International Journal of Informatics Society

07/20 Vol. 11 No.3 ISSN 1883-4566

Informatics Society

Aims and Scope

The purpose of this journal is to provide an open forum to publish high quality research papers in the areas of informatics and related fields to promote the exchange of research ideas, experiences and results.

Informatics is the systematic study of Information and the application of research methods to study Information systems and services. It deals primarily with human aspects of information, such as its qu ality and value as a resource. Informatics also referred to as Information science, studies t he structure, algorithms, behavior, and interactions of natural and a rtificial systems that store, process, access and communicate information. It also develops its own conceptual and theoretical foundations and utilizes foundations developed in other fields. The advent of computers, its ubiquity and ease to use has led to th e study of info rmatics that has computational, cognitive and social aspects, including study of the social impact of information technologies.

The characteristic of informatics' context is amalgamation of technologies. For creating an informatics product, it is necessary to integrate many technologies, such as mathematics, linguistics, engineering and other emerging new fields.

Guest Editor’s Message

Masao Isshiki

Guest Editor of Thirty-third Issue of International Journal of Informatics Society

We are delighted to have the Thirty-third issue of the International Journal of Informatics Society (IJIS) published. This issue includes selected papers from the Twelfth International Workshop on Informatics (IWIN2018), which was held at Salzburg, Germany, Sept. 9-12, 2018. The workshop was the twelfth event for the Informatics Society, and was intended to bring together researchers and practitioners to share and exchange their experiences, discuss challenges and present original ideas in all aspects of informatics and computer networks. In the workshop 26 papers were presented in seven technical sessions. The workshop was successfully finished with precious experiences provided to the participants. It highlighted the latest research results in the area of informatics and its applications that include networking, mobile ubiquitous systems, data analytics, business systems, education systems, design methodology, intelligent systems, groupware and social systems.

Each paper submitted IWIN2018 was reviewed in terms of technical content, scientific rigor, novelty, originality and quality of presentation by at least two reviewers. Through those reviews 20 papers were selected for publication candidates of IJIS Journal, and they were further reviewed as a Journal paper. We have three categories of IJIS papers, Regular papers, Industrial papers, and Invited papers, each of which were reviewed from the different points of view. This volume includes six papers among those accepted papers, which have been improved through the workshop discussion and the reviewers’ comments.

We publish the journal in print as well as in an electronic form over the Internet. We hope that the issue would be of interest to many researchers as well as engineers and practitioners over the world.

Masao Isshiki received his Ph.D in heat transfer engineering from the Tokyo University of Agriculture and Technology, and a master's degree in mechanical engineering from the University of Tokyo Institute of Technology. He has been working at Toshiba corporation for 27 years on consumer electronics business. Masao joined the W3C Team in January 2009 as the W3C/Keio site Manager as project professor of KEIO University. He left W3C in April 2014 and changed to professor for Kanagawa Institute of Technology (KAIT) from 2011. His research interests are IoT and smart house business. He has been working for the chairman of smart house standardization meeting project which sponsored by METI of government since 2011. He is a member of IPSJ, IEEE and JSME.

Industrial Paper

A Data Synchronization Method for a Vehicle Using Multimodal Data Features

Yosuke Ishiwatari[*][**], Takahiro Otsuka[*], Masahiro Abukawa[*], and Hiroshi Mineno[**]

[*] Information Technology R&D Center, Mitsubishi Electric Corporation, Japan
[**] Graduate School of Science and Technology, Shizuoka University, Japan
{Ishiwatari.Yosuke@dr, Otsuka.Takahiro@dw, Abukawa.Masahiro@bx}.MitsubishiElectric.co.jp, mineno@inf.shizuoka.ac.jp

Abstract - The installation of multiple sensors in vehicles for acquisition and analysis of data is gaining popularity, owing to the increasing diversity, miniaturization and inexpensiveness of sensors. However, these sensors are not necessarily integrated into the same overall system. For instance, an owner-installed dashboard camera may not be connected to the factory-installed network of the vehicle. Therefore, it is important to synchronize data from multiple sensor systems to analyze the relation between the time series datasets of multiple sensors. A time-synchronization method is popular for this purpose, but this approach is not suitable for synchronizing offline sensor data. In this study, we propose a method for synchronizing video data with acceleration data from a moving vehicle's onboard sensors that use image features to detect synchronization points, which are then matched to corresponding points in the acceleration data. We evaluated the performance of our method by comparing video data with acceleration data (both collected via smartphone) when the vehicle turns right or left. Using this approach, we found the error to be 39.103 milliseconds. We intend to expand and further optimize our methodology by extracting and comparing data from different driving scenarios.

Keywords: autonomous driving, multimodal, data synchronization, motion estimation of a vehicle.

1 INTRODUCTION

The installation of multiple sensors in vehicles for acquisition and analysis of data is gaining popularity, owing to the increasing diversity, miniaturization, and inexpensiveness of sensors. Autonomous driving is one of the applications using this approach, as autonomous vehicles have multiple sensors, such as global positioning system (GPS) receivers, cameras, and acceleration, laser, and radar sensors [1][2]. In addition, many consumer-grade vehicles use dashboard cameras (dashcams), which include a GPS receiver and/or some acceleration sensors. However, these sensors are not necessarily connected to the same central system, for instance, a dashcam is usually separated from other sensors within a vehicle (Fig.1). In this situation, multiple systems are therefore used for analyzing data acquired from multiple sensors. Vehicles that do not possess any sensors are rare. For some events such as a traffic accident in which the car is involved, if someone who was not involved in the accident (e.g., policeman or insurance representative) is investigating the driver's role in the accident, even if not at fault, the driver has to provide some form of proof of innocence. If the driver uses unsynchronized dashcam video and other sensing data, the proof depended on using the data separately. So, even if the driver could prove innocence by using synchronized data (for example, proving that the driver "was correctly braking when the accident occurred" by using integrated camera video and acceleration data), that driver may not be able to prove it by using the data separately. Many consumer-grade vehicles do not come with a camera installed, so it is common for drivers to equip a car with an after-market dashcam. For commercial-transportation vehicles [3], synchronization among multiple sensors (such as a dashcam and other sensors that are original equipment in a vehicle) is important.

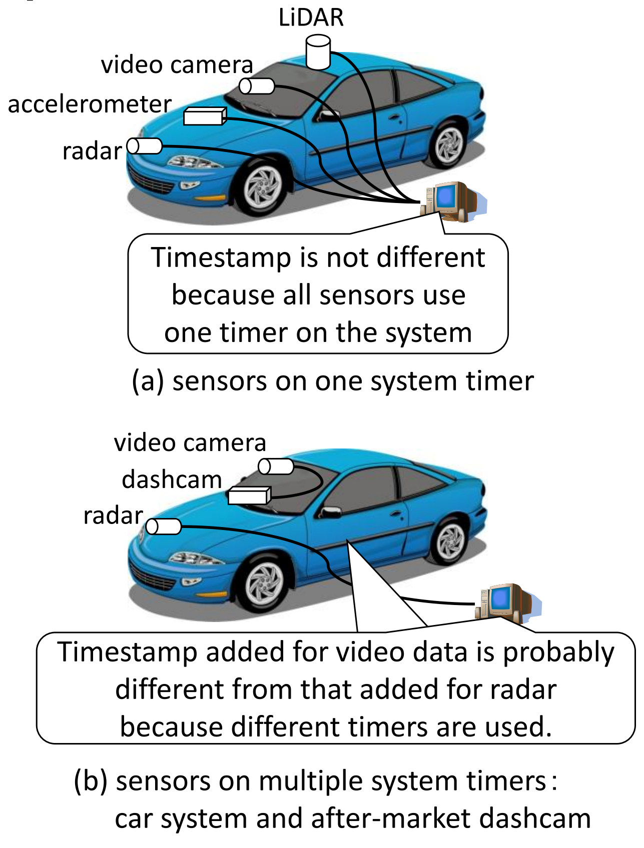

LiDAR
video camera
accelerometer
radar
Timestamp is not different because all sensors use one timer on the system
(a) sensors on one system timer
video camera
dashcam
radar
Timestamp added for video data is probably different from that added for radar because different timers are used.
(b) sensors on multiple system timers: car system and after-market dashcam

Figure 1: System timer and timestamps of sensor

Synchronization of acquisition times of different sensor data is very important for analyzing and enabling correlation between these data. If sensor data are acquired in a single integrated system, acquisition times are synchronized seamlessly; however, data obtained by multiple systems may not be synchronized because of different times recorded by the separate systems. Because this commonly occurs when a vehicle owner installs a dashcam or some other after-market device, as vehicles generally do not have a synchronization method for such situations, an extra system must be added.

Precise time differences between systems are needed to correct the discrepancies and enable data synchronization. Typical correction methods involve synchronizing system times or inserting timestamps in the sensor data [4][5], but because separate system clocks are different, system times become incorrect after multiple synchronizations over a long period of time. Too, when data from multiple systems are analyzed after all the data are recorded, differences between data-acquisition times cannot be obtained because the unsynchronized system times are not known when the data are recorded. Thus, if time synchronization is not in effect during data acquisition, it is impossible to synchronize data at a specific time to enable full use of all data.

Furthermore, if external time, such as GPS time, is used for synchronization, GPS receivers are needed for all systems, but if this external time is in error, such as when vehicles are passing through a tunnel or an urban area containing many buildings, this error is not detected during synchronization. Also, the cumulative cost of a GPS receiver for each sensor can be considerable, so it would be advantageous for synchronization to be possible without having to purchase multiple GPS receivers.

We have developed a method for synchronizing sensor data by extracting the data ranges of different vehicle motions through analysis of the characteristics of sensor data, without needing additional devices to record time data, thereby providing a more cost-effective method for synchronizing data from multiple sensors in a vehicle.

2 OUR PROPOSED METHOD AND RELATED STUDIES

2.1 Proposed Method

In this study, we aimed to synchronize video and acceleration data recorded by sensors in a vehicle, where each sensor was connected to a different system (Fig. 2). We intended to perform synchronization after rather than simultaneously with data acquisition.

Although the data-acquisition times for the various devices were approximately the same, the exact differences in times between the systems were unknown because the system clocks differed. Thus, for instance, because the differences in time between the camera and acceleration data were not always constant, if the synchronization was performed at one data point, it did not necessarily mean that all data points could be synchronized in a similar way. This was resolved by synchronizing some points of the data and correcting the data between these synchronizing points using the differences in time at the said points. Video and acceleration data also have many different characteristics, therefore reference points related to characteristics common to both data types are required for synchronization to be possible. Therefore, we propose a synchronization method that is based on the detection of vehicle-motion behavior from sensor data and matches ranges of data. The fact that we did not use time data to detect vehicle motion meant that the inconsistency in the differences in time (resulting from different clocks in the systems for each sensor) was negligible.

2.2 Proposed Vehicle-Motion Events

Because our method requires the use of multiple synchronization points, vehicle-motion events must be easily detectable and occur frequently while driving. However, easily detectable events such as passing over a bump (video result: vertical displacement; acceleration result: vertical vibration) and heavy braking (video result: variation in moving vector of objects; acceleration result: variation of acceleration in the direction the vehicle is moving) do not necessarily occur frequently in a given journey. Accordingly, we focused on extracting the required data from very common events such as when a vehicle is turned right or left, with the beginning and end of these turning events treated as synchronization points. It rarely occurs that a car is driving without turning right/left (i.e., the car is moving only straight ahead or back). In car-related studies, such as those concerning automonous driving, those algorithms/methods are evaluated by driving a variety of car motions, such as changing lanes, turning right/left at an intersection, or passing [6]-[12]. That means a car is not just moving straight ahead. Therefore, we chose to use the time when the car is turning right/left as a synchronizing point.

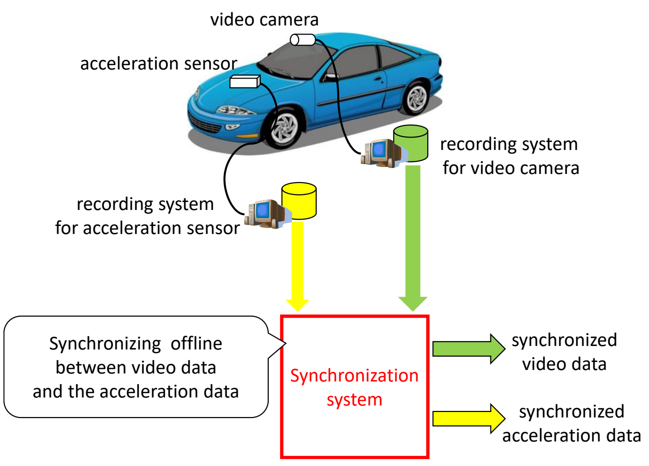

video camera
acceleration sensor
recording system for video camera
recording system for acceleration sensor
Synchronizing offline between video data and the acceleration data
Synchronization system
synchronized video data
synchronized acceleration data

Figure 2: Proposed system

2.3 Proposed Method

For synchronization between data that have different characteristics, it is necessary to convert the data and/or isolate the characteristics that make the data mutually comparable. For example, in order to match with acceleration data, car speed can be calculated by a bird's-eye view created by camera images [13]. If the speed is calculated correctly, synchronization of video data (each frame being the converted speed of the car) and acceleration data is possible, but according to Morimoto et al. [13], calculation accuracy is not good when the car is moving slowly. For example, when the speed is 20 km/h, the error is >10%, which means the synchronization error is >10%. Sometimes synchronization is impossible because the pattern of the speed differences and the values of the acceleration data are very different. Because such low-speed driving sometimes occurs in urban areas, their study [13] is not suitable for our research.

Another method is to convert optical flows to be comparable to events characterized by other sensor data. Fridman et al. [14] devised a method to synchronize sensor data using optical flow by detecting vibrating events or steering events; this method is good for relative synchronization but cannot enable absolute synchronization. Giachetti et al. [15] developed a method for estimation of egomotion using optical flow; however, it is obviously not useful for vehicles traveling on a flat road, where vertical displacement is negligible.

There are a lot of synchronization methods that focus on acceleration and camera imagery. For example, Tanaka et al. [16] described a method using a correlation value between sensor data without consideration of time, but it is applicable to acceleration data only and thus is not suitable for our system.

3 DETERMINING SYNCHRONIZAION POINTS USING IMAGE FEATURES AND CHARACTERISTICS OF ACCELERATION DATA

3.1 Overview

Our method consists of three functions, 1) detecting "turning right" and "turning left" events using image features, 2) detecting "turning right" and "turning left" events using acceleration data, 3) detecting synchronization points using detected events.

We describe the functions in the following sections.

3.2 Detecting "Turning Right" and "Turning Left" Events Using Image Features

We used the optical flows of image features for detecting behaviors of the vehicle as "turning right" or "turning left" from camera images. Because the optical flows of image features of stationary objects have vectors that are opposite in direction to those of a moving vehicle, we can acquire vectors from stationary objects that correspond to those of a moving vehicle. We can calculate the tendencies of the vectors from the optical flows of image features using whole frames. These image features are not solely on stationary objects; however, there are not many objects that move around the vehicle, so the tendencies of the vectors that can be regarded as a vector are the same as those of a vector that shows the movement of the vehicle. Fig. 3 depicts the optical flow when a vehicle turns right, showing how at this point a vector that is opposite to that of the moving vehicle can be acquired (i.e., a vector in which the direction is from the left to the right of the image as drawn).

When a vehicle is turning right or left, the optical flows from the image features on stationary objects are opposite to the direction in which the vehicle is moving. Thus, if a camera is recording in front of a vehicle, when the vehicle is turning right, optical flows turn left. Conversely, when a vehicle is turning left, optical flows turn right. Moreover, if a vehicle is not moving right or left, optical flows do not change, meaning that the direction of optical flow is very useful for detection of a vehicle-motion event.

