ACCSE 2018

The Third International Conference on Advances in Computation, Communications and Services

ISBN: 978-1-61208-658-3

July 22 - 26, 2018

Barcelona, Spain

ACCSE 2018 Editors

Charles Liu, California State University, Los Angeles, USA

ACCSE 2017

Foreword

The Third International Conference on Advances in Computation, Communications and Services (ACCSE 2018), held between July 22 - 26, 2018- Barcelona, Spain, is targeting the progress made in computation, communication and services on various areas in terms of theory, practices, novelty, and impact. Current achievements, potential drawbacks, and possible solutions are aspects intended to bring together academia and industry players

The rapid increase in computation power and the affordable memory/storage led to advances in almost all the technology and services domains. The outcome made it possible advances in other emerging areas, like Internet of Things, Cloud Computing, Data Analytics, Smart Cities, Mobility and Cyber-Systems, to enumerate just a few of them.

We take here the opportunity to warmly thank all the members of the ACCSE 2018 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to ACCSE 2018. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the ACCSE 2018 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ACCSE 2018 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of computation, communication, and services.

We are convinced that the participants found the event useful and communications very open. We hope that Barcelona provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

ACCSE 2018 Chairs:

ACCSE Advisory Committee
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Mario Freire, University of Beira Interior, Portugal
Avi Mendelson, Technion, Israel
Young-Joo Suh, POSTECH, Korea
Michael Hübner, Ruhr-University of Bochum, Germany
Sotirios Kentros, Salem State University, USA
Shigeaki Tanimoto, Chiba Institute of Technology, Japan

ACCSE Industry/Research Advisory Committee
Daniel Ritter, SAP, Germany
Hiroaki Higaki, Tokyo Denki University, Japan
Danda B. Rawat, Howard University, USA
David Nelson, University of Sunderland, UK

ACCSE 2018

COMMITTEE

ACCSE Steering Committee

Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Mario Freire, University of Beira Interior, Portugal
Avi Mendelson, Technion, Israel
Young-Joo Suh, POSTECH, Korea
Michael Hübner, Ruhr-University of Bochum, Germany
Sotirios Kentros, Salem State University, USA
Shigeaki Tanimoto, Chiba Institute of Technology, Japan

ACCSE Industry/Research Advisory Committee

Daniel Ritter, SAP, Germany
Hiroaki Higaki, Tokyo Denki University, Japan
Danda B. Rawat, Howard University, USA
David Nelson, University of Sunderland, UK

ACCSE 2018 Technical Program Committee

Ali Ben Abbes, University of Manouba, Tunisia
Mai Abdelhakim, University of Pittsburgh, USA
Kishwar Ahmed, Florida International University, USA
Baris Aksanli, San Diego State University, USA
Markus Aleksy, ABB AG, Germany
Alessia Amelio, University of Calabria, Italy
Piotr Artiemjew, University of Warmia and Mazury in Olsztyn, Poland
Mozhgan Azimpourkivi, Florida International University, USA
Aleksander Bai, Norwegian Computing Center, Norway
Harald Baier, Hochschule Darmstadt / CRISP, Germany
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
An Braeken, Vrije Universiteit Brussel, Belgium
Zaki Brahmi, Taibah University, KSA
Alfonso Mateos Caballero, Universidad Politécnica de Madrid, Spain
Juan-Carlos Cano, Universitat Politecnica de Valencia, Spain
Robert Charles Green, Bowling Green State University, USA
Daniel B.-W. Chen, Monash University, Australia
Si Chen, West Chester University, USA
Arun Das, Visa Inc., USA
Gabriel Falcao, IT / University of Coimbra, Portugal
Gianluigi Ferrari, University of Parma, Italy
Stefano Ferretti, European Space Agency / European Space Policy Institute, Austria
Mario Freire, University of Beira Interior, Portugal
Hong Fu, Chu Hai College of Higher Education, Hong Kong

Table of Contents

Implementation of a Super-resolution Algorithm to Improve Spatial Resolution of Optical Signals

Byung-Jin Lee, Sung-Woo Kim
Department of Electrical and Electronic Engineering, Chungbuk National University, Cheongju, Chungbuk, Rep. of Korea
e-mail: Byung2487@naver.com, onlyparadise@nate.com

Bong Guk Yu
GCI INCORPORATED, 307, 218, Gajeong-ro, Yuseong-gu, Daejeon
e-mail: bgyu2@nate.com

Kyung Seok Kim
Department of Electrical and Electronic Engineering, Chungbuk National University, Cheongju, Chungbuk, Rep. of Korea
kseokkim@cbnu.ac.kr

***Abstract*—An Optical Time Domain Reflectometer (OTDR) is the most widely used instrument to monitor problems in currently installed optical fiber. It evaluates the physical characteristics of optical fiber, such as transmission loss and connection loss. It is important to increase the spatial resolution to accurately detect optical path problems using an OTDR. When the pulse width is greater than twice the distance between the two reflectors, the two reflected pulses overlap, and it is not possible to distinguish the reflected signal. To overcome these limitations, this paper proposes a spatial resolution–enhancement method by applying a super-resolution algorithm. As a result of digital signal processing implementation in TMS320C6455T DSP board, it is possible to analyze the event interval more precisely by improving the resolution when applying the super-resolution algorithm.**

Keywords—optical time domain reflectometer; spatial resolution; cross-correlation; super-resolution algorithm.

I. INTRODUCTION

The wired communications market is switching to fiber to the home (FTTH). With downsizing of base stations due to the competition of fourth-generation (4G) LTE services and the spread of optical repeaters, the mobile communications market has widely spread to antenna towers within a 25 km radius. With the advent of 5G and revitalization of the Internet of Things market, the use of optical fiber will increase sharply in the future. The OTDR is the most widely used instrument for monitoring problems in currently installed optical fiber. The OTDR is designed to test FTTx networks and evaluate the physical properties of the fiber, such as transmission loss and connection loss. The OTDR is widely used in the manufacture, construction, and maintenance of fiber optic communications systems. It is important to increase the spatial resolution to accurately detect optical path problems using an OTDR. Spatial resolution is the most representative parameter for OTDR performance, along with dynamic range. When the pulse width is less than twice the distance between two reflectors, the signals from them are reflected without overlap, and the reflected signal can be distinguished. However, when the pulse width is greater than twice the distance between the two reflectors, the reflected pulses overlap, and it is not possible to distinguish the reflected signal. Spatial resolution must be increased to distinguish the signal. To increase the spatial resolution, a narrow pulse width of 10 ns or less should be used. However, as the width of the pulse narrows, the distance of measurable optical fiber is shortened, and a receiver having a large reception bandwidth is required. As the receiving bandwidth increases, the amount of noise in the receiver increases proportionally, so the dynamic range of the OTDR decreases. That is, a conventional OTDR based on a single pulse has a tradeoff relationship between spatial resolution and dynamic range. To overcome this problem, a cross-correlation–based OTDR method was proposed [1]. Although this method has several advantages, such as increasing the dynamic range without sacrificing spatial resolution, it is still limited by the electron modulation bandwidth [2]. Therefore, a new super-resolution algorithm is needed to overcome these limitations and improve resolution.

In this paper, we propose a method of improving the spatial resolution of an OTDR based on cross-correlation by applying a super-resolution algorithm. We evaluate the spatial resolution performance of the OTDR by comparing the proposed method with a cross-correlation–based OTDR via Digital Signal Processing (DSP) implementation.

This paper is organized into five sections. Section 2 describes the cross-correlation–based OTDR system, and Section 3 analyzes the super-resolution algorithm for resolution enhancement. Section 4 describes the algorithm’s implementation and offers an analysis of the results. The main conclusions are in Section 5.

II. CROSS-CORRELATION BASED OTDR SYSTEM

A cross-correlation–based OTDR can be divided into three major parts, as shown in Figure 1: the transmitter, which mixes the downstream data signal with a pseudo noise signal, the part of the optical fiber that experiences various losses and reflections, and the receiver, which receives and analyzes the signal. The data signal is separated into a reference signal for correlation using a tap, a signal back-reflected from the optical fiber and received through a photodetector, and an OTDR trace that can be drawn by cross-correlation with the reference signal. A cross-correlation–based OTDR using a Golay signal can obtain the OTDR trace through cross correlation of $v(\mathrm{t})$ and Golay signals received through a Photo Diode (PD). The

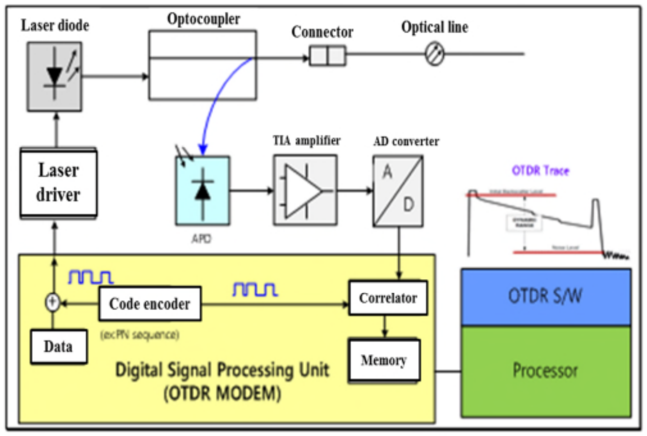


Figure 1. Cross-correlation-based OTDR system model

cross-correlation result of the two signals, $C(\gamma)$, can be expressed as:

$$\begin{aligned} C(\gamma) &= \int_0^{m\Delta t} v(t)s(t-\gamma)dt \\ &= \int_0^{m\Delta t} v_0(t)s(t-\gamma)dt + \\ &\int_0^{m\Delta t} n_{thm}(t)s(t-\gamma)dt + \int_0^{m\Delta t} n_q(t)s(t-\gamma)dt \\ &= C_0(\gamma) + C_{thm}(\gamma) + C_q(\gamma) \end{aligned} \quad (1)$$

Cross correlation between the signal returned from the OTDR and the Golay signal has a range of *0* to $m\Delta t$, where *m* is the length of the Golay signal, given as $m = 2^{\text{golay}} - 1$, and Δt is the interval between one bit in the Golay signal. That is, $m\Delta t$ is the time corresponding to one period of the Golay signal. $C_0(\gamma)$ denotes the measured reflectivity, $C_{thm}(\gamma)$ denotes the thermal noise generated from the TIA(Transimpedance Amplifier), and $C_q(\gamma)$ denotes the ADC(Analog to Digital Converter) quantization term. $C_{thm}(\gamma)$ and $C_q(\gamma)$ are terms related to the noise level when there is no backscattering signal on the OTDR trace, and $C_0(\gamma)$ determines the shape of the OTDR trace.

III. ANALYSIS OF THE SUPER-RESOLUTION ALGORITHM TO IMPROVE RESOLUTION

When the pulse width is more than twice the distance between the two reflections, the signal in which the two reflected pulses overlap and are reflected is not distinguishable. Therefore, a super-resolution algorithm to distinguish it is necessary. Among the signal processing methods available, the Multiple Signal Classification (MUSIC) algorithm is the most widely used. Although computational complexity is high, it is known as a technology that can provide highly accurate estimations [3]. The MUSIC algorithm is used to estimate the spatial spectrum of an incoherent signal. When the signal is coherent, the coherent signal will be combined into one signal, and the received independent signal is reduced in size due to the interference signal. This leads to less covariance matrix rank reduction and a larger number of eigenvalues than the incoming signal. To solve this problem, a method for reconstructing a conjugate matrix of a data matrix was proposed [4]. The proposed technique is based on the eigenvalue decomposition of the autocorrelation matrix of received signal $C(\gamma)$. The procedure for implementing the proposed OTDR-based super-resolution algorithm is as follows. The energy value of signal $C(\gamma)$, cross-correlated to find the reflection point of the raw data signal, is calculated, and the peak value extracted. This is a result of not applying the algorithm. Second, $C(\gamma)$ is applied to the resolution algorithm to derive reflection point results. We compare the performance of the second resolution algorithm by outputting the first and second results together. First, covariance matrix R of the received signal is expressed with Equation (2):

$$R_x = \mathrm{E}\{C(\gamma)C^H(\gamma)\} = \mathrm{APA}^H + \sigma_n^2 I = R_s + R_w \quad (2)$$

where *H* denotes the Hermitian transpose, and A is the signal gain value (A = 1). From Equation (1), covariance matrix *R* can be represented by the combination of signal covariance matrix R_s and noise covariance matrix R_w. $P = \mathrm{E}\{C(\gamma)C^H(\gamma)\}$ is the $k \times k$ covariance matrix of the signal. Assuming that additive noise is not correlated. The mean of noise is 0, and variance is equal to σ_n^2, and the covariance matrix can be expressed as $R_w = \sigma_n^2 I$. R_s is an *m*-by-*m* matrix with explicit matrix coefficients (Rank), where *m* is the number of signals reflected in the event interval. Therefore, each *(m-k)* eigenvector corresponds to a noise vector *m* by a *k* that has eigenvalue 0. As a result, the signal vector is orthogonal to the (*m-k*) noise vector. If there is no correlation between *k* signals, rank(APA^H) = *k*. Using this, covariance matrix R of the received signal is divided into a signal subspace and a noise subspace, and can be expressed by Equation (3):

$$R_x = \mathrm{APA}^H + \sigma_n^2 I = U_S \Lambda_S U_S + U_N \Lambda_N U_N \quad (3)$$

where $U_S = [u_1, u_2, \cdots, u_k]$ is the signal eigenvector matrix, and $U_N = [u_{k+1}, u_{k+2}, \cdots, u_m]$ is the noise eigenvector matrix, which then make transformation matrix *T*, where *T* is an m-order inverse unit matrix referred to as a transition matrix:

$$T = \begin{bmatrix} 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & 0 \end{bmatrix} \quad (4)$$

when $\mathrm{Y} = TC_0^*(\gamma)$, $C_0^*(\gamma)$ is a complex conjugate of $C_0(\gamma)$. The covariance matrix Y is as follows:

$$R_y = \mathrm{E}\{YY^H\} = TRY^*(r)T \quad (5)$$

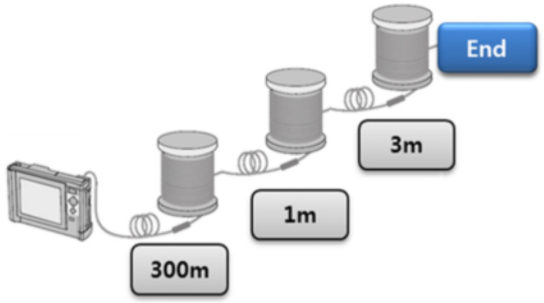


Figure 2. OTDR measurement environment

Figure 3. TMS320C6455T DSP board

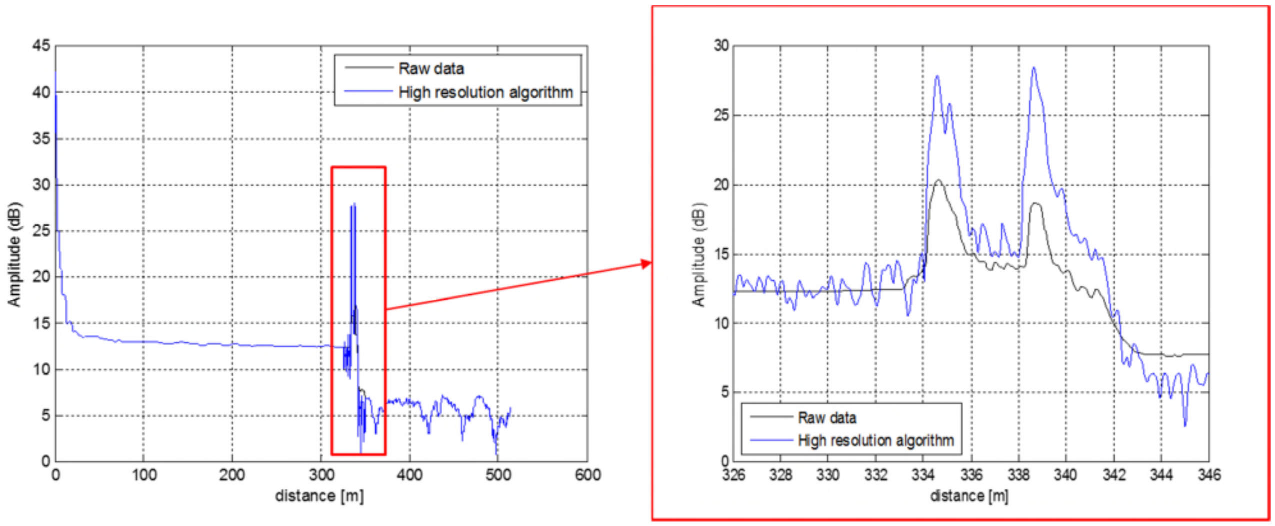


Figure 4. High resolution algorithm DSP implementation result

Here, R is the sum of R_x and R_y, and the reconstructed conjugate matrix can be obtained using the above equation:

$$R = R_x + R_y = \mathrm{APA}^H + T[\mathrm{APA}^H]^*T + 2\sigma_n^2 I \quad (6)$$

According to matrix theory [5], if **u** is an eigenvector corresponding to the zero eigenvalue of matrix APA^H, **u** must be an eigenvector corresponding to the eigenvalue of matrix $T[\mathrm{APA}^H]^*T$. Thus, the matrices R_x, R_y, and R has same noise subspaces. When eigenvalue decomposition is performed on R, eigenvalues and eigenvectors are obtained.

The noise subspace between the eigenvectors can be distinguished according to the number of signals to be estimated, as follows:

$$R = \hat{U}_S\hat{\Lambda}_S\hat{U}_S + \hat{U}_N\hat{\Lambda}_N\hat{U}_N \quad (7)$$

The MUSIC spatial spectrum is composed of the new noise subspaces, as follows:

$$F_{improved} = \frac{1}{A^H\hat{U}_N\hat{U}_N^H A} \quad (8)$$

IV. Algorithm Implementation and Results Analysis

In this section, we analyze performance by implementing the super-resolution algorithm in TMS320C6455T DSP board. First, the measurement environment is shown in Figure 2. Three events were created, 334 m and 335 m, and 338 m from the starting point, followed by the endpoint. The measurement parameters were set to a wavelength of SM1310 nm at a distance of 500 m. The pulse width was 3 ns, with the average amount of data at 2^{14}. The refractive index was 1.46, the total size of the data was 10,000 samples, and the data sampling interval, 5 cm. Figure 3 shows the TMS320C6455T DSP board that was used. The DSP chip used in the emulator implementation is a Texas Instruments TMS320C6455; the CPU operates at 1.2 GHz, flash memory was 2 MB, and Dynamic Random Access Memory (DRAM), 512 MB. However, with flash memory at 2 MB, if the program becomes complicated and the amount of computation increases, operation of the program becomes possible through DRAM. In this case, although the amount of DRAM is large, it is possible to expand the program, but the operating speed is slower than operation through flash memory. To configure the real-time system, consideration of memory management is also required [6].

The results of the DSP implementation in the above measurement environment are shown in Figure 4. Results from the DSP were stored and represented in Matlab. The black graph represents the raw data, and the graph outlined in red is from the super-resolution algorithm. It can be confirmed that an event occurs near 330 m in the total length of 500 m. If you look at the enlargement, you can see that the first event occurred at 334 m, and there is an event after another 1 m, and you can see that an event occurs after another 3 m. With raw data, you cannot decompose 1 m of 334 m and 335 m. From applying the resolution algorithm, the resolution improves, and the 1 m event interval that the raw data cannot decompose is correctly decomposed. The size of the signal level is also improved by about 8 dB, so the event interval can be more accurately distinguished.

V. CONCLUSION

In general, the performance of an OTDR is determined by its dynamic range and spatial resolution. In this paper, we analyzed the MUSIC algorithm to improve the resolution performance of the OTDR, and we implemented an improved MUSIC algorithm to compensate for the incoherent characteristics of the MUSIC algorithm. As a result of analyzing the performance of the algorithm through DSP implementation, we correctly decomposed events of 1 m intervals that could not have been resolved before, and the variation of the signals was also reduced. Compared with the raw data, the signal level improved by about 8 dB.

REFERENCES

[1] S. Furukawa, K. Tanaka, Y. Koyamada, and M. Sumida, “Enhanced coherent OTDR for long span optical transmission lines containing optical fiber amplifiers,” IEEE Photon. Technol. Lett., vol. 7, no. 5, pp. 540–542, May 1995

[2] X. Ai, R. Nock, J. G. Rarity, N. Dahnoun, “High-resolution random modulation cw lidar”, Applied Optics, Vol. 50, No. 22, pp. 4478-4488 2011.

[3] Z. I. Khan, M. MD. Kamal, N. Hamzah, K. Othman, and N. I. Khan, “Analysis of Performance for Multiple Signal Classification (MUSIC) in Estimating Direction of Arrival,” Preceeding of RFM, pp. 524-528, 2008.

[4] S. Su, W. Liu, W. Zheng, “Estimation of direction of arrival for correlation signals based on modified MUSIC algorithm”, JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE, Vol. 10, No. 13 , pp. 4027-4035, 2013

[5] Nering, Evar D. (1970), Linear Algebra and Matrix Theory (2nd ed.), New York: Wiley, LCCN 76091646

[6] Y. B. Jang, “DSP theory and practice”, Saeng neung Publisher, 2004.05

Radar-Camera Sensor Fusion Based Object Detection for Smart Vehicles

Eugin Hyun
ART Lab.,
DGIST,
Daegu, Korea
e-mail: braham@dgist.ac.kr

Young-Seok Jin
ART Lab.,
DGIST,
Daegu, Korea
e-mail: ysjin@dgist.ac.kr

Hyeong-Cheol Jeon
ART Lab.,
DGIST,
Daegu, Korea
e-mail: hamtooree@dgist.ac.kr

Young-Nam Shin
DAS Engineering Team,
SL Corporation,
Gyeongsanbuk-do, Korea
e-mail: ynshin@slworld.com

***Abstract*—We propose a post-data processing scheme for radar sensors in a fusion system consisting of a camera and radar sensors. The proposed scheme is divided into the recursive least square filter, the ghost target and clutter cancellation, and region of interest (ROI) selection. Especially for the recursive least square filter, we determine whether detections are valid tracks or are new tracks to initialize or update the filter parameters. Next, we apply the valid detections to the filter to reduce detection errors. Next, we cancel ghost targets as comparing the current tracks and the last tracks, and suppress clutter using the detected radial velocity. Finally, we select the ROI and determine the transfer coordinates to provide these values to the camera sensor. To verify the proposed method, we use Delphi commercial radar and carry out the measurements in a chamber and on the real road.**

Keywords- Post processing; Sensor fusion; ADAS.

I. INTRODUCTION

At present, the Advanced Driver Assistance System (ADAS) is one of the main issues for smart vehicle safety in road traffic. To support effective ADASs, the target detection sensors used are very important. Among currently available sensors, camera and radar sensors are commonly used in ADASs [1][2].

Because radar detects objects by emitting radio signals and analyzing the echo in the reflected signal, this system can operate robustly in different weather conditions [1][2]. Cameras are also widely used because they can provide rich data, similar to that by the human eye [1][2]. However, radar measurements are limited in terms of the angle resolution and this data is rather noisy due to false alarms. Cameras are also sensitive to light and weather conditions, and they have low detection accuracy levels, such as for the velocity and range detections.

Owing to these limitations, sensor fusion technology is considered as an efficient means of increasing target detection performance levels [1]-[3]. Because previous works have provided sensor fusion outcomes in the end stage [4][5], the computational complexity of camera classification remains high. Thus, in order to improve the detection performance and reduce the computational intensity, early-stage-based sensor fusion was proposed in previous works [6]-[8].

In the previous works [6][7], radar is used to detect targets in the region of interest (ROI) of captured image and searches for vehicle features within the ROI. However, the detection accuracy is unsatisfactory because the camera sensor detects the range of ROI as the pre-processing. In order to overcome the limitation, in another work [8], a more robust and efficient vision-based vehicle detection method was presented. In that case, the radar sensor provides ROIs for the camera sensor. Compared to the [6][7], because the radar sensor detects the range of target, the method can improve the detection accuracy and reduces the false alarm rate.

For the fusion method, the radar system provides precise ROI information to the camera sensor. Specifically, because the coordinates between the radar and the camera have differences, coordinate matching is also required. Finally, because the both sensors' fields of views (FOVs) are also different, the overlap area should be considered.

Thus, in this paper, we propose a radar post-processing scheme, which takes these issues into account for fusion of the camera and radar sensor. Section II briefly presents the proposed radar post-data processing scheme. Section III describes measurement results under the real field. Section IV presents the conclusion.

II. POST-DATA PROCESSING SCHEME

Figure 1 presents the expected fusion results of systems incorporating both camera and radar sensors.

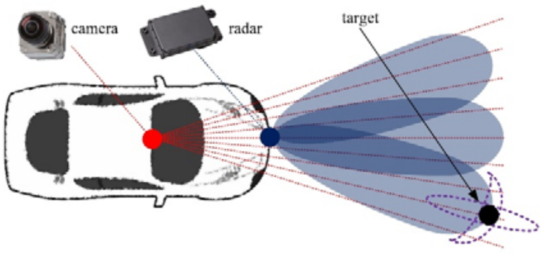


Figure 1. Expected fusion results of camera and radar sensors.

The angle detection of the radar sensor has a low resolution while range detection error of the camera sensor is very high. Here, the overlap between the two sensors is the final detected target position. In order to overcome the limitations of each sensor, we propose the sensor fusion based processing concept shown in Figure 2.

From the radar sensor, the detection information is received through a controller area network (CAN) with radar start commend. After packet receiving and decoding, the track data and the corresponding flag (or status) data are saved in the registers. Subsequently, through the proposed post data processing and projective transformation steps, the ROI information is transferred to the image processing path. In the camera sensor, based on these ROIs, feature extraction and target classification are carried out.

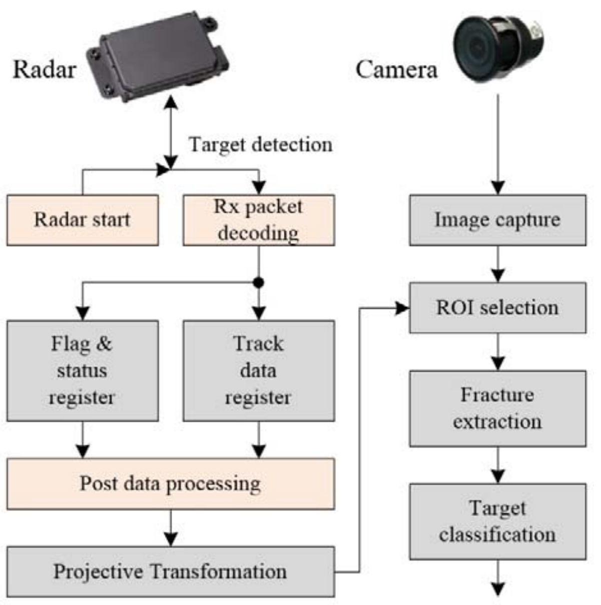


Figure 2. Proposed camera and radar sensor fusion processing concept.

The proposed post data processing scheme for radar is illustrated in Figure 3. First, we determine where detection is valid or not valid using the flag and status values.

In the second step, we initialize the parameters for error minimizing filter if the current track is new. On the other hand, for existing tracks, the corresponding parameters are calculated using the values updated in previous state. In this paper, we employed the recursive least square second order filter [9] to improve the angle detection error. On the other hand, the range and velocity values of the first step are passed without any modifications because the detection errors are very low.

The filter processing is expressed by (1) and (2), where $\mathrm{K1} = 2(2\mathrm{k}-1)/(\mathrm{k}^2+\mathrm{k})$, $\mathrm{K2} = 6/(\mathrm{k}^2+\mathrm{k})$, and $\mathrm{Res} = x[k] - \hat{x}[k-1] - \hat{x}_d[\mathrm{k}-1]$. In this equation, $\hat{x}[\mathrm{k}]$ is the k^{th} the estimated angle and $x[k]$ is the k^{th} measurement. For a new track (*k* is 1), we define $\hat{x}[0] = x[1]$ and $\hat{x}_d[0] = 0$.

$$\hat{x}[\mathrm{k}] = \hat{x}[k-1] + \hat{x}_d[k-1] + K1 \cdot Res \quad (1)$$
$$\hat{x}_d[k] = \hat{x}[k-1] + K2 \cdot Res \quad (2)$$

Next, the sudden tracks are cancelled. That is, as comparing the current track status and the previous information, we can determine the received detection is ghost or not.

