow paths between data analytics software components on the fly, during each stage of data analysis. SDN enables better QoS between the virtual machines by dedicating more cross-links between them depending on the type of the analysis process [7]. However, this approach brings several challenges, such as: 1) enabling all vendor technologies to implement both SDN controller integration with virtual machine schedulers. and 2) how to instrument SDN devices with monitoring agent. The implementation of Network Function Virtualization (NFV) can decrease the amount of hardware required to launch and operate network services. However, NFV also brings some challenges of its own, such as: 1) performance degradation, 2) the elasticity of service provisioning may require the consolidation and migration of VNFs based on traffic load and user demand. 3) Implementation of NFV brings a new set of security concerns, as virtual applications running within data centres might not owned by network operators.

Monitoring a network is a top concern within IT departments. This is especially the case as monitoring efforts are ubiquitously leveraged to meet network security and performance goals. Monitoring can be effected by the unavailability of useful tools and alerting capabilities. In different networks, it is necessary to capture, store and analyse the vast amount of monitoring data. In short, we can say that monitoring is quite difficult and challenging due to different problems like Integration of service context and Dynamic or real time change.

An ideal scenario would be one where the capabilities of SDN and NFV are combined to provide a holistic monitoring solution. SDN has capacity of building and managing large IP/Ethernet networks by separating the network's control, whereas NFV (with the purpose of reducing deployment costs) has capability of virtualizing network functions and migrate them to generic servers [13].

There are several commercial monitoring tools available for big data monitoring, they are: Monitis, RevealCloud, LogicMonitor, Nimsoft, Nagios, SPAE by SHALB, CloudWatch, OpenNebula, CloudHarmony, Windows Azure FC etc. These tools can provide a report in the form of a graph or chart about the load on servers and other information. Each of these tools have their own advantages and drawbacks. For example "RevealCloud" can only provide information for the last 30 days. These tools use different communication protocols for performing tasks, which may or may not work perfectly with SDN and/or NFV communication protocols.

3 Configuration Management Challenges

Configuration management is the detailed recording and updating of information that describes an enterprise's hardware and software. In a Cloud computing context, configuration management supports the management of services by providing information about how the services are being assembled or bundled together. This information is crucial to the other service management processes, especially for change management, incident management, or problem management. It is also crucial to ensure meeting all agreed-to service levels. Many tools are made for automating the cloud applications configuration management. Among these tools, CFEngine, Puppet, Chef, Ansible are well known. Different cloud-based services (e.g., IaaS, PaaS, or SaaS), will require different levels of configuration management. In the IaaS layer, the consumer of IaaS services usually has control over the configuration aspects of the resources, such as which operating system to run on a virtual machine, or how to utilize the storage resources, or how to assign IP address to a provisioned virtual machine. In the PaaS layer, configuration management could be performed on the individual components of the platform, such as the automated integration between database and application server layer in context of multi-tier web applications. In the SaaS layer, configuration management operations include configuration end user's account and access credentials with the CDC-hosted SaaS application.

To allow information flow between the management plane and other planes (Figure 1), configuration management interfaces need to be in place. These interfaces allow settings to be enforced across network devices, and information to be sent back into the management plane, for example, for reporting to a network administrator [9]. In traditional networks, the logic of the data and control planes is confined inside each network device according to a well-defined set of standardized protocols. With the separation of planes, as SDN promotes, the need to bootstrap the communication between the data and control planes becomes a basic requirement. Configuring this communication can be particularly complex, considering that both planes can operate under protocols defined by software. Moreover, changes in any plane may affect such communication directly. Ideally, a new management interface is required than can manage configuration properly in SDN.

Given the need to deploy and setup novel data forwarding devices into a traditional network, administrators must configure these devices in order to have them operational. In the most basic and common scenario, the administrator would interact with a command line interface on the device and have it configured by performing many intricate proprietary commands. In the case of SDN data forwarding devices, there are still a few parameters to be configured, such as the communication policy with the control plane (e.g., drop every packet when the control plane is unavailable or follow the last valid set of rules). In response to these necessities, the Open Networking Foundation (ONF) proposed the OpenFlow Management and Configuration Protocol (OF-Config). This protocol allows operational staff to assign controllers to switches, set ports up/down, configure queues, assign certificates for the communication with the control plane, set up tunnels, handle versioning, and retrieve device capabilities. OF-Config is already a significant step toward tackling dataplane-related management requirements. On the other hand, this protocol is targeted specifically to OpenFlow networks; therefore, it is tied to the limited view of SDN employed by this technology (e.g., fixed dataplane and logically centralized control plane).

The challenge of configuration management in SDN is to automate the management of configurations in each plane (application plane, control plane, data plane) by an administrator via management interfaces. SDN and its most known realization, OpenFlow, has enabled widespread and vendor-neutral programmability of the control plane of the network environment. However, despite such proposals from research and standardization bodies, the management plane still lacks a comparable interface and protocol for capability discovery, device management and monitoring [10]. Therefore, network management is still heavily human-centred with minor autonomous behaviour. The Organization for the Advancement of Structured Information Standards (OASIS) is working on a Topology and Orchestration Specification for Cloud Applications (TOSCA), to enable the creation of portable cloud applications and the automation of their deployment and management. This standard provides a concept named management plan, which makes the management of complex enterprise applications automated, repeatable, traceable, and less error prone. However, the abstraction focuses on higher-level network services such as database management systems (DBMS) and their relationship to other entities, not on details of the individual forwarding elements.