Meanwhile, the vertical direction of the optical flow varies with the position of each image feature within the camera image. Fig. 4 is an example of "turning left," which shows that the vertical direction of the optical flow for each image feature varies from one to another.

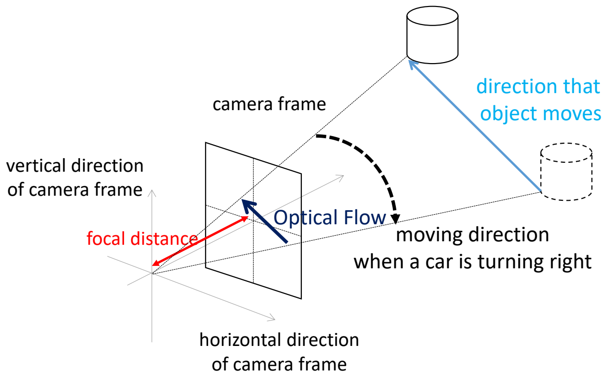

camera frame
direction that object moves
vertical direction of camera frame
Optical Flow
focal distance
moving direction when a car is turning right
horizontal direction of camera frame

Figure 3: Optical flow when car is turning right

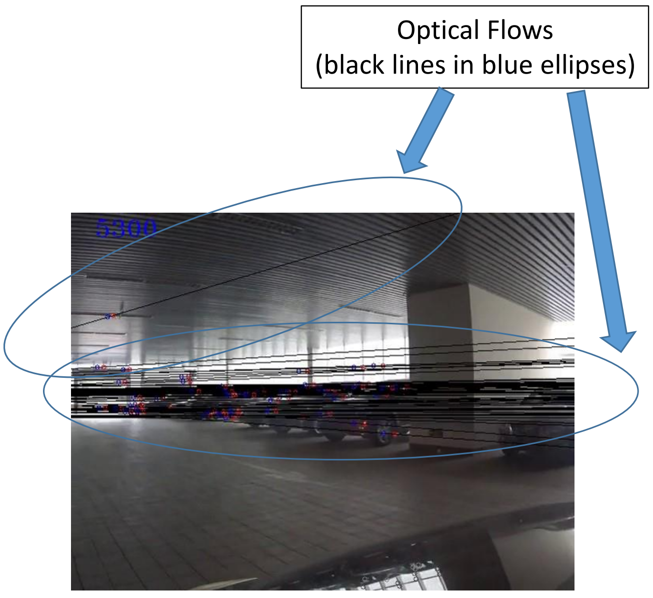


Figure 4: Optical flows of "turning left" event

3.3 Detecting “Turning Right” and “Turning Left” Events Using Acceleration Data

In this study, we hypothesized that we could acquire acceleration data in the horizontal direction. If a vehicle is moving and the velocity in the horizontal direction is always zero unless a vehicle does not turn right or left, we can detect the event of right/left turning by detecting whether the velocity in the horizontal direction is zero. This can be achieved using sensor data by calculating integration of the acceleration value. However, a vehicle is moving in the horizontal direction even if it is not turning right/left; therefore, we cannot detect vehicle-motion events by confirming that the velocity of the vehicle is zero.

When a vehicle turns right or left, the driver operates the steering wheel to move in the horizontal direction. Thus, this operation is equal to accelerating the vehicle in the horizontal direction, so the start of this operation causes a significant change in acceleration (Figs. 5 and 6). This means that this operation can be detected from the change in acceleration. The start and the end of this turning operation can be detected as a peak or inflection in the acceleration

In general, raw acceleration data from acceleration sensors includes some noise and bias that must be removed before calculating the peak or the inflection.

3.4 Detecting Synchronization Points

Based on the above method for detecting right/left turning events from video or acceleration data, we propose a method to synchronize video and acceleration data.

Our method comprises two functions. The first is a function that detects ranges of the frame that indicate the vehicle is turning right or left. The second is a function that calculates the difference between these ranges and the range of acceleration data by searching points that correspond with the desired data. To reduce the searching range, the times of the systems are approximately similar and the difference in time is not known. However, the start of the searching point (the point that would match the point of the other sensor if the difference were zero) can be determined.

In videos, “turning right/left” events have characteristics such that the optical flows tend to turn left/right. We therefore use these characteristics for detecting the ranges of the frames (Fig. 7), as follows:

1) Obtain the optical flows of the image features between successive frames;
2) Calculate the tendency of the vectors of the optical flows by classifying the vectors into 16 bins based on the direction of the vector, and select the bin that includes the majority of the vectors;
3) Calculate the range by counting the frames in which the bin of the start frame is the left bin (bins 7 and 8 in Fig. 7) or the right bin (bins 1 and 16 in Fig. 7) if the bin is near the former frame (within two consecutive bins).

The ranges calculated by the method are regarded as “turning left (continuously classifying left bin)” or “turning right (continuously classifying right bin).” Fig. 7 shows an example of classifying a “turning right” event and the event preceding it.

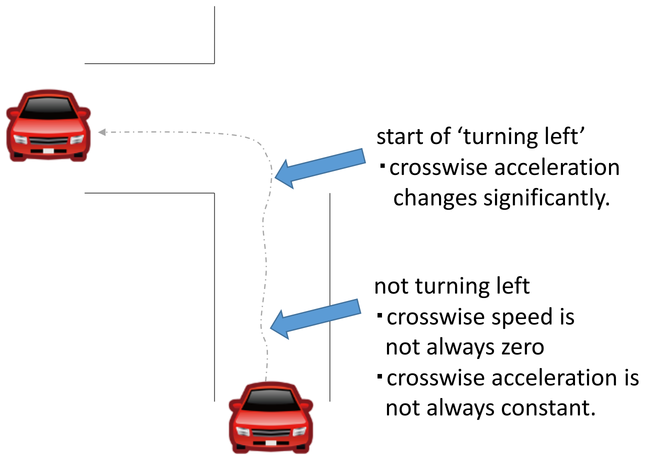

start of ‘turning left’
・crosswise acceleration changes significantly.
not turning left
・crosswise speed is not always zero
・crosswise acceleration is not always constant.

Figure 5: Acceleration change in “turning left” event

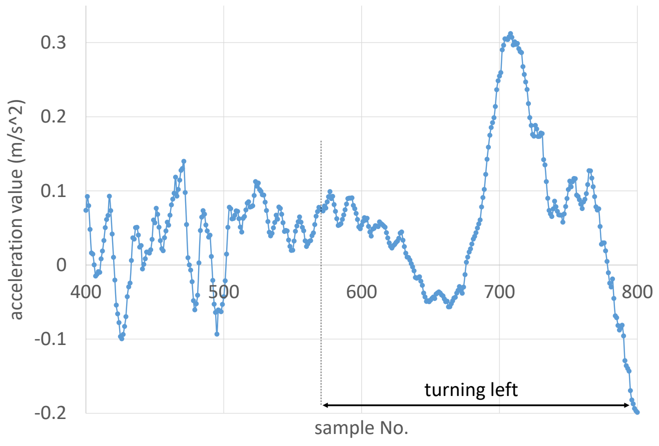


Figure 6: Change of acceleration in “turning left” event

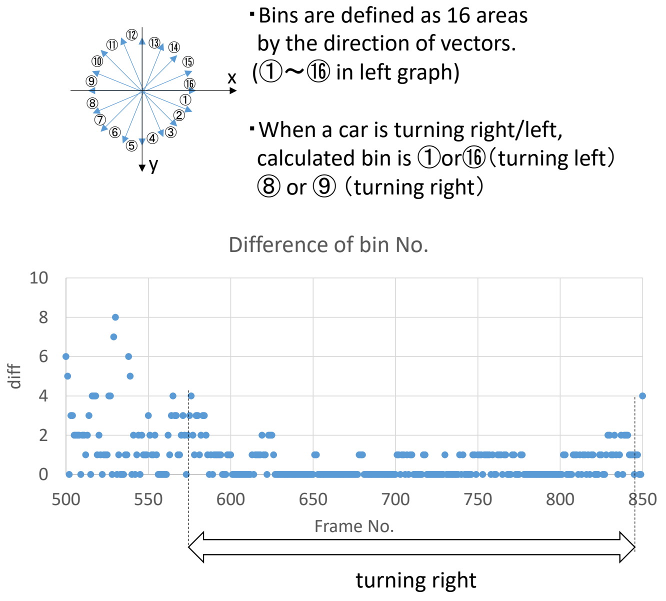


Figure 7: Bin differences in “turning right” event

In our earlier research [17], we discussed how the possibility of incorrect detection of a turning right/left event was controlled for checking whether such an event was a combination of more than one event. This may arise when another action occurs simultaneously with the vehicle turning right or left, such as riding on a curb or heavy braking. Thus, if the number of frames between two of the "turning right/left" events is very small (a few frames) and if these situations are the same (i.e., these situations are "turning right/right" and vice versa), these situations should be regarded as one event (Fig. 8).

After calculating the range of the frames, the matching points between the start/end frame of the desired range and the acceleration data are calculated. The data is examined for the presence of peak or an inflection point of the acceleration. A peak or an inflection point of the acceleration data closest to the start point is regarded as the corresponding point of the start/end frame.

After calculating the range of the frames, the matching points between the start/end frame of the desired range and the acceleration data are determined. The data is examined for the presence of acceleration peaks or inflection points. Either one closest to the start point is regarded as corresponding to the start/end frame.

After calculating the corresponding point of the start frame (point C_1) in the acceleration data (point A_1) and the corresponding point of the end frame (point C_2) in the same data (point A_2), A_1 and A_2 are corrected to have the same range of time between the range from C_1 to C_2 and the range from A_1 to A_2. In detail, point A_1 is moving to A_1' and point A_2 is moving to A_2'. Therefore, "(time of A_2')−(time of A_1') = (time of C_2)–(time of C_1)" and "(time of A_1)−(time of A_1') = −[(time of A_2)−(time of A_2')]." Accordingly, A_1' and A_2' are calculated as follows:

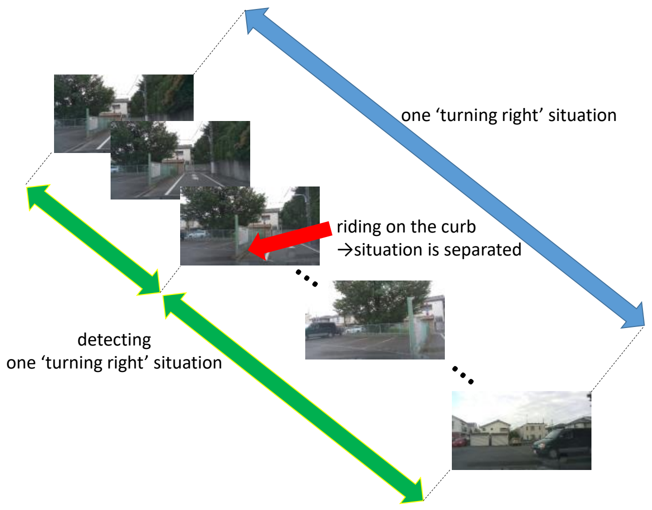


Figure 8: Separating one "turning right" situation into two situations

$$\Delta t_c = (\text{time of } C_2) - (\text{time of } C_1)$$
$$\Delta t_a = (\text{time of } A_2) - (\text{time of } A_1)$$
$$\text{diff} = (\Delta t_c - \Delta t_a)/2$$
$$\text{time of } A_1' = (\text{time of } A_1) - \text{diff}$$
$$\text{time of } A_2' = (\text{time of } A_2) + \text{diff}$$

Hence, D_1 = (time of C_1)−(time of A_1'), which is the difference in time between the start frame of the video and the start of the acceleration data of that range, and D_2 = (time of C_2)−(time of A_2'), which is the difference in time between the end frame of the video and the end of the acceleration data of that range. D_1 and D_2 are not always identical. Therefore, the difference value (ΔE) at time E (between A_1' and A_2') is calculated as follows:

$$\Delta \mathrm{E} = \frac{D_1 * ((time\ of\ A_2') - (time\ of\ E)) + D_2 * ((time\ of\ E) - (time\ of\ A_1'))}{(time\ of\ A_2') - (time\ of\ A_1')}$$

For synchronizing point X in the range other than at a right/left turning event, the difference (Δd_1) between the point X and point X_1 (the end frame of the range of the right/left turning that occurs immediately before the point X), and the difference (Δd_2) between point X and point X2 (the start frame of the range of the right/left turning event that occurs immediately after the correcting point X) is used. The time difference ΔX at point X is calculated as follows:

$$\Delta \mathrm{X} = \frac{\Delta d_2 * ((time\ of\ X) - (time\ of\ X_1)) + \Delta d_1 * ((timeof\ X_2) - (time\ of\ X))}{(time\ of\ X_2) - (time\ of\ X_1)}$$

4 FUNDAMENTAL EVALUATION OF OUR METHOD

We evaluated the fundamental accuracy of our method. The data and our evaluation method are described as follows.

4.1 Data Setting

We used video data and acceleration data acquired by a smartphone in a vehicle. The camera recorded the front view of the vehicle. The acceleration sensor recorded accelerations in three dimensions: the direction in which the vehicle moved, the horizontal direction of the vehicle, and the vertical direction of the vehicle. Fig. 9 shows the course we traversed with the smartphone-equipped vehicle.

The recorded time was also acquired for these data (camera frames and acceleration data).

For an evaluation, we decided on evaluation events (synchronization timing) to check a video and extract frames that correspond to the start or end point of turning right/left. We also calculated the acceleration data at each frame by linear interpolation. If those data are to be used in applying our method, the difference must be zero at all evaluation points, but some points had nonzero difference values because of noise. Accordingly, in this evaluation, we investigated robustness of our method.

In this evaluation, the number of right/left turning events was 26, and these were all used for evaluation of the method.

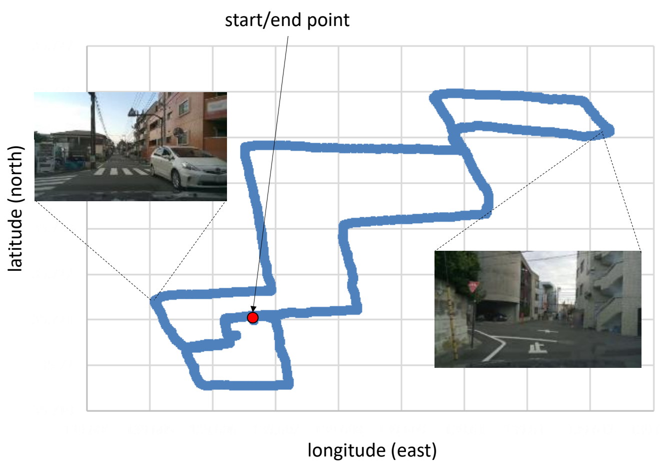


Figure 9: Course used in driving experiment

To reduce noise in the acceleration data, we used a smoothing process, and we applied an interpolation process to adjust the sampling rate of the acceleration data to the interval between the frames of the video (30 fps).

The ranges calculated by the method in some cases had overlapping subranges, which showed ranges that correlated with the same right/left turning event. Accordingly, we added a merge process within our method so as to match one range to one event. In that process, some ranges that have the same frames got merged into one range.

In addition, a range of one right/left turning event was sometimes split, so one event was sometimes described by more than one range. Accordingly, we added a concatenate process within our method so as to ascribe one range to one event. In that process, two ranges having an interval of one or two frames got merged into one range.

4.2 Parameter Setting

We searched for peaks or inflection points from the start point in the acceleration data. During evaluation, the search range was defined as 30 samples (corresponding to 1 sec.).