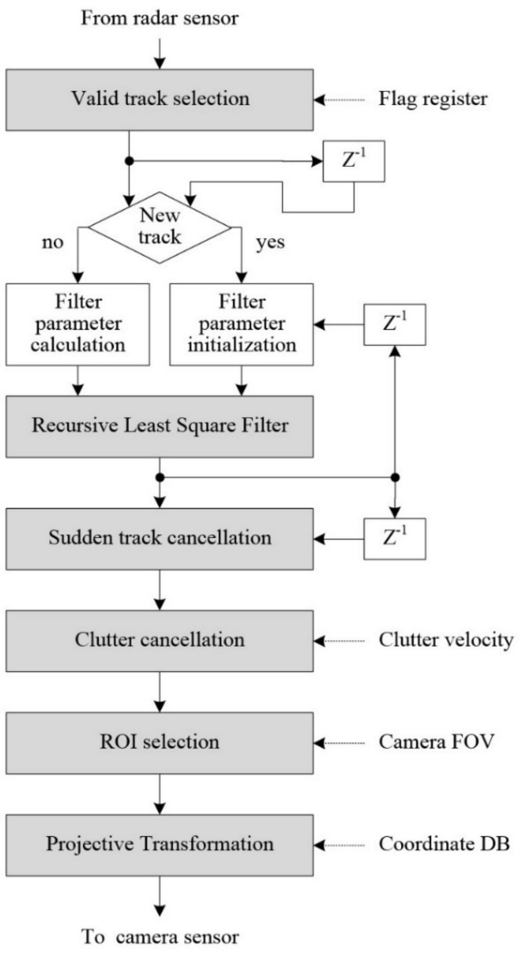


Figure 3. Post-data processing scheme for radar.

In the next step, we distinguish whether the current track is a target or a clutter. If the velocity of the current track, *v[k]*, meets (3), the current track is target, otherwise it can be regard as clutter. Here, v_{min} is maximum velocity of clutter and v_{ego} is ego velocity of subject vehicle. In addition, v_{min} is statistically estimated through the experiment through trade-off between the detection probability and false alarm rate.

$$v[k] < -v_{ego} - v_{min} \ or \ v[k] > -v_{ego} + v_{min} \quad (3)$$

Then we select the ROI information (range, radial velocity, and angle) in the overlap area of the radar and the camera. Finally, the projective transformation is carried out. In that case, the error bound is also fed to camera sensor considering the range detection error of the camera sensor and angle detection error of the radar sensor. The camera sensor will process images within window size, which reduces the complexity of image processing.

The steps described above are repeated until the scanning of final tracks is completed. The number of tracks is dependent on the type of commercial radar and the corresponding parameter setting.

III. MEASUREMENT RESULTS

In this paper, we employ the *Delphi* 77GHz ESR (Electronically Scanning Radar) system. In this radar system [10], because the maximum number of tracks is 64, the post-data processing described in Section II are repeated until the scanning of 64 tracks is completed.

TABLE I. DELPHI ESR SPECIFICATIONS

Category	Values
Size	173.7 × 90.2 x 49.2 mm (L x W x H)
Weight	575 g
Scanning frequency	76.5 GHz
Field of View	+/- 45°
Range	~ 60m
Target	64
Update rate	<= 50 ms

In order to verify the post-data processing method for radar, we configured a moving target measurement scenario in a chamber room, as shown in Figure 4. First, we install the *Delphi* ESR on the positioner and a single target is placed on the rail approximately 3.5 m away from the radar. The target then moved from 3.5m to 5.9 m in a round trip.

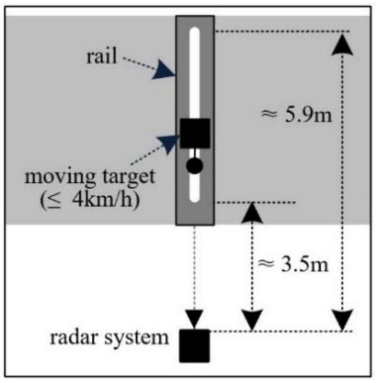


Figure 4. Moving target measurement scenario in chamber room.

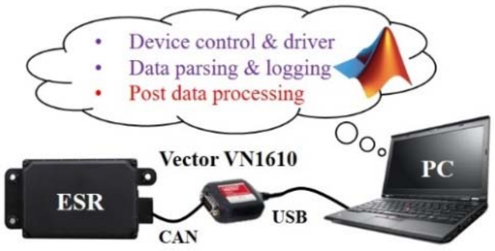


Figure 5. Radar measurement set-up.

We also utilized the measurement set-up shown in Figure 5. Here, a PC is connected to the ESR device through CAN to a USB converter. We coded the device driver software to start the radar sensor and data parsing program so as to log the received data using the Matlab simulator. In addition, we also completed the aforementioned post-data processing algorithm.

Figure 6 shows the detection result for several frame times: range (top), radial velocity (middle), and angle (bottom). As shown in the results, the angle detections contain numerous errors. Even when the moving target is placed on the middle line of the radar, the detection results were found to vary. Thus, we filtered the detection outcomes through a recursive least square filter. The corresponding outputs are indicated by the red line in Figure 6.

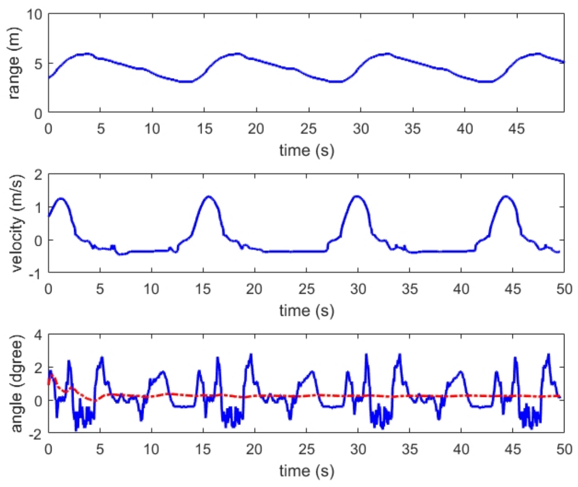
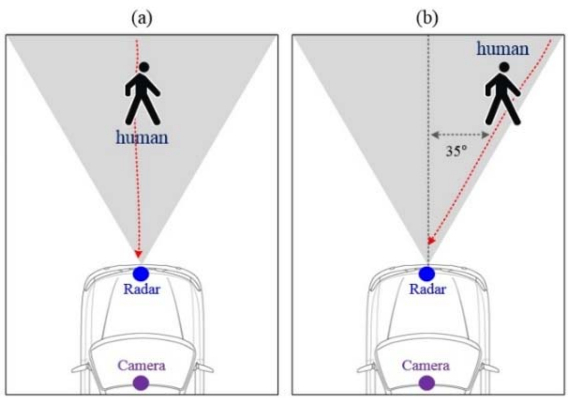


Figure 6. Results of post data processing for radar.

Next, in order to verify the algorithm on the real road, we install the radar and camera on the middle of the vehicle front bumper. The electrical power of vehicle was supplied to the both sensors. Signal lines were built in the vehicle to acquire radar signals and capture camera images together with a PC.

In addition, we considered four scenarios as shown in Figure 7. First, a single human is moving along the middle line of the radar sensor at approximately 6 m (a). In the second scenario (b), a human is walking along the right edge of radar FOV. In next scenario (c), the human is walking 3m away from the middle line in the longitudinal direction. Final case (d) shows the pedestrian who moves laterally at about 3 m away from the radar.



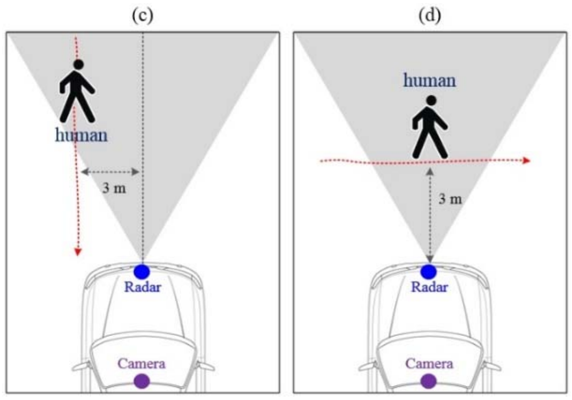


Figure 7. Results of post data processing for radar.

Figures 8-13 present the measurements and post-processing results for each scenario. We monitor moving human for about 7 seconds. In all figures, the blue points are the detected tracks in each frame.

First, Figures 8 and 9 show the results when the human is walking and running for the first scenario, respectively. Here, Figures (a) ~ (c) present the valid tracks received from radar sensor and Figures (d) ~ (f) show the post-data processing results. In Figures (a) and (d), the x-axis is the frame index and the y-axis indicates the range (meter). In Figure (b) and (e), the x-axis indicates the frame index and the y-axis is the angle (degree). Figures 8 (c) and (f) express the corresponding x- and y-positions (meter) of tracks over the whole frames. The results are calculated using range and angle values of each track. Here, the black line indicates FOV of radar. Figures 9 (c) and (f) show the radial velocity (m/s) over each frame.

In the results of Figure 8, we can see that angle errors are compensated through the recursive least square filter. Moreover, in Figure 9, we can find that the ghost targets and clutter are cancelled and the multiple scattering points oriented from one target are grouped together.

Next, in Figures 10, 11, and 12, we present the processing results for the scenarios 2, 3, and 4. In the results, we describe the x-y positions (meter) of target as calculating with the detected range and angle for all frames. We also mark black line to be able to see the detectable angle of radar. From all results, it was proved that the angle errors are minimized, the ghost and clutter were canceled, and the grouping was completed.

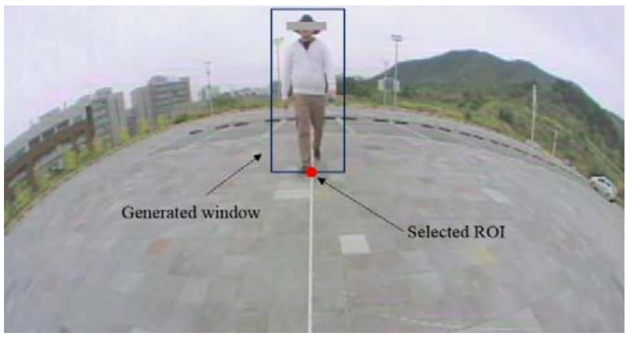


Figure 13. Example of ROI selection and window generation for the camera sensor

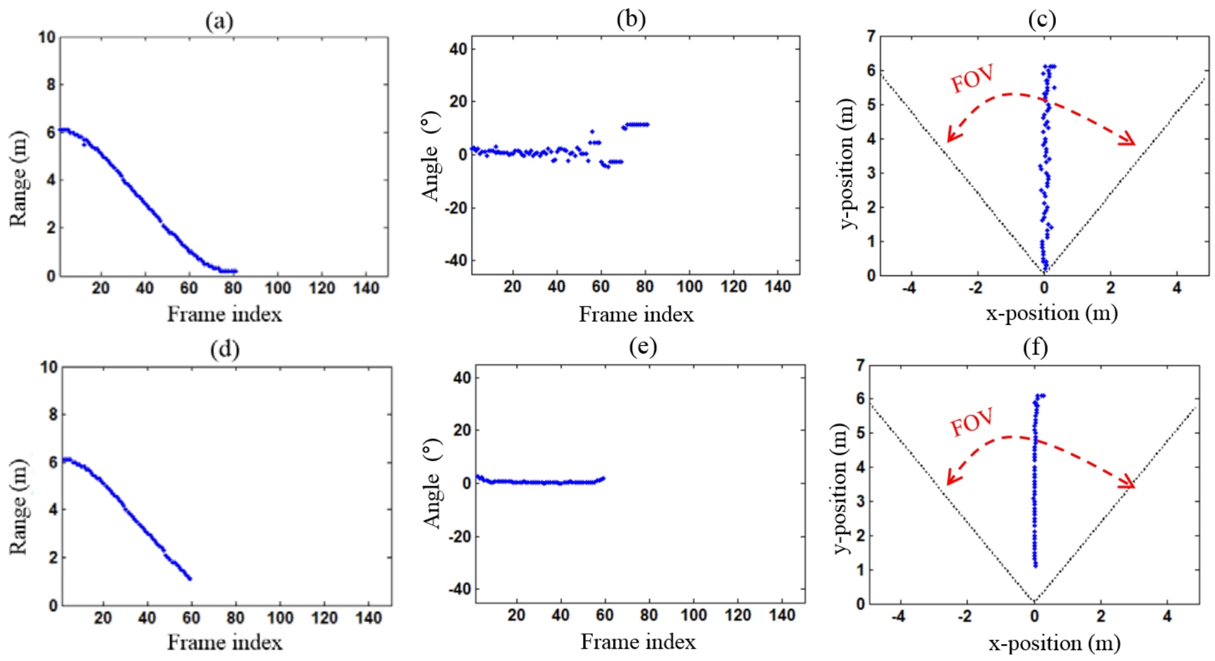


Figure 8. Detected target tracks (range, angle, and the corresponding xy-position) for the first scenario: (a)~(c) tracks before post-processing and (d)~(e) tracks after post-processing.

Last, Figure 13 shows an example of the ROI selection process (red circle) on the captured image for the first scenario. Here, the camera with wide angle was developed by the *SL* Corporation. In the image processing, the window can be generated based on selected ROI including the range and angle such the example of Figure 13.

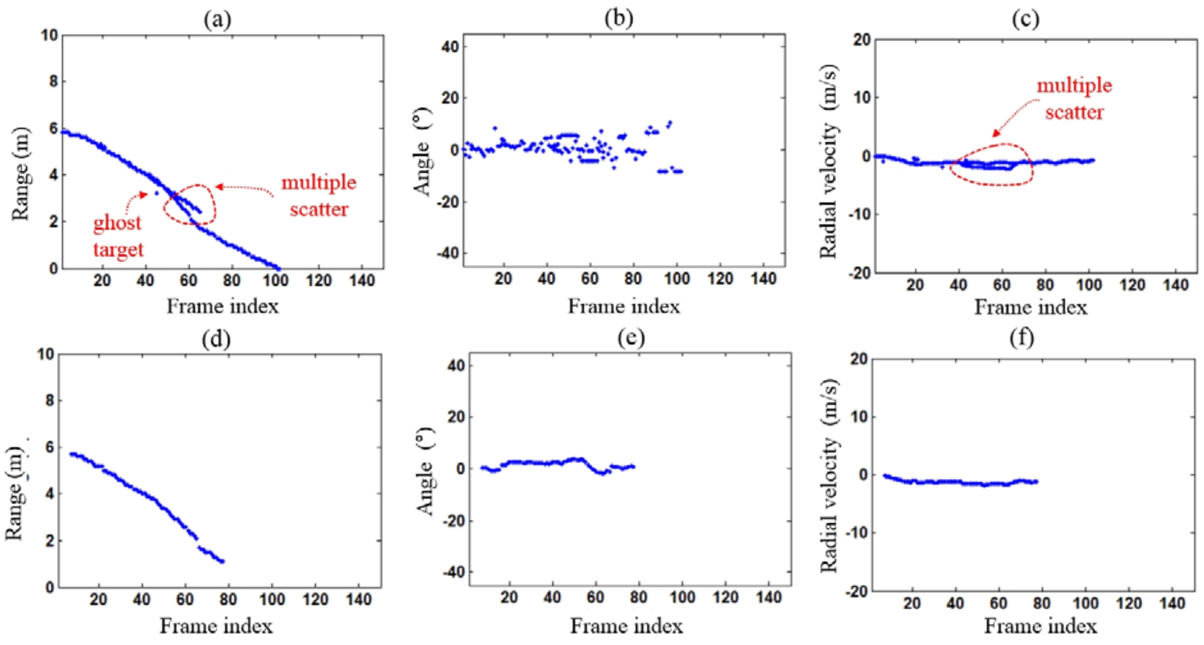


Figure 9. Detected target tracks (range, angle, and radial velocity) for the second scenario: (a)~(c) tracks before post-processing and (d)~(e) tracks after post-processing.

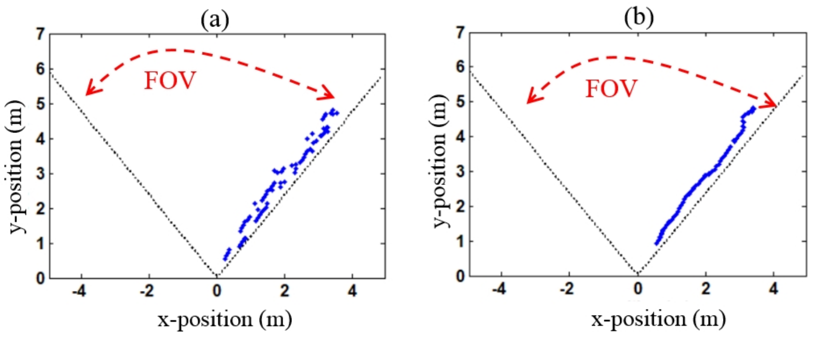


Figure 10. Detected target tracks (xy-position) over the whole frames for the third scenario.

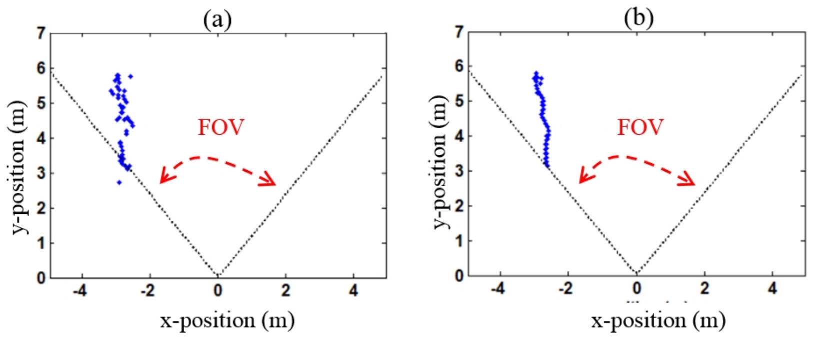


Figure 11. Detected target tracks (xy-position) over the whole frames for the fourth scenario.

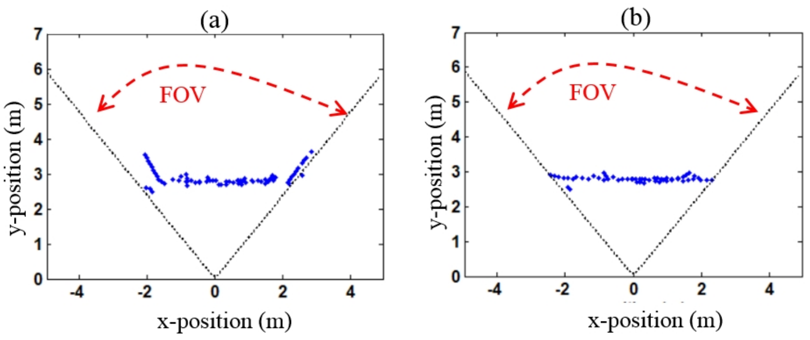


Figure 12. Detected target tracks (xy-position) over the whole frames for the fifth scenario.

IV. CONCLUSION

In this paper, we proposed post radar data processing for a camera and radar sensor fusion system. To do this, we utilized a Delphi 77GHz automotive commercial radar system.

First, using the flag values received from the radar, we determined instances of valid detection and new tracks. Next, we employed a recursive least square filter to reduce the detected angle error. Next we cancelled the ghost target and clutter using the received track information. Finally, based on the selected ROI information, the projective transformation is carried out for the camera sensor. The performance capabilities of the proposed scheme were assessed in a chamber and in the outdoor environment.

In the future, we will verify the proposed processing scheme in various scenarios on the real road. Thus, we will provide the meaningful results. Moreover, together with the camera sensor, we will develop methods of sensor fusion processing. Thus, we will compare the results of sensor fusion and them obtained by camera database alone.

ACKNOWLEDGMENT

This research was supported by the Technology Transfer and Commercialization Program through INNOPOLIS Foundation (2017-DG-0001) and the DGIST R&D Program (18-FA-07) funded by the Ministry of Science and ICT, Korea.

REFERENCES

[1] A. Gavriilidis, D. Müller, S. Müller-Schneiders, J. Velten, and A. Kummert, "Sensor System Blockage Detection for Night Time Headlight Control Based on Camera and Radar Sensor Information," IEEE ITSC 2015, Anchorage, USA, Sep. 2012.

[2] E. Hyun and Y. S. Jin, "Multi-level Fusion Scheme for Target Classification using Camera and Radar Sensors," IPCV'17, Lasvegas, USA, July 2017, pp. 111-114.

[3] J. Laneurit, C. Blanc, R. Chapuis, and L. Trassoudaine, "Multisensorial data fusion for global vehicle and obstacles absolute positioning," IEEE Intelligent Vehicles Symposium, Columbus, USA, Jun. 2003, pp. 138–143.

[4] R. O. Chavez-Garcia, J. Burlet, T. D. Vu, and O. Aycard, "Frontal object perception using radar and mono-vision", IEEE Intelligent Vehicles Symposium 2012, Alcala de Henares, Spain, June 2012

[5] U. Kadow, G. Schneider, and A. Vukotich, "Radar-vision based vehicle recognition with evolutionary optimized and boosted features," IEEE Intelligent Vehicles Symposium, Istanbul, Turkey, June 2007, pp. 749–754.

[6] A. Sole, O. Mano, G. Stain, H. Kumon, Y. Tamatsu, and A. Shashua, "Solid or not solid: Vision for radar target validation," IEEE Intelligent Vehicles Symposium, Parma, Italy, Jun. 2004, pp. 819–824.

[7] G. Alessandretti, A. Broggi, and P. Cerri, "Vehicle and Guard Rail Detection using Radar and Vision Data Fusion," IEEE Transactions on Intelligent Transportation Systems, Vol. 8, Iss. 1, pp. 95-105, Mar. 2007.

[8] X. Wang, L. Xu, H. Sun, J. Xin, and N. Zheng, "On-Road Vehicle Detection and Tracking Using MMW Radar and Mono-vision Fusion," IEEE Transaction on Intelligent Transportation Systems, Vol. 17, No. 7, pp. 2075-2084, July 2016.

[9] J. Lee and V. J. Mathews, "A fast recursive least squares adaptive second order Volterra filter and its performance analysis," IEEE Transactions on Signal Processing, Vol. 41, Issue 3, pp. 1087-1102, Mar. 1993.

[10] Autonimoustuff, "Delphi ESR Startup Guide version 2.1", Oct. 2015.

Machine Learning-Based Object Detection System Using PIR Sensor

Dong Hyun Kim, Jung Bin Park, and Jong Deok Kim*
Department of Computer Science and Engineering
Pusan National University
Pusan, South Korea
e-mail: {dhkim1106, jbpark, kimjd}@pusan.ac.kr

***Summary*—An intrusion prevention system using a digital Pyroelectric Infra-Red (PIR) sensor produces an error with an object, not a human. To solve this error, this research suggests an analog PIR sensor and an object detection system using machine learning. The analog PIR sensor provides an output based on various voltage scales within a certain area rather than producing binary outputs using a threshold value. From samples of an analog signal attained by using an analog PIR sensor, a Fast Fourier Transform (FFT) processed frequency is produced and used as a feature vector of the Artificial Convolutional Neural Network (CNN). The artificial CNN then studies the signal patterns of human motion and animal motion and detects whether it is a human or animal that intruded.**

Keywords—machine learning; object detection system; PIR sensor

I. Introduction

A Pyroelectric Infra-Red (PIR) sensor uses the pyroelectric effect of electromotive forces that occur when it absorbs infrared rays and polarization changes result in electronic charges abandoned. The PIR sensor then detects an object that has a temperature differential with surrounding environments and produces signals. Using these signals, the sensor can detect human or animal motion [1].

In security systems, there are a number of ray detection machines developed that prevent intrusion and give an alarm using a PIR based motion sensor [1][2]. However, the PIR based motion detection sensor that works with a temperature differential between objects and surrounding environments is highly sensitive when the object moves closer to the sensor; hence, it is radically less sensitive when the object is close enough to warm up the surrounding environment, which presents a problem [3]. Therefore, during the summer, when the ambient temperature is closer to the temperature of the human body, the sensor will experience more problems than would occur in the winter. In addition, even if a human body is moving slowly or has a cover to block the heat, the sensor has a propensity to decrease in sensitivity. For example, if a person is holding an umbrella or wearing a raincoat, his or her umbrella or raincoat blocks the heat generated by their body, and the PIR sensor has difficulty detecting motion. Sunlight has various rays that the PIR sensor can detect and which have motion; therefore, the sensor will have problems if the sunlight touches it.

The existing intrusion prevention system detects motions based on the threshold values of the PIR sensors. If the digital logic values go above the fixed threshold value, they produce a HIGH, which is considered an intrusion; otherwise they produce a LOW, which is not considered an intrusion. The threshold values vary by objects and situations; therefore, they can only identify whether there is an object [3].

This research suggests a new form of PIR sensor and machine learning based object detection algorithm to identify a human and an object. First, the paper describes the extraction of a range of signals attained from PIR sensors and the technique by which PIR data are processed into signals and frequency components of the signals are extracted into feature vectors. This research is still underway; therefore, this paper only covers the learning method through an Artificial Convolutional Neural Network (CNN) that is a machine learning algorithm and the design of a classification method after the learning.

II. Gathering sensor and CNN layers

In this section, we describe the signal acquisition process using PIR and the operation process of CNN.

A. Gathering sensor data

To collect infrared lights in pyroelectric devices, a Fresnel lens was used. The advantage of using a Fresnel lens is that owing to its extremely thin construction it behaves exactly like a convex lens.

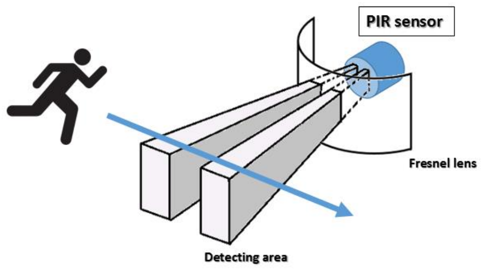


Figure 1. Concept of recognizing an object using PIR sensor and Fresnel lens[5]

The infrared lights detection area is separated into two parts, as seen in Figure 1, which are the detection area and non-detection area. The detection area describes a distance wherein a human can be detected, while the non-detection area describes a distance wherein a human cannot be detected; however, the movement of an object smaller than a human, which has a heat source higher in temperature than that of a human's body temperature, can be detected. Generally, the detection distance of an infrared detection sensor means the distance to the detection area [4]. However, when the infrared detection system is being installed, the non-detection area should be considered.

B. *Convolutional Neural Network (CNN)*

A Convolutional Neural Network (CNN) that was firstly proposed by Yann Lecun in 1998 is an artificial neural network modeling the training process for recognizing cursive writing [6]. CNN has contributed to simplifying the complex calculation structure of existing Multi-Layer Perceptron (MLP) by adding a convolutional layer. Henceforth, the CNN has been studied in many researches by proving its high performance for studying image [7].

1) *Convolutional Layer*

The big difference between MLP and CNN is the convolutional layer. For the existing MLP, every neuron is connected to those of next layer when every layer is transferred to the next one. For example, assume that we perform training having an image as inputs. When the size of input image is 32x32x3 (32 wide, 32 high, 3 color channels), the number of neuron of first input layer becomes 3,072 by 32x32x3 = 3,072. The number of neuron of second hidden layer is 120,000 by 200x200x3 = 120,000. If every neuron of input and hidden layer is connected then the total becomes 368,640,000 by 3,072x120,000 = 368,640,000. Accordingly, every layer has a quite complex structure, which requires huge amount of calculation. In other words, investigating every pixel when recognizing image is not possible and even a waste of time. Recognizing an image requires a method of extracting features of various pixels. The convolutional layer is used in this method. The calculation of convolutional layer is shown in Figure 2.

2) *Pooling Layer*

A pooling layer takes the role of reducing width, height and size, which in turn reduces the amount of calculation of neural network and the number of parameter and controls the overfitting.

The left side of Figure 3 shows the result volume of pooling layer. It shows that 224*224*64 size volume decreased to 112*112*64 volume. The right side of Figure shows the example of Max Pooling process that results in the maximum value in every area. If the result value has the average value of filters, then it is called an average pooling.

3) *Fully Connected Layer*

A fully connected layer has the same structure of MLP that was previously explained. Every neurons of each layer is connected and the results go through the activation function; therefore it has the same structure with the one of MLP.

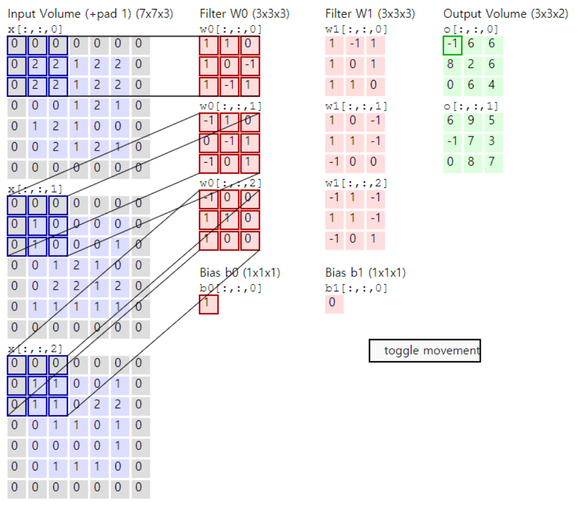


Figure 2. Calculation process of convolutional layer

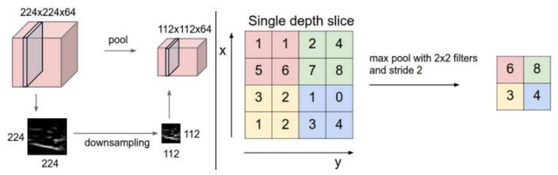


Figure 3. Example of Pooling Layer

4) *ReLU Function*

$f(x) = \tanh(x)$ and $f(x) = (1 + e^{-x})^{-1}$ have been widely used after passing through the neural network. However, looking at the function from the calculation of training process using gradient descent, the training process is quite slow because of nonlinear aspect. Therefore, to improve the calculating speed, $f(x) = \max(0, x)$ is used. Rectified Linear Units (ReLU) has much faster calculating speed than hyper tangent or sigmoid function which were referred when training Deep Convolutional Neural Networks (DCNN).