In terms of Network Function Virtualisation (NFV), the agile and automated management of virtualized network functions (VNFs) throughout their lifecycles becomes a foremost objective [11]. TOSCA operationalizes the deployment of VNFs and triggers their initial configuration. A TOSCA template is a file-based description of VNF-related things and a set of execution guidelines for deploying and operating them as a single entity on cloud infrastructure. Each TOSCA-defined node has interfaces through which it can be manipulated by an initial configuration application or script, which is referenced in the TOSCA template. However, each VNF instance needs to be further configured at runtime to fulfil the specific needs of a consumer and service. Runtime configuration is beyond the scope of a TOSCA template.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of internet services and applications* 1.1 (2010): 7–18.

[2] D. Kreutz, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *IEEE*, pp. 14–76, January 2015.

[3] S. Sukhpal, C. Indervee. "Q-aware: Quality of service based cloud resource provisioning," *Computers & Electrical Engineering*, pp. 138–160, October 2015.

[4] S. Sukhpal, C. Indervee, "QRSF: QoS-aware resource scheduling framework in cloud computing," *The Journal of Supercomputing*, pp. 241–92, September 2014.

[5] D. Warneke, O. Kao, "Nephele: efficient parallel data processing in the cloud," *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, 2009.

[6] L. Cui, F. R. Yu and Q. Yan, "When big data meets software-defined networking: SDN for big data and big data for SDN," *IEEE Network*, vol. 30, no. 1, pp. 58–65, January–February 2016.

[7] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 210–217, June 2014.

[8] D. Raumer, L. Schwaighofer and G. Carle, "MonSamp: A distributed SDN application for QoS monitoring," *Federated Conference on Computer Science and Information Systems*, Warsaw, pp. 961–968, 2014.

[9] Wickboldt, Juliano Araujo, et al. "Software-defined networking: management requirements and challenges." *IEEE Communications Magazine*, 53.1 (2015): 278-285.

[10] Sieber, Christian, et al. "Towards a programmable management plane for SDN and legacy networks." *NetSoft Conference and Workshops (NetSoft)*, 2016.

[11] Mijumbi, Rashid, et al. "Network function virtualization: State-of-the-art and research challenges." *IEEE Communications Surveys & Tutorials* 18.1 (2016): 236–262.

[12] S. R. Chowdhury, M. F. Bari, R. Ahmed and R. Boutaba, "PayLess: A low cost network monitoring framework for Software Defined Networks," *IEEE Network Operations and Management Symposium (NOMS)*, Krakow, pp. 1–9, 2014.

[13] G. Gardikis et al., "An integrating framework for efficient NFV monitoring," *IEEE NetSoft Conference and Workshops (NetSoft)*, Seoul, pp. 1–5, 2015.

[14] E. Solaiman, R. Ranjan, P. P. Jayaraman, K. Mitra. "Monitoring Internet of Things Application Ecosystems for Failure", *IT Professional*, IEEE, 2016.

[15] E. Solaiman, I. Sfyrakis, C. Molina-Jimenez. "A State Aware Model and Architecture for the Monitoring and Enforcement of Electronic Contracts". *18th Conference on Business Informatics (CBI)*, IEEE, 2016.

Emerging Research Topics for Intelligent and Connected Vehicles

Chung-Wei Lin, Nikos Arechiga, Siyuan Dai, BaekGyu Kim, and Shinichi Shiraishi
Toyota InfoTechnology Center U.S.A.

1 Introduction

Automotive systems are very typical cyber-physical systems. As Advanced Driver Assistance Systems (ADAS), autonomous driving, and connected vehicles emerge, automotive systems are becoming more software-defined. Along with this trend, more and more cyber technologies tend to be applied to automotive systems and their development. In this article, we point out several research topics for next-generation vehicles. First, we demonstrate the need for assurance of automotive systems with Artificial Intelligence (AI) components. Next, we introduce the potential applications and challenges of Virtual Reality (VR) and Augmented Reality (AR) in the automotive domain. Then, we present a platform for plug-and-play software architecture which allows software to be installed and updated in a flexible way. To support this on-demand kind of automotive systems, safety and reliability need to be addressed by on-demand software certification. Lastly, we provide some potential design challenges considering security.

2 Assurance of Systems with AI Components

ADAS and autonomous driving frequently rely on AI components for perception and decision-making applications. Since these features are safety-critical, it is paramount to have solid assurance techniques, such as formal verification or automatic test-case generation with good coverage. In the specific context of automotive systems, three concrete challenges arise.

First, existing assurance techniques struggle to handle the sheer scale of industrial models, even when these do not contain AI components [1, 2]. This difficulty will only be compounded by the increased size of software for ADAS and autonomous driving. Next, AI components frequently make use of transcendental functions (such as sigmoids or arctangent activation functions in neural networks). These functions have a high arithmetic complexity and are computationally expensive to analyze [3, 4]. Lastly, formal specifications in the traditional sense cannot easily be given for AI components used for perception applications. We cannot, for example, formally specify that a vision algorithm should correctly detect a stop sign in every image that contains a stop sign, since that would require a formal characterization of images that contain stop signs. Indeed, AI components are commonly deployed for applications in which it is difficult to write a precise characterization, and a reinforcement learning technique is used to find patterns in example data. This challenge requires shifting our paradigm to thinking about properties of the overall system and decomposition into appropriate properties at the interface level [2], which may take the form of robustness or error rate requirements instead of traditional formal specifications.

Addressing these challenges will require the development of new and more powerful techniques to automatically analyze large and complex models.