From the ranges calculated by the method, we extracted those spanning ≥120 frames (i.e., >4 sec.). This parameter is based on the result of an examination described by Fukuda et al. [18], using elapsed time of a turning right event as being 3–6 sec.

4.3 Experimental Results

We evaluated the 26 situations by calculating the difference between the start time of camera frames and the synchronized time of the acceleration data during right/left turning events, after applying our method (Fig. 10).

In Fig. 10, the calculated value (difference between camera frame and acceleration data) for each event is shown, together with the absolute value of each calculated data point. The average value is 39.103 milliseconds (= 1.173 frames), and the standard deviation is 46.026 milliseconds (= 0.824 frames). Event 14 has a large error compared to that of other events. That is because acceleration data are not varying at that point, so the shift in synchronization points is larger. We consider the cause to be data-acquisition error or bad noise filtering, and we intend to further investigate this kind of error.

In our experiment, we obtained a timing error of ~40 milliseconds between the video data (from an added dashcam) and the acceleration data measured by the factory-installed sensor in the vehicle. We propose a method for synchronization of the data from each of these sensors. If another sensor is added to the vehicle (e.g., a radar sensor at the front of the vehicle) is synchronized with acceleration data, this sensor will also be synchronized to video data.

For evaluating the accuracy of our method, we examined the accuracy of object detection by multiple sensors. When multiple sensors detect the same object, the detection results should theoretically be the same if sensor data are correctly synchronized. However, if synchronization is not correct, there is an error that, in the case of a moving vehicle, equates to a measurable distance. For example, in detecting pedestrians moving at 80 m/min, an error of 40 milliseconds equates to a 5.3 cm difference in distance, whereas in detecting cars moving at 60 km/h, this error is 66.7 cm. Accordingly, these distances can be considered negligibly small.

For calculating optical flows, we have to select two frames. If the interval between those two frames is short, the length of the optical flow tends to shorten. This means that the optical flow is significantly affected by the error resulting from the matching. By enlarging the interval between these two frames, the impact of the matching error can be smaller. However, enlargement of the interval means that the accuracy of the detecting ranges of the right/left turning event decreases, because the accuracy depends on the intervals. This may result in the process failing to detect some ranges. To avoid this problem, we evaluated the accuracy by alternating the intervals. Fig. 11 shows the average and the standard deviation of the errors at some intervals from no frame (optical flow is calculated using consecutive frames) to nine frames.

In Fig. 11, the average of errors is greatest at the interval “0frame” (more than 120% of the average at other intervals), and the average of errors and sum of average and standard deviations is smallest at the interval “6frames”. Although some intervals have greater averages or standard deviations, Fig. 11 seems to show that the magnitude of the average is inversely proportional to the magnitude of the interval. Fig. 12 shows the ratio of a number of corrected events to a number of detected events, which shows that the smaller the magnitude of an interval, the larger the ratio.

Figs. 11 and 12 show that both increasing and decreasing intervals have disadvantages, which suggest that use of the same optical flow for both detecting ranges of data and synchronizing data is not optimal. However, in our study, the use of the interval “0frame” for detecting range and the interval “6frames” for synchronizing data does appear to be acceptable.

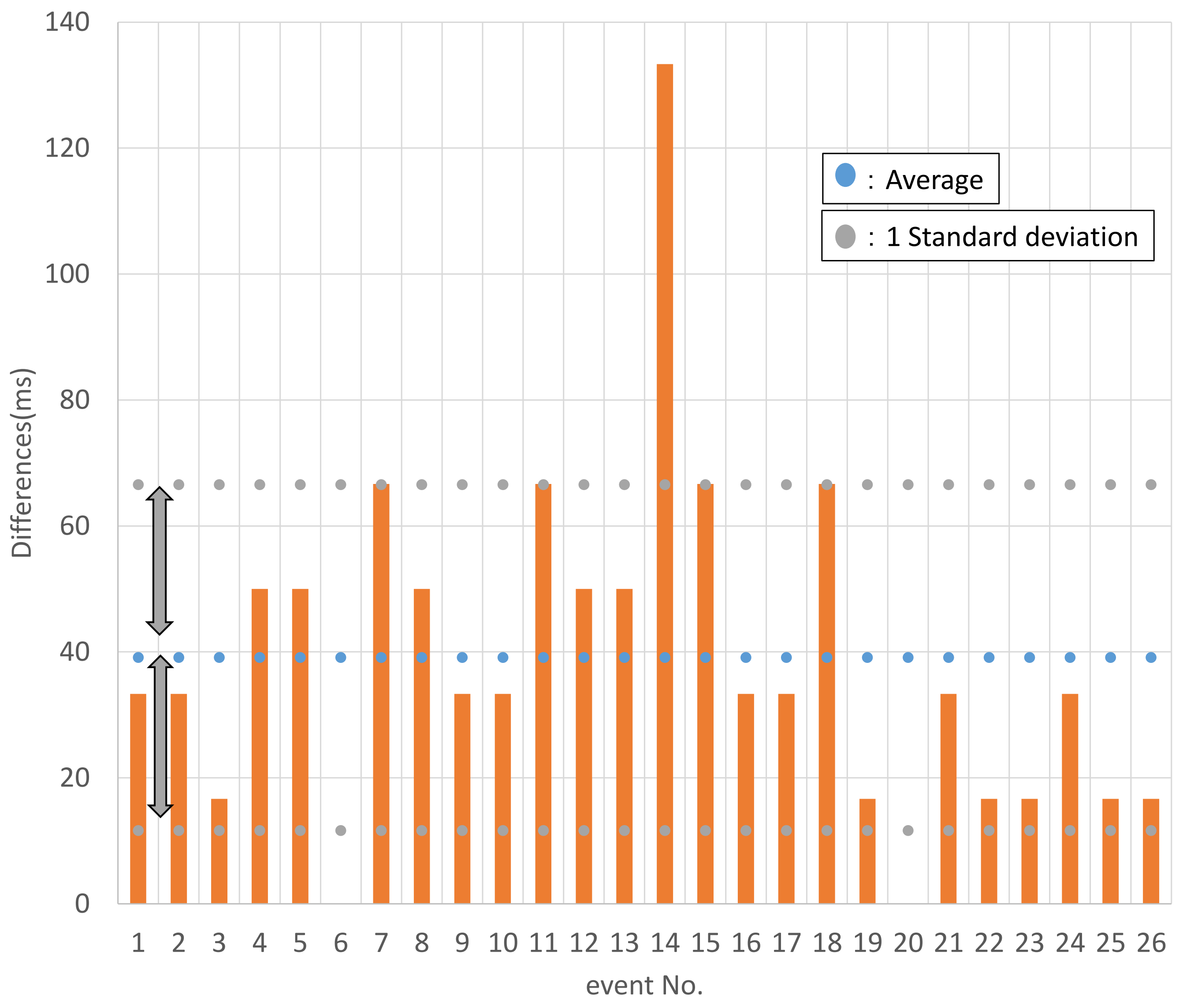


Figure. 10: Evaluation results

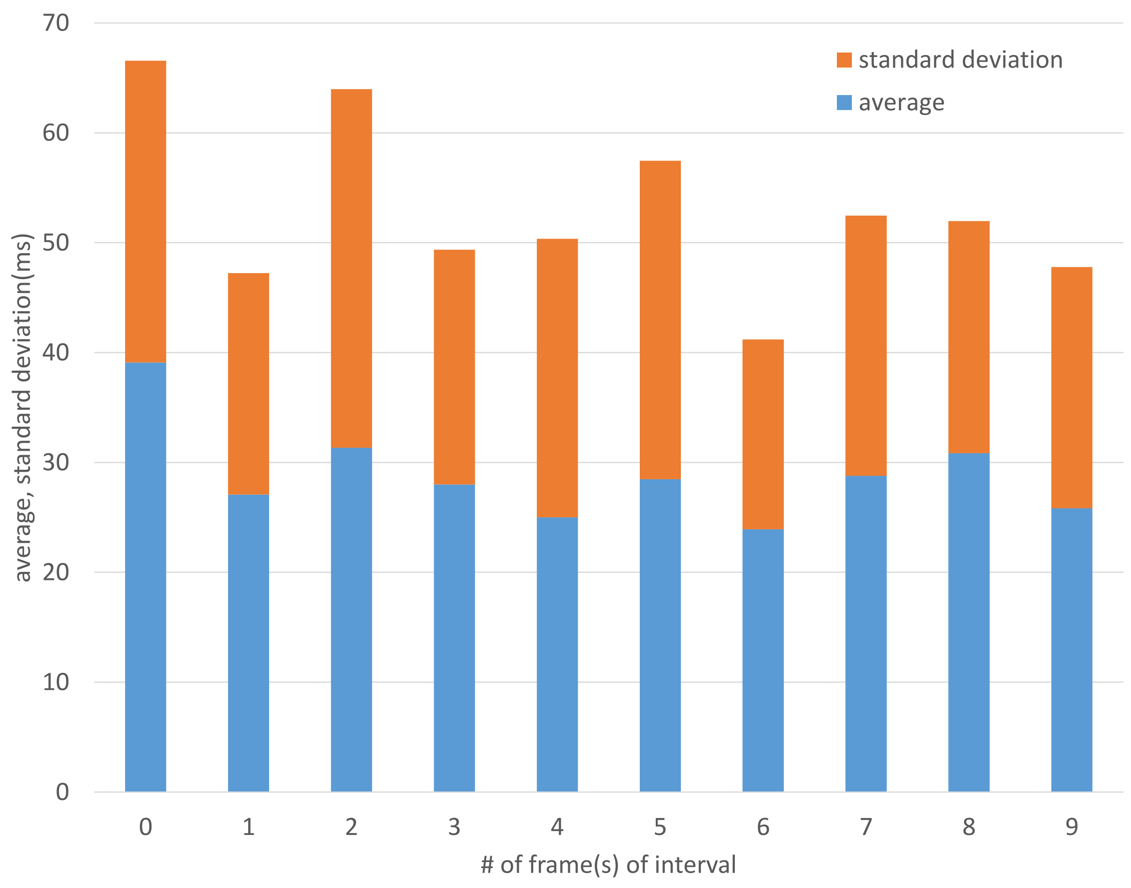


Figure. 11: average/standard deviation of errors and frame intervals

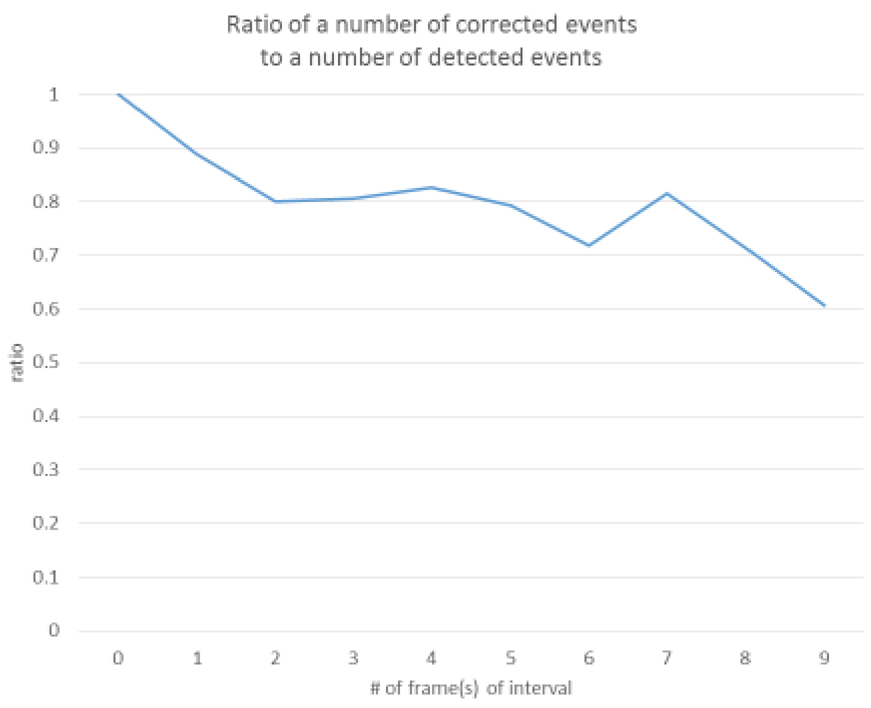
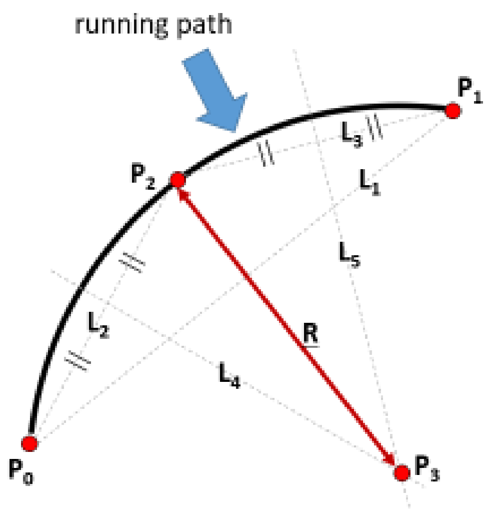


Figure. 12: Ratio of corrected events and frame intervals

In right/left turning events, moving radius and velocity of a car varies by situation. When the car is slowly turning, noise in the acceleration data may affect the result. Table 1 shows the moving radius (calculated by using the positions and algorithm (Fig. 13), velocity calculated approximately by integrating acceleration data and subtracting the integrated data at the point that the car is stopped nearest to but before the point because of drifts of the acceleration, and the difference value (Fig. 10)). In this evaluation, we applied noise reduction to the acceleration data, so it is assumed that the effect of the noise is small. Too, the correlation between errors and velocities is −0.151. That value means velocities are affected but only slightly. However, the correlation between errors and moving radius is 0.554, meaning that large radii tend to produce larger errors. We will investigate this more precisely in the future.

Table. 1: radius, velocity and error

event No.	radius [m]	approx. velocity [m/s]	error [# of frames]
1	5.32	1.96	1
2	5.25	2.22	1
4	8.69	0	1.5
5	2.04	0	1.5
6	37.37	0.70	0
7	18.52	0	2
8	31.20	0.91	1.5
9	31.54	1.74	1
12	45.94	1.00	1.5
14	355.41	0.80	4
15	232.86	2.36	2
16	143.80	0.50	1
23	135.38	0.60	0.5
24	210.75	1.73	1
25	98.92	1.76	0.5



Point P_0: start point of turning
Point P_1: end point of turning
Point P_2: a point on the running path
Point P_3: intersection of L_4 and L_5
Line L_1: joints P_0 and P_1
Line L_2: joints P_0 and P_2
Line L_3: joints P_1 and P_2
Line L_4: perpendicular bisector of L_2
Line L_5: perpendicular bisector of L_3

Turning Radius R: distance between P_3 and $P_2(P_0, P_1)$

Figure. 13: Calculating radius using points

4.4 Future Work And Discussion

In the 26 events, the average of the error frames is short, so it can be concluded that our methodology adequately accounts for the data. In future work, we will consider aspects such as the following:

1) Detecting the range of the frames more precisely

Our method detects the range of frames based on the tendency of the optical flow direction. If the range is short, the event corresponding to the range is not right/left turning but is similar in action to a part of the right/left turning event, e.g., an S-shaped curve.

2) Handling the difference between Δt_a and Δt_c (see section 3.4)

In our method, the difference between Δt_a and Δt_c is divided equally and used for A_1 and A_2. This is not always true, as the difference between A_1' and A_1 is not always the same as that between A_2' and A_2. We can potentially resolve this issue by matching correlation values.