III. Suggested PIR Sensor-Based Human and Object Detection System

In this section, we propose a PIR sensor based human and object detection system.

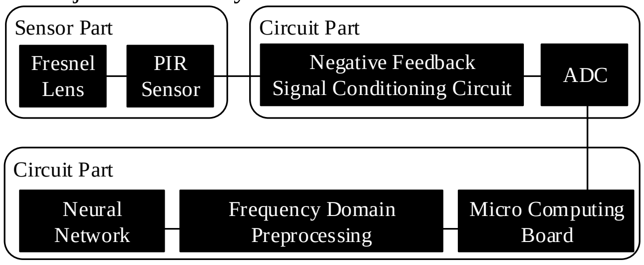


Figure 4. Construction of PIR sensor-based human and object detection system

Figure 4 is used to represent the entire data processing part. The overall data processing structure is divided into a sensor part, circuit part, and processor part. The sensor part has a role in extracting signals from an external stimulus. The circuit part amplifies and transforms the signals that came from the sensor part up to the dynamic range. Additionally, it transforms the analog signals into digital signals so that they can be processed in the processor part. The processing part transforms the signals of the digital time area that came from the circuit part into signals of the frequency area so that they can be easily classified into machine learning class. The signals transformed into the frequency area go through the artificial CNN and are classified by models that study newly given inputs.

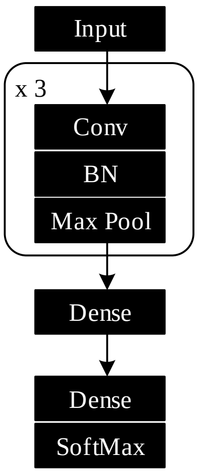


Figure 5. CNN based human and object detection algorithm

This research uses a five-layered structure that consists of input and output layers and three CONV LAYERs, while each input layer uses its own feature information. The output layers consist of two nodes that are the criteria for distinguishing between a human and a pet. Figure 5 describes the structure of the artificial CNN that was used.

IV. Implementation

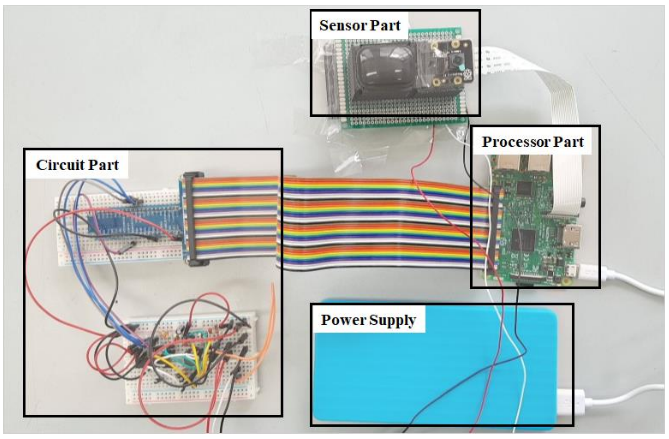


Figure 6. Object detection part of the realized PIR sensor-based human and object detection system

The circuit part of the suggested system amplifies the signal by 69 dB through a non-inverting two-stage amplifier. This filtered the low and high frequency noise that could have led to incorrect detection. Thus, the direct current components were removed. After confirming that the input signals in the analog-to-digital converter (ADC) were ranging from 0 to 3020, the reference voltage was set to 0.37 times the maximum voltage.

An LHI-878 was used as a PIR sensor in the sensor part and a PD23-6020 was used for the Fresnel lens. A Raspberry Pi 3 Model B was used in the processor part. To check object detection, a Raspberry Pi NoIR Camera V3 module was connected to the process. Figure 6 shows the actual equipment of proposed system.

The software development tool of machine learning used keras library-based deep learning studio, CPU used Intel®Core(i)i5-6600 and GPU used Geforce GTX 1050 Ti.

V. Experiments and Evaluation

We had conducted experiments based on several factors that affect the amount of ambient infrared that PIR sensor detects.

In Figure 7, the *t* represents the number of samples and the sample rate is equal to 14.5 ksps; f(t) describes values ranging from 0 to 5 v in quantization rate of 3020. Data from the PIR sensor were collected when a human was at a distance of 1 m, 2 m, and 5 m and in different positions. Additionally, data were collected when a dog at a distance of 2 m was in motion. These data were Fast Fourier Transform (FFT) processed to detect whether the object was human or animal using an artificial CNN.

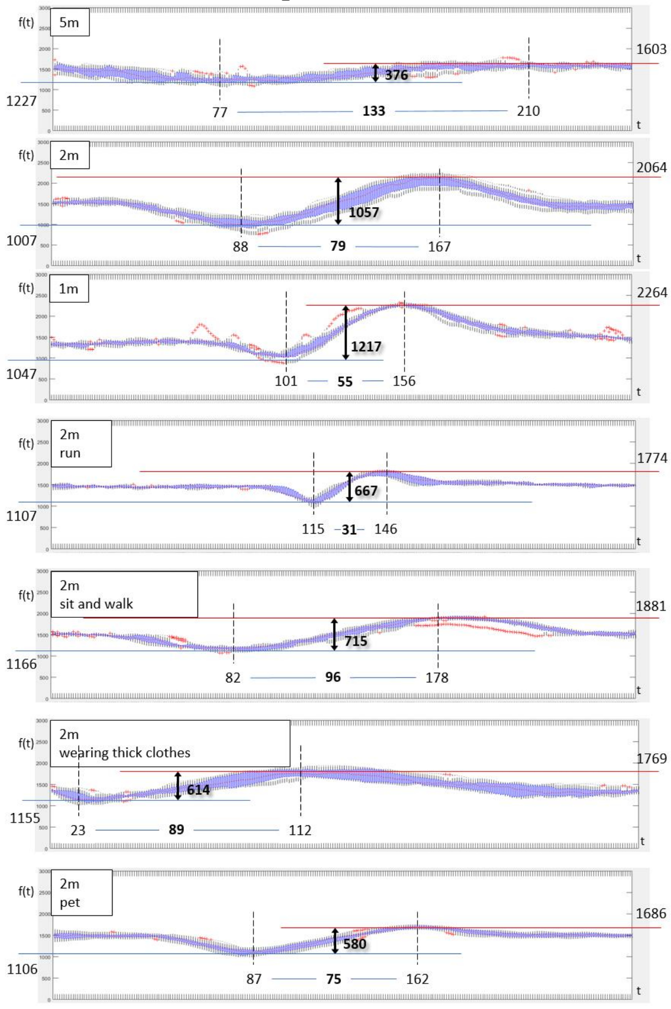


Figure 7. Distribution of PIR sensor values depending on distance of human and animal

TABLE I. TRAINING, VALIDATION, TEST DATA CONDITION

	Group	Human	Nothing
Training	A group	9,000	9,000
Validation		3,000	3,000
Test	B group	3,000	3,000
Total	-	15,000	15,000

By using the developed Object Detection System, data were collected by distinguishing between when there being a person and when there being none. The acquisition cycle was 1 second and was labeled with 1000 data per second. Table 1 shows the dataset configuration.

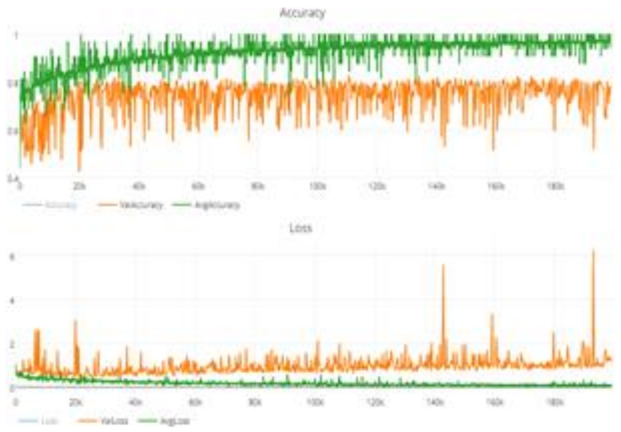


Figure 8. Accuracy and loss rate distribution toward each step

The accuracy and the loss rate were identified by distinguishing between when there being a person and when there being none toward the training, validation and test.

Training accuracy was 97.62%, validation accuracy was 80.5% and test accuracy was 72.8%. Figure 8 shows the accuracy and loss rate distribution toward the training, validation and test.

VI. CONCLUSION

This study designed a PIR signal process and CNN based learning algorithm using analog signals to improve a PIR sensor-based intrusion detection system. The signal processing algorithm was realized and used in experiments to distinguish between a human and an object in various situations.

In the future, this research will transform signals of a certain time range into a frequency range to express a certain frequency component of a human or an object as data. Thereafter, using this data as a parameter of the machine learning algorithm, the accuracy of an object detection system using a PIR sensor will be improved.

ACKNOWLEDGMENT

"This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1A2B4013150)."

* : Corresponding Author

REFERENCES

[1] P. Shpter, “Passive infrared motion detector and method”, US Patent 6, 215,399, November 1997

[2] S. Soliman and M. Srinath, “Continuous and Discrete Signals and System”, Prentice Hall, 1998.

[3] P. Shpter, “Infrared motion detection signal sampler”, US Patent 6, 111,256, April 1997

[4] J. H. Park, H. G. Yeom, B. G. Jung, I. H. Jang and K. B. Sim, “Soundsource Localization and Tracking System of Intruder for Intelligent Surveillance System”, Journal of Korean Institute of Intelligent Systems, Vol.17, No.6, pp.786-791, December 2007.

[5] F. Nelli, “PIR motion detector - a sensor for Arduino and Raspberry Pi”, http://www.meccanismocomplesso.org/en/pir-motion-detector/, February 2016.

[6] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, Vol. 86, No. 11, pp. 2278-2324, November 1998.

[7] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proc. 27th International Conference on Machine Learning, 2010.

Evaluation of IoT Device Management Tools

Biliyaminu Umar[1], Hamdan Hejazi[1], László Lengyel[1], Károly Farkas[1,2]
[1]Budapest University of Technology and Economics,
[2]NETvisor Ltd
email: biliyaminu.umar@edu.bme.hu, hamdan.hejazi1@gmail.com, lengyel@aut.bme.hu, farkask@hit.bme.hu

***Abstract*— Industry 4.0 with Internet of Things (IoT) is the next wave in technology revolution, which is expected to change our everyday life. This digitalization is having great impact on all the domains (energy, healthcare, transportation, manufacturing etc.) in addition to the Information and Communication Technologies (ICT) sector. In IoT scenarios, numerous sensors measure and report several phenomena and diversified IoT solutions are deployed to collect huge amount of data. IoT platforms, such as Amazon AWS, IBM Watson or Microsoft IoT Suite, have been available to aid the development of such services/applications. However, one of the major challenges faced by IoT solutions providers is the supervision and management of the large number of deployed sensors/devices. Presumably, the magnitude and heterogeneity of the IoT systems makes it difficult to manage them with conventional IT management tools and techniques. New techniques and tools have to be explored and developed or the traditional management solutions have to be adapted to the new challenges. In this paper, we identify and formulate the essential challenges of IoT device management and supervision, review the actual state-of-the-art IoT device management and supervision techniques and tools available on the market, and briefly evaluate their features and typical use cases.**

Keywords- Internet of Things; Device Management; Platforms; Sensors.

I. INTRODUCTION

Internet of Things (IoT) enables numerous devices around the world to communicate and transfer data collected from different environments to the IoT platforms. According to Cisco, 25 - 50 billion ‘things’ will be connected to the Internet by the year 2020 [1]. This aggressive growth of emerging smart devices connected to the Internet infrastructure poses one of the most challenging tasks in the IoT space. IoT management tools need to provide solutions to meet the requirements of connectivity, heterogeneity, security, scalability and data handling [2].

The global relevance of IoT and its application to several domains, such as home and industrial automation, intelligent energy management, automotive applications, healthcare, works of life, brings in another dimension of heterogeneity as these diverse applications use a plethora of things (sensors, actuators, devices) to communicate via the Internet [3]. However, the lack of a unified approach of handling heterogeneous devices from several vendors presents a major challenge in IoT device management. Several solutions using different techniques, such as Lightweight Machine to Machine (LwM2M) [4], which manages devices remotely, have been proposed to solve these shortcomings. Unfortunately, these approaches are limited only to devices that have enough resources to implement the required management protocols and to connect directly to the Internet [5][6]. SNMP [7] and NETCONF [8] standards have also been used in monitoring IoT devices, but the heterogeneous nature often leads to waste of resources and inefficiency.

Finding an appropriate IoT management tool from the available options for a given field of application is a challenge a customer faces. Although the functionality and the performance provided by the tools are similar, their techniques and implementations are quite different. Thus, a comprehensive analysis of requirements and possible solutions is necessary to facilitate the tool selection process. In this paper, we identify and formulate the essential challenges of IoT device management and supervision, review the actual state-of-the-art IoT device management and supervision techniques and tools available on the market, and briefly evaluate their features and typical use cases.

The rest of the paper is organised as follows. Section II introduces the basics of IoT system architecture and IoT device management challenges. Section III discusses the requirements and our evaluation benchmark for comparing the management tools. The selected and investigated IoT management tools are introduced in Section IV and compared in Section V. Finally, Section VI draws the conclusions.

II. BACKGROUND

To effectively identify and evaluate the existing solutions in IoT device management, it is imperative to clearly understand the structure and challenges faced in the IoT systems. In this section, the generic architecture of IoT systems and its common challenges with regards to device management are introduced.

A. IoT System Architecture

IoT systems consist of numerous devices, such as smartphones, temperature sensors, actuators, connected in various environments. These sensors, devices, gateways are connected via communication networks to cloud services and applications. These things could be surrounded or distributed by long distances in different environments but controlled and managed centrally in the cloud, thus named cloud computing. On the other hand, a decentralized solution known as edge/fog computing is an alternative to be realized when processing is required to be carried out closer to the source of the data to improve the quality of service provided [9].

To understand the IoT system architecture, identifying and investigating its logical layering can help. In this paper, the fundamental blocks of the IoT system architecture are presented as layers and every layer forms an interesting field of research. These layers are: Sensing layer, Communication layer, Cloud layer, Management layer, and Services and Applications layer in Figure 1. The Sensing layer consists of sensors, actuators and smart devices that collect the data from surrounding environment. The Communication layer provides a means of transferring the collected data to the cloud, or the application layer. The Cloud layer aggregates the data for processing and storage, and makes it available for use to services and applications. Management data are separated from service data and collected in the management system from proper operation and administration of the entire IoT system. The Services and Applications layer presents the output data as services, applications and features offered to the end-user depending on the use-case.

The IoT communication protocols in the Communication layer and the low latency computing in the Cloud layer in addition to the provided Quality of Service (QoS) and management tools of the system determine the strengths and weaknesses of the IoT platforms and system architectures.

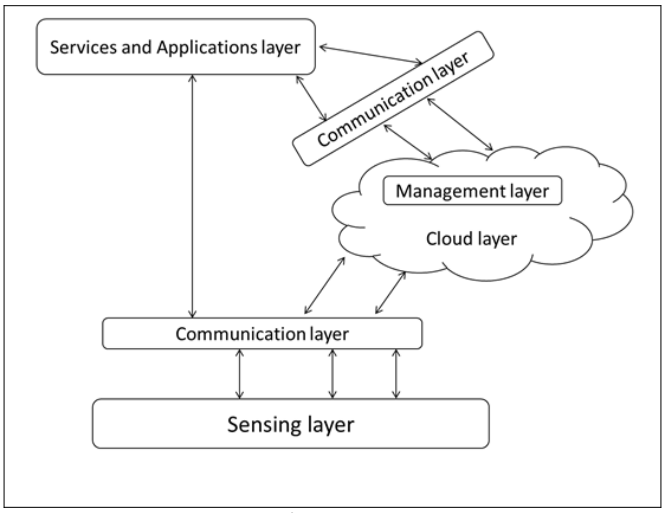


Figure 1. Layers of the IoT System Architecture.

1) Sensing layer: The main function of the Sensing layer is to detect changes in the physical status of the connected things in real-time. It includes sensors, which are the main components of this layer. The task of the sensor is to measure the physical environment, identify and localize the smart objects, collect the data and send them to the Cloud layer for processing and storage. The actuators in this layer are usually mechanical devices, such as switches, that execute the desired actions in response to changes [10].

2) Communication layer: The Communication layer is responsible for interaction between the layers of the IoT architecture. It transfers the data collected in the Sensing layer to the Cloud or the Services and Applications layer directly. It includes routers, switches and gateways, which are connected to devices that cannot connect directly to the cloud. Protocols, such as Constrained Application Protocol (CoAP) [11], Message Queuing Telemetry Transport (MQTT) [12] and Lightweight Machine to Machine (LwM2M) connect various IoT devices to send data to upper layers [13].

3) Cloud layer: It is also known as the processing unit of the IoT system. The collected data from sensors and devices are ingested in the Cloud layer. Its tasks are storing, processing, and analyzing data. In general, the cloud employs a data centre as a central server to process data generated by the edge devices. There is ongoing research on next generation cloud computing to decentralize some of the processing tasks from the cloud to edge nodes to improve computation performance [14].

4) Management layer: It is responsible for monitoring and operating all other layers, providing the features for the management tools usually implemented in the cloud.

5) Services and Applications layer: The Services and Applications layer provides the applications and a variety of the services, such as data collection, data analytics, data visualization and security. They depend on the use cases and desired functionalities provided to the end users.

B. IoT Device Management Challenges

Consequent to the accelerated evolution in IoT, service providers encounter several challenges in satisfying the management requirements. These challenges include the following ones.

1) Connectivity of Heterogeneous IoT Devices: The IoT paradigm requires widespread connectivity of billions of heterogeneous devices. This heterogenity in connectivity is considered as a significant challenge to the interoperability of protocols and solutions developed by different vendors [15]. The accessibility from anywhere can be achieved via the Internet, either by gateways or direct connection and opens the IoT system to a large environment of products and services. Moreover, remote control, which enables the management, monitoring and control of devices, is of high significance to the solutions. This will further lower operational costs by collecting data and implementing maintanance remotely [16]. The IoT system architecture is designed for use in different physical environments. Thus it requires the capability to handle many heterogeneous devices. Wherefore, a considerable concern within developing IoT solutions is handling the interaction with heterogeneous IoT devices [17].

2) Device Management Challenges: Device management is one of the most significant features expected from any IoT management tool. Retaining the device information, status and logs is important. Provision of detailed reports and information about the device level statistics is desired for numerous things [18]. As IoT devices scale to billions, the current centralized network mangement model could present bottlenecks. In an IoT system, the device integration support is required because some tasks or requirements can be done by implementing one service, while other tasks will be executed via the integration of several services [19].

3) Security Limitations: Security is a critical challenge in IoT systems because of the consequences of security breaches, such as financial and credibility losses. For instance, hackers often target the edge devices of the IoT system, which are considered as entry points [20]. Efficient IoT systems with billions of devices connected should have protection, detection mechanisms and secure procedures in case of unusual events and anticipate vulnerabilities [17]. The integration of IoT into our life extend the security concerns from information and assets to human life and health. With the rate at which technology is growing, vendors could focus more on functionality rather than security. Therefore, IoT management tools need to implement alternative techniques to handle different issues while using the identification and authentication for multiple types of IoT communication protocols used for data communication and transfer. These need to be encrypted and secured with a robust encryption algorithm to prevent possible risks [21].

4) Next Generation of IoT Management Tools: The accelerated development of IoT is impacting various scientific areas, thus inducing many trends in the next generation of IoT systems. Changing infrastructure is one of these trends because the centralized computing prototype is impressionable to single point of failures and large data centres consume huge amount of energy to keep them operating [14]. Alternate technologies being developed to reduce failures in the cloud include multi-cloud, micro cloud and cloudlet, ad hoc cloud and heterogeneous cloud [9]. In addition, minimizing the workloads for low-latency and resource processing has been a considerable challenge for cloud computing [22]. The new trend known as edge/fog computing brings processing closer to the data source [23], and the management tool is required to suit these changes and subsequently scale with the architecture and devices.

III. EVALUATION BENCHMARK

Based on the highlighted challenges, we draw a requirements/features table to serve as a benchmark for comparing the management tools reviewed in this paper (Table II). Therefore, performance and relevance of a tool have been evaluated by investigating and comparing the following requirements/features.

- *Device Management:* This is one of the most important features expected from any management tool. The tool should maintain a list of connected devices and track their operation status; it should be able to handle configuration, firmware (or any other software) updates and provide device-level error handling and reporting [18].
- *Protocols Supported:* Things require a direct communication path to the platform in both the forward and reverse direction for information exchange and sending commands. Thus, a management tool should support application and management protocols that the device can work with to exhibit a 'device agnostic' property. Some widely used application protocols include MQTT, CoAP, REpresentational State Transfer (REST) and eXtensible Messaging and Presence Protocol (XMPP). Other Protocols, such as LwM2M and Open Mobile Alliance - Device Management (OMA-DM) are classified as management protocols [24].
- *Product Lifecycle Management:* This involves the management of a device from installation and commissioning till its decommissioning. During the lifetime of this thing, it is necessary to make some software/firmware updates to implement new features, remove bugs and fix security vulnerabilities [25]. Thus, it is a major challenge in IoT, based on its scale of millions of devices, to individually perform these important tasks. Over-the-Air (OTA) upgrades, downgrades and option of force updates for super critical firmware are expected features of the management system.
- *Troubleshooting and Maintenance:* Diagnostics features are required in the operation of IoT devices [26]. The tool should also allow the sending of custom and system level commands to a device, such as reboot or factory reset.
- *Security and Access Control:* The security measures required for IoT systems are higher than those of general software and applications [27]. The connection of millions of devices to a network increases the vulnerabilities proportionally. Since the devices are low cost and low power, these security requirements need to be met from the platform end of the management system in the form of message-level security and data encryption [28].
- *Localisation and Mapping:* Location support is essential especially when a device's location is not static rather dynamic. The continuous tracking of the location will thus help generate the historic location view. In some applications, GPS locations or network triangulation is necessary for fleet management and asset tracking solutions [24].
- *Scalability:* This is one of the most important non-functional features [24]. As most of the management systems are web applications, it is expected to be highly scalable to the order of millions of things. Support auto scaling feature could also be included by the application developers, so a scalability magnitude could also be defined for customers to provide some limit.
- *Device Monitoring:* Tools that can provide device monitoring and performance data visualizations are also very helpful in supervising the network of things. Alarm indications to provide alerts in case of faults and critical events should be embedded into the tool for easy and efficient monitoring of the whole network [26].
- *Integration:* Provision of standard/open APIs for integration has high importance in a management tool. As most vendors already have an existing enterprise platform, the seamless integration of a management tool via a standard API will make the operations and management much easier. The importance of the interoperability of IoT management tools cannot be over-emphasized as this is the source of a platform/device agnostic management system [28].

IV. IoT DEVICE MANAGEMENT TOOLS

In this study, we have selected a variety of tools on the market that have the potential to play an essential role in monitoring smart things in the IoT solutions. These tools were selected because they are suitable stand-alone IoT device management tools with extensive implementation in several industrial use cases. We shortly describe the selected tools in this section, while a summary highlighting their key features and example use cases is shown in Table I.

A. *Xively CPM*

Xively Connected Product Management (CPM) is a tool that offers solutions for enterprises building connected products and services. Moreover, it enables companies to easily build and manage IoT security, connected devices and products including home automation, and capturing their IoT data. It provides a simple and scalable platform enriched with tools necessary to connect, manage and engage things. It has standard APIs for integrating data with primary enterprise systems, such as Customer Relationship Management (CRM) [29][30].

B. *DevicePilot*

DevicePilot implements locating, monitoring and managing connected devices at scale. It is completely agnostic, providing platform connectivity to any device, and easily integrates with IoT platforms. It is a cloud-based application, which scales with the deployed infrastructure, schemaless and provides all functionalities via a REST API [31].

C. *Wind River HDC*

Wind River Helix Device Cloud (HDC) is a tool that helps reduce the complexities of building and managing large-scale IoT deployments. It enables device health monitoring, bi-directional file transfer, remote access to help service engineers detect and diagnose problems before they impact critical data collection. HDC provides tools one needs for deploying, monitoring, servicing, updating, and decommissioning IoT devices [32].

D. *QuickLink IoT*

QuickLink is a resource efficient device management solution based on LwM2M and OMA-DM standards. It supports device provisioning, configuration, diagnostics management and over-the-air updates. It has a plug-in API architecture with encrypted data collection using CoAP with Transport Layer Security (TLS) [33].

E. *ThingWorx Utilities*

ThingWorx Utilities is a set of tools, rich in features that enable and support the rapid deployment and adoption of powerful IoT applications. It provides device management capabilities for day to day management of the connected devices and includes utilities to provision, remotely monitor and update the connected devices and assets. With its standard framework, it is also possible to integrate new IoT applications into existing business systems [34].

F. *Particle*

Particle is a full-stack IoT device management platform that provides all the necessary tools to securely and reliably connect IoT devices to the web/cloud. The solution can be used on different scales of deployment from large enterprises to innovative start-ups and everyone in between. It is secured by using encrypted communication protocols, easy to use and provides an interface to see devices, push software updates, and make changes and improvements on an ongoing basis. It offers several development tools, such as Web IDE, Desktop IDE and a Command Line Interface (CLI). The device management console can manage team permissions from a single administrative interface. Support for cross-vendor devices is limited and continuously developed [35].

G. *Losant Helm*

Losant Helm is a fully integrated IoT device management and connectivity tool directly embedded in the Losant IoT platform, an enterprise-ready cloud platform that enables developers to easily make use of real-time data by rapidly developing smart, connected solutions for IoT. It serves as a control hub for connected production facilities and its hardware-agnostic platform is easily integrated with a broad variety of sensors, controllers, machines, and device gateways. This enables many-to-many interoperability across disparate systems and technologies. Its open communication standards (REST, MQTT) provide simple connectivity to millions of devices [36].

H. *DataV IoT Device Management*

This tool makes equipment and device management a priority as industrial companies connect more business-crucial assets together with IoT. It gives the power to manage the full lifecycle of all assets from a centralized location, including configuration, inventory, and OTA software updates and configuration [37].

V. COMPARISON OF MANAGEMENT TOOLS

Today, none of the selected and evaluated tools claims to support all the features we used in the benchmark. Interestingly, all of them support the basic features of device management, remote monitoring, product lifecycle, scalability and integration to IoT platforms. The protocols supported, localization and mapping, troubleshooting and maintenance, security and access control features are available in a limited number of these tools.

DevicePilot stands out as the star performer from this study because it supports more features than the other tools we have evaluated. Localisation of devices, access control of the connected things and support to REST protocols are its added features. Its only drawback is the lack of maintenance and troubleshooting function.

QuickLink IoT follows closely with troubleshooting and maintenance features with OTA updates added to the basic functionalities. It also supports LwM2M and OMA-DM device management protocols. It does not support mapping of devices and the security features are limited.

Particle, Losant and Wind River HDC have very good maintenance features with remote diagnostics and updates but lack localization and access control functionalities. Particle supports CoAP, MQTT and its proprietary Particle subscribe protocols. Losant integrates remote management with audit and log files from devices. HDC manages devices via MQTT protocol with security extensions.

Xively is also a very good management tool with robust security features and support to MQTT, REST and HTTP protocols. Unavailability of localization, mapping and troubleshooting and maintenance of devices are its major drawbacks.

ThingWorx Utilities and DataV both integrate well with IoT platforms. While ThingWorx supports protocols such as MQTT, CoAP and XMPP, DataV provides limited support to standard protocols. Both tools lack localization and access control features, but DataV supports troubleshooting and error log management. None of the reviewed tools fully supports all IoT-related protocols.

TABLE I. KEY FEATURES AND TYPICAL USE CASES OF THE EVALUATED IOT MANAGEMENT TOOLS

Tool	Vendor	Key Features	Typical Use Cases
XIVELY CPM [29]	LogMeIn Inc.	Device agnostic connectivity (MQTT, REST and HTTP protocols), scalability, security and IoT platform integration	Agriculture, energy management and DNA research improvement
DEVICEPILOT [31]	DevicePilot	Device management, security, scalability, mapping, real-time monitoring and easy integration	Energy management, construction, healthcare and smart cities
WIND RIVER HDC [32]	Wind River	Thing management via MQTT with security, device health monitoring, remote diagnostics and software upgrade	Smart homes, healthcare, industrial, automotive and energy management
QUICKLINK IOT [33]	SmithMicro Software	LwM2M and OMA-DM supported device management, securty, diagnostics and OTA updates	Asset management, smart monitoring, connected cars and smart cities
THINGWORX UTILITIES [34]	ThingWorx	Device management using MQTT, XMPP or CoAP, remote control and monitoring, product lifecycle and IoT platform integration	Manufacturing, healthcare, transportation and utilities
PARTICLE [35]	Particle.io	Connectivity, OTA updates, security, IoT platform integration, remote diagnostics, monitoring, reports and alerts. It supports MQTT, CoAP and Particle subscribe	Smart homes, environment monitoring, infrastructure and supply chain management
LOSANT HELM [36]	Losant	Remote provisioning, agnostic management, audits and logs, 3rd party IoT platform integration	Manufacturing, logistics and retail management
DATAV IOT DEVICE MANAGEMENT [37]	BSquare	Device health monitoring, device and error logs, real-time monitoring, performance issue resolution and IoT/enterprise platform integration	Smart metering, intelligent vending, fleet management and transportation

TABLE II. COMPARISON OF THE EVALUATED IOT MANAGEMENT TOOLS/PLATFORMS
(LEGEND: ● – SUPPORTED; ○ – NOT SUPPORTED; ◐ – PARTIALLY SUPPORTED)

Tool	Device Management	Protocols Supported	Product Lifecycle Management	Trouble-shooting and Maintenance	Security and Access Control	Localisation and Mapping	Scalability (to millions)	Device Monitoring	Integration
XIVELY	●	◐	●	○	●	○	●	●	●
DEVICE PILOT	●	◐	●	○	●	●	●	●	●
WIND RIVER HDC	●	◐	●	●	◐	○	●	●	●
QUICKLINK IOT	●	◐	●	●	◐	○	●	●	●
THINGWORX UTILITIES	●	◐	●	○	◐	○	●	●	●
PARTICLE	●	◐	●	●	◐	○	●	●	●
LOSANT HELM	●	◐	●	●	◐	○	●	●	●

DATAV IOT DEVICE MANAGEMENT	●	◐	●	●	◐	○	●	●	●

Table II compares the eight selected and evaluated IoT management tools/platforms taking into consideration that due to their continuous development some requirements will be met by the products in the nearest future.