3 Applications with Virtual Reality and Augmented Reality

The emerging VR technology has been revolutionizing the gaming industry with many companies having devoted large amounts of resources in the development of the technology. Many industry experts claim that the impact of VR technology will be as significant as the impact of the smartphone or the Internet [5]. Currently, the most popular way to experience VR is through head-mounted displays, such as the PlayStation VR by Sony and the Oculus Rift by Facebook. These headsets use stereoscopic lenses and motion tracking sensors to produce a visually appealing virtual experience [6]. In addition to the gaming industry, VR technology has been applied to the automotive domain and provides a new way of experiencing driving. Many popular racing titles have developed plugins which allow the gamer to experience the game through VR headsets. We have also developed some VR environments as shown in Figure 1.

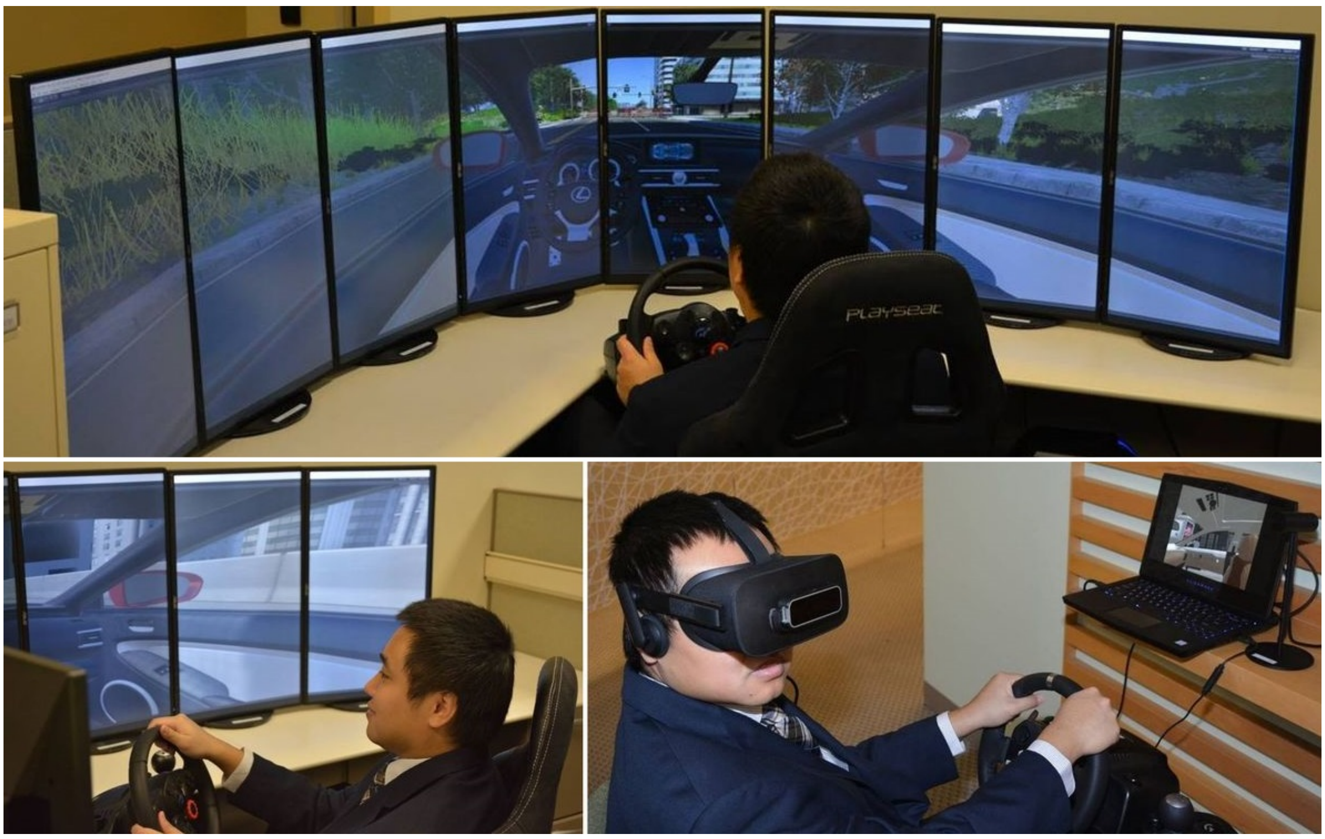


Figure 1: Our VR environments

Another important technology that has been gaining traction over the past decade is AR, where the physical world is augmented by virtual objects that the user can view through a device. The 2016 release of Microsoft's HoloLens brought AR technology into the mainstream, and since then many companies have developed a wide array of AR applications. AR technology is especially beneficial to the automotive domain because it allows the design of a wide variety of applications specific to assisting the driver. Recently, many automotive applications for AR have been developed ranging from pedestrian warning systems [7] to automated driving assistance [8].

VR and AR technology contains many important engineering challenges that must be solved. A disadvantage of any VR environments is a phenomenon known as VR sickness [9]. VR sickness is similar to motion sickness and is caused by a visual perception of motion when no physical motion is experienced. This is often the result of a VR system not being totally immersive and lacking some important tactile feedback [10]. Studies often show that humans are most responsive to acceleration and jerk; in the case of automotive applications, seeing the vehicle accelerate or jerk but not feeling either will often cause dizziness [11]. An important research topic from this that we are trying to address is the impact of force feedback to the alleviation of motion sickness. For AR technology, safety is a major concern, and application designers must ensure that the virtual objects are placed in an optimal location on the AR device. This research is especially important for the automotive domain because the obstructing of an important object in the environment can often result in an accident occurring.

4 Plug-and-Play Platform and Hardware Virtualization

In comparison to a vehicle that significantly relied on mechanical operation, a modern vehicle is considered a software platform that can be reconfigured upon various customers' needs. This trend has a large potential to create new personalized business models cooperating with other existing businesses. One example is the business model linked to the car-sharing service. Third-party companies can develop various vehicle applications independently of Original Equipment Manufacturers (OEMs), ranging from in-vehicle infotainment system to advanced vehicle warning systems (e.g., lane depart alert). Then, customers purchase their preferred applications, and, when they rent a shared vehicle, they can plug in those applications to the rental car for more personalized driving experiences.