3) Special situations: a lot of objects move in the same direction but other than the car's direction, etc.

In our method, we assumed that a lot of characteristic points have characteristics that are the same as for the motion of the car, and extracted frames as the right/left turning frames by checking that the frame had optical flows of mainly left/right direction. If the car is not turning left/right but other objects are moving mainly left/right, and the characteristic points are mainly on the objects, our method may detect that frame as right/left turning. However, that means other objects are moving across the car because a camera is recording in front of the car, thus the situation could occur that when the car is stopping, it crashes into those objects. We expect to be able to detect that situation and remove it from among the right/left turning situations.

In our evaluation, the video data has a situation where the car is stopping and waiting at the railroad crossing, and the situation occupies 1.4% of the frames, and our method was not detect that situation as right/left turning.

4) Problems in obtaining image features from objects

Our method assumes that the tendency of the optical flow direction is almost the same as the direction of the optical flows, based on the image features of the stationary object, so we assumed that multiple image features are acquired from an object that does not move in any frame. We currently use all the optical flows, but selecting the appropriate optical flows should enable improved detection and analysis. However, because our method applies all the frames, extensive calculation processes using optical flows will be required. We do not intend to use a heavily calculated method for, say, object detection, so this aspect will be carefully explored.

5) Problems associated with insufficient number of image features

Some frames do not possess multiple image features, for example when data is acquired at night. A frame may also be occupied by the sky or the ground with no lines, signs, and other objects.

6) Utilization of camera images from other than the front

As shown in Chapter 4, the installed camera is taking pictures of the front of the vehicle. Since the front is photographed, it is assumed that the optical flow swings from side to side when turning left to right. However, when using a camera image taken in a direction other than the front, this premise changes. In this situation, it would not be possible to use the optical flow obtained from the captured image, so it would be necessary to convert these optical flows to match those in the front direction.

The following is a matching example. It is assumed that a feature point (u_1, v_1) in frame F_i is corresponding to a 3D relative position $\overrightarrow{x_1} = (x_1, y_1, z_1)$ (where the X axis represents the right direction of the camera, the Y axis represents the optical axis of the camera, the Z axis represents the upward direction of the camera, and the origin is the camera sensor), and at the time in frame F_j, the same point as $\overrightarrow{x_1}$ is represented the 3D relative position $\overrightarrow{x_2} = (x_2, y_2, z_2)$, and (u_2, v_2) in frame F_j, and the vehicle advances $\triangle\vec{X}$ (described using vehicle coordinate system, X axis represents the right direction of the car, Y axis represents the front direction of the car, Z axis represents the upward direction of the car, and the origin is the position of the vehicle) from frame F_i to frame F_j (Fig.14). In that case, the following equations hold, where f is the focal length of the camera, p is the pixel interval of the camera image, and (C_X, C_Y) is the central position of the camera image.

$$u_1 = \frac{f}{p \cdot y_1} x_1 + C_X, v_1 = \frac{-f}{p \cdot y_1} z_1 + C_Y$$
$$u_2 = \frac{f}{p \cdot y_2} x_2 + C_X, v_2 = \frac{-f}{p \cdot y_2} z_2 + C_Y$$

And, an equation $\overrightarrow{x_2} = \overrightarrow{x_1} - R^{-1}\Delta\vec{X}$ holds, where is R represents the transformation matrix, which transforms the vehicle coordinate system to the camera coordinate system. So, if R and $\Delta\vec{X}$ is acquired, $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$ can be calculated from (u_1, v_1) and (u_2, v_2). When $\overrightarrow{x_1}$ and $\overrightarrow{x_2}$ are calculated, the feature points (u_1', v_1') and (u_2', v_2') in the front images are calculated, where $R^{-1}\overrightarrow{x_1} = (x_1', y_1', z_1')$ and $R^{-1}\overrightarrow{x_2} = (x_2', y_2', z_2')$.

$$u_1' = \frac{f}{p \cdot y_1'} x_1' + C_X, v_1' = \frac{-f}{p \cdot y_1'} x_1' + C_Y$$
$$u_2' = \frac{f}{p \cdot y_2'} x_2' + C_X, v_2' = \frac{-f}{p \cdot y_2'} x_2' + C_Y$$

Figs.15-17 are examples of this calculation. In the illustrated range, the vehicle is going straight after a right turn. Fig. 15 shows the most frequent bins calculated by the algorithm presented in this paper by using the optical flows of the same range obtained from the front camera. Fig. 16 shows the most frequent bins by using the optical flows obtained from camera images taken from the left side of the vehicle without matching. In Fig. 15, the optical flows after the right turn is mainly 'right to left', that means the vehicle moves the left side, but it is not shown at all in Fig. 16. Meanwhile, Fig. 17 shows the most frequent bins by using the same optical flows but matched using $R = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ (, where R means rotation by an angle 90° around the Z axis, as well as$\triangle\vec{X} = (v_x\Delta t, v_y\Delta t, v_z\Delta t)$, where (v_x, v_y, v_z) is the velocity of the vehicle and Δt is the interval of timestamp acquisition of the optical flows. Fig.17 has a similar tendency as Fig. 15. However, in this example, the velocity of the vehicle must correspond with the frame. In the assumptions of this paper, there is a gap between the timestamps of the velocity of the vehicle and those of the image frames, so the deviation will be included by the conversion method described above. In this paper, the gap is small, so we consider that the impact is minor.

7) Expanding our method to other situations

Our method uses all the optical flows in the frame to calculate the tendency of their direction. A right/left turning event is an appropriate situation to be detected using the method. However, our method is not suitable for some situations, such as moving straight ahead. In that situation, not all the optical flows turn in the same direction. The direction is determined according to the point of the image feature within the frame.

To expand our methodology, we can split a frame into subframes and calculate the tendencies within the subframes, followed by detection of the range based on the characteristics of those tendencies. This should enable data to be obtained when a vehicle is moving in a straight line, and thus enable sensors to be synchronized at any time during the journey of a vehicle.

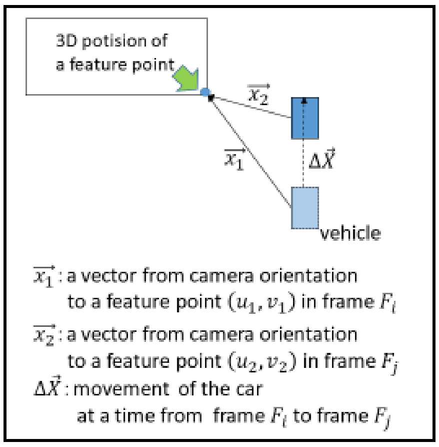

3D potision of a feature point
$\vec{x_2}$
$\vec{x_1}$
$\Delta\vec{X}$
vehicle
$\vec{x_1}$: a vector from camera orientation to a feature point (u_1, v_1) in frame F_i
$\vec{x_2}$: a vector from camera orientation to a feature point (u_2, v_2) in frame F_j
$\Delta\vec{X}$: movement of the car at a time from frame F_i to frame F_j

Figure. 14: The case of similar feature points at different frames

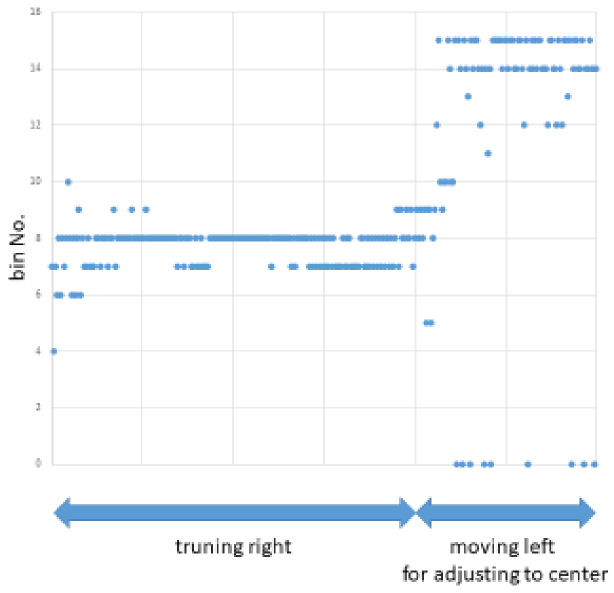


Figure. 15: Changes of bins of the case from images of the front of the car

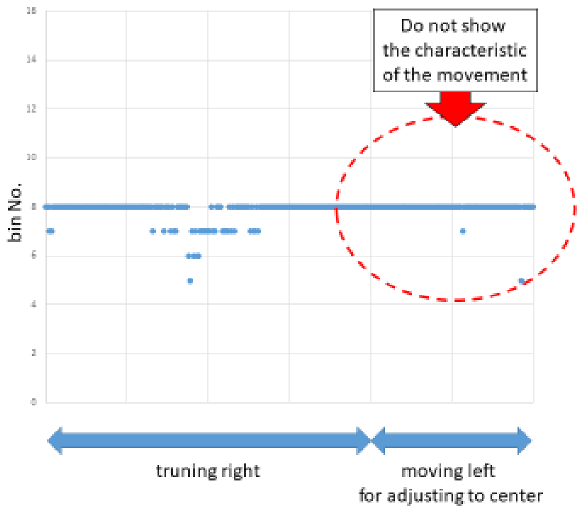


Figure. 16: Changes of bins from images taken from the left side of the car

5 CONCLUSION

In this study, we have proposed and evaluated a method to synchronize video and acceleration data from different sensors, connected to different systems, on a moving vehicle. In our method, we calculated the synchronization points by determining a right/left turning event from camera image data and acceleration data. From the camera image, we used the tendency of optical flows of the camera frame to detect the range of the event by continuously detecting the specific tendency of the corresponding vectors.

From the acceleration data, we detected the situation by identifying the peak or inflection point of the acceleration.

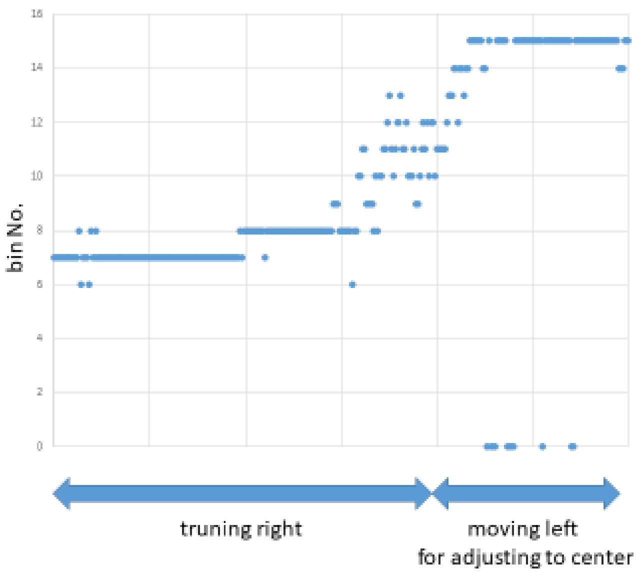


Figure. 17: Changes of bins from images from images of the left side of the car and matched

We evaluated the fundamental performance of our method using the camera image and the acceleration data acquired from a smartphone in a vehicle, and the error-frame average was 39.103 ms. However, some problems need to be addressed, such as improving the precision of detecting the range of an event from the camera image.

In addition, even though we determined that the difference between the acquisition time of video data and that of acceleration data is small, we will investigate a means of entirely removing this time difference, so as to further enhance the overall accuracy and utility of our method.

REFERENCES

[1] “Autonomous long-distance drive”, Mercedes-Benz, https://www.mercedes-benz.com/en/mercedes-benz/innovation/autonomous-long-distance-drive/

[2] Google Self-Driving Vehicle Project, https://www.google.com/selfdrivingvehicle/

[3] https://www.mlit.go.jp/common/001217658.pdf

[4] T. Kantonen, “Sensor Synchronization for AR Applications”, the 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp.245-246 (2010).

[5] S. Schneider, *et al.*, “Fusing Vision and LIDAR – Synchronization, Correction and Occlusion Reasoning”, 2010 IEEE Intelligent Vehicles Symposium, pp.388-393 (2010).

[6] S. Kitazawa, *et al.*, “Study of Update Timing of Control Target Calculation for an Autonomous Vehicle using Risk Potential”, 2018 JSAE Annual Congress (Spring), No. 54 (2018).

[7] R. Yanase, *et al.*, “Self-Localization based on Elevation Map for Autonomous Driving”, 2018 JSAE Annual Congress (Spring), No. 55 (2018).

[8] K. Sato, *et al.*, “Changes in Drivers’ Levels of Wakefulness during Automatic Driving in an Actual Driving Environment”, 2018 JSAE Annual Congress (Spring), No. 139 (2018).

[9] M. Obayashi, *et al.*, “Fast Optimization for Motion Planning Using MPC based on Approximated Problem”, 2018 JSAE Annual Congress (Spring), No. 238 (2018).

[10] S. Kitajima, *et al.*, “Development of Proving Ground Testing of Safety Related Performance Evaluation for an Automated Driving Prototype Car before Field Operation Tests”, 2018 JSAE Annual Congress (Spring), No. 340 (2018).

[11] K. Okamura, *et al.*, “Comparison of Trajectory Accuracy between RTK GPS and QZSS for the Autonomous Driving”, 2018 JSAE Annual Congress (Spring), No. 373 (2018).

[12] T. D. Son, *et al.*, “Autonomous Valet Parking Planning and Control Developments”, 2018 JSAE Annual Congress (Spring), No. 435 (2018).

[13] M. Morimoto, *et.al.*, “Estimating Vehicle Speed from In-Vehicle Monocular Camera Footage”, The Journal of The Institute of Image Information and Television Engineers, Vol. 68, No. 1, pp.J47-J54 (2014).

[14] L. Fridman, *et al.*, “Automated Synchronization of Driving Data Using Vibration and Steering Events”, Pattern Recognition Letters, Vol. 75, pp.9-15 (2016).

[15] A. Giachetti, *et al.*, “The Use of Optical Flow for Road Navigation”, IEEE Transactions on Robotics And Automation, Vol. 14, No. 1, pp.34-48 (1998).

[16] S. Tanaka, *et al.*, “Study of Time Synchronization Method between Multiple Wearable Sensors”, the 13th of the SOFT Kyushu Chapter Annual Conference. (2011).

[17] Y. Ishiwatari, *et al.*, “A Synchronization Method Between Movie and Sensor Data Using Directions of Vectors Determined by Corresponding Points Between Video Frames”, IPSJ SIG Technical Reports, 2018-ITS-73 (2018).

[18] K. Fukuda, *et al.*, “Effect of Experience on Estimate of Time for Turning across Opposite Traffic”, Transactions of Japan Society of Kansei Engineering, Vol. 11, No. 1, pp.97-102 (2012).

(Received January 16, 2019)
(Revised September 2, 2019)

Yosuke Ishiwatari received the B.E. and M.E. degrees from the University of Tokyo, Japan, in 1997 and 1999, respectively. In 2002, he completed coursework in Ph.D. program in information science at the Graduate School of Science, University of Tokyo. In 2003, he joined the Research and Development Center of Mitsubishi Electric Corporation as a researcher. He is mainly engaging in risk estimation for autonomous driving. He is also a member of IEEE, ACM, IEICE and IPSJ.

Takahiro Ohtsuka received his B.E., M.E., and Ph.D. degrees from the Utsunomiya University, Japan, in 1997, 1999, and 2002, respectively. Since 2002 he has been primarily engaged in signal processing, machine learning, and artificial intelligence at the Research and Development Center of Mitsubishi Electric Co., Japan.