VI. CONCLUSIONS

The current growth trends adumbrate that IoT will gain higher and higher importance in several industries in the coming years. This expands its influence on the interaction between man and technology, and the role of a functional and robust management system is getting more importance.

This paper presents the basic and fundamental requirements of an IoT management and supervision solution based on the generalized architecture of an IoT implementation. Using these requirements as a benchmark, we have selected, evaluated and compared eight industrial IoT management tools. Unfortunately, the complex structure of IoT implementations due to their numerous applications, heterogeneous devices and diverse use cases makes it challenging to come up with a generic 'one for all' management tool. However, our comparison matrix, given in Table II, can help IoT solution providers choose the most appropriate management tool for their target system assuming a good understanding of the requirements. In future, we plan to develop/extend IT management tools to meet the needs of the IoT ecosystem.

ACKNOWLEDGMENTS

The work presented in this paper has been carried out in the frame of project no. 2017-1.3.1-VKE-2017-00042, which has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2017-1.3. funding scheme.

This work was performed also in the frame of FIEK_16-1-2016-0007 project, implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the FIEK_16 funding scheme.

REFERENCES

[1] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," CISCO IBSG, 2011.

[2] C. Pham, Y. Lim, and Y. Tan, "Management architecture for heterogeneous IoT devices in home network," 2016 IEEE 5th Glob. Conf. Consum. Electron. GCCE 2016, 2016, pp 1-5.

[3] Y. Jie, J. Y. Pei, L. Jun, G. Yun, and X. Wei, "Smart Home System Based on IoT Technologies," in 2013 International Conference on Computational and Information Sciences, 2013, pp. 1789–1791.

[4] "Lightweight machine to machine technical specification," OMA-TS-LightweightM2M-V1_0-20170208-A, Open Mobile Alliance, Feb 2017.

[5] S. Rao, D. Chendanda, C. Deshpande, and V. Lakkundi, "Implementing LWM2M in constrained IoT devices," in 2015 IEEE Conference on Wireless Sensors (ICWiSe), 2015, pp. 52–57.

[6] M. Ersue, D. Romascanu, and J. Schoenwaelder, "Management of Networks with Constrained Devices: Problem Statement and Requirements," RFC 7547, May 2015

[7] D. Harrington, R. Preshun, and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," RFC 3411, December 2002.

[8] R. Enns, M. Bjorklund, and A. Bierman, "Network Configuration Protocol (NETCONF)," RFC 6241, June 2011.

[9] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick and D. S. Nikolopoulos, "Challenges and Opportunities in Edge Computing," 2016 IEEE International Conference on Smart Cloud (SmartCloud)," New York, NY, 2016, pp. 20-26.

[10] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," IEEE Commun. Surv. Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.

[11] J. Guth, U. Breitenbücher, M. Falkenthal, F. Leymann, and L. Reinfurt, "Comparison of IoT platform architectures: A field study based on a reference architecture," 2016 Cloudification of the Internet of Things (CIoT), 2016, pp. 1-6.

[12] C. Bormann et al, "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets," RFC 8323, 2018.

[13] OASIS.org, "MQTT version 3.1.1," OASIS Standard, October 2014, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, [retrieved: June 2018].

[14] B. Varghese, and R. Buyya, "Next Generation Cloud Computing: New Trends and Research Directions," Future Generation Computer Systems, 2017, pp. 1–22.

[15] M. Elkhodr, S. Shahrestani, and H. Cheung, "The Internet of Things : New Interoperability, Management and Security Challenges," International Journal of Network Security & Its Applications, vol. 8, 2016, pp. 82-102.

[16] "Remote Monitoring and Maintenance: Mission-Critical Operations at the Competitive Edge," Oracle, 2016, [Online], Available: http://www.oracle.com/us/solutions/internetofthings/iot-remote-monitoring-brief-2881653.pdf, [retrieved: May, 2018].

[17] T. Perumal, S. K. Datta, and C. Bonnet, "IoT device management framework for smart home scenarios," 2015 IEEE 4th Glob. Conf. Consum. Electron. GCCE 2015, 2016, pp. 54–55.

[18] V. Gazis et al., "A survey of technologies for the Internet of Things, 11th International Wireless Communications and Mobile Computing Conference (IWCMC), Dubrovnik, 2015, pp. 1090-1095.doi: 10.1109/IWCMC.2015.7289234.

[19] J. Lin et al., "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," IEEE Internet of Things Journal, vol. 4, no. 5, 2017 pp. 1125–1142. doi: 10.1109/JIOT.2017.2683200.

[20] K. Shea, "Device Management in the Internet of Things – Why It Matters and How to Achieve It," WindRiver, 2017, [Online] Available: http://www.new-techeurope.com/2017/06/07/device-management-internet-things-matters-achieve/, [retrieved: May 2018].

[21] Jasper (2014), "Achieving End-to-End Security in the Internet of

Things," [online], Availabe: http://pages.jasper.com/White-Paper-Cellular-IoT-Security_Cellular-IoT-Security.html, [retrieved: May 2018].

[22] P. Varshney and Y. Simmhan, "Demystifying Fog Computing: Characterizing Architectures, Applications and Abstractions," Proc. - 2017 IEEE 1st Int. Conf. Fog Edge Comput. ICFEC 2017, 2017, pp. 115–124.

[23] D. C. Klonoff, "Fog Computing and Edge Computing Architectures for Processing Data from Diabetes Devices Connected to the Medical Internet of Things," J. Diabetes Sci. Technol., vol. 11, no. 4, 2017, pp. 647–652.

[24] P. Ganguly, "Selecting the right IoT cloud platform," 2016 International Conference on Internet of Things and Applications (IOTA), Pune, 2016, pp. 316-320. doi: 10.1109/IOTA.2016.7562744.

[25] B. Moran, M. Meriac, H. Tschofenig, "IoT Firmware Update Architecture," IETF Internet-Draft, [online], Available: https://tools.ietf.org/html/draft-moran-suit-architecture-00, [retrieved: May 2018].

[26] "Internet of Things will require remote monitoring solutions", Opengear, [online], Available: https://opengear.com/articles/internet-things-will-require-remote-monitoring-solutions-order-succeed, [retrieved: May 2018]

[27] L. Atzori, A. Iera, G. Morabito, "The Internet of Things: A survey," Computer networks, 54(15), 2010, pp.2787-2805.

[28] "What makes device management a core compatibility of an IoT platform," Softweb Solutions, [online], Available: https://www.softwebsolutions.com/resources/IoT-device-management-platform.html, [retrieved: May 2018].

[29] "Connected Product Management," Xively, [online], Available: https://www.xively.com/xively-iot-platform/connected-product-management, [retrieved: May 2018].

[30] "Guide to Connected Product Management (CPM)," Xively, [online], Available: https://www.xively.com/resources/guide-to-connected-product-management, [retrieved: May 2018].

[31] "Device Pilot Features," Device Pilot, [online], Available: https://www.devicepilot.com/about/features/, [retrieved: May 2018].

[32] "Wind River Helix Device Cloud," Wind River, [online], Available: https://www.windriver.com/products/helix/device-cloud/, [retrieved: May 2018].

[33] "Overview of QuickLink IoT Services Platform," SmithMicro Software, [online], Available: https://www.smithmicro.com/iot-oem/products/quicklink-iot-services-platform/overview, [retrieved: May 2018].

[34] "Manage Your Industrial IoT with ThingWorx," ThingWorx [online], Available: https://www.ptc.com/en/products/iot/thingworx-platform/manage, [retrieved: May 2018].

[35] "Device Cloud," Particle, [online], Available: https://www.particle.io/products/software/device-cloud, [retrieved: May 2018].

[36] "Data and Device Management," Losant, [online], Available: https://www.losant.com/iot-platform/data-and-device-management, [retrieved: May 2018].

[37] "DataV IoT Device Management, BSquare, [online]. Available: https://www.bsquare.com/iot-device-management/, [retrieved: May 2018].

Data Stream Optimization of Sum of Absolute Differences Algorithm on a Graphics Processing Unit

Tom Pudpai, Tae Kyun Kim, and Charles Liu
Electrical and Computer Engineering, California State University, Los Angeles, California, United States
Email: tompudpai@gmail.com, ttkkus@gmail.com, and cliu@calstatela.edu

***Abstract*— This paper describes the data streaming approaches to performance optimization of the Sum of Absolute Differences (SAD) algorithm on an NVIDIA Graphics Processing Unit (GPU) using the OpenCL programming paradigm. The SAD algorithm forms one of several steps required to implement stereo vision. It creates pixel-based disparity maps from two concurrent images captured by a pair of cameras positioned with a distance in between. The disparity maps can be used to derive depths of objects in the scenes of interest. The massively parallel architecture of a GPU can take advantage of the highly parallelizable SAD algorithm. OpenCL programming framework was chosen to develop the parallel algorithm on the GPU. Performance gains are realized by explicitly mapping data from the slower global memory to the faster shared local memory of the GPU. Local memory is loaded by either a centralized or distributed approach from the OpenCL-defined work-items operating in a workgroup. The resulting performance improvements were discussed based on the architectural features of the GPU and the data streaming approaches used in this research work.**

Keywords - data streaming; Sum of Absolute Differences algorihtm; massive parallel architecture.

I. Introduction

Computer vision is a field of study concerned with extracting information from visual data through computers in a variety of applications, such as robotics, augmented reality, and face detection [1]. Computer vision algorithms typically step through the stages of a vision pipeline. A vision pipeline generally starts from image processing methods to improve results from feature extraction and image analysis. Global and local feature metric extraction form the next stages. Different operations are used on rows or blocks of pixels. In this paper, the SAD algorithm takes place in the local feature metric stage, performing an area operation on the GPU's Single Instruction Multiple Thread (SIMT) architecture.

Advanced Driver Assistance Systems (ADAS) leverage computer vision to increase road safety. One ADAS application is stereo vision, which constructs a three-dimensional image by finding corresponding pixels in image frames from two adjacent cameras [2]. The SAD algorithm is one method to generate the matching costs functions that finds point correspondence in stereo vision. This paper focuses on the use of OpenCL, a generic parallel programming paradigm, to develop the SAD algorithm while utilizing the locality of data reference in the memory hierarchy of a GPU. This research supports the efficient use of restricted memory space in an embedded system for data streaming applications.

Prior research has studied the implementation of the SAD algorithm on different hardware platforms. One study evaluated performance on the FPGA platform with respect to embedded systems [2]. More emphasis was placed on the validity of the algorithm itself, and finding the optimal window size and accuracy over different test image pairs. One of the image pairs was the Venus image sequence, an established stereo vision benchmark that was chosen in this research as well [3]. Another study experimented on the SAD algorithm using an SoPC (System-on-Programmable-Chip) heterogeneous architecture [4]. Their work is similar to ours in that they optimize performance by leveraging on-chip memory and selectively transfer data to off-chip memory. By drawing from the parameters and benchmarks of these previous works, we would like to survey performance speedup of the SAD algorithm on the GPU architecture through optimal data mapping. The rest of this paper is organized as follows. Section II describes the SAD algorithm in relation to computer vision-based ADAS applications. Section III introduces the NVIDIA GPU platform. Section IV describes the OpenCL parallel programming paradigm. Section V described the design and the implementation of the data streaming methodology behind the SAD algorithm implementation and optimization. Section VI described the data streaming approaches. Section VII presents and analyzes the observed results. Section VIII concludes this paper.

II. SAD Algorithm in Advanced Driver Assistance Systems (ADAS)

An ADAS increases driver situational awareness and safety by providing important information to warn the driver of any dangerous events. However, humans are not infallible, and ADAS must eventually advance to take control tasks such as braking or steering, mitigating the errors human drivers make. Eventually, as ADAS applications grow more robust, we can expect fully autonomous vehicles to enter the consumer market.

A variety of sensors enable ADAS applications by providing timely and relevant feedback of the environment. We can roughly categorize these sensors into two categories: time-of-flight and camera [1] (see Figure 1). For front-facing imaging sensors, there are applications available such as lane detection, traffic sign and pedestrian recognition, forward collision warning, and adaptive front-lighting. Imaging sensors that face the rear or side of the vehicle can support ADAS applications, such as parking assistance, rear collision warning, and blind spot detection. Imaging sensors inside the vehicle can even detect occupancy and the alertness of the driver [1]. Detection of vehicles, pedestrians, and traffic signs require substantial computing power. Adding an additional imaging sensor can allow for more accurate and robust detection system by the addition of depth information. The means for extracting depth information from a stereo camera setup, also known as stereo vision. Stereo vision allows 3D information to be extracted from a pair of 2D images taken from adjacent cameras and is an important application of ADAS for vehicles. The fundamental problem with stereo vision analysis is finding the corresponding elements within the image pair. For correct correlation of image pair elements, rectification is required [5]. It ensures that the images are horizontally aligned, allowing for the epipolar curve between each image to be a linear. This means that any algorithm that matches pixels from one image to the next will only need to search horizontally across a row of pixels.

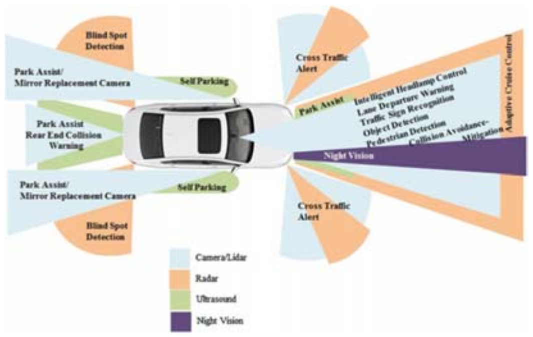


Figure 1. Key applications for ADAS [1].

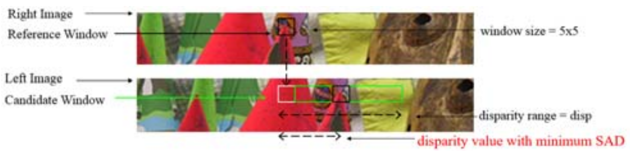


Figure 2. SAD value calculation example; $h \times k = 5 \times 5$, and *disp*=64.

After rectification, each pixel in one image is matched with a pixel in the other image. Then, a disparity map can be generated, indicating the disparity level of each pixel, to be referenced for acquing depth information. The Sum of Differences (SAD) algorithm is the method chosen to calculate a disparity map in this paper. The benefit of this algorithm is computation efficiency, since the calculations involve primarily addition and subtraction operations. The operational form of the SAD addresses window size and the disparity range because area operations are less computationally costly and depth range is physical limited to the distance between the cameras. For instance, if the disparity range is 64 pixels and the window 5x5, a SAD value may be defined as follows:

$$SAD(i,j,disp) = \sum_{h=-2}^{2}\sum_{k=-2}^{2}|P_R(i+h,j+k) - P_L(i+h,j+k+disp)|$$

Where i and j are the indices of the reference pixel in the right and left images, P_R and P_L respectively, *disp* (the disparity range) is the number of candidate windows that are evaluated in the left image, and h and k define the size of the window. Note that the matching pixel is only searched horizontally after image rectification. Thus, the *disp* is only applied in the second dimension of the left image in the SAD calculation. Figure 2 illustrates the SAD value calculations for matching the tip of a red cone between the right and left images. After the 64 SAD values have been calculated for every pixel from coordinate (i, j) to (i, j+*dist*), the disparity level selected is based on the minimum cost function:

$$Disp(i,j) = \operatorname{Arg\,Min}(SAD(i,j,disp)), 0 \leq disp \leq 63$$

Using the Argument Minimum (ArgMin) function, the index of the candidate window with the smallest computed SAD value will be treated as the disparity level for the pixel coordinate (i, j). The disparity range and window size should be scaled based on the parameters of the application where the SAD algorithm is used. The disparity range will depend on the distance between the two cameras, as well as the distance from the camera to the object of interest.

III. NVIDIA GPU PLATFORM

The NVIDIA GeForce 940M graphics card is the primary hardware architecture used to run the SAD algorithm. The GM 108 has three Maxwell Streaming Multiprocessors (SMMs). Figure 3 shows the architecture of an SMM.

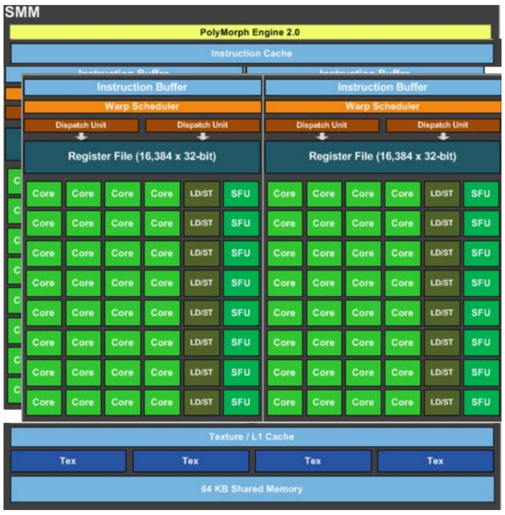


Figure 3. Maxwell Streaming Multiprocessor (SMM) block diagram. (excerpted from [6]).

There are 128 cores in each SMM. Each SMM is partitioned into four separate processing blocks, each with its own instruction buffer, scheduler, and 32 cores, as well as a 16,384 x 32-bit register file [6]. There are two L1/texture caches per SMM that act as coalescing buffers for memory accesses. There is also 64 KB of shared memory that can be programmed and allocated by the programmer. Since it is located on-chip like cache memory, the shared memory can be accessed very quickly. Thus, the explicit streaming of data to shared memory is the focal point of the SAD algorithm optimization for this paper.

IV. Programming Paradigm

OpenCL was used as the Application Program Interface (API) in developing the parallel SAD program on the GPU. It is a heterogeneous programming framework [7]. OpenCL kernels are modeled in a similar manner to Single Program Multiple Threads (SPMT), where parallel threads (i.e., work-items) execute instances of the kernel to map effectively on both scalar and vector hardware. The OpenCL specification can be divided into four models: *Platform model, Execution model, Programming model,* and *Memory model [8].* The *Platform model* specifies that there is one host processor that coordinates execution of kernels, and that there are one or more device processors that actually execute the kernels. Each device is modeled as a group of compute units, which are further divided into processing elements where each element can execute instances of kernels. The *Execution model* defines how the OpenCL environment is configured by the host, and how the host may direct the devices to perform work. The *Programming model* defines how concurrency is mapped to physical hardware. Each unit of concurrent execution is defined as a work-item, which executes the kernel function body. The work-items are indexed in an n-dimensional range, also known as NDRange. To achieve scalability, the work-items of an NDRange can be divided into equally-sized workgroups. Synchronization of work-items is only possible within workgroups (see Figure 4). The workgroup and global work-item size dimensions are specified by the programmer and must be a power of two number. Also, the global work-item size must be evenly divisible by the workgroup size [8].

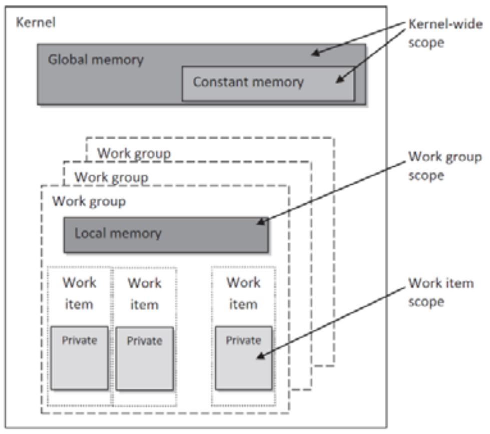


Figure 4. OpenCL Programming and Memory models [5].

The *Memory model* defines memory object types, and the abstract memory hierarchy that kernels use regardless of actual underlying hardware architecture. Memory in OpenCL is divided into host memory and device memory [8]. Device memory is divided into global memory, local memory, private memory, and constant memory (see Figure 4). Global memory can be read from or written to by all work-items running on the device. Data transferred to or from the host will reside in global memory. Reads and writes may be implicitly cached depending on the capabilities of the device [7]. Local memory is shared by work-items in a workgroup only. It is typically mapped to on-chip memory that has shorter latency and higher bandwidth than global memory. Private memory is visible only within a work-item. Constant memory is a region of global memory that remains constant during kernel execution.

The memory model of OpenCL is well suited for NVIDIA GPUs. Each core running a thread, or OpenCL work-item, contains dedicated private memory. All workgroups can communicate through global memory located in off-chip GPU memory. SMMs have dedicated shared memory for communication between work-items in a workgroup, which fits the role of OpenCL's local memory. Accessing this shared memory is fast as long as there are no bank conflicts between threads [9]. Shared memory is divided into equally sized memory banks, which can all be accessed simultaneously. If there are multiple requests to the same bank, the requests become sequential, incurring memory access delays. Therefore, for maximum performance, bank conflicts should be minimized by considering how the memory addresses are mapped.

V. Design and Implementation

In the OpenCL Platform Model, the host sends commands to the device to transfer data between host and device memories, as well as to execute the parallel device code. The host (an Intel Core i5) executes serial code and is typically a CPU. The host is responsible for setting up the execution pathway to and from the device, and requires a lengthy setup process which begins by identifying the platform and device. Memory buffers must be created to link objects on the host to objects in the kernels executed on the device. For this research, memory buffers are needed for the left and right input image values, the output SAD values, the image dimensions, the padded image dimensions (to round up to the closest power of 2 number in each image dimension as explained below), disparity level, window dimensions, work item dimensions, and conditional values.

The device is responsible for execution of the kernel as directed by the host. Initially, the input images, disparity output, and other kernel parameters defined in the previous section are transferred from the host memory and allocated to the global memory of the device. In this paper, the images used are the Venus pair, used in several benchmarks amongst stereo vision researchers. Given the 384 pixels x 434 pixels image size, there are 162,222 Disparity Levels to be calculated based on the SAD algorithm (see Section II). Each one is executed on a work-item. Because of the size restriction by OpenCL, we must round up the image dimensions to the nearest power of 2 in order to process every pixel of the image pair. Thus, the image values of L and R are padded with values of 0 to reach dimensions of 512 x 512. The partitioning of work-items into workgroups is determined by the OpenCL local-item size and global-item size dimensions. In this NVIDIA GeForce 940M GPU architecture, each work-item from OpenCL is operated on a GPU core. Each workgroup is operated on an SMM with 128 cores. Thus, 128 work-items can run in parallel. Since this GPU has 3 SMMs, a total of 3 * 128 = 384 work-items can run in parallel. If the GPU were to run at maximum occupancy, there would be $\lceil 162222/384 \rceil = 423$ iterations of the SAD algorithm needed.

To differentiate and track each work-item, the OpenCL API function get_global_id() is used to return its unique global ID value [8]. This is important because the instances of the kernel operating on SAD values of the image edges must be treated differently. In this paper, without losing the generality of the parallel algorithm, we use a common window size with 5 pixels x 5 pixels for the SAD algorithm for performance analysis. A reference window in an image compares to 64 iterations of candidate windows in the counterpart image. Note that the SAD values on the border cannot be computed because the 5 x 5 windows will be incomplete. Thus, the SAD values cannot be computed for kernels two pixels within each border. In the device kernel code, this padding is implemented through a conditional statement with a reference to the global ID of the kernel to avoid the incomplete calculations of such close-to-border SAD values. Upon finding the minimum SAD value for all 64 candidate windows, the corresponding disparity level must be saved to the disparity map output. The output matrix is stored as an integer array in global memory.

The performance of the SAD algorithm can be first enhanced through loop unrolling. Loop unrolling involves the rewriting of loops into a repeated sequence of similar independent statements. This helps eliminate the loop overhead and also hides stalls due to data dependencies [10]. The original implementation of the SAD algorithm in this paper consists of a nested *for loop* that increments the kernel's SAD value a total of 25 times, one for each pixel in the 5 x 5 window. The disadvantage to this approach is much lengthier code, which is particularly harmful to embedded systems with limited instruction memory.

The other important factor to affect performance is the workgroup size; i.e., the number of work-items defined in a workgroup. In our design, the workgroup size varies from 32 to 512. Based on the feature of the GPU hardware, there are 128 x 3 = 384 cores. 512 is that number's next power of two value. Thus, a workgroup size greater 512 is not considered due to the mismatch to the hardware.

VI. Data Streaming Optimization

The first SAD algorithm in this paper was implemented to access all data from the GPU's global memory. Global memory is visible to all of the Streaming Multiprocessors in the GPU, but is located off-chip, so accesses to global memory incur heavy delays. As addressed in Section III, The OPENCL local memory is mapped to the SMM's shared memory, which is shared by all of the cores in that single SMM, and is stored on-chip (see Figure 3). By taking advantage of this local memory and the mapping scheme for utilizing the spatial and temporal localities of data, significant speedup can be achieved for the SAD algorithm.

Datatype optimizations are possible through OpenCL. As mentioned in Section III, memory copying incurs performance penalties because bandwidth and power wasted on data transfer. We must consider the input format of our algorithm, which is made up of pixel intensity values between 0 and 255. This means that only an 8-bit unsigned integer is required to store the input value. Previously, we have used 32-bit signed integers to transfer from the CPU host to the GPU device. OpenCL does not provide support for 8-bit unsigned integer types, but it does allow for an 8-bit unsigned char type. By typecasting the 32-bit integer pixel intensity input values to type unsigned char, we can reduce the memory copied to the GPU by 75%. This produces a noticeable decrease in execution time.

A. Centralized Memory Access

The first implementation of the data streaming optimization requires that the first work-item in the first row of a work-group to process its kernel will populate local memory with the necessary pixel values needed by the workgroup row. This is considered the centralized memory access approach to data streaming optimization. The centralized approach of having one work-item load local memory for its workgroup is depicted in Figure 5.

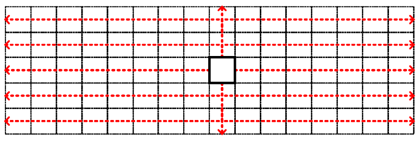


Figure 5. Centralized loading approach visualization.

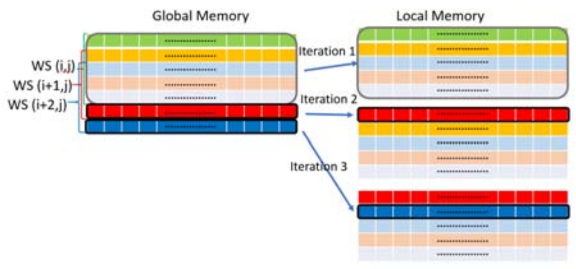


Figure 6. Round-robin local memory loading.

Figure 6 shows the data streaming from the global memory to the local memory. The working set, *WS(i, j)*, shows the amount of memory needed to determine *Disp(i,j)*. In Iteration 1, 5 rows of global memory are loaded to local memory. In subsequent iterations (for determining *Disp(i+1, j)*, *Disp(i+2, j)*, etc.), only 1 new row needs to be loaded and to replace an existing row in the local memory in a round-robin fashion to fulfill the local memory accesses to their corresponding working sets. This approach reduces the data accesses by utilizing the spatial and temporal localities in local memory.

Modifications to the host-side code are required for implementation of local memory optimization. The size of local memory allocated on the device must be specified by the host code. For the NVIDIA GeForce 940M GPU, the local memory capacity is 49,152 bytes. For this implementation, using a 5 x 512 local memory size, where each value is represented by a 4-byte integer, leads to 5 x 512 x 4 = 10,240 bytes allocated in local memory. Both left and right images require their own local memory allocations, leading to 20,480 bytes allocated total. This local memory allocation is explicitly declared when setting the kernel argument for the device code kernel. Normally, this kernel argument is linked to a memory buffer previously defined in the host code. However, data in local memory is private to the workgroup in the device. Thus, data is never read from or written to local memory from the host directly.

Another modification needed for the host code is the declaration of a Boolean array named rowDone, which keeps track of completed rows of work-items in each workgroup. The size of this array is equal to the height of the padded image times the number of workgroups along the width of the original image. The implementation of this array allows work-items in consequent rows to check the status of the work-items in the previous rows prior to completing execution. This array must be declared as a readable and writable memory buffer since it must be read from and written to by different work-items.