However, there are several challenges towards the plug-and-play automotive software platform. First of all, traditional automotive software architecture is not designed to accept third-party software. Typically, only software engineers inside OEMs or suppliers can design, develop, and test the software. This nature makes it hard to let vehicle software be accessed and implemented by third-party companies. Another challenge is the safety issue. Typically, OEM software follows a standardized and thorough test procedure to assure the safety of software. This requires, for example, whenever a software component undergoes upgrade or modification, engineers need to drive the subject vehicle for a long distance to make sure such changes do not impact to the vehicle safety. However, it is difficult to expect third-party software to go through such a strict process to assure its safety and meet OEM's expectation.

We believe more research needs to be done to open some part of OEM software to the public so that third-party companies can develop compatible applications, while assuring safety of co-existence of OEM software and third-party software. One relevant research topic is the hardware virtualization. Many OEM software components are tightly coupled with sensors and actuators of a vehicle hardware platform. However, it is difficult to make such physical platforms available to third-party companies to develop their applications. This implies that OEMs need to provide a means for third-party companies to independently develop their applications of a particular physical vehicle platform. A well-designed programming interface would appropriately abstract heterogeneous sensors and actuators across a range of vehicle platforms and provide enough expressiveness to develop various personalized applications ranging from driving comfort to vehicle control.

In addition, the OEM software architecture should be designed in a way to isolate the impact of third-party software from safety-critical software. Since it is hard to expect third-party software to have the same level of safety quality as OEMs', the software platform should make sure that any malfunction or malicious behavior does not affect the operation of safety-critical components. For example, several industrial virtualization technologies, such as OS hypervisor, aims at isolating the performance impacts across different virtual components in mixed-critical systems. We believe incorporating such virtualization technology into the vehicle software platform is an appropriate research topic to realize the plug-and-play software architecture.

5 On-Demand Software Certification

Considering the plug-and-play scenarios above, correctness and quality of automotive software are extremely important. To prove the fulfillment of regulations and enhance customers' confidence, software certification is a promising technique, and it should integrate verification, simulation, and testing results in systematic and rigorous ways. However, compared with other safety-critical domains such aviation and medical devices, the software certification in the automotive domain has not been well-established. This is because of many challenges including lack of corresponding regulators, multifarious process-based checklists, the gap between produce-based testing and rigorous certification, and complicated system composition. Expecting that software certification will become a necessary step in future automotive design, we are considering assurance cases as a systematic way to describe system requirements. We need clear and efficient guidelines to derive assurance cases and maintain their reusability, extensibility, and other properties. At the same time, we have to consider the scenarios that some software source codes are confidential and inaccessible [12]. Although the overall certification process is difficult to be fully-automatic, research should be conducted to make it as automatic as possible.

6 Security-Aware System Design

Automotive security threats and protections have been discussed for many years. To support ADAS and autonomous driving, automotive electronic systems become more distributed and connected than ever, no matter from the perspectives of in-vehicle networks or Vehicle-to-X (V2X) communications. These connections create a variety of interfaces, such as direct or indirect physical access, short-range wireless access, and long-range wireless channels, which become breeding grounds for security attacks [13, 14]. Especially, V2X communications bring more challenges to automotive systems. Most of current safety-critical functions do not rely on external information, and thus a gateway can provide certain separation between internal and external networks. However, when connected services are implemented, a safety-critical function may behave based on external information. For example, a vehicle decides its speed based on the external messages containing the status of a traffic light or the speeds and the accelerations of other vehicles.

Although there are many existing general security mechanisms in the literature, it is difficult to apply them to automotive systems. One reason is that automotive systems do not have sufficient computational resource and communication bandwidth to support those security mechanisms. Another reason is that, without considering security in early design stages, those security mechanisms cannot meet other system requirements, such as timing constraints, which are usually strict in automotive systems. A complete security solution should consider other system requirements in early design stages. Besides, it should cover from the physical layer to the application layer as well as from in-vehicle components and networks to V2X communications.

7 Conclusion

In this article, we covered many emerging research topics in automotive systems and software. These directions provide broad and deep research space and need multidisciplinary effort to address their corresponding challenges. We truly believe that the research can propel the development of more intelligent, more user-friendly, more customized, safer, and securer automotive systems.

References

[1] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts, “Powertrain Control Verification Benchmark,” *ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, pp. 253–262, Apr. 2014.

[2] S. Seshia, D. Sadigh, and S. Sastry, “Towards Verified Artificial Intelligence,” *Technical Report*, Jul. 2016. https://people.eecs.berkeley.edu/d̃sadigh/Papers/seshia-verifiedAI-arxiv.pdf

[3] B. Akbarpour and L. C. Paulson, “MetiTarski: An Automatic Theorem Prover for Real-Valued Special Functions,” *Journal of Automated Reasoning*, vol. 44, no. 3, pp. 175–205, Mar. 2010.

[4] S. Gao, J. Avigad, and E. M. Clarke, “Delta-Complete Decision Procedures for Satisfiability Over the Reals,” *International Joint Conference on Automated Reasoning (IJCAR)*, 2012.

[5] P. Rosedale, “Virtual Reality: The Next Disruptor: A New Kind of Worldwide Communication,” *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 48–50, Jan. 2017.