Masahiro Abukawa received his B.E. degree from Yokohama National University in 1990. He joined Mitsubishi Electric Corp. Currently he is a general manager of Human Intelligent Technology Dept. at Information Technology R&D Center. He is also a member of IPSJ.

Hiroshi Mineno received his B.E. and M.E. degrees from the Shizuoka University, Japan in 1997 and 1999, respectively. In 2006, he received his Ph.D. degree in information science and electrical engineering from the Kyushu University, Japan. Between 1999 and 2002, he was a researcher at the NTT Service Integration Laboratories. In 2002, he joined the Department of Computer Science of Shizuoka University as an Assistant Professor. He is currently a Professor. His research interests include intelligent IoT systems as well as heterogeneous network convergence. He is also a member of ACM, IEICE, IPSJ, and the Informatics Society.

Regular Paper

Detection of Malware Infection based on the Similarity between Malware Infected Traffic

Masatsugu Ichino[†], Yuuki Mori[†], Mitsuhiro Hatada[‡], and Hiroshi Yoshiura[†]

[†]Graduate School of Informatics and Engineering, The University of Electro-Communications, Japan
[‡]NTT Communications, Japan
ichino@inf.uec.ac.jp, y-mori@uec.ac.jp, m.hatada@ntt.com, yoshiura@uec.ac.jp

Abstract - New types of malware are appearing every day, and malware attacks have become an urgent problem. Current methods of detecting malware use malware signatures, which need to be identified and registered in advance. However, the daily appearance of new types of malware makes such identification and registration impractical. A more practical approach is to identify malware on the basis of traffic behavior since each malware type displays a unique behavior. We have developed a method for detecting malware infection using traffic models based on the similarity between traffic of malware samples. Malware-infected traffic is divided into clusters on the basis of traffic behavior, and a model representing each cluster is created. These models are used to identify target traffic samples as infected or normal. This method should enable the detection of infection caused by a new type of malware if the malware's traffic behavior is similar to that represented by one of the models. Simulation evaluation demonstrated that the proposed method can effectively identify malware-infect traffic with high accuracy. And we discussed the created models and effectiveness using the models created by proposed method. We also discussed the detection of unknown malware using the models created by proposed method.

Keywords: security, malware, malware detection, traffic, clustering

1 INTRODUCTION

New types of malware are appearing every day, and malware attacks have become an urgent problem. Current methods of detecting malware use malware signatures, which need to be identified and registered in advance. However, the daily appearance of new types of malware makes such identification and registration impractical. A more practical approach is to identify malware on the basis of traffic behavior.

This paper focuses on infection detection, which we broadly classify as malware detection, intrusion detection, and infection detection. Intrusion detection typically involve techniques for detecting unauthorized access from a network before a malware infection occurs. Infection detection involves techniques for detecting an existing malware infection from network traffic as usual. Malware infections have become more difficult for users to detect, so infections have spread more widely without users knowing that their computers are being used maliciously. Therefore, infection detection for personal computers and middleboxes in the network such as routers and firewalls is an important measure for preventing the spread of infection.

The research reported here focused on the use of traffic data to detect infection. This approach determines the features of normal communication traffic and of infected traffic and uses pattern recognition techniques to detect infections. Infection detection based on traffic data uses only the incoming and outgoing communication traffic of the target machine. Basically, traffic is generated if there is an infection, so this method holds promise as a means of detection from outside the target machine. That is, malware infections are externally detected by observing the target machine's traffic patterns when it connects to a network.

Each malware type displays a unique behavior and a unique communication pattern when an infected terminal is connected to a network. The unique communication pattern comprises association confirmation of the infected terminal to the internet, surveying of the network environment, communication between the command and control (C&C) server and the infected terminal, and so on. Whether the malware is known or unknown, some malwares exhibit common communication behavior when the terminal is infected with malware.

In this paper, we propose malware infection detection using the traffic models based on the similarity between malware-infected traffic samples. It works by creating feature values on the basis of the time series of the traffic data, clustering malware-infected traffic samples in accordance with the similarities between them, and creating a representative model for each malware cluster. Detection of unknown malware infection is important research theme. The target of this paper is to classify the malware infection traffic as malware infection. This paper focuses on the detection method of malware infection included in the unknown malware infection. Malware-infected traffic is divided into clusters on the basis of traffic behavior. A model of each cluster is created, and the models are used to identify target traffic as infected or normal. This method will enable infections caused by new types of malware to be detected if the resulting traffic behavior is similar to that represented by one of the models.

This paper is organized as follows. Section 2 introduces related work, Section 3 describes the proposed method, Section 4 describes the evaluation method, Section 5 presents the key results, Section 6 discusses the results, and Section 7 concludes the paper with a brief summary of the key points and a mention of future work.

2 RELATED WORK

There have been various studies of malware detection using traffic data. Some used the definitions provided by security vendors for detecting malware infection, and some did not.

Studies in the first group (e.g.,[1]-[5]) classified malware traffic samples into groups on the basis of the definitions and created models of infected traffic for each group. However, security vendor definitions are not always based on the characteristics of infected traffic. It is thus better to create models on the basis of the characteristics of infected traffic.

Some studies in the second group ([6]-[8]) created models of infected traffic on the basis of the characteristics of infected traffic but did not consider the time series of the traffic data and the similarities between malware-infected traffic samples. Other studies ([9]-[11]) created models of infected traffic on the basis of the characteristics of infected traffic considering similarities between malware-infected traffic samples but did not consider the time series of the traffic data. Still other studies ([12], [13]) created models of infected traffic on the basis of the characteristics of infected traffic considering the time series of the traffic data but did not consider the similarities between malware-infected traffic samples.

Traffic data is a stream of network information, and previous studies have demonstrated the effectiveness of considering the time series of traffic data. Consideration of the similarities between malware-infected traffic samples is also necessary for representing common characteristics of infected traffic.

Therefore, in our study, we created feature values by considering the time series of each malware traffic sample. Next, we divided the malware samples into clusters on the basis of their similarities. Then, we created models representing the common characteristics of the infected traffic for each cluster.

3 PROPOSED METHOD

Our method is based on the detection of malware-infected traffic by using models representing each common traffic characteristic of malware. Proposed method conducts three parts shown in Fig. 1. As outlined above, feature values are created by considering the time series of the traffic data, malware samples are clustered by considering the similarities between them, and a representative model is created for each malware cluster. Labels **(Step 1)** $\cdots$ **(Step 6)** used in the following subsection correspond to the step numbers in Section 4.2.1 , in which we describe the proposed method as an experimental procedure with experimental datasets.

3.1 Create Feature Values by Considering Time Series of Traffic Data

(Step 1) Extract features of training data

To create a feature vector representing the time series of the traffic data, the time series is divided into 1-s time slots, as shown in Fig. 2. The traffic data is a set of packets captured from network traffic. The time slots are then grouped into intervals lasting a defined number of seconds for analysis.

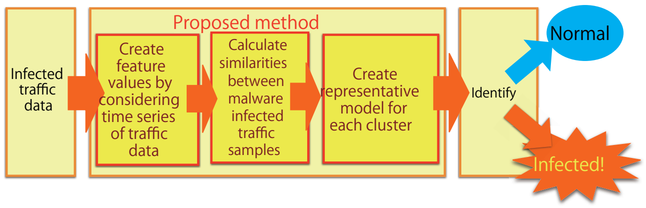

Infected traffic data
Proposed method
Create feature values by considering time series of traffic data
Calculate similarities between malware infected traffic samples
Create representative model for each cluster
Identify
Normal
Infected!

Figure 1: Outline of proposed method

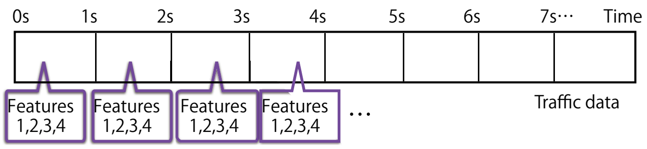


Figure 2: Division of time series of traffic data into 1-s time slots

Dividing time-varying traffic into time slots with a fixed duration and monitoring that traffic in units of time slots enables normal and infected traffic to be distinguished by focusing on the overall temporal variation in that traffic. In this work, we set the time-slot width to 1 s and determined the features for every time slot. The feature values are calculated for each time slot, and a feature vector concatenating the feature values is created for each time slot. In this study, we used minimum packet size per time slot, number of SYN packets per time slot, ratio of SYN packets to TCP packets per time slot, and number of ACK packets per time slot as the feature values. In our previous study, we analyzed traffic data after a malware infection and clarified which features would be the most effective in the detection of infection. It focused on using traffic data to detect infections and on the use of features that do not change much over time from those of the training data. In the evaluation, minimum packet size per time slot, number of SYN packets per time slot, ratio of SYN packets to TCP packets per time slot, and number of ACK packets per time slot as the feature values were effective features[9]. In this study, we represented the time-slot information as a four-dimensional feature vector by concatenating the feature values.

3.2 Cluster Malware Samples by Considering Similarities Between Malware Infected Traffic Samples

(Step 2) Create codebook for training data

To represent the traffic data as a code sequence, the set of feature vectors created as described in Section 3.1 is clustered (in this study, we used the LBG + splitting vector quantization algorithm [14]), and code is created for each cluster. A cluster is a mass of feature vectors and is divided on the basis of the distribution of data in the feature space. The code is the representative value of the cluster. The codebook is the code set. For example, when four-cluster clustering is applied, four codes (a, b, c, and d) are created. And we call the set of four codes the codebook.

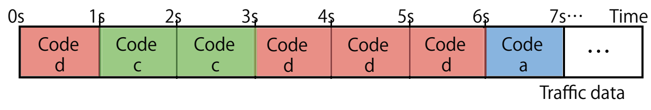


Figure 3: Example nearest code

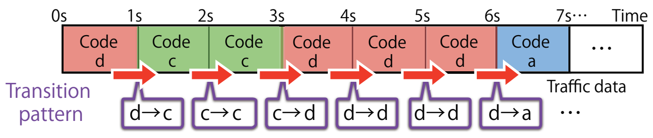


Figure 4: Example transition pattern

(Step 3) Create time-series representation of training data

The distances between the feature vector of target time slot and the code for each cluster is calculated, and a search is made for the nearest code. The time slot is then shifted, and a search is again made for the nearest code shown in Fig. 3. This series of nearest codes is called a "transition pattern." An example transition pattern is shown in Fig. 4.

The number of occurrences of each transition pattern per time interval (for example 40 s) is counted, and the ratio of each target transition pattern to all types of transition patterns is calculated, as shown in Fig. 6. The time interval is then shifted, and the number of occurrences of each transition pattern per time interval is again counted, and the ratio of each target transition pattern to all types of transition patterns is calculated.

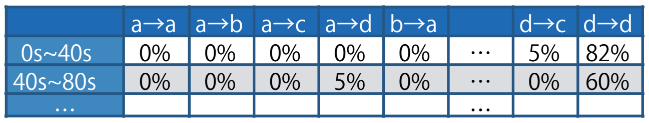
(Step 4) Calculate similarity(correlation coefficient) between each pair of samples in training data

To evaluate the similarities between two malware traffic samples, their correlation coefficient is calculated using the occurrence frequency ratios, like those shown in Fig. 5. The correlation coefficient represents the correlation between each sample's digital sequence of the number of transition patterns $\times$ the number of time intervals. The coefficient is calculated using

$$\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}, \tag{1}$$

where x and y are the probability variables, $\bar{x}$ is the mean value of x, $\bar{y}$ is the mean value of y, and n is number of transition patterns $\times$ the number of time intervals.

Calculation of the correlation coefficient requires that the n of x equals the n of y, as shown in Fig. 6. However, each malware traffic sample has a variable number of time intervals because each malware sample has a variable number of infected time intervals. The number of time intervals is thus adjusted by applying dynamic programming matching (DP matching) to the digital sequences of the two samples. The correlation coefficient is calculated using the adjusted digital sequences. DP matching adjusts the time lengths of the two samples by considering the time-series information and stretching the parts that are similar between the samples.



	a→a	a→b	a→c	a→d	b→a		d→c	d→d
0s~40s	0%	0%	0%	0%	0%	…	5%	82%
40s~80s	0%	0%	0%	5%	0%	…	0%	60%
…						…		

Figure 5: Example occurrence frequency ratios

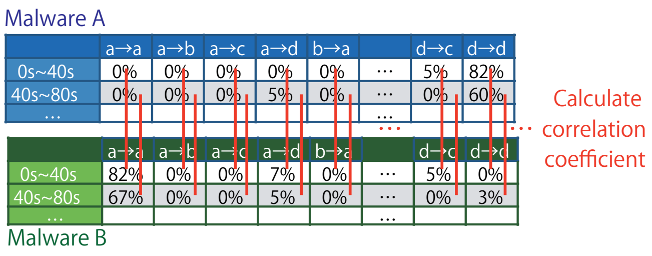

Malware A
a→a a→b a→c a→d b→a … d→c d→d
0s~40s 0% 0% 0% 0% 0% … 5% 82%
40s~80s 0% 0% 0% 5% 0% … 0% 60%
…
Calculate correlation coefficient
a→a a→b a→c a→d b→a … d→c d→d
0s~40s 82% 0% 0% 7% 0% … 5% 0%
40s~80s 67% 0% 0% 5% 0% … 0% 3%
…
Malware B

Figure 6: Calculate correlation coefficient

3.3 Create Representative Model for Each Malware Cluster

(Step 5) Create representative model for each malware cluster for training data

A representative model is created for each malware cluster by extracting a representative malware sample.

The malware samples are classified using hierarchical clustering based on correlation coefficients (we used the nearest neighbor method as hierarchical clustering). A high correlation coefficient means the similarity is high. The malware sample that has the most traffic samples with a correlation coefficient greater than an upper threshold is selected as the initial representative malware sample for the cluster. Since the optimal number of clusters is unknown in advance, hierarchical clustering is used as it does not require advance setting of the number of clusters. We extracted the malware sample that had the most malware's traffic samples with a correlation coefficient greater than the threshold(upper threshold) as the initial representative malware sample for the cluster. By the same token, the malware samples for which the correlation coefficient between the two malware traffic samples is less than the threshold (lower threshold) are deselected in each cluster to remove the samples for which the correlation is weak. This clustering is repeated until all training samples are divided into clusters.

The extracted malware traffic sample for a cluster is used as a representative model for that cluster in order to model sequential traffic data that actually occurred.

3.4 Detection of Infection

(Step 6) Calculate similarity between two samples in testing data

The time-series features of the testing data are created using Steps 1 and 3. The similarity between each representative cluster model and the target malware traffic sample is calculated, and the similarity between the model of normal traffic and the target malware traffic sample is calculated. The two similarities for each sample are compared, and the sample is identified as infected or normal.

4 EVALUATION

4.1 ExperImental Datasets

We used the anti-Malware engineering WorkShop (MWS) Datasets [15] for our evaluation. In particular, we used the CCC (Cyber Clean Center) DATAset and the D3M (Drive-by Download Data by Marionette) Dataset for training. As malware-infected traffic data, we used 317 malware samples (151 from CCC DATAset 2010, 156 from CCC DATAset 2011, and 10 from D3M 2012) for the training data such that the data used for training were older than the data used for testing. The normal traffic data used for training were captured between 2011 and 2015.

We also used the CCC DATAset and D3M Dataset for testing. As malware-infected traffic data, we used 200 malware samples (177 from CCC DATAset 2011, 15 from D3M 2013, 5 from 2014, and 3 from D3M 2015) for the testing data such that the data used for testing were newer the data used for training.

The CCC 2010 and CCC 2011 attack data include communications prior to malware infection. Thus, given the purpose of our evaluation, we extracted from this attack traffic only the traffic following malware infection using the method described by our group et al. [9].