The first implementation of this data streaming optimization requires that every work-item populate local memory with the relevant data for its workgroup. Each work-item begins by defining boundaries for the data that must be loaded to local memory. The work-items in the first row of the workgroup will load local memory first and then perform the SAD algorithm. The following row will wait until this previous row has finished execution, and will then replace one row of local memory with the next row of data from global memory. When the last work-item of a row has finished execution, it will set rowDone to "true" for its corresponding row and workgroup. This is possible because in OpenCL, work-items execute in order along rows of work-items in a workgroup.

B. Distributed Memory Access

The Centralized Memory Access approach will introduce increased workload to the first work-item as the number of work-items in a workgroup increases. This is due to the pre-load of a larger number of working sets. Thus, it may eventually cause workload imbalance among the work-items. In this paper, the second approach to data streaming optimization is distributed memory access. We attempt to distribute the task of loading to local memory equally among all of the work items. In this manner, the work is divided evenly within each workgroup, and no work-items are left idle. Figure 7 depicts this process of distributed loading for one row of work items. In Iteration 1, each work-item loads 5 pixels, where the center pixel has the same image coordinates as the global ID of that work-item. Then, in the subsequent iterations, the pixels from the next rows will be read by the corresponding work-item and be located to the local memory buffer in a round-robin fashion. The mapping is the same as shown in Figure 6.

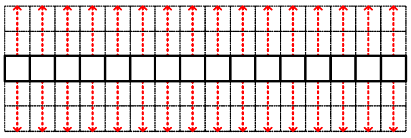


Figure 7. Distributed loading approach visualization.

Conceptually, this implementation is much simpler than the centralized approach. Previously, the centralized approach involved multiple if-else statements to check the row position and whether the previous row of work-items completed execution. Without these conditional statements, the distributed approach saves execution time

For implementation of this distributed approach to the data streaming optimization, modifications are made to the kernel code alone. The scenario in Figure 7 where each work-item loads exactly the same number of pixel values from to local memory is ideal, but not feasible. The work-items are executing in parallel, but the latency to load the pixels across the boundaries of local memory is significantly higher than it of the others due to the lack of spatial locality of accessing their local memories across the boundaries. This may result in some work-items attempting to calculate SAD disparity values before these lagging work items have completed loading the required values. Therefore, they must be synchronized with a barrier. All work-items in a workgroup must execute the OpenCL function "barrier(CLK_LOCAL_MEM_FENCE)" before they can proceed, and the CLK_LOCAL_MEM_FENCE flag ensures that local memory accesses are visible to all work-items in the workgroup [8].

VII. Experimental Results

The performance of the three implementations 1) global (the first implementation with global memory access) 2) centralized local (Section VI A), and 3) distributed local (Section VI B) are compared.

There is a trend of declining execution time as the workgroup size increases in Figure 8. It can be explained as the better mapping of the parallel SAD algorithm to the GPU hardware. A larger workgroup will allow more work-items to share local memory, and hence, have better temporal and spatial locality in memory accesses. As expected, the SAD algorithm with global memory access had the worst performance as the workgroup size was set smaller than 256. At the largest workgroup size of 512, the two aforementioned approaches have similar execution times at 18.03 ms, and 18.01 ms, respectively. This is explained as the increase of overhead due to the workload imbalance on the first work-item, which is responsible for pre-loading all working sets to the local memory for the entire workgroup. In contrast, the distributed approach to the local memory data streaming optimization remains consistently faster than the others, ending up at 4.88 ms for the same workgroup size of 512. 4.88 ms for one disparity calculation would lead to 1 / 0.00488 ≅ 205 frames per second, without taking into consideration the overhead between frames.

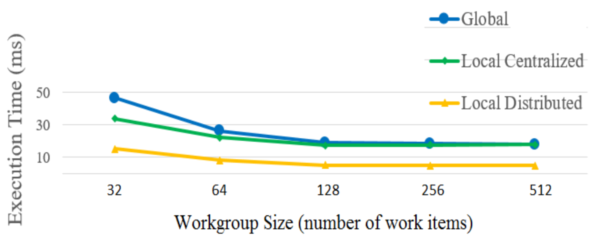


Figure 8. Comparison of SAD algorithm Performance across Different Optimizations.

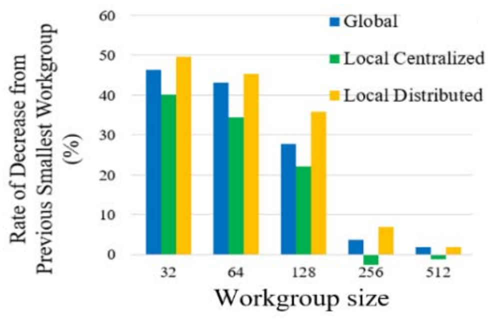


Figure 9. Comparison of Rate of Decrease from Previous Smallest Workgroup for SAD Algorithm Across Different Optimizations.

As the workgroup size gets larger, the rate of decrease in execution time generally decreases, as depicted in Figure 9. The centralized local approach consistently decreases the rate of decrease until it achieves a negative rate from workgroup size 128 to 256 and workgroup size 256 to 512. A negative rate of decrease means that the execution time actually increased. The other approaches have a consistent rate of decrease between 40 and 50%, until reaching a workgroup size of 128. For the three approaches described previously, the rate of decrease is diminished but still positive when transitioning from workgroup sizes of 128 and above. Performance is expected to peak at workgroup size 128 and drop off as the workgroup size increases, but performance continues to increase. These results can be partially attributed to the implicit use of spatial/temporal locality of memory accesses stored in caches by OpenCL. The continuing performance gain may also be explained by the number of kernels queued to an SMM exceeding the number of cores available, leading to a queuing delay. Each SMM has 4 instruction buffers that delegate instructions to their respective cores, and they are loaded with kernel instances each time a workgroup is executing. Larger workgroup sizes mean fewer workgroups, and fewer times the instruction buffers must be loaded.

VIII. Conclusion

This paper has shown that the SAD algorithm can be optimized on a GPU platform through OpenCL by explicit programming of local memory data loading and implicit data caching. Code optimizations and explicit caching of global memory have been observed to increase performance. Switching from a centralized approach to a distributed approach to local memory loading further improves performance. This work can be applied to embedded systems running ADAS applications where immediate distance calculation of objects is crucial and life-saving. With a maximum disparity map calculation rate of roughly 205 frames per second on a CPU-GPU heterogeneous environment, this algorithm optimization will surely make a beneficial impact when implemented on real-time embedded systems in automobiles. The work can scale to GPUs with more cores, and to higher resolution images. The code would be very similar in either case. In the future, we hope to continue the distributed memory access optimization approach by parallelizing the loading in a vertical fashion for each workgroup, which will enable us to compute multiple disparity values from the same kernel. We would also like to port this code to an embedded platform to see if the real-time performance gain will carry over as suspected.

References

[1] B. Kisačanin and M. Gelautz, Eds., Advances in Embedded Computer Vision, Springer International Publishing, 2014.
[2] Citron, C. (2014). “Stereo vision system module for low-cost FPGAs for autonomous mobile robots”. doi:10.15368/theses.2014.149
[3] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," International Journal of Computer Vision, vol. 47, pp. 7-42, 2002. doi: 10.1109/SMBV.2001.988771,
[4] X. Zhang and Z. Chen, "SAD-Based Stereo Vision Machine on a System-on-Programmable-Chip (SoPC)," Sensors, no. 13, pp. 3014-3027, 2013. doi: 10.3390/s130303014.
[5] K. Konolige, “Small Vision Systems: Hardware and Implementation”, Proceedings of the 8th International Symposium in Robotic Research, pp. 203–212, 1997.
[6] NVIDIA, "NVIDIA GeForce GTX 750 Ti Whitepaper," 2014.
[7] D. Kaeli, M. Perhaad, D. Schaa, and D. P. Zhang, Heterogeneous Computing with OpenCL 2.0, Morgan Kaufmann, 2015.
[8] Khronos Group, "The OpenCL Specification Version 2.0," 2015.
[9] NVIDIA, "OpenCL Best Practices Guide," 2011.
[10] A. Nicolau, "Loop quantization : unwinding for fine-grain parallelism exploitation," Cornell University, Dept. of Computer Science, Ithaca, N.Y., 1985.

Context-based Information Visualization for Smart AR Glasses

Choonsung Shin, Youngmin Kim, Jisoo Hong, Sung-Hee Hong, Hoonjong Kang
VR/AR Research Center
Korea Electronics Technologies Institute
Seoul, Republic of Korea
email:{cshin, rainmaker, jshong, hshong, hoonjongkang} @keti.re.kr

***Abstract*—This paper presents context-based information visualization architecture for smart augmented reality (AR) glasses. In order to support context-based information visualization, the proposed system consists of four parts: object recognition and tracking, context management, interaction management, and AR visualization. With these components, the proposed architecture understands nearby objects and the user's situation, then composes and visualizes virtual information over a real environment. We implemented the proposed architecture on Hololens and tested it with an example AR scenario.**

Keywords-augmented reality; context-aware; smart glasses.

I. Introduction

With recent technical advances, augmented reality has received a great deal of attention not only from researchers but also from consumers. These advances have resulted from both the software and hardware used to realize augmented reality. Global companies, such as Google and Apple have released their own AR software development kits (SDKs) for rapid AR app development that support natural feature tracking and recognition. Additionally, Microsoft released Hololens, which supports spatial tracking and mapping while Meta released a similar AR glass that supports a wide field of view and hand interaction [1][2]. With these advanced in AR, users have the ability to engage in a more immersive AR experience. More interestingly, these wearable AR glasses are promising for industry and daily life since these devices are able to more practically and intuitively assist workers and users with free hand interaction.

Several studies have examined supporting context-based information visualization in a mobile and wearable AR environment. An early work, the touring machine supported 3D visualization based on localization and interaction with a heavy desktop system combined with external sensors [4]-[6]. Later research focused on more lightweight systems, including a system that used an AR head mounted display (HMD) connected to a smartphone and a wearable interaction device, although this combination was still limited to a lab study [7]. Google Glass later showed the possibility of mobile and wearable information visualization for consumers [3]. Recently, Hololens has shown more immersive and impressive 3D visualization by enabling spatial mapping and head tracking technologies [1]. While previous work has solved crucial problems related to practical AR systems for end-users, these systems have still been limited to using contextual information about users and environment.

In this paper, we introduce context-based information visualization architecture for smart AR glasses. The proposed architecture combines context-awareness with wearable AR. It thus understands user's situation and detects nearby objects; then, it proactively visualizes with virtual object and information related to the objects detected. We implemented the proposed system on Hololens and tested it with a representative use case scenario.

II. Context-based Information Visualization

Our aim is to investigate context-based guidance using wearable AR systems that provide contextual information by understanding nearby objects and environments. These types of applications require the recognition of physical objects and location while also providing related information with end-user services. The touring machine used an external global positioning system (GPS) and magnetic sensors to determine the position and orientation of the user [4]. However, recent AR devices, such as Google Glass and Hololens have been equipped with a number of sensors such as a GPS, orientation sensor, touch pad, and accelerometer. In our work, we focus on how to architecturally combine the information from the sensors with augmented reality.

In order to achieve this aim, we introduce context-based information visualization architecture to provide architectural support for intelligent visualization on smart AR glasses. The proposed system consists of an object recognition and tracking component, context management, interaction management, and AR visualization. Figure 1 shows the architecture of the proposed information visualization system.

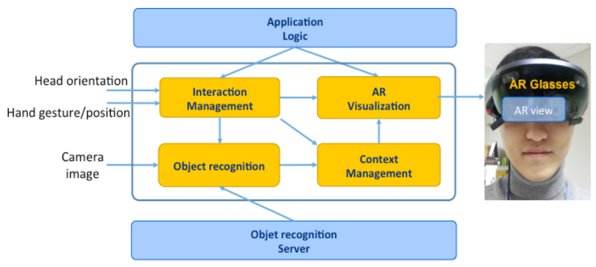


Figure 1. Context-based information visualization for smart AR glasses

The object recognition and tracking component detects, identifies, and tracks physical objects from the camera image stream. These objects include visual markers, 2D patterns, and 3D objects. This component thus includes a marker recognition/tracking library. For 2D recognition, feature detection and matching are applied to a given camera image while optical flow is applied to support frame-by-frame tracking. To deal with 3D objects, it is connected to object recognition server that classifies the object based on a deep learning model. The context management component stores and offers a wide range of contextual information about a user and his/her device and environment from sensory information and the object recognition component. The interaction management component detects and deals with user input and events with respect to the AR scene being rendered. The AR visualization component has a role in presenting 3D information based on the AR scene and contextual information. This component thus aligns the geometric relationship between the display screen and virtual scene. It then visualizes the 3D information over a real space.

III. Implementation

We implemented the proposed context-based information visualization system on Hololens. Hololens is a stand-alone AR glass platform, which contains the components required for AR such as head tracking, spatial mapping, gaze tracking, hand gesture recognition, and inertial measurement unit (IMU) sensors. In order to support image processing, we included the OpenCVForUnity library, which is a C# wrapper library connected to the raw OpenCV library [8]. We also added a camera image-grabbing library since the Hololens application programing interface (API) itself does not provide a method to retrieve the camera stream [8]. We then implemented maker recognition and tracking and 2D image recognition and tracking based on the OpenCV library in a Unity 3D environment. Figure 2(left) shows an example of marker-based visualization.

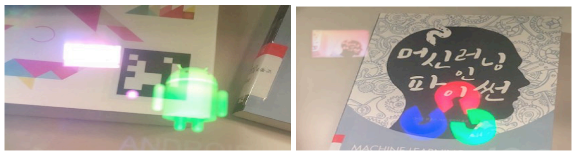


Figure 2. Example of maker and image recognition and tracking.

As seen in Figure 2(left), the proposed system detected and tracked a marker and the visualized a 3D object. In this example, the proposed system detected visual marker pattern from a camera image and then extracted the identification information. It then calculated 3d camera pose and applied it to Hololens camera matrix for visualizing the 3D object. We then also tested the 2D image-based recognition and tracking. As seen in Figure 2(right), the proposed system recognized and tracked a 2D image pattern. It first captured 2D images and stored them in the local folder in the Hololens. The proposed system then loaded the images and matched them to the camera stream from Hololens. The proposed system computed geometrical transformation related to the image it matched when the image from the camera stream was found in the images loaded. The system then estimated the 3D camera pose and applied it to the Hololens camera matrix in order to visualize the 3D contents over the object detected.

Last, we evaluated the performance of the proposed system. As marker-based tracking is simple and produces real-time performance, we focused more on 2D recognition and tracking. We thus evaluated the performance of the 2D image-based recognition and tracking. For this purpose, we measured the time to recognize and track 2D image patterns and compared the Hololens performance with an Android smartphone and a desktop PC. Hololens has a 1 GHz CPU while the Android smartphone has a 2.7 GHz octal core and the Desktop PC has a 3.3 GHz quad core.

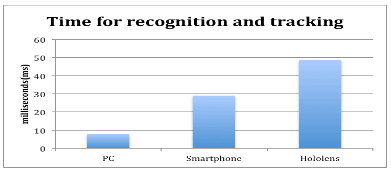


Figure 3. Time required for recognition and tracking.

As seen in Figure 3, Hololens took more time to recognize and track 2D patterns than the smartphone and the desktop PC. While the desktop PC's time was 16 ms (60 fps) and the smartphone's time was 25 ms (40 fps), Hololens took 46 ms. When we consider the camera frame rate (30 fps) of the Hololens, the overall time for tracking and rendering was 77 ms (15 fps), which needs to be improved for practical use.

IV. Conclusion

This work is the first step toward supporting information visualization for smart AR glasses. There are still technical problems that need to be improved. We first would like to add deep learning to improve 3D object tracking and recognition. We would also like to improve tracking and recognition performance on smart AR glasses.

Acknowledgement

This work was supported by the Ministry of Science and, ICT (MSIT) (Cross-Ministry Giga Korea Project).

References

[1] Hololens, https://www.microsoft.com (access date: 2018 Mar. 24)

[2] Meta Glass, http://www.metavision.com (accse date: 2018 Mar.24)

[3] Google Glass, https://www.x.company (accse date: 2018 Mar.24)

[4] S.Feiner, B.MacIntyre, T.Höllerer, and A.Webster, "A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment," PUC, Vol. 1, No 4, pp. 208–217, 1997.

[5] T. Starner, et al., "Augmented Reality Through Wearable Computing", In Presence, Vol. 6, No. 4, pp. 386-398, 1997

[6] R. Tenmoku, M. Kanbara, and N. Yokoya, "A Wearable Augmented Reality System for Navigation Using Positioning Infrastructures and a Pedometer," Proceedings of the ISMAR '03, pp. 344.

[7] J. K. Steven, et al., "Wearable mobile augmented reality: evaluating outdoor user experience," In Proceedings of VRCAI '11, pp. 209-216.

[8] https://github.com/EnoxSoftware/HoloLensWithOpenCVForUnityExample (access date: 2018 Mar. 24)

Application-Specific Memory Access by Prefetching via High Level Synthesis

İsmail San and Atakan Doğan

Dept. of Electrical and Electronics Engineering
Faculty of Engineering
Anadolu University
Eskişehir, Turkey
Email: {isan, atdogan} @anadolu.edu.tr

Kemal Ebcioğlu

Global Supercomputing Corporation
Yorktown Heights, NY, USA
Email: kemal.ebcioglu@global-supercomputing.com

***Abstract*—High Level Synthesis (HLS) allows an automatic translation from high level C/C++ descriptions into Register Transfer Level (RTL) hardware designs. HLS enables to design at a high level of abstraction that offers one to focus on high level concepts within less amount of design time. Once a specific data intensive application is considered to be accelerated in hardware, its memory access pattern must be exploited for higher performance. Most of the time, an application suffers from a high amount of memory access latencies. To reduce the memory access latencies, we use widely known prefetching technique to mask the latencies. In this paper, we enabled a data prefetching scheme specified at the C/C++ descriptions level via Vivado$^{\text{TM}}$HLS, which overlaps double-buffered prefetching and computation.**

Keywords–Prefetching; High level synthesis; Memory access time; Off-chip memory latency; Application-specific memory hierarchy.

I. Introduction

Modern VLSI technology allows Integrated Circuit (IC) manufacturers to build a single IC chip consisting of billions of transistors [1]. This advancement in VLSI technology requires complex Computer-Aided Design (CAD) tools to place and route billions of transistors. Advanced CAD tools must deal with this complexity and offer an acceptable design time. High Level Synthesis (HLS) tools allow hardware and software designers to create Register Transfer Level (RTL) hardware architectures from C/C++ high level descriptions [2]–[4]. In turn, current RTL synthesizers convert RTL specifications into Field-Programmable Gate Array (FPGA) bitstreams or IC chips. When performance is critical for the applications, designing application-specific hardware accelerators becomes inevitable. FPGAs offer reconfigurability for implementing arbitrary digital circuits: one can exploit all the parallelism in the algorithm to get more performance. FPGAs are used in many different fields from networking to data centers. Currently, they can also be accessed in the cloud. One can run customizable Xilinx FPGAs in the Amazon Elastic Compute Cloud (Amazon EC2) F1 instances [5].

When we consider a specific application that needs to be accelerated in hardware, it is of utmost importance to consider how the application is accessing the memory, since memory access time limits the peak performance of many data intensive applications. To reduce the memory access time, the memory access pattern of the application is utilized by designing application-specific memory hierarchies. Attaching caches in between memory and the processor, or hardware and software prefetching helps to reduce the memory access penalties. Since HLS generates an application-specific hardware accelerator from a high level description, it is very important to exploit the static memory references pattern inherent in the algorithm to be accelerated by the HLS. For this purpose, we questioned whether today's HLS tools generate prefetching hardware from C/C++ high level description. The key idea of our methodology is to take advantage of the fixed static memory pattern which is inherent in the algorithms, by requesting the data, which will be referenced soon, beforehand.

The rest of the article is organized as follows: Section II overviews the related work on caches and prefetching. Our motivating example for prefetching via Vivado$^{\text{TM}}$ is given in Section III. Design philosophy is described in Section IV. We discuss our implementation results on a Xilinx FPGA device in Section V and conclude in Section VI.

II. Related Work

Microarchitectural optimizations have been utilized to improve the on-chip cache bandwidth and hit rate for a given set of applications in state of the art methods on FPGA memory hierarchies. Caches exploit temporal and spatial locality by keeping recently used data in the cache and also bringing the nearby data into the cache, respectively. General memory hierarchies use the locality in the memory references of the algorithms via caches. In fact, caches eliminate several memory accesses, however, they will not eliminate the memory latency [6]. When a referenced data at a given address is not in the cache, a cache requests the data from the main memory and causes the processor to wait until the requested data arrives at the cache. Caches are efficient ways of connecting a general purpose processor to a memory.

In [7], the authors automatically construct a multi-cache architecture by automating the cache sizing to achieve higher cache bandwidth and hit rate. In contrast to [7], the method here presented focuses on fetching data before they are used in the execution in order to hide the memory access latency.

Prefetching is a known technique that accesses and stores data into a temporary buffer before it is needed; it aims to hide memory latencies. Applications having regular memory accesses, which are deterministically known prior to the execution, can take advantage of prefetching whereas the other applications with random memory accesses cannot utilize prefetching optimizations. Hardware-based data prefetching reduces the processor stall time by fetching data into the local memory before its use in the processor [6].

Recent work has used two data preloading techniques as prefetching and access/execute decoupling for accelerator-based systems [8]. The framework adds tags to accelerator memory accesses so that hardware prefetching can effectively preload data for accesses with regular patterns. On the other hand, our work does not include any tag inclusion for memory requests. Our technique employs the double-buffered prefetching and computation.

A software prefetch mechanism, which exploits the access pattern of multimedia and image processing applications, by using the DMA mode is proposed in [9] to improve the performance and reduce the overall power consumption. In contrast to [9], the present study focuses on high level synthesis of an application-specific hardware architecture with a data prefetch unit.

Automatic insertion of application-specific prefetching units is valuable in terms of hiding memory access latencies [10]. As stated in [10], automated synthesis of prefetching units can be enabled if the information is provided about when data is available for prefetching and when it is used by the application. Automatic synthesis of application-specific prefetching units, which fetch data from off-chip memory and store it in the on-chip caches in advance, is also proposed as a future work in [11]. LEAP scratchpads [12] are extended to automate the construction of application-specific memory hierarchies [11]. As emphasized by Winterstein *et al.* [11], “Knowledge about access patterns will also be used to implement application-specific prefetching and request merging”. With exploiting access patterns that is inherent in the algorithm, application-specific prefetching is a key to improve the performance of memory intensive applications.

The key element of our approach is to take advantage of the static memory access pattern at high level, which is inherent in the algorithm, by requesting the data in advance. The prefetching technique, which is shown by the PMM algorithm (Algorithm 2), uses a manual fusing of two inner loops and double-buffered prefetching, so that a few memory latencies are eliminated.

III. Motivating Example

Our motivation relies on the fact that data that will be referenced in the future can be prefetched into a buffer on the chip before they are used in a computation, which is widely known in the area as a hardware prefetching technique. We investigate whether such a hardware prefetching technique can be enabled at a high level of abstraction and whether the corresponding RTL design can be generated using an HLS tool. For our experiments, we use the Vivado™ HLS tool version 2017.4.

Let us consider matrix multiplication, since it is employed in many different fields of study from control theory to machine learning algorithms. Matrix multiplication involves several load operations that can be prefetched before they are used in the multiply and add execution unit. Matrix multiplication is defined as

$$\mathbf{C} = \mathbf{A} \times \mathbf{B}$$

where $\mathbf{A} \in \mathbb{Z}^{M\times K}$, $\mathbf{B} \in \mathbb{Z}^{K\times N}$ and $\mathbf{C} \in \mathbb{Z}^{M\times N}$. In order to compute $\mathbf{C}$, Equation 1 is used.

$$c_{ij} = \sum_{k=1}^{K} a_{ik} b_{kj} \tag{1}$$

where $1 \leq i \leq M$ and $1 \leq j \leq N$.

Once it is infeasible to store the entire input and output matrices within on-chip memories, one should store the matrices in main memory (which usually is outside the chip), fetch the data into the chip, do the computation and store back the result into the off-chip main memory.

While designing application-specific hardware, prefetching data from memory instead of caching provides more performance since a prefetching design avoids cache misses. Cache misses occur once referenced data is not in the cache memory, but, in a prefetching technique, the data is usually brought to be near the computation so that the computation never waits for the data.

While accessing the memory via caches, there is a miss penalty. The miss penalty decreases the peak performance. With perfect prefetching that masks all the memory latencies within the application, there is no miss penalty, when compared to matrix multiplication with caches. Caches exploit the locality information inside consecutive memory references. Instead of such a general solution that is provided by caches, we exploit application-specific memory patterns and generate memory references to data before the data is needed by computation. We propose a prefetching mechanism that is implemented via HLS, for a matrix multiplication operation. This prefetching mechanism can also be applied to other application-specific hardware designs if data reuse exists and data references are static.

IV. Methodology

Algorithm 1 shows the pseudo code which is very close to C/C++ description for the matrix multiplication that we choose as a reference. Translating this C/C++ description using

Algorithm 1 RMM: Ordinary Matrix Multiplication

```
Input: $A_{n\times n}$ and $B_{n\times n}$ matrices
Output: $C_{n\times n}$ matrix
1.  for $i = 0$ to $n-1$ do
2.      for $j = 0$ to $n-1$ do
3.          $S \leftarrow 0$
4.          for $k = 0$ to $n-1$ do
5.              $S \leftarrow S + A[j][k] * B[k][i];$
6.          end for
7.          $C[j][i] \leftarrow S;$
8.      end for
9.  end for
10. return $C$
```

Vivado™ HLS, generated hardware accelerator accesses to the main memory through three Advanced eXtensible Interface (AXI) interfaces for its three matrices. In main memory, three matrix addresses are not overlapped and referenced with different offset values. Our RMM algorithm, which we use as a reference, loads values of the input matrices from memory, does the computation, stores the result back to the main memory and continues with the next iteration. There is always a memory latency for each reference to data; this latency is the minimum time interval between the point in time a data is requested and the point in time the data is used in a computation.

Algorithm 2 describes our proposed prefetched matrix multiplication method. While one of the matrix row (column) buffers is used for the computation, the other matrix row (column) buffer is filled up with the next row (column) of the matrices that will be used next. This kind of double-buffered prefetching avoids stalls. As a result of converting the imperfect loop nest to a perfect loop nest, the HLS tool flattens the loop nest and saves extra cycles that would otherwise be spent when entering or exiting the loops [13]. A perfect loop nest refers to a nested loop where only the innermost loop has a body. Thus, in Algorithm 2, we have three nested loops and only the innermost loop has the body.

Algorithm 2 PMM: Prefetched Matrix Multiplication

```
Input: $A_{n\times n}$ and $B_{n\times n}$ matrices, $CB$ and $C2B$ refers to the
    column prefetch buffers, $RB$ and $R2B$ refers to the row
    prefetch buffers. $S$ stands for the accumulated sum.
Output: $C_{n\times n}$ matrix
1.  for $r = 0$ to $n-1$ do
2.    $CB[r] \leftarrow A[r][0]$
3.    $RB[r] \leftarrow B[0][r]$
4.  end for
5.  for $i = 0$ to $n-1$ do
6.    for $j = 0$ to $n-1$ do
7.      for $k = 0$ to $n-1$ do
8.        if $k == 0$ then
9.          $S \leftarrow 0$;
10.       end if
11.       $S + \leftarrow ((j\%2 == 0)\ ?\ RB[k]\ :\ R2B[k]) *$
              $((i\%2 == 0)\ ?\ CB[k]\ :\ C2B[k])$;
12.       if $j\%2 == 0$ then
13.         $R2B[k] \leftarrow A[(j+1)\%n][k]$;
14.       else
15.         $RB[k] \leftarrow A[(j+1)\%n][k]$;
16.       end if
17.       if $i\%2 == 0$ && $k == 0$ then
18.         $C2B[j] \leftarrow B[j][i+1]$;
19.       else if $i\%2 == 1$ && $k == 0$ then
20.         $CB[j] \leftarrow B[j][i+1]$;
21.       end if
22.       if $k == (n-1)$ then
23.         $C[j][i] \leftarrow S$;
24.       end if
25.     end for
26.   end for
27. end for
28. return $C$
```

V. Results

We described the RMM and PMM algorithms in C language and generated the hardware designs using Vivado™ HLS tool version 2017.4. We also prototyped the RMM and PMM accelerators on the Nexys 4 Double Data Rate (DDR) FPGA board which contains an Artix-7 FPGA and measured the actual timings on the FPGA board. We designed a system-on-chip (SoC) architecture (see Figure 1) to verify and measure the latencies on FPGA.

Figure 1 illustrates the block design of the system-on-chip used for our experiments on the FPGA. The MicroBlaze processor is employed in this SoC to configure all the peripherals and measure the elapsed time of the RMM and PMM accelerators. The design contains a timer peripheral to measure the elapsed time and a Universal Asynchronous Receiver-Transmitter (UART) peripheral to see the data and the values taken from the timer. The Nexys4 DDR FPGA board contains a Micron MT47H64M16HR-25:H DDR2 memory component and can be accessed through a DDR controller provided by Vivado™ design tool. We deployed this peripheral in our SoC and accessed it through an AXI interconnect.