[6] J. Wang, Y. Xu, H. Liang, K. Li, H. Chen, and J. Zhou, “Virtual Reality and Motion Sensing Conjoint Applications Based on Stereoscopic Display,” *Progress in Electromagnetic Research Symposium (PIERS)*, pp. 3382–3386, Aug. 2016.

[7] S. Langlois and B. Soualmi, “Augmented Reality Versus Classical HUD to Take Over from Automated Driving: An Aid to Smooth Reactions and to Anticipate Maneuvers,” *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1571-1578, Nov. 2016.

[8] M. T. Phan, I. Thouvenin, and V. Fremont, “Enhancing the Driver Awareness of Pedestrian Using Augmented Reality Cues,” *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1298–1304, Nov. 2016.

[9] F. P. Brooks, “What’s Real about Virtual Reality?” *IEEE Computer Graphics and Applications*, vol. 19, no. 6, pp. 16–27, Nov. 1999.

[10] G. Robles-De-La-Torre, "The Importance of the Sense of Touch in Virtual and Real Environments," *IEEE MultiMedia*, vol. 13, no. 3, pp. 24–30, Jul. 2006.

[11] F. Wang, N. Ma, and H. Inooka, "A Driver Assistant System for Improvement of Passenger Ride Comfort Through Modification of Driving Behavior," *International Conference on Advanced Driver Assistance Systems (ADAS)*, pp. 38–42, Sep. 2001.

[12] C.-W. Lin, S. Shiraishi, and B. Kim, "An Amanat-based multi-party certification protocol for outsourced software in automotive systems," *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 13–16, Oct. 2016.

[13] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental Security Analysis of a Modern Automobile," *IEEE Symposium on Security and Privacy*, pp. 447–462, May. 2010.

[14] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," *USENIX Conference on Security*, Aug. 2011.

Resource Constrained Real-time Lane-Vehicle Detection for Advanced Driver Assistance on Mobile Devices

Tianchen Wang, University of Notre Dame

1 Introduction and Motivation

With the fast growing smart vehicle industry, advanced driver assistance system (ADAS) has attracted a lot of attention in recent years. Acting as a critical component in a closed human-in-the-loop cyber-physical system (CPS) [1], a qualified ADAS should keep drivers away from dangers by monitoring fast changing environment in real-time, as shown in Figure 1. Two basic functions in ADAS are the lane departure warning system (LDWS) and forward colli-

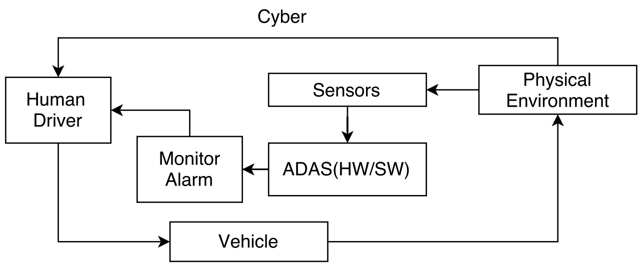


Figure 1: Scheme of ADAS with human-in-the-loop in CPS

sion warning (FCW), enabled by various software implementations that extract and locate lane marks [2] and detect on-road vehicles [3] respectively. To reduce latency and allow real-time application, hardware implementations are also investigated in FGPAs [4] and ASIC. However, these works all assume that a powerful microprocessor, FPGA or ASIC chip is available to process the data and run the algorithms. Unfortunately, many vehicles today do not have them pre-installed and a rebuild for ADAS upgrade is costly and not always feasible. To address this issue, we propose to utilize the widely available mobile devices such as smart phones or tablets, which typically have embedded processors and cameras that are needed by ADAS.

However, implementing ADAS in mobile devices is significantly different from that in PCs, FPGAs or ASICs. First of all, the architectures of mobile devices are typically different from those of PCs, FPGAs or ASICs, leading to different design considerations and performance optimization strategies. Second, mobile devices are highly resource constrained, temperature sensitive and poor in self-cooling. The increase in hardware temperature due to battery consumption, hardware computation or sunlight irradiation would trigger protection mechanisms such as frequency scaling leading to performance degradation, which is not desired for ADAS where latency is critical. As such, an accurate, flexible yet light-weight ADAS implementation dedicated to mobile devices is needed.

In this paper, we propose a real-time lane departure warning and vehicle detection system on mobile devices with embedded cameras, which reacts with little latency, consumes a small portion of system resource, works with uncalibrated cameras, and still achieves high accuracy. We have implemented the system on an Android smart phone and run it under various real road conditions, camera angles, vehicle types, countries and time of a day. Experimental results suggest that an average latency of 15 fps can be achieved with a high accuracy of 12.58 average pixel offset for each lane in all scenarios and 97% precision for vehicle detection. To the best of the authors' knowledge, this is the very first ADAS implementation on mobile devices with uncalibrated embedded camera to achieve the performance.

2 Real-time Lane and Vehicle Detection Modules

2.1 Lane Detection

To achieve fast and accurate lane detection and lane departure warning, only the left and right lane marks of the current driving road are detected in region of interest (ROI). The flowchart of lane detection and LDWS is shown in Figure 2. Although IPM is a common approach to enhance vertical line extraction, it needs careful calibration on