4.2 Experimental Methods

To evaluate the effectiveness of the proposed method, we compared its detection performance with that of three reference methods.

4.2.1 Detection Using Proposed Method

As shown in Fig. 7, using our proposed method, we performed six basic steps .

(Step 1) Extract features of training data

Step 1-1 We divided the traffic data into 1-s time slots. We used the packet header information because the payload information was often encrypted.

Step 1-2 From each time slot, we extracted four features that we had determined to be effective for infection detection: minimum packet size per time slot, number of SYN packets per time slot, ratio of SYN packets to TCP packets per time slot, and number of ACK packets per time slot. The four features are evaluated as effective features for infection detection in [9].

Step 1-3 We normalized the extracted feature values by using the min-max method.

Step 1-4 We represented the time-slot information as a four-dimension feature vector by concatenating the normalized values.

(Step 2) Create codebook for training data

Step 2-1 We applied the LBG+splitting vector quantization algorithm to the vectors with the cluster number set to four.

Step 2-2 We calculated a representative vector (cluster center) for each malware cluster and collected the vectors into a codebook representing the characteristics of each cluster.

(Step 3) Create time-series representation of training data

Step 3-1 We calculated the distances between the feature vector of time slot and each code. And we assigned code of minimum distance to the time slot. The time slot is then shifted, and a search is again made for the nearest code.

Step 3-2 We assigned a code to all time slots.

Step 3-3 We counted the frequency of each transition pattern in each time interval and represented the ratio of the frequencies as time-series information. There were 16 transition patterns (a$\rightarrow$a, a$\rightarrow$b, $\cdots$, d$\rightarrow$c, d$\rightarrow$d) because we used four codes. We set the time interval to 10, 20, 30, 40, or 60 s. For example, when we set the time interval to 10 s, we calculated the frequency of each transition pattern in each 10-s interval (comprising ten time slots) and calculated the ratio of the frequencies of each transition pattern. We then shifted the time interval and calculated the frequency of each transition pattern per interval and calculated the ratio of each target transition pattern to all types of transition patterns.

(Step 4) Calculate similarity (correlation coefficient) between each pair of samples in training data

We calculated the correlation coefficient between each pair of malware samples. A total of 50,086 ($={}_{317}\, C_2$) correlation coefficients were calculated for each interval. We adjusted the time length (number of transition pattern $\times$ number of time interval) of each pair of malware samples by using DP matching.

(Step 5) Create representative model for each malware cluster for training data

We performed hierarchical clustering using the correlation coefficients calculated in Step 4. In multi-variant analysis, the correlation between each pair of samples was evaluated using the following criterion based on correlation coefficients C [16].

$0.0 \leq$ C ≤ 0.2 : barely correlated

$0.2 \leq$ C ≤ 0.4 : weakly correlated

$0.4 \leq$ C ≤ 0.7 : a little strongly correlated

$0.7\leq$ C ≤ 1.0 : strongly correlated

The calculated correlation coefficients were used as measures of the similarity between malware-infected traffic samples. The higher the coefficient, the stronger the correlation. The malware samples for which the correlation was very high were grouped together. On the basis of the above criteria, a coefficient greater than 0.7 generally means that the correlation is very strong. Therefore, we set the upper threshold to 0.7. The lower the coefficient, the weaker the correlation. The malware samples for which the correlation was very low were removed from the cluster. On the basis of the above criteria, a coefficient less than 0.2 generally means that the correlation is very weak. Therefore, we set the lower threshold to 0.2. Given these criteria, we selected the malware sample that had the most traffic samples with a correlation coefficient greater than 0.7 as the initial representative malware sample for the cluster. To keep a somewhat high correlation between each pair of malware traffic samples in the cluster, we deselected the malware samples that did not correspond to more than 70% of samples in the cluster; that is, the correlation coefficient was more than 0.2.

(Step 6) Calculate similarity between two samples in testing data

Step 6-1 We created the time series features of the testing data using Steps 1 and 3.

Step 6-2 We created a model of normal traffic using Steps 1 to 5.

Step 6-3 We calculated the cumulative minimum distance between each representative cluster model and the target malware traffic sample and calculated the cumulative minimum distance between the model of normal traffic and the target malware traffic sample.

Step 6-4 We compared the two distances for each sample. If the distance between the representative cluster model and the sample was greater than that between the model of normal traffic and the sample, the sample was identified as normal. Otherwise it was identified as infected.

4.2.2 Detection Using Time-Slot Method

For detection using time slots, we did not use both the time-series information and the similarity between malware-infected traffic samples. Instead, we created four codes from the malware infection training data using Steps 1 to 2 and created four codes from the normal training data using Steps 1 to 2.

We calculated the distances between the vector for the target time slot and the four codes for infection. Of the four distances calculated, the minimum one was selected as the similarity for infection. Moreover, the distances between the vector for the target time slot and the four codes for normal were calculated. Of the four distances calculated, the minimum distance was selected as the similarity for normal. Next, we compared the two similarities. If the one for infection was smaller than the one for normal, the time slot was identified as infected. If the one for normal was smaller than the one for infection, the time slot was identified as normal.

We applied the same process to all time slots of each malware traffic sample. If the ratio of the number of infected time slots to number of all time slots was more than the threshold (20%, 50%, or 70%), we identified the target traffic sample as malware-infected.

4.2.3 Detection Using One Representative Model

For detection using one representative model, we used the time-series information. We did not use the similarity between pairs of malware samples. The average malware traffic sample of the training data was treated as the representative model of malware-infected traffic.

We created the time-series information for the target malware traffic samples using steps 1 to 3 above. We calculated the mean ratio of the frequencies of each transition pattern for all malware traffic samples and selected the sample that was closest to the mean as the representative model of malware-infected traffic.

For testing, we created time-series information for the malware traffic samples using steps 1 to 3 above. We calculated the cumulative minimum distance between the target sample and the model of infected traffic. We also calculated the cumulative minimum distance between the sample and the model of normal traffic and identified the sample as normal or malware-infected on the basis of the two distances.

4.2.4 Detection Using Models Based on Security Vendor's Definitions

For detection using models based on a security vendor's definitions, we used the time-series information and clusters for classification. We did not use the similarity between malware-infected traffic samples.

We created time-series information for the target malware traffic sample using steps 1 to 3 above. Next, we divided the training malware traffic samples into clusters defined by the security vendor: BKDR, PE, Mal, TROJ, andWORM. We calculated the mean ratio of the frequencies of the transition patterns of the malware samples in each cluster. We selected the sample in each cluster with the frequency closest to the mean as the representative model of malware-infected traffic.

We calculated the cumulative minimum distance between the model of malware-infected traffic and target traffic sample and calculated the cumulative minimum distance between the model of normal traffic and target sample. We identified the sample as normal or infected on the basis of the distances.

5 RESULTS

5.1 Identification Rate of Proposed Method

The identification rate of the proposed method by changing the time interval is summarized in Table 1. The time interval is the duration during which the code transitions were counted, as described in section 3.2. The number of patterns of infected traffic is the number of hierarchical clusters, as described in section 3.3. The identification rate is the number of correctly identified malware-infected traffic samples divided by the total number of such samples in the testing data.

The identification rate was 100% for time intervals of 10, 20, 30, and 40 s, meaning that it is robust against the time interval.

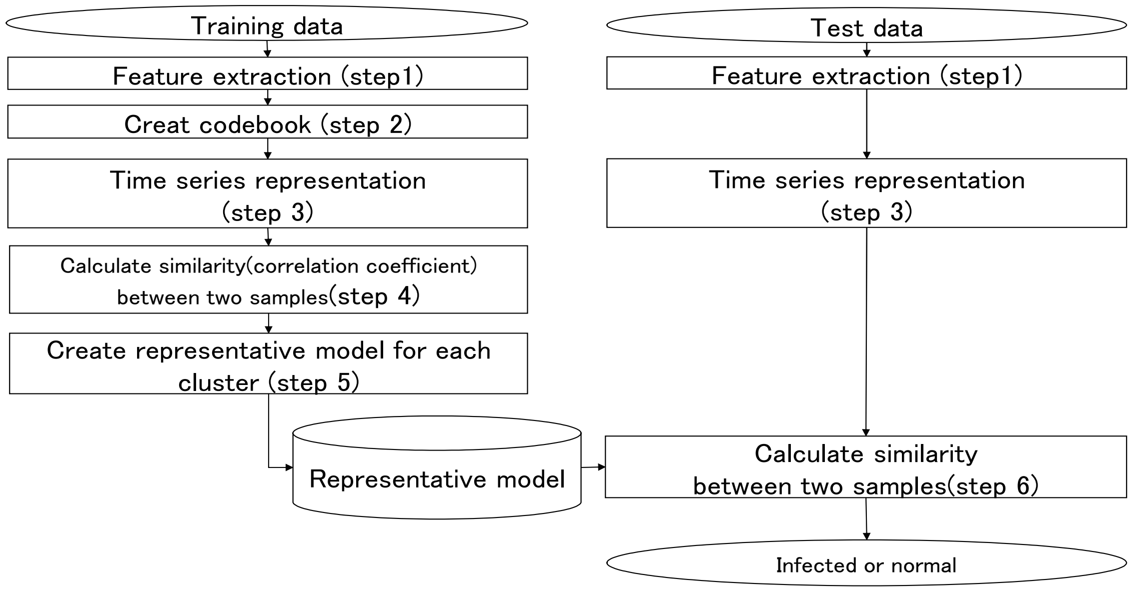

Training data
Feature extraction (step1)
Creat codebook (step 2)
Time series representation (step 3)
Calculate similarity(correlation coefficient) between two samples(step 4)
Create representative model for each cluster (step 5)
Representative model
Test data
Feature extraction (step1)
Time series representation (step 3)
Calculate similarity between two samples(step 6)
Infected or normal

Figure 7: Overview of experiment

Table 1: Identification rate of proposed method

Time interval (s)	No. of patterns of infected traffic	Identification rate (%)
10	17	100
20	15	100
30	12	100
40	12	100
60	11	99.0

Table 2: Identification rate of time-slot method

Rate of infected time slots (threshold)	Identification rate
20%	87.0%
50%	22.5%
70%	16.5%

5.2 Identification Rate of Other Methods

To evaluate the effectiveness of proposed method, we compared its identification rate with those of the three reference methods. The identification rate of the time-slot method (section 4.2.2) is shown in Table 2. The rate of infected time slots is the number of time slots identified as infected divided by the total number of infected time slots in each traffic data. It is used for identifying whether a traffic sample is infected or normal. The identification rate of the one-representative-model method (section 4.2.3) is shown in Table 3. The identification rate of the security-vendor-definition-based method (section 4.2.4) is shown in Table 4. The number of patterns of infected traffic is five because the data used included five malware families.

Table 3: Identification rate of one-representative-model method

Time interval (s)	No. of patterns of infected traffic	Identification rate (%)
10	1	12.5
20	1	14.5
30	1	25.5
40	1	32.5
60	1	52.5

Table 4: Identification rate of security-vendor-definition-based method

Time interval (s)	No. of patterns of infected traffic	Identification rate (%)
10	5	47.5
20	5	78.0
30	5	92.0
40	5	87.5
60	5	98.5

6 DISCUSSION

6.1 Representative models

We first discuss the effectiveness of increasing the number of models, i.e., the number of patterns of infected traffic. As shown in Tables 1, 2, 3, 4, the proposed method had the highest identification rate.

To evaluate [the effectiveness to represent some models of infected traffic, we analyzed the main transition pattern of each model. As described in section 3.3, a transition pattern is the transition of the codes in a time slot. To analyze the main transition pattern of each model, we show the traffic features of each code by analyzing the traffic data near the code. The

Table 5: Traffic Features of Each Code

Code	Feature(s)
a	SYN send
b	ACK send
c	UPnP, CBrowsing, SSL communication (digital sign etc.)
d	DNS communication (name resolution), RST/ACK send, UPnP

Table 6: Main transition pattern of each model with proposed method

Traffic pattern	No. of malware samples	representative model	Main transition pattern
0	219	BKDR_IRCBOT	dd
1	79	WORM_DOWNAD	a → a
2	6	WORM_DOWNAD	a → a, Cd → d
3	3	WORM_DOWNAD	b → b
4	2	TROJ_KRYPTIK	c → d, Cd → c
5	2	WORM_DOWNAD	d → d, Ca → a
6	1	WORM_DOWNAD	a → a, Cd → d
7	1	TROJ_MAILBOT	a → a, Cd → d
8	1	WORM_DOWNAD	b → b
9	1	BKDR_SMALL	b → d, Cd → b
10	1	WORM_DOWNAD	d → d, Cc → c
11	1	WORM_ALLPLE	a → a

results are summarized in Table 5. Each code represents one or more traffic features.

The main transition pattern of each model with the proposed method is shown in Table 6.The proposed method created 12 representative models, and each traffic pattern had bias of main transition pattern. In contrast, the one-representative-model method created one representative model, as shown in Table 7. A comparison of the main transition patterns of proposed method with that of the one-representative-model method shows that the latter is included in the main transition patterns of proposed method. It also shows that there is a big difference between the identification rate of the two methods. Therefore, the number of traffic patterns with the latter method is insufficient. A greater number of models is needed to represent the infected traffic. The transition pattern depends on the number of feature classes. If the feature is classified to more complex classes, it is unavoidable that combination explosion will occur. In this study, we set to four codes for vector quantization algorithm.

6.2 Effectiveness of Detection

We discuss the effectiveness of detection by proposed method. As mentioned, the proposed method had the highest identification rate. To evaluate the effectiveness to represent the model of proposed mehotd, we calculated the minimum value of the cumulative minimum distance between each representative model and the target malware traffic sample in the training data. We calculated the mean value of the minimum value of all combinations of representative models and all malware traffic samples in the training data. The shorter the distance between the representative model for each traffic pattern and all malware traffic samples in the training data, the better the representative models represent all malware traffic samples in the training data. We calculated the minimum value of the cumulative minimum distance between each representative model and the malware traffic samples in the testing data. We also calculated the mean value of the minimum value of all combinations of representative models and all malware traffic samples in the testing data. We did the same for the model based on the vendor's definitions and the one representative model. These results are summarized in Table8.

Table 7: Main transition pattern of one representative model

Representative model	Main transition pattern
WORM_DOWNAD	d → d

Table 8: Mean value of minimum distance between representative model and all malware traffic samples

Method	Training data	Testing data
Proposed method (time interval 10 s)	10.34	27.17
Proposed method (time interval 20 s)	3.02	4.45
Proposed method (time interval 30 s)	10.19	13.61
Proposed method (time interval 40 s)	5.30	5.99
Security-vendor-definition-based method (time interval 60 s)	12.67	15.62
One-representative-model method (time interval 60 s)	5603.65	3534.65

The proposed method (20-s time interval) had the minimum distance for the training and testing data. It was about a quarter that of the security-vendor-definition-based method for the training data. It is about 1/1800 that of the one-representative-model method (the method without clustering of malware samples) for the training data. The shorter the cumulative minimum distance, the better the models of infected traffic patterns represent the features of all the traffic. Therefore, the proposed method represents the infected traffic pattern better than the two other methods shown in Table 8.

We investigated the transition pattern of malware samples classified as each traffic pattern. To show the difference of transition pattern, we show a example of the histogram of transition pattern of two worms in Fig. 8 and Fig. 9. In these figures, horizontal axis is transition pattern and vertical axis is ratio of appearance transition pattern. The worm shown in the Fig. 8 is classified as traffic pattern 0. The worm shown in the Fig. 9 is classified as traffic pattern 1.