In order to verify whether the generated PMM hardware does prefetching, we set up an experiment on the Nexys 4 DDR FPGA board and monitored the DDR2 slave port to check the incoming read and write addresses and their order. Figure 2 shows the waveform of the DDR2 S_AXI signals including s_axi_araddr (slave read address) and s_axi_awaddr (slave write address). One can see that the first result (c_{00}) of the matrix product is available at the 2048th cycle, which we marked on the Figure 2. The beginning address of the matrix B, A and C is 0x80000000, 0x80100000 and 0x80200000, respectively. Slave write address at this point is 0x0200000, which is the address of c_{00}, as exactly expected. The address of the beginning element of the second row, a_{10}, is 0x0100100. As one can observe that the 0x0100100 address is being fetched before the write operation of the first result c_{00}. It means that while execution stage is computing the c_{00} element, it starts to prefetch the second row of one of the input matrices which will be used in the next j iteration. Here, only one element of the second column of the other input matix is prefecthed on every j iteration.

Figure 3 shows the waveform of the same DDR2 signals as in Figure 2. Now, the first result (c_{00}) of the matrix product is available at the 2047th cycle which we marked on the Figure 3 with a marker. Slave write address at this point is 0x0200000 which again indicates the address of c_{00}. The address of the beginning element of the second row, a_{10}, is again 0x0100100. In this case, 0x0100100 address is being referenced after the first computation is finished. So, there is no prefetching in this RMM design.

We take advantage of the intrinsic memory access pattern of a specific application to decrease the total memory access latencies by overlapping double-buffered prefetching with computation. Thanks to this approach, we implemented prefetching via the HLS tool and achieved a smaller amount of total main memory access time as compared to an an algorithm without prefetching, as listed in Tables I and II. A careful scheduling technique in our prefetched matrix multiplication algorithm also helps us to totally avoid memory access stalls. As a

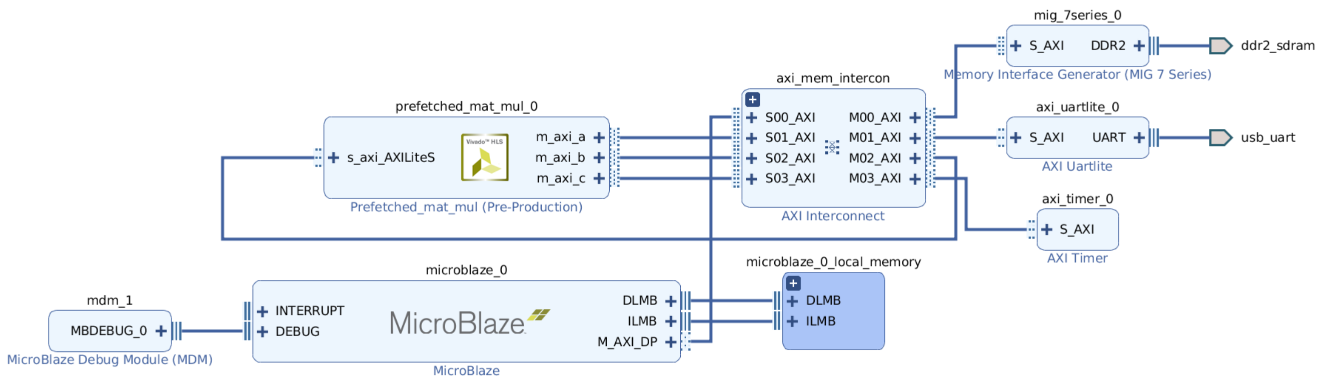
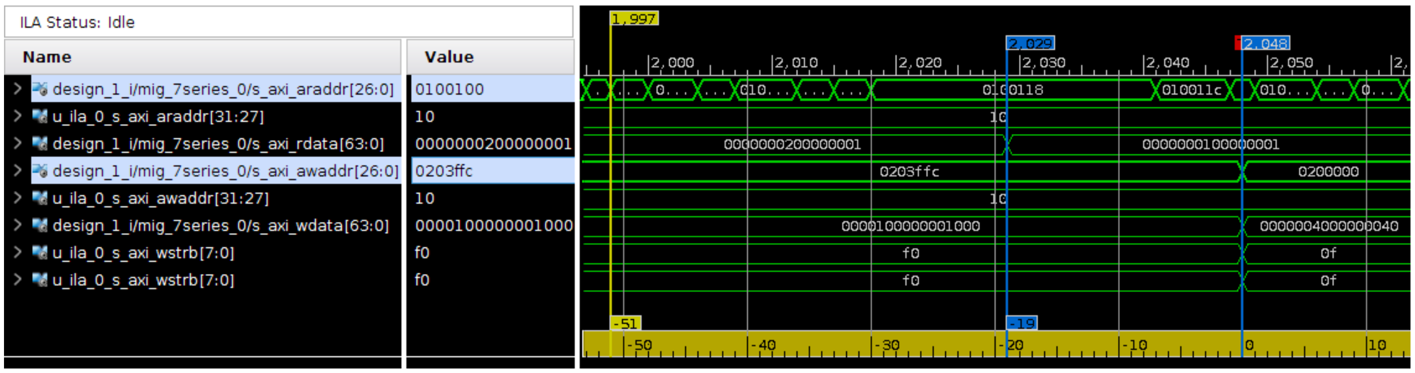


Figure 1. Block design view captured from Vivado^{TM} Design Suite 2017.4 version. MicroBlaze is the processor where we configure and control all the peripherals including DDR2 controller, Timer and UART.



ILA Status: Idle

Name	Value
design_1_i/mig_7series_0/s_axi_araddr[26:0]	0100100
u_ila_0_s_axi_araddr[31:27]	10
design_1_i/mig_7series_0/s_axi_rdata[63:0]	0000000200000001
design_1_i/mig_7series_0/s_axi_awaddr[26:0]	0203ffc
u_ila_0_s_axi_awaddr[31:27]	10
design_1_i/mig_7series_0/s_axi_wdata[63:0]	0000100000001000
u_ila_0_s_axi_wstrb[7:0]	f0
u_ila_0_s_axi_wstrb[7:0]	f0

Figure 2. A window of the timing diagram for the PMM captured from Chipscope tool of the Vivado^{TM} design suite 2017.4. Slave port of the DDR2 component signals are captured in this diagram.

TABLE I. RESULTS OF PMM AND RMM DESIGNS ON ARTIX-7 FPGAS. THE VALUES ARE LATENCY ESTIMATES GATHERED FROM VIVADO^{TM} HLS 2017.4.

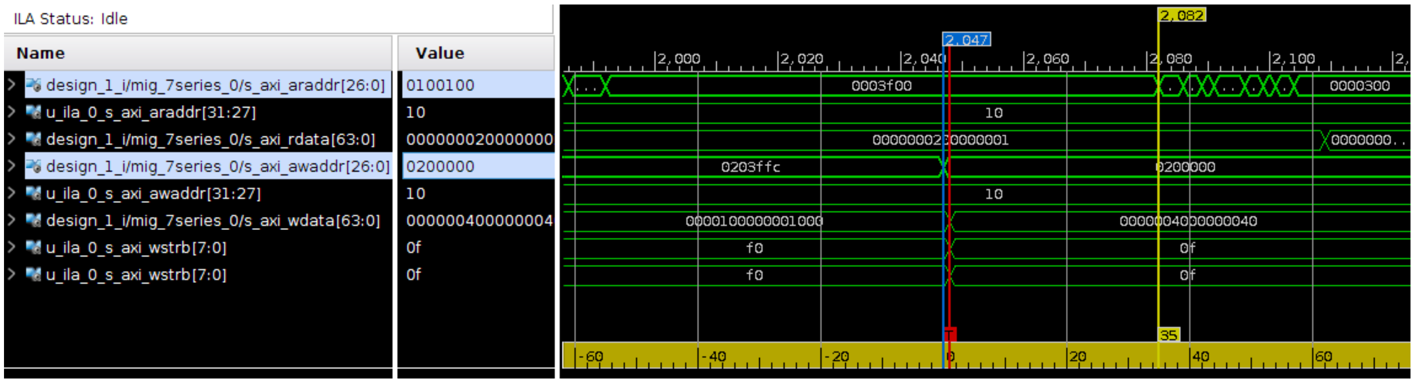
Design	Size	BRAM_18K	DSP48E	FF	LUT	Latency [cycles]
RMM	16	6	3	1908	2311	10497
PMM		10	3	3108	4359	4325
RMM	32	6	3	1924	2319	58369
PMM		10	3	3126	4384	33013
RMM	64	6	3	1940	2327	364545
PMM		10	3	3144	4398	262421
RMM	128	6	3	1956	2341	2506753
PMM		10	3	3162	4416	2097493
RMM	256	6	3	1972	2357	18415617
PMM		10	3	3180	4444	16777685

†All designs are targeted to 10 ns clock period.

consequence, the proposed prefetching method is achieving higher performance thanks to the HLS tool.

Table I summarizes the utilization and performance results taken from Vivado^{TM} HLS 2017.4 for PMM and RMM hardware designs with different matrix sizes. For instance, when we consider 16×16 matrices for the matrix multiplication, PMM-16 design takes 4325 cycles to complete the operation which is better than the RMM-16 in terms of latency. Since we use block memories (BRAM_18K) to implement four buffers in the PMM, it requires 4 more buffers than the PMM for all sizes. Approximately, 1.5 times more Flip-Flop (FF) and 2 times more Lookup Table (LUT) is required in the PMM. Once application performance is critical (which is often the case in the hardware accelerators), prefetching provides more performance, but with an area overhead.

Table II lists the performance estimates and the actual hardware results in terms of cycles of PMM and RMM with matrix size of 64. As one can see, the PMM-64 is completing the matrix multiplication approximately 1.6 times faster than the RMM-64. Due to the latencies in the AXI interconnect and the DDR2 memory, actual hardware measurements are higher than the Software (SW) estimates. The software estimates are measured without considering the latencies caused by the accelerator interfaces.



ILA Status: Idle

Name	Value
design_1_i/mig_7series_0/s_axi_araddr[26:0]	0100100
u_ila_0_s_axi_araddr[31:27]	10
design_1_i/mig_7series_0/s_axi_rdata[63:0]	000000020000000
design_1_i/mig_7series_0/s_axi_awaddr[26:0]	0200000
u_ila_0_s_axi_awaddr[31:27]	10
design_1_i/mig_7series_0/s_axi_wdata[63:0]	000000400000004
u_ila_0_s_axi_wstrb[7:0]	0f
u_ila_0_s_axi_wstrb[7:0]	0f

Figure 3. A window of the timing diagram for the RMM captured from Chipscope tool of the Vivado^{TM} design suite 2017.4. Slave port of the DDR2 component signals are captured in this diagram.

TABLE II. RESULTS OF PMM AND RMM DESIGNS ON ARTIX-7 FPGAS. THE VALUES ARE GATHERED FROM FPGA IMPLEMENTATION.

Design	SW Estimates [cycles]	HW Results [cycles]
RMM-64	364545	3188847
PMM-64	262421	1960804

†All designs are targeted to 10 ns clock period.

VI. Conclusion

HLS provides an improved design methodology because it reduces design time and because it allows specifying an algorithm at a high level. Design space explorations becomes prominent and applying high level concepts without touching the difficult low level abstraction layer is possible. Memory bandwidth is the bottleneck that limits the performance of many different data intensive applications. In order to decrease the memory latencies, we enabled a prefetching mechanism using the HLS tool and obtained results with a case study of the matrix multiplication algorithm. The main advantage of prefetching is that one can mask all the memory latencies if computation is taking more time than total memory access time and if dependences permit. If the memory access pattern of an application is static, prefetching is an efficient way to improve performance and can be described by Vivado^{TM} HLS.

Note that while currently the Vivado^{TM} HLS tool cannot automatically add prefetching to a C algorithm as we have accomplished in the present paper, we are not claiming that an HLS compiler in general cannot automatically create an application-specific hardware with prefetching from a C algorithm. For example, a compiler technique called *loop fusion* [14] applied to two loops in sequential C code at an intermediate step of possible compiler transformations starting from the original RMM algorithm and ending with the PMM algorithm:

1) loop to do computation on the current row (column) buffer
2) loop to do prefetching to fill the next row (column) buffer

can result in a single loop where the the next row (column) is prefetched into the next row (column) buffer while computation proceeds at the same time on the current row (column) buffer.

However, one compiler challenge in achieving perfect prefetching with loop fusion is to *fuse loops of unequal shapes*, which is in effect used by the prefetching technique of the present paper: consider a loop (1) to perform computations on the current column buffer which is a doubly nested loop (the j loop, which includes the k loop as an inner loop) and consider a loop (2) for filling the next column buffer which is an ordinary singly nested loop. Loops (1) and (2) must be fused to accomplish effective prefetching. In the present PMM algorithm a manual fusing has been done by moving an incremental prefetching of 1/n'th of the next column of B, all the way into the inner k loop, which is an example of fusion of loops with unequal shapes.

Further work is needed to automatically design an application-specific perfect prefetching mechanism through HLS in order to fully mask memory latency time (i.e., so that referenced data is always in the SRAM of the processor). A worthwhile research direction is to schedule and software pipeline a general sequential code loop nest, and to generate parallel hardware and a memory hierarchy from the code such that each memory operand is already in a register or SRAM location when it is needed for computation, whenever dependences permit [15].

References

[1] S. K. Kundu, S. Karmakar, G. S. Taki, A. Roy, C. D. Choudhuri, M. Basu, A. Basak, R. Upadhyay, A. Raj, and S. Mandal, "Progress in submicron device technology," in 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), Aug 2017, pp. 318–320.

[2] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-Level Synthesis for FPGAs: From Prototyping to Deployment," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 4, April 2011, pp. 473–491.

[3] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J. H. Anderson, S. Brown, and T. Czajkowski, "LegUp: high-level synthesis

for FPGA-based processor/accelerator systems," in Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays. ACM, 2011, pp. 33–36.

[4] R. Nane, V. M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels, "A Survey and Evaluation of FPGA High-Level Synthesis Tools," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 10, Oct 2016, pp. 1591–1604.

[5] "AWS FPGA Developer AMI," https://aws.amazon.com/marketplace/pp/B06VVYBLZZ, Accessed: 2018-04-04.

[6] T.-F. Chen and J.-L. Baer, "Effective hardware-based data prefetching for high-performance processors," IEEE Transactions on Computers, vol. 44, no. 5, May 1995, pp. 609–623.

[7] F. Winterstein, K. Fleming, H.-J. Yang, J. Wickerson, and G. Constantinides, "Custom-sized caches in application-specific memory hierarchies," in Field Programmable Technology (FPT), 2015 International Conference on. IEEE, 2015, pp. 144–151.

[8] T. Chen and G. E. Suh, "Efficient Data Supply for Hardware Accelerators with Prefetching and Access/Execute Decoupling," in The 49th Annual IEEE/ACM International Symposium on Microarchitecture, ser. MICRO-49, 2016, pp. 46:1–46:12.

[9] M. Dasygenis, E. Brockmeyer, B. Durinck, F. Catthoor, D. Soudris, and A. Thanailakis, "A Combined DMA and Application-specific Prefetching Approach for Tackling the Memory Latency Bottleneck," IEEE Trans. Very Large Scale Integr. Syst., vol. 14, no. 3, Mar. 2006, pp. 279–291.

[10] F. J. Winterstein, S. R. Bayliss, and G. A. Constantinides, "Separation Logic for High-Level Synthesis," ACM Trans. Reconfigurable Technol. Syst., vol. 9, no. 2, Dec. 2015, pp. 10:1–10:23.

[11] F. Winterstein, K. Fleming, H.-J. Yang, S. Bayliss, and G. Constantinides, "MATCHUP: Memory Abstractions for Heap Manipulating Programs," in Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, ser. FPGA '15, 2015, pp. 136–145.

[12] M. Adler, K. E. Fleming, A. Parashar, M. Pellauer, and J. Emer, "Leap scratchpads: Automatic memory and cache management for reconfigurable logic," in Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, ser. FPGA '11, 2011, pp. 25–28.

[13] "Vivado Design Suite User Guide - High-Level Synthesis," 2017. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug902-vivado-high-level-synthesis.pdf

[14] Wikipedia, "Loop fission and fusion," 2018, [Online; accessed 8-June-2018]. [Online]. Available: https://en.wikipedia.org/wiki/Loop_fission_and_fusion

[15] K. Ebcioglu, E. Kultursay, and M. T. Kandemir, "Method and system for converting a single-threaded software program into an application-specific supercomputer," Feb. 24 2015, US Patent 8,966,457.

Fake GPS Defender: A Server-side Solution to Detect Fake GPS

Yu-Heng Chang

Department of Computer Science
and Information Engineering
National Central University
Taoyuan, Taiwan 32001
Email: bill0722001@gmail.com

Yan-Ling Hwang

School of Applied Foreign Languages
Chung Shan Medical University
Taichung, Taiwan 40201
Email: yanling_h@yahoo.com

Chih-Wen Ou

Department of Computer Science
and Information Engineering
National Central University
Taoyuan, Taiwan 32001
Email: chihwen.frankou@gmail.com

Chih-Lin Hu

Department of Communication Engineering
National Central University
Taoyuan, Taiwan 32001
Email: clhu@ce.ncu.edu.tw

Fu-Hau Hsu

Department of Computer Science
and Information Engineering
National Central University
Taoyuan, Taiwan 32001
Email: hsufh@csie.ncu.edu.tw

***Abstract*—Smart phones, tablets, and wearable devices are equipped with Global Positioning System (GPS) sensors in order to obtain devices geographical location. Many conventional network-based applications provide the specific content and service to users according to their locations. The correctness of the location provided by the device's GPS module is certainly important to these Location-Based Service (LBS) providers. However, most service providers are unable to effectively authenticate GPS values provided by their users. This becomes an issue because device users can manipulate GPS values with their desired latitude and longitude through installing the specific firmware on their devices. For a popular LBS game like *Pokemon GO*, fake GPS values bring negative impact on the system stability and the fairness among other service users. This issue is so called "Fake GPS" problem. In this paper, we propose a pure network-based detection solution for the LBS provider who has to verify the correctness of their users' GPS values. Our mechanism is based on Internet Control Message Protocol (ICMP), and is able to provide the detection precision to state/city-level by using location-IP mappings of devices' edge routers. As a server side solution, our approach makes the malicious GPS manipulators more difficult to perform the trick. According to the implementation and experiment, the major contribution of FGDefender is that it does have better detection precision. Its server-side nature of deployment is competitive as well.**

Keywords–Location; Fake GPS; Spoofing; Router.

I. INTRODUCTION

With the fast development of Internet and telecommunication technology, electronic devices are usually equipped with Global Positioning System (GPS) module, giving them the ability to perceive location of themselves. To catch up with the trend of Internet of Things (IoT), more and more network services rely on devices location, which forms a kind of network service: *Location-Based Service* (LBS). *Pokemon GO* [1] is a well-known example of LBS game for its heavily usage of the GPS sensor on mobile devices. This kind of network applications suffers from a specific data spoofing attacks. LBS servers accept all the GPS values obtaining from user devices by default, typically without additional mechanisms to check whether these GPS values have been manipulated or not. Such issue is usually referred as the Fake GPS problem. Lack of data authentication and integrity checking not only brings doubt about the system, but also gives chances to malicious users to spoof the location, which is known as the *Location Spoofing Attack* (LSA), to obtain illegal interests from the application. The Mobile Network Operator (MNO) may aware of where their subscribers are, but LBS providers still hardly acquire GPS information from the MNO, especially while the LBS provider is not a domestic regulated enterprise. Moreover, most of conventional IoT-related devices can assume that physical access, changing/reversing the firmware of the device, all require lots of effort and professional skills. Unlike conventional IoT-based applications, LBS applications running on smartphones, which usually belong to user themselves, only take relatively low effort for owners to manipulate the firmware on their devices. Many open and online resources are provided for smartphone owners capable of jailbreaking, rooting, flashing new custom ROMs on their smartphones. *LineageOS* [2] is a famous and easy-to-install third-party smartphone ROM for Android. For Apple's iPhone, those without the newest version of iOS can also be jailbreaked easily by just one click. All these conditions indicate that LSA can be conducted by malicious LBS users without too much cost. These methods allow GPS value manipulating before returning them to the LBS servers are listed below:

Android mock location: By surfacing developer options in Android, one can easily find the mock location option lying on the setting menu. Emulating GPS value is a partially build-in function in Android framework [3]. It is originally designed for the LBS application developer whose device is not equipped with GPS module. As a result, creating fake GPS result is not difficult since Android OS provides such function for developing purpose.

Emulator: *BlueStacks* [4], and *Genymotion* [5] are well-known Android emulators running with custom Android ROMs. Those emulators are usually running on the desktop PC instead of the handheld device. Therefore, the build-in GPS module emulation is obviously necessary. Attackers who use these emulators to run LBS applications can easily spoof the location information by assigning desired GPS values to the emulator's configuration.

Xposed Framework [6]: Xposed is a framework that can change the behavior of the system and apps without touching any APKs. Xposed Framework is a famous tools using system call hooking techniques in order to control the system behavior. By replacing `/system/bin/app_process` in Android framework, most the sensor-related APIs can be intercepted and modified before returning it to the caller application. Other data provided by other sensors is also unconvincing.

Software Define Radio (SDR): Software defined radio [7] is a radio communication system implemented by means of software on a personal computer or embedded system. *HackRF* [8] is an open source SDR platform. Interfering GPS signals on hardware level helps malicious users to evade most of the software-based defense mechanisms for both Android and iOS devices. Due to the requirement for considerably resources and professional knowledge, few civilian cases other than military activities of such attack were found.

In this paper, a fake GPS defender (FGDefender) is proposed. It is a server-side, pure network-based LSA detect mechanism for the LBS provider who has to verify the correctness of their users GPS value. Most existing systems provide IP-location lookup by querying databases containing these IP registry data. This is static data mapping, and many IP address owners assign this field with the owners location. If the owner is MNO providing Internet access service across the country, the location is obviously different between the IP address owner and the practical user. Our mechanism will be based on Internet Control Message Protocol (ICMP), which is widely supported by most network equipments and systems. In addition, most modern detection methods can only authenticate GPS values with country-level precision. FGDefender can effectively increase the detection precision to state/city-level by using location-IP probings for the device's edge router. Because GPS values are authenticated by those features obtained from applications server side, It makes the GPS value manipulators (attacker) more difficult to perform the trick. Our study of FGDefender also analyzes several situations that may cause false positive and negative, and gives possible improvements for each case as well. The major contribution of FGDefender is that it has better detection precision and lower deployment cost. Meanwhile, it still also works along with other existing detection systems.

II. Related Work

Saroiu and Wolman proposed the location proofs [9] in 2009, a simple primitive that allows mobile devices to prove their locations to mobile applications and services. A location proof is issued by the wireless infrastructure, such as a Wi-Fi access point, to the device within the communication range. The proof can be transmitted by the device to the application that wishes to verify the location of that device. This approach verifies user location at the client side and depends on the deployment for the target infrastructure. The detailed comparison between this approach and ours is discussed in Section VI-A. Recent studies [10] [11] [12] deal with the fake GPS signal, which is one of the most common issues that is researched. Our approach aims at issues not restricted to the fake GPS signal. We also focus on the issue that users may use tools on the device to spoof GPS data for LBS applications.

To verify the location of a device, one potential idea is to check its IP-location mappings. For each IP address on Internet, there are databases maintaining registration data [13] [14] listing companies or organizations who owns these IP addresses. The registration data often includes the organization name, location, registration date, and administrators contact information. Users can obtain country, and state/city (probably), in the location field usually. Hence, most basic detect mechanisms to prevent LSA is to compare the location of the IP registration data with the location that the device claims. However, the registration data for IP address is not accurate, especially for IP address when the device operates inside the mobile network. As a case shown in Figure 1, a public IP address (114.137.156.248) of a device actually locates in Taoyuan. But, when we search the Whois database, this IP's location is marked as the location of the MNO's headquarter in Taipei. As the result, only the Country field in the registration database is relatively convincing. Such precision is not accurate enough for most LBS providers.

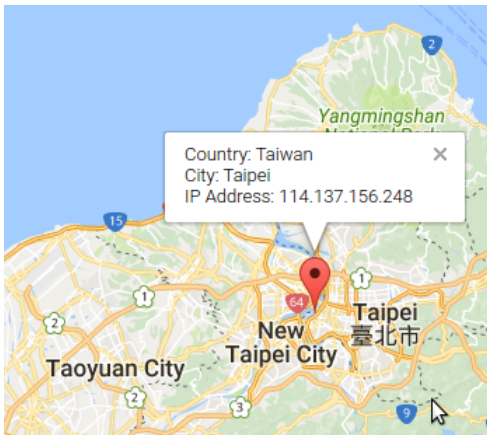


Figure 1. GeoIPtool database result for IP address 114.137.156.248

Some studies, like [15], are trying to use other sensors on the mobile devices to detect LSA. Devices with same moving path can represent similar pattern on gyroscope, accelerometer, magnetometer or other sensors. When it comes to such detection mechanisms, problems will occur: First, acquiring more sensors' information on the mobile device indicates more permissions required for accessing these sensors. In addition, since the GPS sensor can be manipulated by some system call hooking techniques provided by Xposed Framework, it means that values from other sensors can also be contaminated by the same skill. Due to the attackers ownership of the mobile device, our detection mechanism will not depend on any information provided by devices.

Another LSA detection [16] is trying to use devices that are adjacent to the target device. The main idea is that two adjacent devices must be able to communicate with or detect some signals from the other if these two devices are close enough geographically. This can be done by making use of the random service set identifier on portal Wi-Fi hotspot, or the ability to communicate with other hosts under the same Network Address Translation (NAT) by a random private IP. Although turning on Wi-Fi hotspot on mobile devices can be achieved programmatically, it causes current network to be disconnected. Due to the fact that Wi-Fi can be functioned either on STA mode or AP mode, it means that users who are chosen to turn on the hotspots, are able to connect to Internet through Wi-Fi before the other devices find it. We believe that it results in lots of inconvenience to those affected users. Another disadvantages is that it is restricted by the Wi-Fi transmission range (which is often 10-50m with normal antenna). We have tried to use the Carrier-grade Network Address Translation (CGNAT) to identify whether two devices connect the same base station or the same CGNAT. Instead of using normal NAT established by Wi-Fi hotspots, we tried to apply the same method on telecommunication networks to eliminate these disadvantages. In the end, it came to a failure that most MNOs may not allow packets forwarded to a private IP address, even these packets are from the same base station.

III. System Design

FGDefender aims at effectively increasing the detection precision to state/city-level, and providing a server-side, pure network solution to LBS provider. As mentioned in Section I, traditional IP location database systems cannot guarantee the location precision which often assigned by the IP address owner in the registry. The original main idea of FGDefender is to verify whether the location of the user's IP address identical to the location which the user delivers to LBS application. Since the static mapping is not real time and precise, ICMP-based probing technology may satisfy the needs of FGDefender. Since mobile devices do not directly connect to Internet, they often do so by their MNO 3G/4G mobile networks or WiFi hotspots. This is the reason why the correlation between the edge router's IP and its location is focused in this study. Since the correctness of an edge router location is difficult to achieve, we use other claimed locations provided by nearby LBS users using this edge router. This design is based on an important assumption that most users are believed to be benign.

As a pure network solution, the main obstacle may be proxies on Internet. Both Virtual Private Network (VPN) and proxy cache service leave us a great number of unknown information behind the front server, and cause inaccuracy to our system. Hence, we verify and exclude known proxy service users from our system first. The reason is that users with the qualified network seldom to use LBS application through VPN service, since VPN cause obvious latency and inconvenience. Some LBS applications prevent users from using VPN or other network proxy service in their application agreements because of the service regional restriction. As a result, we consider such pre-condition of proxy service is acceptable.

The next step is to examine the edge router. As shown in Figure 2, it is unlikely that a device, using "Router A" (denote as R_A in following) as its edge router, has the location obviously far from the area where R_A is responsible for. Suppose that there are other normal users using the same LBS application, and we can get some of them who also use R_A as their edge router (shown in Figure 2 as green dots). By clustering location data of those other R_A users, the possible location of R_A can be inferred. We defined it as "responsible area" of R_A. Hence, among those devices with an edge router, the device who is obviously far away from others can be the possible LSA device we are looking for.

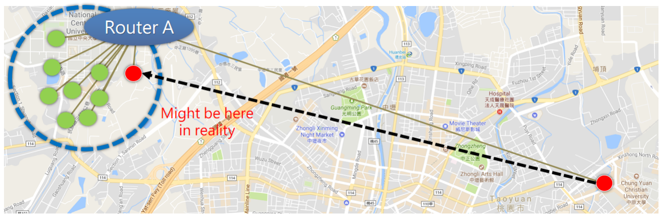


Figure 2. The responsible area of an edge router

When we focus on the result of one edge router, the possible result may be similar to Figure 3 (a). There is an obvious crowd and the attacker resides far away from others. The only way that the attacker can evade our mechanism is to purchase a lot more device or register lots of accounts for the LBS application, as shown in Figure 3 (b).

IV. System Implementation

FGDefender contains two major components. Each component runs a phase, as shown in Figure 4. The first component runs VPN detection phase, and the second component runs the database phase for maintaining edge router locations.