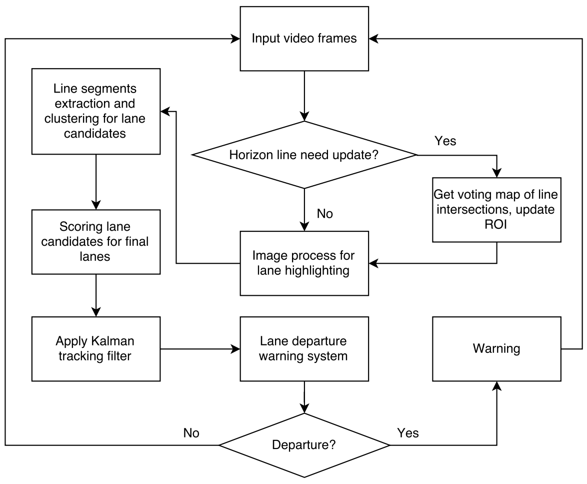

Input video frames
Line segments extraction and clustering for lane candidates
Horizon line need update?
Yes
No
Get voting map of line intersections, update ROI
Scoring lane candidates for final lanes
Image process for lane highlighting
Apply Kalman tracking filter
Lane departure warning system
Warning
No
Departure?
Yes

Figure 2: Lane detection in-process scheme

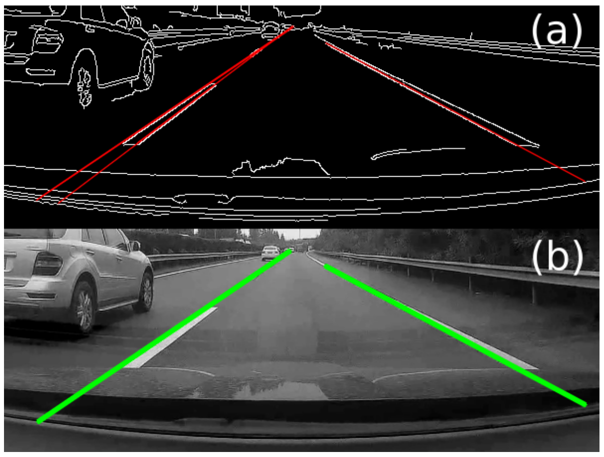


Figure 3: (a) Hough transform detected lines from Canny edge detector, and (b) Lane candidates extraction result

camera to obtain the correct parameters, which is not available for mobile devices. Therefore the ROI is calculated and extracted from the images rather than predefined. Based on the fact that in every frame, all lane detection candidates lie under the horizon, it is reasonable to consider only the bottom part of image area to reduce the computational complexity for mobile computing.

Since the camera is not pre-calibrated, the lane marker detection based on IPM or other similar methods may not work. It is found in various experiments that the x-coordinate of the vanishing point calculated is not always accurate due to the effect of all kinds of noise. The most effective method based on our experiments is described below.

To start processing, the image is first smoothed by Gaussian blur to minimize the image noise and to eliminate the edge cracks. After evaluation, probabilistic Hough transform instead of the standard one is applied to obtain the desired edges since it can significantly reduce the amount of computation by exploiting the difference in the fraction of votes needed to reliably detect lines with different numbers of supporting points. A good selection of the parameters, maximum line gap and minimum line length, can filter out small unwanted noise segments and connect the line candidates of dashed lanes.

After the lines are extracted as shown in Figure 3(a), they need to be merged, verified, and further filtered by certain hypothesis of lane detection. They will be first filtered by their slopes and locations with the assumption that the majority part of the left/right lane should locate on the left/right half of the ROI. It is also observed that the current road lane lies within a range of angles with respect to the driving direction. The remaining segments are then merged by their lateral offset which is shown in Figure 3(b).

To generate the final lane from detection part, the lane candidates are scored and the best one is picked for each side. The locations of lane marks in the current frame are supposed to be identical or close to those in the immediately preceding frames. As such, the closer distance to the previous lane marks, the higher score the lane candidate receives. After all filtering steps, the final lanes from detection part are obtained for further steps.

2.2 Vehicle Detection

In ADAS, vehicle detection can be divided into two parts to implement: identifying the vehicle in the image, and detecting its distance to the current vehicle. Like lane detection, we cannot process the vehicle detection on whole images all the time due to the tight latency requirement and limited computing power available on mobile devices. For the task of identifying vehicles, studies [5] have applied a boosted cascade of simple Haar-like local binary pattern (LBP) edge features in on-road vehicle detection systems. We can safely assume all wheels of vehicles are on the ground plane for daily on-road drives, and they can be regarded as solid bodies in which deformation and rotation are neglected.

To obtain the LBP feature, the pixel code based histogram is built within the image. It only adopts first-order gradient information between the center pixel and its neighbors. The original LBP operator labels the image pixels by applying the threshold of 3×3 neighborhood of each pixel with the center value and considering the result as a binary number. The histogram of these $2^8 = 256$ different labels can then be used as a texture descriptor.

After LBP operator processing, the extracted values are then classified by AdaBoost, which is illustrated and applied in [6]. We use seven stages, each of which has hundreds of weak classifiers. At each step, the method iteratively takes the best simple classifier and adds final set of classifiers. The off-line learning process then follows which has a decisive effect on feature detection accuracy. To make the data diverse for better result, we collect up to $30,000$ images of vehicle rear view on road to train the model.

The sweeping of features for vehicles is implemented by different sizes of sliding windows, which forms a s-factor loop. With the predefined thresholds for detection size, the size of sliding window is increased within certain range to detect the feature, and save the box if detected. When the size exceeds the maximum threshold, boxes are grouped and merged for range determination. By taking advantage of horizon with geometric adjustment and ROI limitation, the number of sliding window can be further decreased to speedup the program and reduce resource consumption. The flowchart of vehicle detection is shown in Figure 4.

3 Experimental Evaluation and Validation

The system with the balanced computation performance and resource consumption is integrated into an Android smartphone and realize both real-time vehicle and lane detections. We use Samsung Galaxy S7 with Qualcomm Snapdragon 820 CPU as the testing device, as shown in Figure 5. The camera of smart phone does not need any calibration for certain installation height, angle or other parameters.