The outline of the histogram of two worm samples is difference each other. There are many transition patterns of d → d in the Fig. 8. There are many transition patterns of a → a in the Fig. 9. Worm malwares are selected as two traffic patterns. So worm malwares are separated into some groups. The results demonstrate that it need to represent infected traffic data with some traffic patterns.

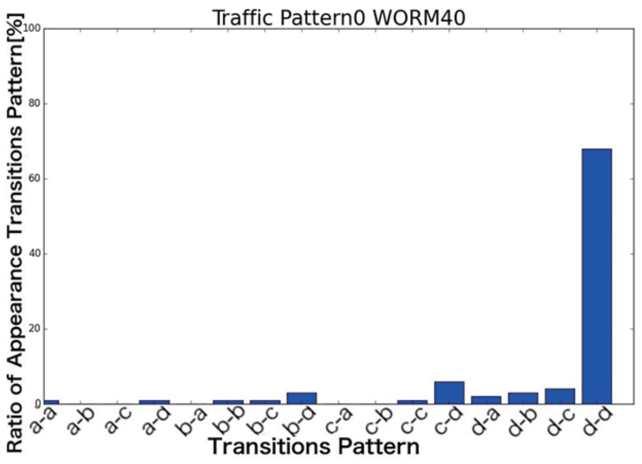


Figure 8: The histogram of worm of traffic pattern 0

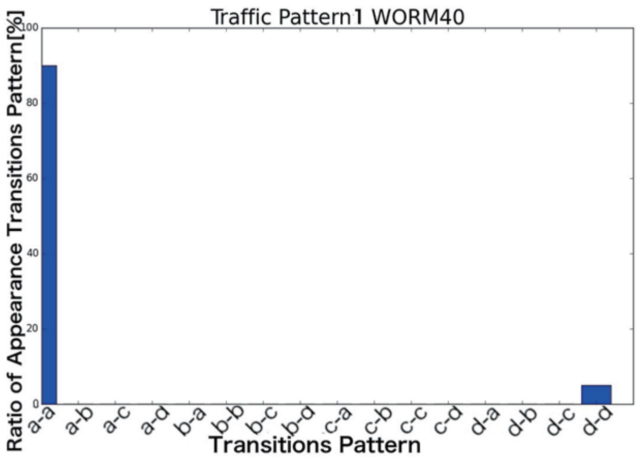


Figure 9: The histogram of worm of traffic pattern 1

6.3 Detection of Unknown Malware

Finally, we discuss the detection of unknown malware. In our experiments, we created models of normal and malware-infected traffic from only training data and used the models to identify malware traffic samples in the testing data. Six classes of ten malwares from the security vendor's definitions were included in the testing data and not in the training data. A malware class corresponds to malware with the same prefix_family name. Malware with the same prefix_family name is considered to be subspecific malware. The ten malware traffic samples represented six malware classes: two were PE_SALITY malware, one was TROJ_KRYPTK malware, one was TROJ_LSADCOM malware, two were TROJ_SPNR malware, three were TROJ_VILSEL malware, and one was TSPY_FAREIT malware. The remaining 190 malware traffic samples were subspecific malware found in the training data. When we focused on the hash value of the malware traffic samples, the training and testing data did not overlap, and the testing data was unknown malware.

As shown In Table 1, the proposed method had an identification rate of 100% for four of the five time intervals. That is, all malware traffic samples in the testing data were correctly identified, including the unknown malware traffic samples of the malware classes included in the training data and the unknown samples of the malware classes not included in the training data. The proposed method is thus able to identify unknown malware samples of a malware class not included in the training data.

7 CONCLUSION

Our method for detecting malware-infected traffic samples is based on the similarity between the pair of malware samples in this paper. Simulation evaluation demonstrated that the proposed method can effectively identify malware-infect traffic with high accuracy.

Future work includes conducting a large-scale experiment to better evaluate the effectiveness of the proposed method.Since normal traffic must be classified as normal when practical, a method for detecting infected traffic combined with a method for detecting normal traffic must be studied. In this paper, we focused on detecting malware infections (including unknown malware infections). Future work includes investigating how to create models of normal traffic for use in classifying unknown normal traffic.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 15H01684.

REFERENCES

[1] Y. Otsuki, M. Ichino, S. Kimura, M. Hatada, H. Yoshiura, “Evaluating payload features for malware infection detection,” Journal of Information Processing (JIP), Vol. 22, No. 2, pp. 376-387 (2014).

[2] K. Kuwabara, H. Kikuchi, M. Terada, M. Fujiwara, “Heuristics for Detecting Types of Infections from Captured Packets,” Computer Security Symposium, pp. 1-6 (2009) (in Japanese).

[3] A. Mohaisen, A. G. West, A. Mankin, O. Alrawi, “Chatter: Classifying Malware Families Using System Event Ordering,” IEEE Conference on Communications and Network Security, pp. 283-291 (2014).

[4] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, D. Garant, “Botnet detection based on traffic behavior analysis and flow intervals,” Computers & Security, Vol. 39, Part A, pp. 2-16 (2013).

[5] M. Piskozub, R. Spolaor, I. Martinovic, “MalAlert: Detecting Malware in Large-Scale Network Traffic Using Statistical Features,” ACM SIGMETRICS Performance Evaluation Review, Volume 46, Issue 3, pp. 151-154, (2018).

[6] D. Chiba, T. Yagi, M. Akiyama, K. Aoki, T. Hariu, “Design and Evaluation of a Profiling Method to Detect Post-infection Communications,” Computer Security Symposium, pp. 960-967 (2014) (in Japanese).

[7] G. Thatte, U. Mitra, J. Heidemann, “Parametric Methods for Anomaly Detection in Aggregate Traffic,”

IEEE/ACM Transactions on Networking, vol. 19, issue 2, pp. 512-525 (2011).

[8] K. Li, R. Chen, L. Gu, C. Liu, J. Yin, "A Method Based on Statistical Characteristics for Detection Malware Requests in Network Traffic," International Conference on Data Science in Cyberspace (DSC), pp. 527-532 (2018).

[9] M. Ichino, K. Kawamoto, T. Iwano, M. Hatada, H. Yoshiura, "Evaluating header information features for malware infection detection," Journal of Information Processing, Vol. 23, No. 5, pp. 603-612 (2015).

[10] T.-F. Yen, M. K. Reiter, "Traffic Aggregation for Malware Detection," 5th International Conference, Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 207-227 (2008).

[11] M. Z. Rafique, J. Caballero, "FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors," Lecture Notes in Computer Science, Volume 8145 (2013).

[12] T. Ichida, M. Ichino, M. Hatada, N. Komatsu, H. Yoshiura, "A study on malware detection method using feature's state transition," Symposium on Cryptography and Information Security, 1E1-2 (2012) (in Japanese).

[13] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, H. Zhang, "An Empirical Evaluation of Entropy-based Traffic Anomaly Detection," IMC '08 Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, pp. 151-156 (2008).

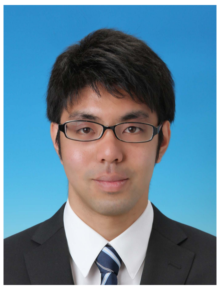
[14] Y. Linde, A. Buzo, R. Gray, "An Algorithm for Vector Quantizer Design, IEEE Trans. on Commun," Vol. 28, No. 1, pp. 84-95 (1980).

[15] M. Hatada, M. Akiyama, T. Matsuki, T. Kasama, "Empowering Anti-malware Research in Japan by Sharing the MWS Datasets," Journal of Information Processing, pp. 579-588 (2015).

[16] T. Akizuki, M. Ichino, N. Komatsu, K. Takeshita, H. Hasegawa, "A study on External Factors to Estimate Network Traffic," Technical Report of IEICE, Vol. 110, No. 455, CQ2010-70, pp. 19-24 (2011) (in Japanese).

(Received October 19, 2019)
(Revised April 20, 2019)

Masatsugu Ichino received his B.E. degree in electronics, information and communication engineering from Waseda University, Tokyo, Japan in 2003, and M.E. and Ph.D. degrees in computer science and engineering from Waseda University, Tokyo, Japan, in 2005 and 2008, respectively. He is currently an associate professor at the Graduate School of Informatics and Engineering, University of Electro-Communications, Tokyo, Japan. His research interests include biometrics, network security and pattern recognition. He is a member of IEICE and IPSJ.



Yuuki Mori received his M.E. degree in engineering from University of Electro-Communications,Tokyo,Japan in 2019.

Mitsuhiro Hatada is currently a manager at NTT Corporation, and an adjunct researcher at Waseda University. He received his B.E. and M.E. degrees in computer science and engineering, and Ph.D in engineering from the Waseda University in 2001, 2003 and 2018, respectively. He joined NTT Communications Corporation in 2003 and has been engaged in the R&D of applied security on anti-malware and threat intelligence. He is a member of IPSJ and IEICE.

Hiroshi Yoshiura received his B.S. and D.Sc. from the University of Tokyo, Japan in 1981 and 1997. He is currently a Professor in the Graduate School of Informatics, the University of Electro-Communications. Before joining UEC, he had been at Systems Development Laboratory, Hitachi, Ltd. He has been engaged in research on information security and privacy where he has published more than 150 refereed papers and has more than 120 registered patents.

Regular Paper

Lightweight Secure Communication Considering Network Path Reliability in IPv6 Wireless Sensor Network

Shunya Koyama[*], Yoshitaka Nakamura[**], and Hiroshi Inamura[**]

[*]Graduate School of Systems Information Science, Future University Hakodate, Japan
[**] School of Systems Information Science, Future University Hakodate, Japan
{g2117020, y-nakamr, inamura}@fun.ac.jp

Abstract - In recent years, IPv6 wireless sensor networks have been widely spread in various fields including IoT environments, because of the development of low-power sensor devices and wireless communication technologies. However, on these sensor networks, it is difficult to use secure communication technologies that can become large overhead, due to power saving of the wireless nodes is important. As one approach to deal with this problem, a method of focusing on Nonce which is one element of security, and separating it from secure communication is proposed, though this method can be used only when the reliability of communication ensured. Therefore, it remains a problem that not suit in environments such as wireless sensor networks where the reliability of communication is not ensured, since multi-hop networks and the like is used. In this paper, we propose a Nonce truncation method that can deal with such environments. Our method is implemented on the nodes that establish secure communication, and transfer information of about several bits that can estimate the Nonce associated the ciphertext as the truncated Nonce value. We also evaluated the effectiveness of our method by comparing the lifetime of the devices between our method and the previous method, and could confirm the effectiveness in a simple secure communication model.

Keywords: IoT, Reliability of Communication, Secure Communication, Nonce

1 INTRODUCTION

Recently, IPv6 (Internet Protocol version 6) wireless sensor networks have been widely spread in various fields including IoT environments, because of the development of low-power sensor devices and wireless communication technologies. The penetration rate of these devices has been increased, and as can be seen from Fig.1, about 50 billion devices will be interconnected in 2020 [1]. It is also expected to be utilized in various fields.

On the other hand, these sensor networks are typically composed of communication devices with limited computing resources such as battery capacity, CPU performance, and little memory. These characteristics are often due to cost constraint and physical constraints on such as size and available energy. Also, these tight limits make it difficult to attain some high load functions like secure communication

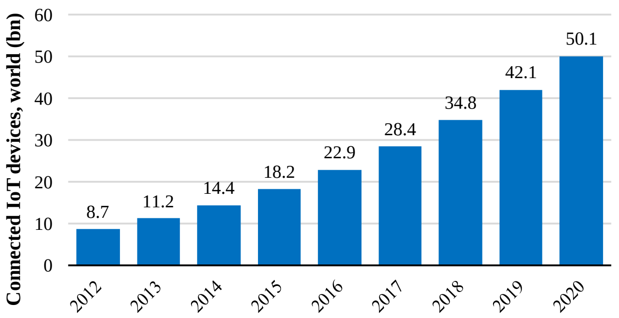


Figure 1: The number of connected IoT devices in the world

that are pretty much taken for granted for conventional networks. So, mechanisms considering these resource constraints are required in the sensor networks.

In addition, wireless sensor networks that called LLNs (Low power and Lossy Networks) are about to become widely spread. LLNs have restrictions not only on the resource constraint of the above-mentioned but also on the networks. The network constraint involves instabilities such as low data rate and high packet loss rate are accompanied in the communication environment, its reliability is not guaranteed. These tight constrained networks are needed to meet the demand for IoT services in various fields. Therefore, there are various factors that impose these restrictions, such as an introduction of simple and highly scalable UDP, and use of a multi-hop network to deal with a wide range of sensing.

In general, in these wireless sensor networks including LLNs, low power consumption wireless communication standard represented by Zigbee [3] is introduced. However, these standards are based on IEEE 802.15.4 [4] as a data link layer technology, and its frame size is small. Therefore, considering resource constraints, it is required some data size reduction scheme for side information like protocol control information. As the information to be reduced, the IPv6 header that supports the IoT service is no exception.

As one of the proposals for introducing IPv6 into such constrained networks, IETF (Internet Engineering Task Force) has established a policy to expand part of these low power consumption wireless communication standards. As a typical example of this, there is a method of providing an adaptation layer for using IPv6 technology on IEEE 802.15.4. Specifically, there are 6LoWPAN (IPv6 over Low-Power

Wireless Personal Area Networks) [5] which compresses IPv6 header or UDP header for alleviating a problem that the frame size of IEEE802.15.4 is too small in the introduction of the IPv6 technology, RPL (IPv6 Routing Protocol for Low Power and Lossy Networks) [6] which is a routing protocol to support the above-mentioned unstable communication environment, and so on. Here, there are many techniques related to the introduction of IPv6, but there are many reasons why this approach is mainly performed. First, IPv6 allows for a huge amount of addresses and provides easy participation in the network by such as SLAAC (State-Less Address Auto Configuration). Thus, it is possible to deal with the interconnection of the aforementioned enormous number of IoT devices, which is difficult in IPv4. In addition, IPv6 based networks can be interconnected readily between the devices including other IPv6 networks because the networks don't need intermediate entities like protocol translation. In consequence, the network scalability is high, and it can deal with various scenarios requested by IoT services. There are many other advantages, but it is said that IPv6 is more appropriate than IPv4 for the reasons mentioned above. Therefore, IPv6 has a high affinity wireless sensor networks including LLNs, and these technologies are expected to be used in various environments including smart grid, smart factory, and others as the core technology.

While it is expected that such communication scheme targeting LLNs based on IEEE 802.15.4 with IPv6 will become widespread, security problems such as unauthorized access aimed at valuable information assets exchanged over the wireless sensor network are also becoming apparent [7]. However, most of the research on this communication scheme is concerned with the network construction, and discussion on security has not been sufficiently done. For example, 6LoWPAN technology described above compresses only the header information, so does not support large size security elements for secure communication. Moreover, although the constraint on the small frame size is relaxed by the compression scheme, the header information occupies much of the frame size remains. Therefore, the importance of considering reduction of side information like secure elements is higher than sensor networks without IP. Therefore, the importance of considering reduction of side information like secure elements is higher than sensor networks without IP. In addition, despite there are proposals for the lightweight secure communication methods for wireless sensor networks without IP which was a major before the spread of IoT service, it has a big different background from recent sensor networks with network constraints like LLNs. For example, a method of separating Nonce which is one security element from communication and reducing its size to zero is proposed. However, in the lossy network, it is difficult to operate the Nonce correctly in this method, so it can become a heavy process. Hence, in an actual scenario, it is necessary to consider a lightweight truncation of Nonce method that can properly operate according to the frame loss rate. For this reason, it is difficult to apply conventional security technology for the LLNs environment. Especially, the problems that cannot support IEEE802.15.4 small frame size, and unstable communication quality are left.