Phase 1: In the first phase, we verify if service user is connecting through a VPN or proxy server. Apart from blacklisting some common VPN/proxy service, we use WITCH [17] or getIPIntel [18], both of them are open web services to help us detect and exclude the VPN/proxy users. Those web-based VPN/Proxy detection service basically check packets MTU with the help of a important principle: A normal MTU is 1,500 bytes, which can deliver 1460 bytes payload in one packet. VPN user was not able to transmit such a long data in one packet due to the extra header used by VPN delivering.

Phase 2: For each LBS application user, we perform reverse *traceroute* [19] to find out which edge router they are using. We built a database. The routers IP address is used as the key denoting which edge router we discover, and store the claimed location for each user. We record three closest routers from client to prevent the case that attackers control the network of the first router (edge router).

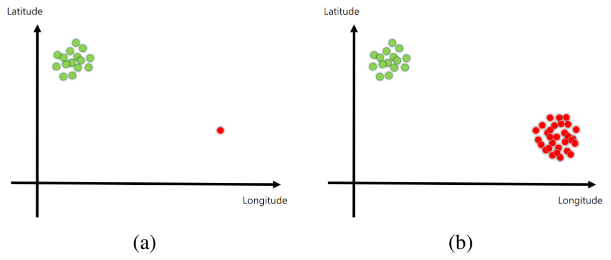


Figure 3. The way for attacker to defeat Fake GPS defender

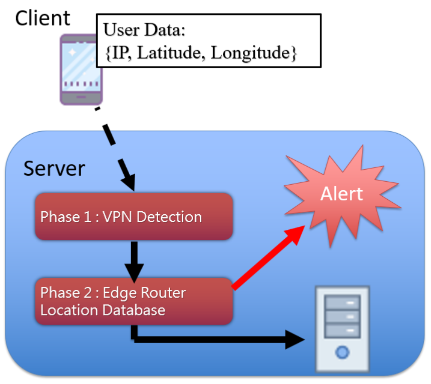

Client
User Data:
{IP, Latitude, Longitude}
Server
Phase 1 : VPN Detection
Phase 2 : Edge Router Location Database
Alert

Figure 4. Fake GPS Defender system

For data analyzing, we try to find out outliers [20] from all users who have the same edge router. Using the mean value of longitude and latitude (gravity) as the reference point, we first calculate the distance between reference point and each users position. Second, after sorting all the distance, we can now obtain upper quartile (Q_3), lower quartile (Q_1) and interquartile range (IQR) of those collected data. Outliers (denote as x) in Tukey's range test [21] are often defined as:

$$x > Q_3 + 3IQR$$
$$x < Q_1 - 3IQR$$

Note that three times of IQR is usually called *major outliers*, which is a restriction in Statistics. As a result, we use this method to find out majority of outliers. For FGDefender, outliers are suspicious LSA attackers. Once FGDefender finds suspicious outliers, it removes the outliers data to prevent large amount of data coming from attackers.

V. Evaluation

In this section, we describe how we deploy and evaluate FGDefender. This experiment is to prove that users who change their location significantly, often result in changing to different edge router to connect to Internet.

A. Environment

In the experiment, we use two smartphones: Sony Xperia Arc S running on Android 4.0.2, and Sony Xperia X Performance running on Android 7.1.1 respectively, and a SIM which provides Internet accessibility from mobile network of Chunghwa Telecom, to gather GPS location and the corresponding network characteristics at a specific location. The detailed technical specifications of experiment are shown in Table I. We also build a server as the database in our laboratory, with one core CPU, 1GB RAM and 20GB hard disk divided from our VMware ESXi (Elastic Sky X integrated) workstation in our campus network.

B. Experiments

We developed and installed an app on both experimental smartphones. This app regularly switch the flight mode on and off, and then report genuine GPS values to our database server. We carried these smartphones to many different cities

TABLE I. EXPERIMENT MOBILE PHONES TECHNICAL SPECIFICATION

Model name	Xperia Arc S (LT15i)	Xperia X performace (F8132)
Android Version	Android 4.0.2	Android 7.1.1
Network tech.	HSPA+	LTE-A
Processor	Qualcomm MSM8255T Snapdragon S2	Qualcomm MSM8996 Snapdragon 820
RAM	512 MB RAM	3 GB RAM

in Taiwan. Once the server gets the reported data, it performed *traceroute* process to the smartphones IP addresses. We performed test on both computer networks and mobile networks, and examine the feasibility and accuracy of FGDefender.

C. Results

On the trip from Taipei to Taichung covering cities of Taipei, New Taipei, Taoyuan, Hsinchu, Miaoli, and Taichung, we collected 63 mobile network IP addresses under the 3G mobile network, which derives six edge routers. The result shows that we can obviously separate the northern Taiwan, which includes cities listed above, into two sub-areas, using these corresponding edge routers as shown in Table II

TABLE II. EDGE ROUTER RESULTS IN 3G MOBILE NETWORK

Place	Edge Router
Taipei-Hsinchu	210.65.126.161 210.65.126.209
Hsinchu-Taichung	210.65.126.193 210.65.126.185 220.128.24.237 220.128.25.169

In the 4G mobile network, we increase the GPS report frequency to making the experiment result more accurately because of the faster transmission data rate. As shown in Figure 5, there is obviously a strong correlation between the location of the users and edge routers they connect to. We collected 303 mobile network IP addresses, which also derives six edge routers. We then separate the northern Taiwan, including cities listed above, into three sub-areas, using corresponding edge routers as shown in Table III.

TABLE III. EDGE ROUTER RESULTS IN 4G MOBILE NETWORK

Place	Edge Router
Taipei	210.65.126.165 210.65.126.161
Taoyuan-Hsinchu	210.65.126.213 210.65.126.209
Hsinchu-Taichung	210.65.126.185 210.65.126.189

The reason causing the difference of reasonability area deployment between 3G and 4G mobile network is that there are fewer people using 3G mobile network than 4G mobile network. For the result of multiple reasonability areas, it must exist a boundary between two adjacent areas. As we performed in our experiments (Figure 7 and Figure 8), when the user

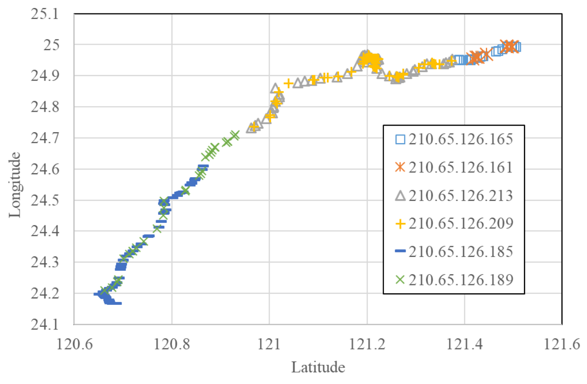


Figure 5. Edge router used in different positions in mobile network

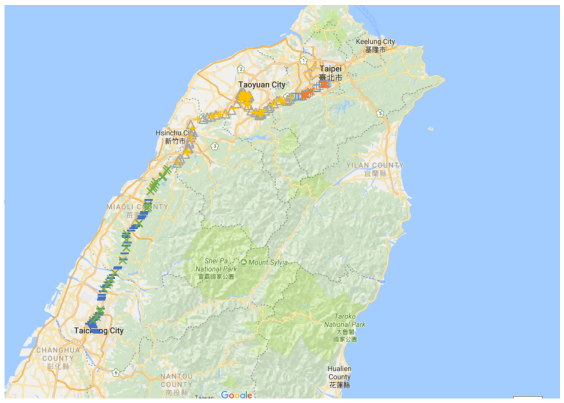


Figure 6. Mapping Figure 5 on Google Map

moves from Taipei to Taichung through the highway, the location that the edge router changes is in Sanxia Distinct. It denotes that the user who stays at the location marked in Figure 7 (a) are using 210.65.126.165 or 210.65.126.161 as the edge router to connect to Internet. The user stays at the location marked in Figure 7 (b) are using 210.65.126.213 or 210.65.126.209 instead. The changing point between Taoyuan-Hsinchu responsible areas and Hsinchu-Taichung responsible area fall in the Zhunan Township. It similarly means that users stay in Figure 8 (b) are using 210.65.126.185 or 210.65.126.189 as the edge router to connect to Internet. When devices handover to the base station (or eNB in LTE 4G networks), which belongs to another responsible area near the changing point, it results in an edge router switching. As a result, when mapping Figure 5 with a map as shown in Figure 6, we can finally figure out how MNO manage their IP address in the mobile network.

Due to the definition of outliers, the radius of the acceptable area we discover is $Q_3 + 3IQR$. It is often much larger than responsible area that the edge router actually manages. Therefore, for the case that client moving cross the boundary, he will locate in the overlapping region between two adjacent responsible areas, and cause little impact to our mechanism.

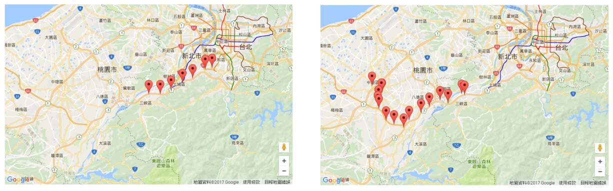
VI. Discussion

In this section, we focus on the accuracy, and potential enhancement if a larger responsible area is encountered. For false negative cases, we specifically discuss this situation and the problems attackers need to deal with for this case. The accuracy of FGDefender depends hardly on the precision we can detect VPN or proxy users, the information of the users edge router we can discover, and the MNOs router and IP addresses management. If FGDefender fails to detect VPN/proxy users, those LSA users will not be detected when they connect through the proxy service. However, FGDefender forces the attacker not to spoof their location with a large distance from the position where the VPN/proxy server locates. Although the attacker successfully evades the VPN detection, they may still be detected once its spoofed location is out of the responsible area where its VPNs router should be. In other words, the attacker who wants to spoof their location must find an undetectable proxy server residing near that spoofed location first before performing fake GPS.

Another situation which might cause false negative case is that when we obtain a much larger responsible area of the mobile network router. Not all MNOs allocate different edge routers for different cities. If only very few sets of edge routers are used for the entire country, it means that there are very few responsible areas. Attackers can successfully spoof the location within cities in the same and large responsible area. Our later discussion of RTT Detection Enhancement is designed to deal with such issue. Attackers using mobile network should present a RTT value in a reasonable range according to other nearby normal users. Since the RTT value may dramatically change, the attacker who wants to evade the detection must emulate a reasonable RTT value for each spoofed location. Such enhancement can bring lots of effort and cost for attackers, and will be implemented in the future.

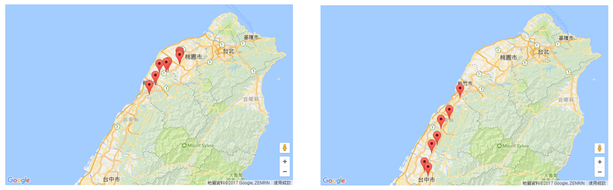
A. Comparison

The location proof, an approach mentioned in the section II, also provides location verification for LBS application. Compared to FGDefender, there are at least two major differences. The first one is that the location proof is a client-side approach, which means obtaining the trustable location data is done on the device. FGDefender does not use this



(a) (b)

Figure 7. Edge router switches from Taipei to Taoyuan



(a) (b)

Figure 8. Edge router switches from Taoyuan to Taichung

design because users can easily modify their device firmware, and making the location data untrusted today. The second difference is that the location proof required deployment to the infrastructure. FGDefender is a pure server-side approach, and no modification to infrastructure is required. FGDefender can be deployed with much lower cost than location proofs.

B. Limitation

ICMP-based approaches suffer from an obvious limitation that most network routers may discard ICMP packets due to security and performance reason. In our prototype system, only ICMP packet is used. More types of probing packets, such as UDP packets, can be used to mitigate this issue for future implementations.

C. Misc

Since the location of the user is considered as a part of privacy, there may be some solutions aiming at providing users functionalities of location privacy protection. In other words, users can decided whether to provide their location to others explicitly. As a server-side location verification solution, like FGDefender, it cannot verify the correctness of the location when users refuse to provide their actual location. But, such location privacy-preserving solutions are designed for stopping unauthorized data transmitted to unauthorized service providers, not for crafting fake locations to them. When users refuse to provide their location for a FGDefender-protected LBS, such identification performed by FGDefender will not be taking place. If the user provides the location, the identification should be conducted. The LBS administrator can receive a report explicitly listing no-location users and fake location users. The connection between the client and the server may suffer from Man-In-The-Middle (MITM) attack if the connection is not well-secured. This is the channel authentication issue, and is not the focus of this study. Most mobile devices support Assisted-GPS (AGPS) to enhance the location precision. According to our survey, there seems to be correlation between the IP addresses of edge routers and device location data from AGPS module. Such correlation is similar to the case of device GPS data, so that we do not distinguished GPS and AGPS data in this study. The device may sometimes obtain GPS location with errors, and causing problems for LBS application. However, most of such errors do not cause problems for FGDefender because FGDefender, which works based on the location of edge routers in the responsible area, can tolerate most margins of error generated by the imprecise GPS location.

VII. Conclusion and Future Work

In this paper, we proposed FGDefender, which is a server-side and network-based method to detect LSA users. We probe the IP address of user's edge router and its inferred location, which can be obtained from the application's server side, to authenticate whether the claimed location by the user is reasonable. As a pure network-based solution, the major contribution of FGDefender is that it has better detection precision and lower deployment cost. It also brings inevitable costs for LSA users. The experiment shows that the accuracy of FGDefender depends on the MNO's network configurations. More mobile network operators, as well as more locations, are planned to be included in the future.

References

[1] Pokmon GO Official Website, May 2018. [Online]. Available: http://www.pokemongo.com/

[2] LineageOS Android Distribution, May 2018. [Online]. Available: https://lineageos.org/

[3] A. Developers, "Location strategies," May 2018. [Online]. Available: https://developer.android.com/guide/topics/location/strategies

[4] BlueStacks, May 2018. [Online]. Available: https://www.bluestacks.com/tw/index.html

[5] Genymotion, May 2018. [Online]. Available: https://www.genymotion.com/desktop/

[6] Xposed framework, May 2018. [Online]. Available: http://repo.xposed.info/

[7] M. Ossmann, "Rapid radio reversing," May 2018. [Online]. Available: https://greatscottgadgets.com/tr/gsg-tr-2016-1.pdf

[8] greatscottgadgets.com, "Hackrf one," May 2018. [Online]. Available: https://greatscottgadgets.com/hackrf/

[9] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in Proceedings of the 10th Workshop on Mobile Computing Systems and Applications, ser. HotMobile '09. New York, NY, USA: ACM, 2009, pp. 3:1–3:6. [Online]. Available: http://doi.acm.org/10.1145/1514411.1514414

[10] J. Magiera and R. Katulski, "Detection and mitigation of gps spoofing based on antenna array processing," Journal of Applied Research and Technology, vol. 13, no. 1, 2015, pp. 45 – 57. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1665642315300043

[11] M. R. Mosavi, A. R. Baziar, and M. Moazedi, "De-noising and spoofing extraction from position solution using wavelet transform on stationary single-frequency gps receiver in immediate detection condition," Journal of Applied Research and Technology, vol. 15, no. 4, 2017, pp. 402 – 411. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1665642317300652

[12] K. Jansen, N. O. Tippenhauer, and C. Pöpper, "Multi-receiver gps spoofing detection: Error models and realization," in Proceedings of the 32Nd Annual Conference on Computer Security Applications, ser. ACSAC '16. New York, NY, USA: ACM, 2016, pp. 237–250. [Online]. Available: http://doi.acm.org/10.1145/2991079.2991092

[13] GeoIPTool, May 2018. [Online]. Available: https://geoiptool.com/

[14] MaxMind, GeoIP2 Databases, May 2018. [Online]. Available: https://www.maxmind.com/en/geoip-demo

[15] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, "Inferring user routes and locations using zero-permission mobile sensors," in 2016 IEEE Symposium on Security and Privacy (SP), May 2016, pp. 397–413.

[16] F. Restuccia, A. Saracino, S. K. Das, and F. Martinelli, "Lvs: A wifi-based system to tackle location spoofing in location-based services," in 2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), June 2016, pp. 1–4.

[17] WITCH - detects OpenVPN via MSS values, May 2018. [Online]. Available: http://witch.valdikss.org.ru/

[18] getIPIntel, May 2018. [Online]. Available: https://getipintel.net/

[19] G. S. Malkin, "Traceroute Using an IP Option," RFC 1393, Jan. 1993. [Online]. Available: https://rfc-editor.org/rfc/rfc1393.txt

[20] G. J. Kerns, Introduction to Probability and Statistics Using R. GNU Free Documentation License, 2011, p. 44.

[21] H. Abdi and L. Williams, "Tukey's honestly signiflcant diflerence (hsd) test," May 2018.

Machine Learning-based Arrhythmia Diagnosis Algorithm

Dong Hyun Kim, Jae Min Lee, and Jong Deok Kim*
Department of Computer Science and Engineering
Pusan National University
Pusan, South Korea
e-mail: {dhkim1106, ashesmin, kimjd}@pusan.ac.kr

Yeong Joon Gil
Department of Research and Development
HUINNO
Seoul, South Korea
e-mail: kyzoon@huinno.com

***Abstract*— As interest in cardiovascular disease increases, there is significant development in real-time healthcare services and devices. This paper suggests a machine learning-based arrhythmia diagnosis algorithm for mobile healthcare systems linked to a wearable electrocardiogram measurement device. The system monitors electrocardiograms in real time and distinguishes among arrhythmia, normal, and noise signals. By regular monitoring using a mobile healthcare system linked to a wearable bio-signal measurement device, users can minimize the risk of chronic disease and maintain a healthy standard of living. In this paper, an arrhythmia diagnosis algorithm is suggested, realized, and evaluated based on bio-signals collected from a watch-type electrocardiogram device. In the process, the efficacy of the algorithm is established**

Keywords- Arrhythmia; Atrial fibrillation; Machine Learning.

I. INTRODUCTION

The U-health care technology has allowed health management that used to be performed in a hospital to become a daily standard of care. This concept has been actively developed along with the dissemination of smart devices. In particular, U-health care devices that assist in the prevention of chronic diseases, such as obesity, diabetes mellitus, and cardiovascular diseases are in active development.

The risk of incidence of cardiovascular diseases demands the need for regular monitoring of electrocardiograms (ECGs), especially for those individuals who have medical history require continuous management regardless of time and place. U healthcare is the newest collection of digital-convergence that combined Information Technology (IT) and Biology Technology (BT) to provide remote services of health and medical care without any limitations of time or place. As real-time electrocardiogram signal processing and classification technologies are realized, early diagnosis of cardiovascular diseases that required real-time monitoring became possible, which is a great achievement [1]-[3].

The electrocardiogram signals are electric waves that are generated by a sequence of depolarization and repolarization of the atria and ventricles, which form repetitive periodic curves. These signals can express various abnormalities of the heart, and each condition exhibits characteristic waveforms. In this paper, a low-layer deep learning algorithm that distinguishes Atrial fibrillation(A-fib), Sinus rhythm, and noise is designed, realized and evaluated, based on data collected from watch-type electrocardiogram devices.

II. RELATED WORK

In this section, we describe existed machine learning platform and machine learning algorithm

A. *Machine learning platform*

In November 2015, Google released its open-source library TensorFlow that was developed for its vital technology of machine learning. Source code and API are open sources built on top of TensorFlow, which can be accessible under the Apache License 2.0 [4].

Anyone with Apache License 2.0 can create new software and transfer its copyright by utilizing the TensorFlow software. In addition, the software developed by utilizing TensorFlow can be used for commercialization purpose.

The engine built for TensorFlow is significantly flexible and can be executed on a CPU that is computer-sourced, thus enabling data analysis and GPU. In addition, it utilizes a multi-core processor of a desktop PC, such as dual-core or quad-core. It is accessible in a server environment that is built by virtualization technology or thousands of big data sensors [5].

TensorFlow works on both a PC architecture, as well as on mobile devices. Google reported that the relative API of this algorithm uses the same code regardless of the device. TensorFlow can be utilized with Python, which is often used in C++ and machine learning. However, there are several limitations of the algorithm. Firstly, it does not facilitate Cloud service [6]. Google has combined TensorFlow with various services and applied machine learning methods, but there are still some systems that cannot utilize this package. For devices that do not support TensorFlow, their systems should be independently developed. Secondly, Google does not provide any guideline regarding TensorFlow or any data to utilize using the algorithm, except the frame that can analyze data. Creation and application of modes are thus realized by individual programmers. TensorFlow is a computing library that realizes an artificial neural network of data through data flow graphs. The data flow graphs consist of nodes and edges. Nodes calculate the output and edges express the relation between the input and output of the nodes. The values calculated from a node are transferred to a multidimensional arrange through an edge, while nodes read the values from the multidimensional arrange and calculate them in parallel [7].

B. *Machine learning algorithm*

Convolution is an operation that is used to extract a feature of interest in a given image using certain filters. s(t) is referred to as a feature map of data xx. Prior to the introduction of deep learning, in order to process an image as an input in other machine learning frameworks, a filter was chosen, preprocessing was performed which involved the convolution of an image with the chosen filter to produce a proper feature map. Then, this result was used as an input to the machine learning framework. This feature engineering process significantly impacted overall performance. Since the choice of filters or the number of filters used are part of the feature engineering process and not a theoretical construct, this area was not viewed as an interesting area in machine learning [8].

The critical idea of Convolution Neural Network (CNN) is to create a model that learns the best convolution filter that can extract the best feature map, and effectively performs preprocessing, since processing significantly affects overall performance. Therefore, there are three key ideas in CNN to realize the best filter with minimum complexity and include; sparse interactions (or sparse weight), parameter sharing (or dense weight), and equivariant representations. In other words, CNN connects every layer but only partially, which is described as the sparse weight. These weights are considered different random variables and updated respectively, while a certain weight group shares parameter to balance the weight value (parameter sharing). Then, based on this idea, a new model that is going to learn representation equivariant to transform such a shift is constructed [9].

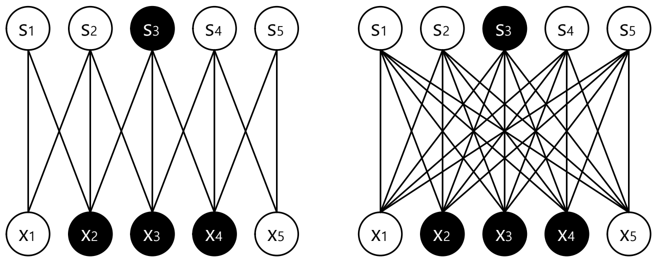


Figure 1. edges and input nodes influence output s3

Figure 1 illustrates the edges that influence the same output s3 and input nodes. The left diagram shows that not every connection is available and only inputs x2, x3, x4 that influence s3. However, in the right diagram, there is a difference of model parameters in available connection and every input influences s3 [10].

However, in the right diagram, there is a difference of model parameters in available connection and every input influences s3.

To process the real image data, the left model works better because only part of information of the image is needed to decide a certain feature. Equivalent representations mean that the changing method of output along with changes of the input becomes equivariant when sparse weight and tied weight are effectively arranged in a certain form. For example, if function f is equivariant to function g, this can be expressed as f(g(x)) = g(f(x)). For image processing, g means a random value of a linear transform. This represents the transform of an image such as shifting, rotation and scaling. The image can still be recognized if it is rotated, moved, or scaled, but a computer cannot identify this same image since the transform results in changes to individual pixel values. If we could create a network *f* that makes an equivariant representation to a certain transform *g*, this network could have a fixed representation as the input regardless of its shift or rotation. Previously, shared parameters were set to process the same filter for each patch. If the image was shifted, then feature map is not distorted but shifted along with the image [10].

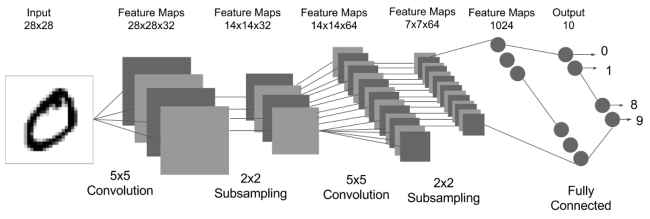


Figure 2. Construction map of Convolutional Neural Network (CNN)[11]

However, every image might need more than one filter. The previous convolution layer expresses only one convolution filter, but several convolution filters might be needed to concatenate and create a feature map. Therefore, the CNN model is in the form of several convolution layers that are combined as subsequently indicated. For reference, each layer or filter is officially called a kernel, and the set of kernels are called one layer [12].

III. DESIGN OF PREPROCESSING AND ARRHYTHMIA ALGORITHM

In this section, we describe suggested ECG data format and deep learning network structure

A. *Preprocessing*

The data used in this research were collected from watch-type wearable devices of HUINNO Co. To use ECG data set stored as time series form in a CNN algorithm, the ECG data stored every 15 min (900 s) were set as 10 s unit of sampling form. The frequency of the watch-type ECG sampling is 300 Hz.

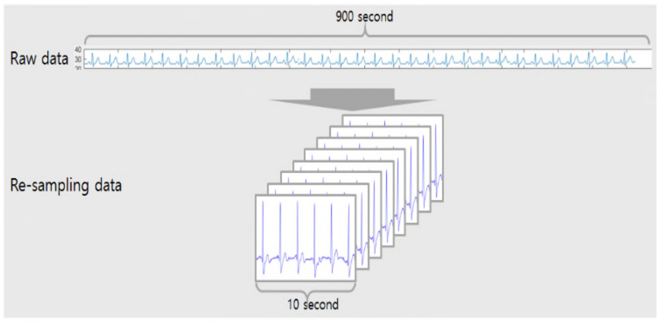


Figure 3. Re-sampling structure with existing ECG data format adjusted to DL engine

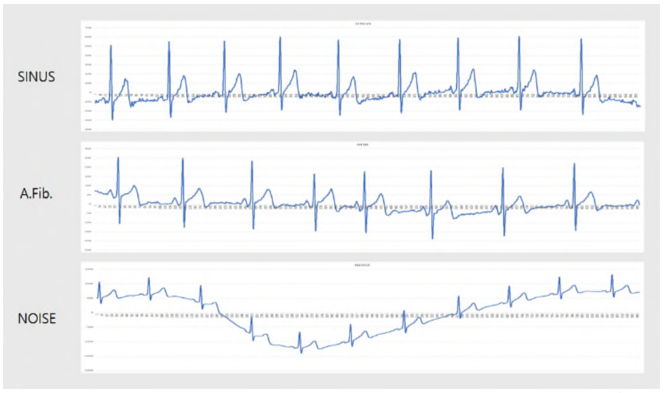


Figure 4. HUINNO SINUS ECG, A.Fib ECG, Noise ECG data sample for HUINNO.

B. *Design of arrhythmia algorithm*

The neural network comprises seven layers: five convolutional layers and two fully connected layers. On every convolutional layer, batch normalization and max-pooling are executed. The first convolutional layer is set with 48 filters and 11 kernels while the second one consists of 128 filters and 5 kernels.

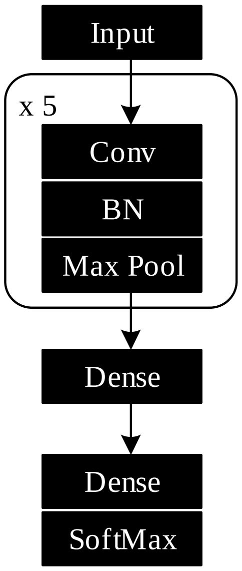


Figure 5. Suggested deep learning network structure

TABLE I. THE NUMBER OF SINUS, A-FIB, NOISE DATA CREATED THROUGH PREPROCESSING

	Sinus	A-fib	Noise	Total
Number of data set	130,496	22,390	18,720	171,606

The first convolutional layer is set with 48 filters and 11 kernels while the second one consists of 128 filters and 5 kernels. The third one contains 192 filters and 3 kernels, the fourth one has 192 filters and 3 kernels, while the last one consists of 128 filters and 3 kernels The Max polling layer moves in between 3 kernels and 2 strides to collect the maximum values in every layer. The two final fully connected layer comprises of 2048 filters. As previously explained, the configuration of each layer is described in Table 2.

TABLE II. THE CONFIGURATION OF EACH LAYER

Layer	Output Shape	Param#
Conv1	(2997, 48)	576
Pooling1	(1498, 48)	-
Conv2	(1494, 128)	30,848
Pooling2	(746, 128)	-
Conv3	(744, 192)	73,920
Pooling3	(371, 192)	-
Conv4	(369, 192)	110,784
Pooling4	(184, 192)	-
Conv5	(182, 128)	73,856
Pooling5	(90, 128)	-
Flatten	(11, 520)	-
Dense6	(2048)	23,595,008
Dense7	(2048)	4,196,352
Dense8	(3)	6,147

IV. PERFORMANCE EVALUATION ENVIRONMENT AND ANALYSIS

In the overall data, the number of data classified into Noise was the smallest. Therefore, the algorithm was created with these 15,000 Noise data, and the rate of Sinus and the A-Fib dataset was adjusted to that of the noise data.

Figure 6 shows the confusion matrix of the sinus, A-fib, noise of the suggested model. It is important to apply the algorithm in a remote medical area to determine the accuracy of the approach in determining the A-fib signal as A-fib. In 3000 test sets, this algorithm identified A-fib signals with 100% accuracy, noise at 82.4%, and sinus at 39.4%. Only three classes were expressed using a simple algorithm to increase efficiency.