With the 1080p (1920×1080) input video streaming, the average runtime on the smart phone for each frame can achieve as low as 65 ms, which meets the requirement of real-time ADAS system that runs at 15 fps [7]. The average CPU usage is 40% with four cores, which will not trigger protection mechanisms. In lane detection performance

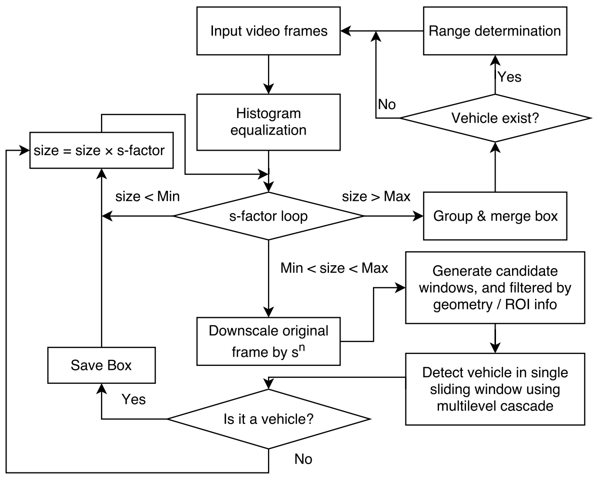


Figure 4: Flowchart of the vehicle detection algorithm

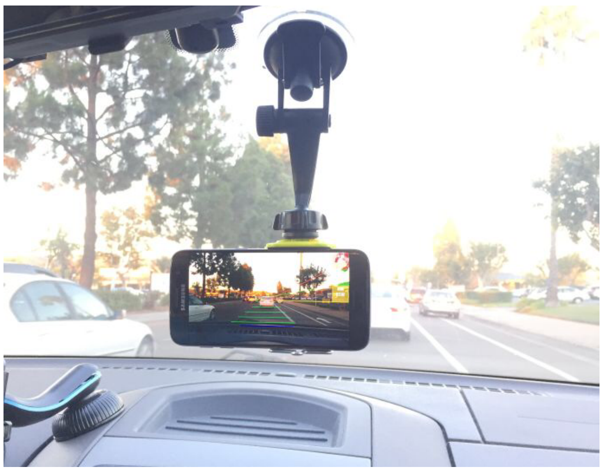


Figure 5: Real-time Android smartphone implementation

evaluation, it is hard to define and get the detection accuracy in percent only by comparing each lane with two given points. To make a more precise lane detection accuracy description, we use the pixel offset measurement between detected result and ground truth. Even with different lengths, if they pass through the same point with the equal slopes, we can consider the detected lanes are correct. Therefore, the detected lanes will be first reduced or extended to have equal y-coordinate as those of the ground truth lanes. Then the equation of average pixel offset S for each lane from n random frames of ground truth is defined as

$$S = \frac{1}{n}\sum_{k=1}^{k=n}\left(|p_{(1,T)}.x - p_{(1,D)}.x| + |p_{(2,T)}.x - p_{(2,D)}.x|\right), \tag{2}$$

where $p_1.x$, $p_2.x$ denote the x-coordinates of two end points of a lane in image; D, T represent detected lanes and ground truth lanes respectively. The pixel offsets of each lane are recorded within sections. For example, offset with 1 to 5 pixels are labeled as 5. Therefore for a set of pixel offset, the less value the better accuracy.

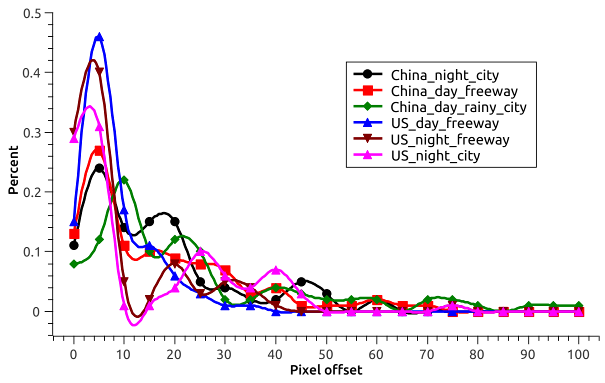


Figure 6: Flowchart of the vehicle detection algorithm

Vehicle size (Pixels)	Precision	Recall
<=50	97%	69%
60	97%	71%
70	97%	81%
80	97%	88%
>=90	97%	95%

Table 1: Vehicle detection evaluation result

Based on this, we test the videos with different scenarios and obtained the testing result shown in Figure 6, where x-coordinate denotes the value of pixel offset, y-coordinate denotes the percent of each pixel offset section. Combined with all scenarios, the calculated total average pixel offset for each lane is 12.58 pixels. In general, the method works better in freeway than in city due to clearer lane marks, less interference from other vehicles and environment. Also, it works better in day time than in night time because of the limited visibility of lane marks and the interference of other vehicles' rear lights. The rainy day has an effect on detection by blurring and decreasing the contrast the image.

For vehicle detection part, we use precision (positive predictive value) and recall (true positive rate) to evaluate the performance. We define a true positive as an over 50% overlapping between the detected bounding boxes and the ground truth ones. Due to the limitation of the training data, we only test the day-time on-road videos and combine the results from different scenarios to analyze. The vehicle detection evaluation result is shown in Table 1. The result shows that the proposed vehicle detection method has a very high accuracy on precision independent of vehicle size; on the other hand, the recall increases as the vehicle size increases. This indicates that our method has a great performance preventing the identification of non-existing vehicles, while would miss vehicles that do present, especially when they are small and far away. The first reason for this is the lack of tracking procedure during the vehicle detection. Without it, the bounding boxes may lose the target during video shaking or illumination variation. Lack of vehicle training data is also the reason of failed detection, such as the data with large trucks, pickups, old-styled vehicles. However, this is the cost we have to pay when implementing the framework with limited resources available on mobile devices.

4 Conclusions

The fast growing industry of smart cars accelerates the demand for reliable and practical ADAS, which forms a human-in-the-loop CPS involving complicated physical environment. To build a reliable, accurate and robust ADAS on mobile device, a new lane and vehicle detection framework is proposed that utilizes microprocessors with embedded camera. Low computation complexity, constrained resource and high accuracy requirements are balanced and optimized to serve the Android mobile devices with uncalibrated cameras. Evaluation results from all kinds of scenarios indicate that an average latency of 15 fps can be achieved with a high accuracy of 12.58 average pixel offset for each lane, and 97% precision for vehicle detection. To the best of the author's knowledge, this is the very first ADAS implementation on mobile devices with uncalibrated embedded camera to achieve the performance.

References

[1] S. Munir, J. A. Stankovic, M. Chieh-Jan and S. Lin, "Cyber physical system challenges for human-in-the-loop control," *Presented as part of the 8th International Workshop on Feedback Computing*, 2013.

[2] A. B Hillel, R. Lerner, D. Levi and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, vol. 25, no. 3, pp. 727–745, 2014.

[3] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[4] R. Marzotto, P. Zoratti, D. Bagni, A. Colombari and V. Murino, "A real-time versatile roadway path extraction and tracking on an FPGA platform," *Computer Vision and Image Understanding*, vol. 114, no. 11, pp. 1164–1179, 2010.

[5] F. Moutarde, B. Stanciulescu and A. Breheret, "Real-time visual detection of vehicles and pedestrians with new efficient adaBoost features," *2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV), at 2008 IEEE International Conference on Intelligent RObots Systems (IROS 2008)*, 2008.

[6] A. Khammari, F. Nashashibi, Y. Abramson and C. Laurgeau, "Vehicle detection combining gradient analysis and AdaBoost classification," *Proceedings. 2005 IEEE Intelligent Transportation Systems*, pp. 66–71, 2005.

[7] A. Kheyrollahi and T. P Breckon, "Automatic real-time road marking recognition using a feature driven approach," *Machine Vision and Applications*, vol. 23, no. 1, pp. 123–133, 2012

Technical Activities

1 Workshops

- The 2nd IEEE International Workshop on Design Automation for Cyber-Physical Systems (DACPS 2017)

2 Special Issues in Academic Journals

- IET Cyber-Physical Systems: Theory & Applications (IET-CPS) special issue on Cyber-Physical Systems for Medical and Life Sciences Applications
- IET Cyber-Physical Systems: Theory & Applications (IET-CPS) special issue on Cyber-Physical Systems in Smart Grids: Security and Operation
- IET Cyber-Physical Systems: Theory & Applications (IET-CPS) special issue on Cyber-Physical Aspects of EVs and HEVs
- IET Cyber-Physical Systems: Theory & Applications (IET-CPS) special issue on Recent Advances in Big Data Analytics and Cyber-physical Systems Security
- IET Cyber-Physical Systems: Theory & Applications (IET-CPS) special issue on Safety-Critical Cyber Physical Systems
- IEEE Transactions on Sustainable Computing (TSUSC) Special Issue on Sustainable Cyber-Physical Systems
- IEEE Transactions on Big Data Special Issue on Big Data for Cyber-Physical Systems
- ACM Transactions on Cyber-Physical Systems (TCPS) Special Issue on Smart Homes, Buildings, and Infrastructure
- Integration, The VLSI Journal Special Session on Hardware Assisted Techniques for IoT and Big Data Applications
- IEEE Transactions on CAD Special Issue on CAD for Cyber-Physical System

3 Special Sessions in Academic Conferences

- ISVLSI-2016 Special Session on Cyber-Physical Systems: Architecture and Security in Smart Buildings and Autonomous Driving
- ISVLSI-2016 Special Session on Emerging Devices for Hardware Security: Fiction or Future

4 Book Publications

- Springer Book "Leveraging Big Data Techniques for Cyber-Physical Systems"

Call for Contributions

Newsletter of Technical Committee on Cyber-Physical Systems (IEEE Systems Council)

The newsletter of Technical Committee on Cyber-Physical Systems (TC-CPS) aims to provide timely updates on technologies, educations and opportunities in the field of cyber-physical systems (CPS). The letter will be published twice a year: one issue in February and the other issue in October. We are soliciting contributions to the newsletter. Topics of interest include (but are not limited to):

- Embedded system design for CPS
- Real-time system design and scheduling for CPS
- Distributed computing and control for CPS
- Resilient and robust system design for CPS
- Security issues for CPS
- Formal methods for modeling and verification of CPS
- Emerging applications such as automotive system, smart energy system, internet of things, biomedical device, etc.

Please directly contact the editors and/or associate editors by email to submit your contributions.

Submission Deadline:

All contributions must be submitted by **July. 1st, 2017** in order to be included in the August issue of the newsletter.

Editors:

- Helen Li, University of Pittsburgh, USA, hal66@pitt.edu

Associate Editors:

- Yier Jin, University of Central Florida, USA, yier.jin@eecs.ucf.edu
- Rajiv Ranjan, Newcastle University, United Kingdom, raj.ranjan@ncl.ac.uk
- Yiyu Shi, University of Notre Dame, USA, yshi4@nd.edu
- Bei Yu, Chinese University of Hong Kong, Hong Kong, byu@cse.cuhk.edu.hk
- Qi Zhu, University of California at Riverside, USA, qzhu@ece.ucr.edu