TABLE III. TRAINING, VALIDATION, TEST DATA CONDITION

	Group	Sinus	A.Fib	Noise
Training	A group	9,000	9,000	9,000
Validation		3,000	3,000	3,000
Test	B group	3,000	3,000	3,000
Total	-	15,000	15,000	15,000

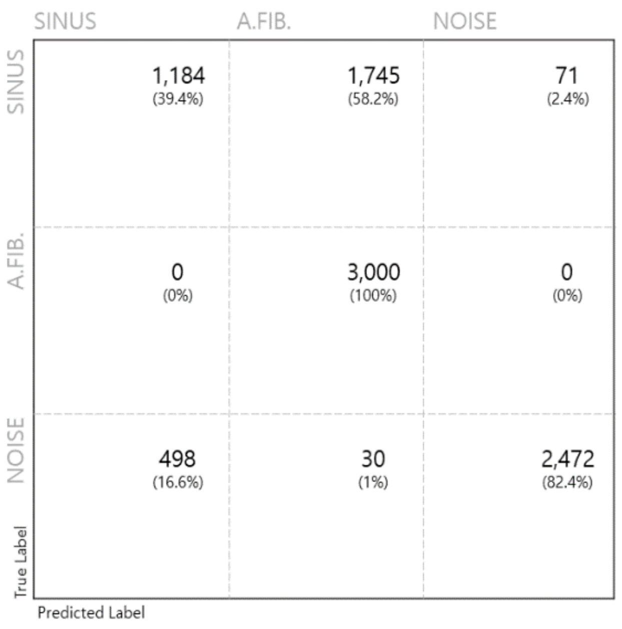


Figure 6. Confusion matrix for the model predictions on the test set.

V. CONCLUSION

In this paper, we designed and realized a deep learning algorithm in a mobile healthcare system for real-time processing of electrocardiogram data. A simple deep learning network was designed with Sinus, A-fib, and noise.

Then, the collected information from watch-type electrocardiogram device was learned, evaluated and tested. The result showed a high accuracy for A-fib signal detection, which could be applied to a remote medical system. There are approximately 12 signals, which indicate an abnormality in the functioning of the heart. Based on our findings, a deep learning algorithm to detect these signals could be designed and developed and its accuracy may be improved through further research.

ACKNOWLEDGMENT

"This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (No. 2016R1A2B4013150)."

* : Corresponding Author

REFERENCES

[1] K Itao, T Umeda, G Lopez, and M Kinjo, "Human recorder system development for sensing the autonomic nervous system", IEEE Sensors, pp423-426, 2008.

[2] Corventis Corporation. corventis products [Internet]. Available: http://www.corventis.com. [retrieved: Jun, 2018]

[3] Isansys Lifecare Corporation. isansys products [Internet]. Available: http://isansys.com/. [retrieved: Jun, 2018]

[4] Tensorflow [Internet]. Available : http://www.tensorflow.org/ [retrieved: Jun, 2018]

[5] Jeffrey Dean and Rajat Monga, TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, Preliminary White Paper, November 9, 2015.

[6] Automatic differention [Internet]. Available : https://en.wikipedia. org/wiki/Automatic_diff erention [retrieved: Jun, 2018]

[7] Four Reasons Why Google's Tensor Flow Disclosure Is Important [Internet] Available : http://www.itworld.co.kr/news/96498 [retrieved: Jun, 2018]

[8] R. Collobert et al, "Natural language processing (almost) from scratch," Journal of Machine Learning Research 12(Aug), pp. 2493-2537, 2011

[9] S. Joty and E. Hoque, "Speech act modeling of written asynchronous conversations with task-specific embeddings and conditional structured models," Proc. of the 54th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 7-12, 2016.

[10] M. Kim and H. Kim, "User Utterance Analysis using Convolutional Neural Network," Journal of Telecommunications and information, Vol. 21, pp. 5-7, 2017.

[11] Implement Convolutional Neural Networks (CNNs) for MNIST number classification using TensorFlow [Internet] Available : http://solarisailab.com/archives/1308 [retrieved: Jun, 2018]

[12] X. Wang, W. Jiang, and Z. Luo, "Combination of convolutional and recurrent neural network for sentiment analysis of short texts," Proc. of the 26th International Conference on Computational Linguistics, pp. 2428-2437, 2016.

Adjusting Positions of Vehicle Parts based on Rules for Unknown Drivers

Kwangsoo Kim, Bong Wan Kim, Sunwhan Lim, Dong-Hwan Park
IoT Research Division
Electronics and Telecommunications Research Institute
Daejeon, Republic of Korea 34129
Email: {enoch, kimbw, shlim, dhpark}@etri.re.kr

***Abstract*—This paper describes an inference engine used in a system and semantic representation for the system which automatically adjusts the positions of vehicle parts based on rules. The inference engine has rules stored in a knowledge base, which describe the relation between the position of the vehicle part and the driver's body size. The inference engine receives the driver's body sizes in real time, and finds a rule associated with the input values by matching a pattern between them. According to the value defined in the rule, the position of the vehicle part is changed automatically. This rule is automatically modified by learning the relation between the driver's preferred position and body size. The number of selected rules and reasoning time are selected as performance indicators of the inference engine. Also, an ontology is designed to share the development results with others. Automated vehicle parts control system can be used as a method that improves the driver's satisfaction by automatically recommending the driver's preferred position in an environment where many unknown people use the same vehicle like a shared car or a rental car.**

***Keywords*–Vehicle Part Control; Rule; Inference; Ontology.**

I. INTRODUCTION

The future worlds we have seen in science fiction movies are becoming a reality one by one. The Internet of Things (IoT) is making these changes possible. The IoT has launched many smart IoT products and services, making our lives and work easier and more efficient [1]. One of these products is a connected car that has the ability to connect to the Internet. Internet access allows the vehicle to share various data with other devices within the car, as well as devices and services outside the car including other cars or infrastructure. The growing use of connected cars enables a lot of new applications. These applications can be divided into two groups: individual vehicle applications and cooperative applications. Individual vehicle applications are services and contents used in a vehicle to improve the efficiency of vehicle use. They are a navigation to guide a path to a destination, a gas station finder in an area, a vehicle condition or fault diagnosis service, and so on. Cooperative applications are provided by connecting between vehicles or infrastructure. They are a collision warning, intersection movement assistance, a blind spot or lane changing warning, left turn assistance, and so on. Also, those applications can be divided into several categories: navigation, infotainment, safety, diagnostics, and payments. Figure 1 shows the shipment trend of connected cars. The connected car shipment was 6.98 million in 2017 and is estimated to reach 23.87 million by 2022, at a CAGR of 27.9% for the forecasted period [2]. We can see that the connected car shipments will grow continually.

Car sharing is another change occurred in the automotive industry. It is a short-term car rental service positioned between a privately-owned car and public transportation. It intends to enhance traffic efficiency, change existing traffic behavior, improve social problem caused by vehicles, and reduce the negative environmental impacts. By the service, the paradigm for automobiles is shifting from ownership to sharing. The car sharing is quickly increasing in the world. Therefore, it is necessary to develop a connected car service which is suitable for car sharing.

Several automakers provide a memory car seat which allows a driver to save its positions for later recall. As the driver of a car is restricted to the owner or his/her family of the car, it is sufficient that the positions of the car seat are automatically set to previously stored positions, which the driver adjusted before. However, in a car sharing service, it is impossible to automatically recommend the preferred seat position to individual driver because an unspecified driver drives the car. Therefore, there is a need for a method for automatically recommending the position of a vehicle indoor component to unknown drivers.

In this paper, we explain how to automatically adjust the position of vehicle parts (e.g., seat and mirrors) to meet the needs of a driver who uses a car sharing service. An inference engine, rules, and driver's body dimensions are used to recommend the location of the interior parts of a vehicle to the driver, which the driver is satisfied. If the driver does not accept the recommended positions and adjust new positions, then the new positions and the body dimensions are inserted into a learning engine. The learning engine generates new rules, which accept the new positions.

The rest of this paper is structured as follows. Section II describes the system architecture. Section III introduces the performance of the inference engine used in the system. Section IV introduces the semantic representation for sharing the system with others. Finally, Section V concludes this paper.

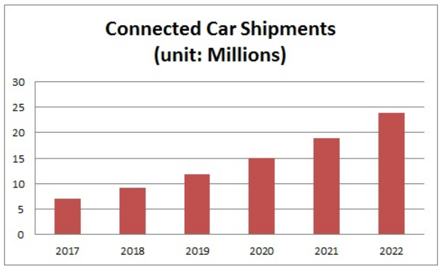


Figure 1. Forecast of connected car shipments (Source: ABI research).

TABLE I. BODY DIMENSIONS AND VEHOCEL PARTS.

Body Dimension	Vehicle Part	Control
height, sitting height, arm length, leg length	driver seat	vertical / horizontal position, tilt
	headrest	vertical position, tilt
	room mirror	left / right position, tilt
	side mirror	left / right position, tilt

II. SYSTEM ARCHITECTURE

A Vehicle Part Position Control System (VPPCS) discussed in this paper consists of an inference engine, rules, a learning engine sensors, and actuators. The inference engine deduces new information by applying logical rules to the knowledge base. The inputs of this engine are rules and body sizes of a driver. The inference engine finds the position of the vehicle part that best suits the driver by pattern matching the driver's body dimensions and rules in real time. The rules describe the relation between body dimensions and positions of vehicle parts. Individual rule stores an initial value that can adjust the position of each vehicle part according to the driver's body size, and the initial value is changed to the optimal value by learning. The learning engine selects and extracts new information from the stored data consisted of body dimensions and positions of vehicle parts. This engine also generates new rules accepting the new information. If a driver adjusts the positions of vehicle parts which are recommended by the inference engine, the learning engine receives the body sizes and the preferred positions of the driver as inputs and executes the learning, and generates new rules. The sensors measure the body sizes (e.g., height) of a driver. The actuators adjust the positions of vehicle parts. The overall system structure is shown in Figure 2, and the body dimensions and position of the vehicle components to be processed in the rule are shown in Table I.

III. INFERENCE ENGINE

In order to apply the VPPCS to a real vehicle, the rule-based reasoner must be able to infer consequences from a set of rules and body sizes in real time. For example, the actuators adjust the position of the drivers seat as soon as the drive is seated. To do this, the reasoner must infer the position of the drivers seat before the driver is seated. In this section, we propose a method to measure the reasoning speed of the inferences engine and to visualize the inferred results.

We use Drools as an inference engine to search for rules controlling vehicle components. Drools is an inference based rule processing engine that uses the Rete algorithm to infer conclusions. The Drools rule consists of a condition and an action, which will be executed if the given condition is true. An example of defined rules is shown in Figure 3.

The advantages of the rule processing engine are as follows [3][4].

- Focus on "what to do" rather than "how to do"
- It is easy to manage and change logics by separating data and logics
- Centralization of knowledge
- It is easy to explain the process which the conclusion is derived

The scenario for applying the VPPCS to an actual vehicle is as follows:

- a driver is close to a vehicle,
- the sensors measure the body size of the driver,
- inferring the positions of the vehicle parts,
- the driver is seated,
- adjusting the position of the vehicle parts,
- the driver adjusts the positions of the vehicle parts if the adjusted positions are inconvenient,
- storing the body sizes and the positions modified by the driver,
- learning using the driver's body sizes and the revised positions,
- producing the new rules reflected the learning results

Therefore, the positions of the vehicle parts deduced by the reasoner must be generated before the driver is seated. In order to measure the reasoning speed of the reasoner, we developed four components. They are a body size emulator that generates a user's body sizes, a set of rules to be used for inference, a rule-based reasoner, and a visualizer to display the reasoning results. In this simulation, 1,122 generated rules are used. The number of conditions included in each rule is ranged from one to four. The target vehicle parts and items of each part are shown in Table I. The range of body sizes used for the initial generation of the rules was the result of the physical examination of the new soldier recruits and the Korean standard body size table published by the Ministry of Commerce, Industry and Energy. The body size emulator randomly generates the height, sitting height, arm length, and leg length. The visualizer displays the number of rules matched with the generated body sizes and rule processing time. As four body sizes are created at one time and each rule contains under four conditions, the number of rules that match the generated

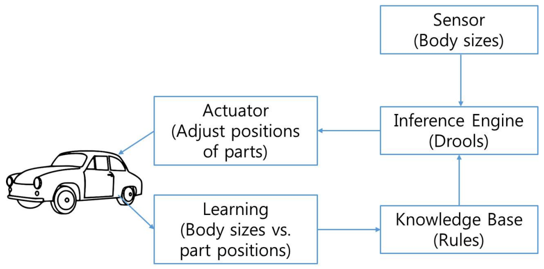
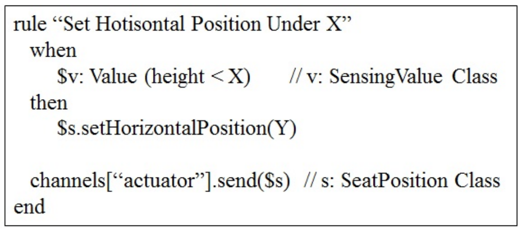


Figure 2. Components of VPPCS.



```
rule "Set Hotisontal Position Under X"
  when
     $v: Value (height < X)      // v: SensingValue  Class
  then
     $s.setHorizontalPosition(Y)

  channels["actuator"].send($s)  // s: SeatPosition Class
end
```

Figure 3. Example of position control rules.

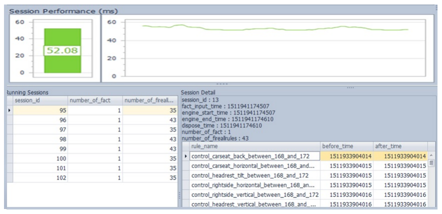
values may be four or more. The visualization tool is shown in Figure 4.

The rule-based reasoner is operated in Raspberry PI 2 which is an embedded computer to be used in a real vehicle. In Figure 4, the upper part shows the running time and the lower part shows the session data. The average processing time is 52ms which is shorter than the initial goal, 100ms. The session data includes the identifier of a session, the number of facts, and the number of fired rules.

Figure 5 shows the evaluation environment. We use five types of devices: a camera, a LiDAR sensor, a pressure sensor, actuators, and a Raspberry PI 2. The camera is used to measure a driver's height, the LiDAR sensor measures the distance between a person and the car, the pressure sensor checks whether a driver is seated or not, the actuators control the positions of the seat, and the Raspberry PI is used as a platform which runs the reasoner and the learner. If someone is closer than a given value, the camera takes a picture of the person and extracts his/her height. The height is sent to the reasoner, and the reasoner finds a suitable rule associated with it. The reasoner sends the positions in the found rule to the actuators. Then, the actuators change the vertical position, the horizontal position, and the backrest position of the driver's seat.

IV. Semantic Representation

As this work described in this paper is a part of an open source project, we will release all results developed in the project. The subjects which will be opened are divided into two groups. One is the source code and the other is the system architecture. The source code will be released onto GitHub or a web site and the architecture will be released as an ontology to improve understanding and sharing. Especially, we want to share the control rules with others connected to the Internet. If they exist on the Internet, we want to download them that someone has published. We will revise those to fit our environment and then publish them to the Internet again. If the rules do not exist, we will publish the rules created in our environment to the Internet. Although the automotive ontology was published [5], it does not contain the vehicle part



Running Sessions

session_id	number_of_fact	number_of_fireall...
95	1	35
96	1	43
97	1	35
98	1	43
99	1	43
100	1	35
101	1	35
102	1	35

Session Detail

```
session_id : 13
fact_input_time : 1511941174507
engine_start_time : 1511941174507
engine_end_time : 1511941174610
dispose_time : 1511941174610
number_of_fact : 1
number_of_fireallrules : 43
```

rule_name	before_time	after_time
control_carseat_back_between_168_and_172	1511933904014	1511933904014
control_carseat_horizontal_between_168_and...	1511933904015	1511933904015
control_headrest_tilt_between_168_and_172	1511933904015	1511933904015
control_rightside_horizontal_between_168_an...	1511933904015	1511933904015
control_rightside_vertical_between_168_and_172	1511933904016	1511933904016
control_headrest_vertical_between_168_and_...	1511933904016	1511933904016

Figure 4. Results of simulation.

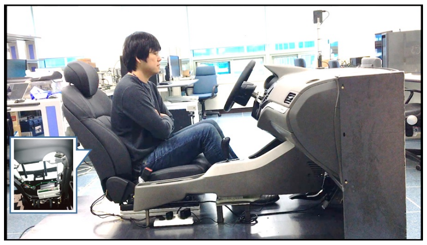


Figure 5. Evaluation environment.

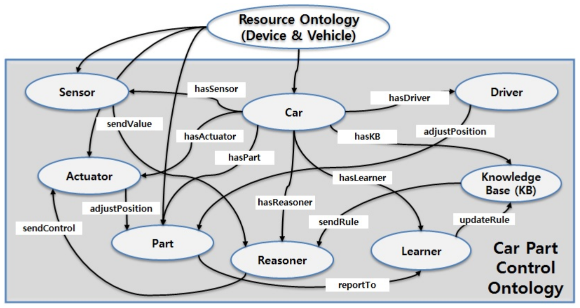


Figure 6. Car part control ontology.

position control. We define a domain ontology related to the vehicle part control. Figure 6 shows the ontology representing the relations among objects. The ontology consists of a car, a sensor, an actuator, a reasoner, a learner, a knowledge base, a vehicle part, and a driver. A driver is inherited from foaf:Person object.

V. Conclusion

In this paper, we describe a vehicle part position control system which will improve the driver's satisfaction in a car sharing service that many unknown people share the same vehicle. An inference engine in the system extracts the positions of vehicle parts from rules according to the driver's body size. Actuators automatically adjusts the positions of the vehicle parts according to the data sent by the inference engine. If the driver adjusts the position, a learner generates a new rule from the driver's body size and the new position. As the performance evaluation index, the number of rules matched the input values and the reasoning time are selected. A total of 1,122 rules are used to measure the inferencing time by using the four body sizes, and the average is recorded as 52 ms. This time is confirmed to be shorter than the initial target value. In the future, further experiments should be conducted to increase the complexity of the rules, to verify the reliability of the system, and to check the scalability when more rules are stored in the knowledge base.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2017-0-00501, Development of Self-learnable common IoT SW engine)

References

[1] J. Choi, J. Park, H. D. Park, and O. Min, "DART: Fast and Efficient Distributed Stream Processing Framework for Internet of Things," ETRI Journal, vol. 39, 2017, pp. 202–212, ISSN: 2-2-3-3-7-3-2-6.

[2] J. Collins and J. Hodgson, Eds., Integrating The Smart Home and The Connected Car. ABI Research, Oct. 2017.

[3] F. Aabedi and G. Etienne, Drools White Paper. LogiCoy, Jan. 2015.

[4] "Business Rules Engines - A White paper," 2012, URL: http://ratakondas.blogspot.kr/2012/06/business-rules-engines-white-paper.html [accessed: 2017-06-02].

[5] M. Feld and C. Muller, "The Automotive Ontology: Managing Knowledge Inside the Vehicle and Sharing it Between Cars," in Proceedings of the 4^{th} International Conference on Automotive User Interfaces and Interactive Vehicular Applications Nov 30–Dec 2, 2011, Salzburg, Austria. Worldwide Publisher, Nov. 2011, pp. 79–86, URL: https://dl.acm.org/citation.cfm?id=2381429 [accessed: 2018-02-15].

A Comparative Study of an Improved pAHP Implementation in SOA Architecture

Aleksander Marianski, Michal Kedziora and Ireneusz Jozwiak
Faculty of Computer Science and Management,
Wroclaw University of Science and Technology,
Wroclaw, Poland.
E-mail: Aleksander.marianski@pwr.edu.pl

***Abstract*—This article presents preliminary findings of practical comparative study of a new improved Probabilistic Analytic Hierarchy Process (pAHP) in comparison to classic AHP method in service selection problem in SOA architecture. The experiment illustrates the possibility of using an pAHP algorithm for selecting a proxy server in SOA architecture. A comparative study shows the use of the pAHP algorithm on data from servers of a real company. The effectiveness of the algorithm was tested and the statistically significance of pAHP over classical AHP was demonstrated in the defined case.**

***Index Terms*—SOA Architecture, Analytic Hierarchy Process.**

I. Introduction

A method for multi-criteria decision making is Analytic Hierarchical Process (AHP), presented by Thomas Saaty [1]. Classic AHP includes the reduction of decision problems to a series of comparisons and the synthesis of results using mathematical equations [2]. The AHP also includes the technique of examining the coherence of the decision-maker's assessments [3]–[8]. Classic AHP considers a set of criterias and a set of alternatives out of which the best alternative is determined [9]. A number of restrictions [10] of classic AHP [11] was our motivation to propose a new pAHP algirithm and to implement and assess its effectiveness for selecting optimal service in SOA architecture.

II. Probabilistic Analytic Hierarchical Process

The first step of pAHP is the acquisition of decision criteria and alternatives. The output of the step is a set of criteria and alternatives denoted by a and c. These are the vectors of alternatives and criteria, where $a^{(i)}$ is the i-th alternative, for $i \in \{1, 2, ..., m\}$, c_i is the i-th criterion for $i \in \{1, 2, ..., n\}$, m is the number of alternatives, n is the number of criteria [12]. In the second step, the process of determining the probability density function for the distribution of a random variable should start by calculation from the distribution of the random variable obtained from the sample based on the formula for Kernel estimation [13]:

$$\tilde{f}(x) = \frac{1}{mh} \sum_{i=1}^{m} K(\frac{x - x_i}{h}), \qquad (1)$$

where: m is the number of sample elements, h - smoothing parameter, x_i are the next numerical values from the sample, K - this function is called kernel [14]. The third step is to set the weights of the criteria and alternatives. The output will be a vector of alternative weights against the criteria: w_j, for $j = 1, 2, ..., n$, where n is the number of criteria or the vector weighting criteria: $w^{(0)}$. Vector w_j, for $j = 1, ..., n$ is m-dimensional, where m is the number of alternatives, while $w^{(0)}$ is n-dimensional [15].

The fourth step is final ranking. Results for each alternative are obtained using the formula:

$$v^{(i),k} = \sum_{j=0}^{n} w_j^{(0)} \cdot w_j^{(i)}, \qquad (2)$$

where: $v^{(i),k}$ is the point score of the i-th alternative in the k-th iteration, and $i \in \{1, ..., m\}$, $k \in \{1, 2, ..., N\}$, m is the number alternatives, N is the number of iterations, in (0) is the weight of the jth criterion, $w_j^{(0)}$ is the weight of the i-th alternative to j-th criterion for $i \in \{1, ..., m\}$ and $j \in \{1, 2, ..., n\}$, n is the number of criteria. The fifth step is to establish a probabilistic final ranking based on knowledge about the distributions. Our method is called P-THRESHOLD and calculates the probability of obtaining a score above a given threshold t: $v_{prob}^{(i)} = P(V^{(i)} > t)$.

III. Service selection with AHP and pAHP

In order to test the effectiveness of the pAHP, an algorithm for selecting a proxy server has been implemented. The environment based on Hetzner dedicated server type PX70-SSD enabled monitoring network traffic parameters such as: connection time (c_1), query response time (c_2), average transfer speed (c_3), success of delivery (c_4) which where used as algorithm criterias. The data forms where sent every minute to the test part of the office service. Four proxy servers were used. The observation lasted 30 days with 1800 parameter values collected for each hour. This was sufficient to determine for

TABLE I
Examples of average values of criteria

Alternative	c_1	c_2	c_3	c_4
$a^{(1)}$	0.54	12.21	25.2	0.87
$a^{(2)}$	2.12	10.12	35.4	0.90
$a^{(3)}$	0.31	3.42	192.3	0.91
$a^{(4)}$	0.87	15.23	20.4	0.90

each criterion, the alternatives and the time of the probability density function of the criterion value. Examples of average values are shown in Table I. Identical comparisons with the pairs of criteria were proposed in Table II. The weight

TABLE II
COMPARISONS IN PAIRS OF CRITERIA

	c_1	c_2	c_3	c_4
c_1	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{9}$
c_2	3	1	2	$\frac{1}{3}$
c_3	3	$\frac{1}{2}$	1	$\frac{1}{3}$
c_4	9	3	3	1

vector for criteria, calculated using the matrix's own vector method, is: [0.062, 0.224, 0.158, 0.556]. This vector of weights was adopted regardless of the hour, because the validity of the criteria does not change over time. Normalization for the pAHP was performed by the expected value method. The P-THRESHOLD method was chosen to determine the final results. Probability density function of ranking random variables is presented in Figure 1. For each hour, one proxy server was selected for the classical and probabilistic AHP. Only hours in which decisions were different were selected for the satisfaction survey. The company's server used alternately

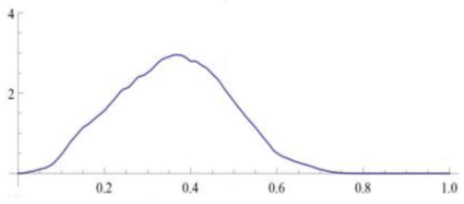


Fig. 1. Probability density function of ranking random variables

proxy servers selected by the classical and probabilistic AHP. The users assessed satisfaction with the shipment (quality of experience) according to their own criteria. The user did not know that he was involved in the study and had no knowledge about the method of sending data forms carried out by the company. The effectiveness of the algorithm is understood as

TABLE III
RESULTS OF THE USER SATISFACTION SURVEY

Grade	AHP	P-AHP
1	2	3
2	2	0
3	20	11
4	42	34
5	142	166
	222	226

the satisfaction of the system user. After sending the form, the user was asked about the satisfaction of shipping and assessed it on a scale of 1-5, where 1 is a lack of satisfaction, 5 is full satisfaction. The study was carried out for one week during the dispatch of annual testimonies. The test results were saved to the database and then made available to the author. The results are summarized in Table III. Classic AHP algorithm obtained the average of: 4.538, the variance 0.617 and the standard deviation: 0.785. The pAHP algorithm obtained an average of: 4.673, variance 0.503 and standard deviation: 0.709. Average rating indicates better results for pAHP.

IV. CONCLUSION AND FUTURE WORK

In this paper, we presented preliminary findings of comparative study of new improved probabilistic Analytic Hierarchy Process (pAHP) in comparison to classic AHP method. The experiment illustrated the possibility of using the implemented pAHP scheme for selecting a service in SOA architecture. The problem of selecting a proxy server to send queries was resolved. The statistically significance of probabilistic AHP over classical AHP was demonstrated in the defined case. The algorithm uses additional knowledge about probability distributions of criteria values for alternatives. This knowledge allows to make automated decisions without the participation of a decision maker. The conclusions confirm the thesis about the possibility of constructing a decision making algorithm in probabilistic conditions, which is significantly better than the classical Analytical Hierarchical Process. Our future work will focus on more detailed performance comparison of classical AHP and new improved pAHP by performing experiments on different case scenarios.

REFERENCES

[1] T. L. Saaty, "Analytic hierarchy process", Wiley Online Library, 1980
[2] M. Aruldoss, T. M. Lakshmi and V. P. Venkatesan, "A survey on multi criteria decision making methods and its applications", American Journal of Information Systems 1-1, 2013, pp. 31-43.
[3] T. L. Saaty, "The analytic hierarchy process - what it is and how it is used", Mathematical modelling 9, 3-5, 1987, pp. 161-176.
[4] T. L. Saaty, "An exposition of the ahp in reply to the paper remarks on the analytic hierarchy process", Management science 36, 3, 1990, pp. 259-268.
[5] T. L. Saaty, "How to make a decision: the analytic hierarchy process", European journal of operational research 48, 1, 1990, pp. 9-26.
[6] T. L. Saaty, "Decision making with dependence and feedback", The analytic network process, vol. 4922. RWS publications Pittsburgh, 1996.
[7] T. L. Saaty, Decision making with the ahp: Why is the principal eigenvector necessary", European journal of operational research 145, 1, 2003, pp. 85-91.
[8] T. L. Saaty, "Decision making-the analytic hierarchy and network processes (ahp/anp)", Journal of systems science and systems engineering, 13, 1, 2004, pp. 1-35.
[9] J. M. Lafleur, "Probabilistic ahp and topsis for multi-attribute decision making under uncertainty", In Aerospace Conference 2011 IEEE, 2011, pp. 1-18.
[10] G. Manassero, Q. Semeraro and T. Tolio, "A new method to cope with decision makers uncertainty in the equipment selection process", CIRP Annals Manufacturing Technology 53, 1, 2004, pp. 389-392.
[11] J. Rezaei, "Best-worst multi-criteria decision making method", Omega 53, 2015, pp. 49-57.
[12] Rosenblatt, M. Remarks on some nonparametric estimates of a density function. The Annals of Mathematical Statistics (1956), 832-837.
[13] I. Jozwiak, M. Kedziora and A. Marianski, "Service selection method with multiple probabilistic QoS attributes using probabilistic AHP", IJCSNS International Journal of Computer Science and Network Security, VOL.18 No.2, February 2018
[14] C. Lacour, P. Massart and V. Rivoirard, "Estimator selection: a new method with applications to kernel density estimation", Sankhya, 2016, pp. 1-38.
[15] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation", Journal of the Royal Statistical Society, Series B (Methodological), 1991, pp. 683-690.