In this paper, we discuss the unstable communication quality and the resource constraints of sensor devices which are the features of LLNs. Then, we design a secure communication method that can deal with these features. To this end, we focus on Nonce (Number used once) which is one of the security elements and address a method to truncate this. Also, in this method, we design a lightweight secure communication scheme that can operate without applying excessive overhead to sensor devices at any frame loss rate.

2 RELATED WORK

As the previous method, a lightweight secure communication method has been proposed, which is focusing on Nonce that is a part of security elements, and completely separating this from the communication. In the following, we describe the mechanism of the previous method and its applicability to LLNs, based on the basic secure communication technology.

2.1 Overview of Basic Secure Communication

In this section, we describe the general secure communication establishment method, and the secure design when applying it to the LLNs based on IEEE 802.15.4, in consideration of the data frame structure.

2.1.1 Establishing General Secure Communication

Strictly speaking, the establishment method of secure communication differs depending on the required security requirements and the Block Cipher Modes of Operation selected according to the requirements. As famous examples of the Modes of Operation, there are CBC (Cipher Block Chaining) mode and CTR (CounTeR) mode that provide confidentiality of communication data, and CCM (Counter with CBC-MAC) mode combines confidentiality and authenticity in an efficient way as authenticated encryption mode [8]. Among them, CCM mode can deal with processing resource constraints and frame size restriction of sensor devices. That's because this mode can process of encryption and decryption in parallel by the same algorithm, and does not expand data size of ciphertext. Moreover, this mode is known as high versatility because has many security requirements that can be provided, can apply various fields. Hence, it also coincides exactly with the design concept of the communication scheme for LLNs. Therefore, we describe how to establish secure communication in CCM mode on the premise of introduction to LLNs. The overview of the operation is depicted in Fig.2.

Figure 2 shows the flow from an establishment of secure communication between sensor devices until the Sender generates an encrypted frame from the plaintext, and then from this frame to the plaintext by the Receiver. In this figure, Key is a secret key, Nonce is a security element to make it possible to use the same Key multiple times without security risk, and MAC (Message Authentication Code) is a security element to provide integrity or authenticity, added

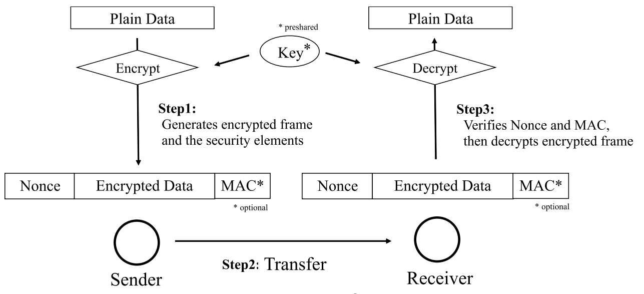


Figure 2: Basic operation of secure communication

only when using CCM mode. As the initial operation of secure communication establishment, sensor devices share the key being secret information, then communicate Nonce, MAC, and encrypted frame as public information. Thereafter, Nonce and MAC that change according to the corresponding encrypted frame are continuously communicated, and these are verified whether each value is correct when the Receiver decrypts the frame.At this time, in particular with respect to the calculation method of Nonce, the value corresponding to each encrypted frame must be unique from the viewpoint of security risk. In the NIST (National Institute of Standards and Technology), they have listed several recommended specifications and calculation methods of Nonce, and the size should be 8 bytes or more [9]-[10]. Further, as one of the calculation methods, a method using a counter value starting from an arbitrary value (for example, zero) is recommended. The value is incremented and shared every time different ciphertexts are generated. SNEP described later in section 2.2 and our proposed method described later in chapter 3 are based on this calculation method.

2.1.2 Secure Communication Design in LLNs (Low power and Lossy Networks)

Figure 3 shows an example of a simple data frame structures when the above described secure communication is applied to LLNs.

In Fig.3, (a)(b)(c) commonly indicate the frame structure when IP technology is introduced on IEEE 802.15.4 and encrypted using CCM mode. In addition, for each frame structure, (a) is introduced UDP into IPv6, (b) is introduced 6LoWPAN and RPL over (a), and(c) is introduced SNEP described later in section 2.2 of the previous method over (b). As can be seen from the figure, the security elements communicated can be large overhead and suppress MAC payload in the environment with limited frame size as LLNs. Therefore, there is a possibility of increasing the processing load of sensor devices through fragment processing, it is desirable to make the size as small as possible. In particular, in each frame structure excluding (c), the ratio of Nonce to MAC payload occupies so large that if Nonce can be completely eliminated, on average about 16% and about 12% of the payload can be expanded.

In order to properly operate the Block Cipher Modes of Operation, there is no strict restriction that each security element must secure a certain size or more. However, if you select the smallest value among the simply selectable sizes, there is also the possibility of impairing the safety of secure

802.15.4 Frame Structure (127 [bytes])

(a) 802.15.4 + IPv6 + UDP

MAC Header	Nonce	IPv6 Header	UDP Header	Data Payload	MAC	FCS
3 ~ 23 [bytes]	*8 [bytes]*	*40 [bytes]*	*8 [bytes]*	*30 ~ 66 [bytes]*	*0 ~ 16 [bytes]*	*2 [bytes]*

(b) 802.15.4 + 6LoWPAN + Compressed UDP + RPL

MAC Header	Nonce	6LoWPAN Header	UDP Header*	Data Payload	MAC	FCS
3 ~ 23 [bytes]	*8 [bytes]*	*3 ~ 36 [bytes]*	*1 ~ 6 [bytes]*	*36 ~ 110 [bytes]*	*0 ~ 16 [bytes]*	*2 [bytes]*

(c) 802.15.4 + 6LoWPAN + Compressed UDP + RPL + SNEP

MAC Header	6LoWPAN Header	UDP Header*	Data Payload	MAC	FCS
3 ~ 23 [bytes]	*3 ~ 36 [bytes]*	*1 ~ 6 [bytes]*	*44 ~ 118 [bytes]*	*0 ~ 16 [bytes]*	*2 [bytes]*

*** Compressed**

Figure 3: Structure pattern of encrypted frames in LLNs (Low power and Lossy Networks)
(a): 802.15.4 + IPv6 + UDP
(b): 802.15.4 + 6LoWPAN + Compressed UDP + RPL
(c): 802.15.4 + 6LoWPAN + Compressed UDP + RPL + SNEP

communication. From that point of view, NIST recommends the size of Nonce is 8 bytes or more. Thus, a method of reducing the size without losing the safety of secure communication is ideal.

2.2 SNEP (Secure Encryption Network Protocol)

Following the previous section, a method of separating Nonce from communication and reducing its size to zero without reducing the safety of secure communication called SNEP (Secure Network Encryption Protocol) has been proposed as a part of a large security schema named SPINS (Secure Protocols for Sensor Networks) [11]. Figure 4 shows the simple operation flow. Specifically, SNEP is the method of sharing only the initial value of Nonce, and thereafter incrementing the Nonce value stored in the sensor devices according to the number of received encrypted frames. If an encrypted frame is lost in the middle due to interruption of communication, decryption fails because the Nonce corresponding to the subsequent encrypted frame does not mesh. For this reason, the resynchronization process is performed to transmit the entire value of Nonce every time the encrypted frames are lost. By taking such a series of procedures, it is shown that in an environment with the stable communication quality, the communication overhead on the sensor devices by secure communication is reduced. On the other hand, in the environment such as LLNs which the communication quality is unstable and the frame loss rate can be high, the resynchronization process frequently occurs. Therefore, this means secure communication overhead of sensor devices is actually increasing, and network congestion problem may occur.

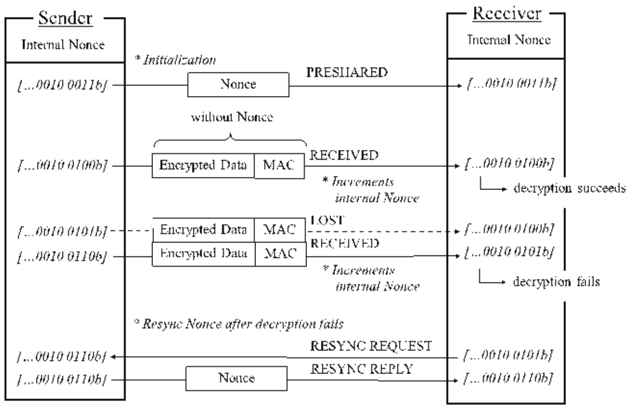


Figure 4: Simple operation flow of SNEP (Secure Network Encryption Protocol)

3 PROPOSAL METHOD

3.1 Research Tasks

In the previous method, if an encrypted frame is lost in the middle due to interruption of communication or the like, it is necessary to repeat the resynchronization process of Nonce for recovery secure communication. Therefore, it is not supported in the environment where the frame loss rate can be high. Also, according to a general secure communication method, the ratio of encrypted frames occupied by Nonce is large, and there is a possibility that a heavy load is applied to the sensor devices and the network itself due to inefficient fragment processing. For this reason, any method is difficult to adapt to LLNs where communication quality is unstable and frame size is limited, and a method capable of dealing with these problems is required.

In this paper, we propose a method to deal with the above problem by estimating Nonce only by sensor devices itself from the truncated value that the size changes according to the frame loss rate.

3.2 Basic Operation

In this section, we describe the basic mechanism for truncating a Nonce. As the block cipher mode of operation for establishing secure communication, CCM mode is used. Also, as a calculation method of Nonce corresponding to each encrypted frame, a counter value that increments the value according to the frame is used.

As a basic idea, we propose the method to minimize Nonce resynchronization processing for frame loss which the problem in the related research. The overview of the proposed method, the device sends a small amount of information that can estimate Nonce as a hint instead of transmitting the entire Nonce value. Then, the receiver estimates the entire Nonce value to be synchronized from this hint and Nonce stored on the device. Hereafter, we describe the operation flow according to Fig.5.

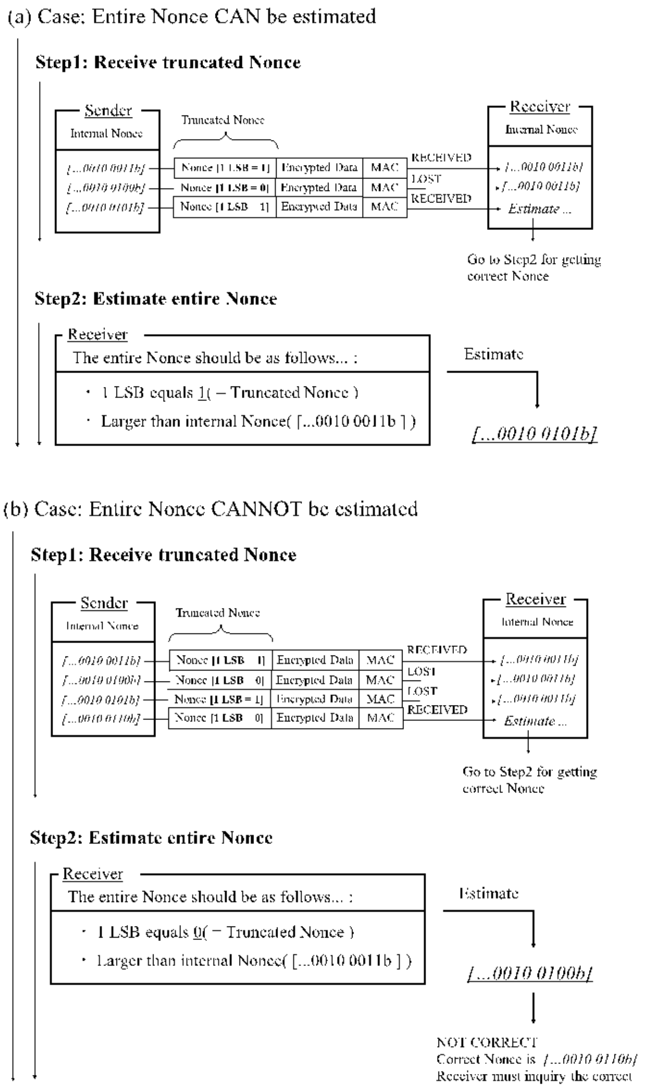

(a) Case: Entire Nonce CAN be estimated
Step1: Receive truncated Nonce
Truncated Nonce
Nonce [1 LSB = 1]
Nonce [1 LSB = 0]
RECEIVED
LOST
Estimate ...
Go to Step2 for getting correct Nonce
Step2: Estimate entire Nonce
The entire Nonce should be as follows... :
1 LSB equals 1(= Truncated Nonce)
Larger than internal Nonce([...0010 0011b])
Estimate
[...0010 0101b]
(b) Case: Entire Nonce CANNOT be estimated
1 LSB equals 0(= Truncated Nonce)
[...0010 0100b]
NOT CORRECT
Correct Nonce is [...0010 0110b]
Receiver must inquiry the correct

Figure 5: Operation flow in the case where the truncated Nonce length is 1
(a): entire Nonce value can be estimated
(b): entire Nonce value cannot be estimated

The first step, the initial value of Nonce is shared between Sender and Receiver, and the whole value is stored in the sensor device as in the previous method. Thereafter, in the sharing of Nonce, only the N of least significant bits (N LSBs) are assigned on communication. Hereinafter, this N bit is called a truncated Nonce length. Figure 5 shows the operation flow in the case where the truncated Nonce length is 1 as the specific example.

In Fig.5, (a)(b) commonly begin already synchronized entire Nonce between Sender and Receiver devices and estimate the entire Nonce value from the truncated value while the devices communicate several encrypted frames. First, (a) shows that the receiver succeeds in receiving the third encrypted frame after losing only the second encrypted frame. At this time, since there is a difference in the entire value of Nonce internally stored between the two sensor devices, receiver fails in decryption the third encrypted frame. At this stage, move on to step 2 of (a). Since the value of the received truncated Nonce is 1, the receiver can decrypt the third encrypted frame by estimating the entire value of

Nonce that is greater than internal Nonce value and 1 LSB equals 1. On the other hand, in the case of (b), the receiver receives the fourth encrypted frame after losing the second and third encrypted frames, hence fails in decryption even if estimates the entire value of Nonce like (a). This is because there was a big difference in the entire values of Nonce internally stored between both sensor devices. In the case (b) shown in this figure, although correct Nonce is "*...0010 0110b*", in fact, it is estimated "*...0010 0100b*" by mistake. In such a case, recovery secure communication by performing resynchronization process sharing the entire value of Nonce. Generally, such resynchronization process occurs only when the truncated Nonce length is *x* bits and the frame is lost consecutively for 2^x times or more. For example, in the case of (b), this process occurs because the frame has been lost 2^1 times that is twice consecutively.

Therefore, depending on the selection of the truncated Nonce length, the same problem as SNEP may still occur. For this reason, it is necessary to select the truncated Nonce length flexibly so as to minimize the number of the resynchronization process according to the frame loss rate of the communication environment. Table 1 shows the occurrence probability of the resynchronization process according to *x* bits of truncated Nonce length and frame loss rate.

3.3 Optimization of the Resynchronization Process Occurrence Count

Considering the characteristics of LLNs, it is necessary to minimize the occurrence probability of the resynchronization process as much as possible so that the same problem as SNEP does not occur. For that purpose, it is ideal to flexibly select the truncated Nonce length as short as possible according to the frame loss rate. For example, referring to Table 1, if we fix the truncated Nonce length to 4 bits, it seems that can support any frame loss rate. However, in actual fact, there is a possibility that the frame loss rate suddenly changes due to temporary noise or the like, so it is required to deal with dynamic link quality.

In order to deal with this problem, we use ETX (Expected Transmission Count) [12] adopted in routing protocols used in many wireless sensor networks including RPL.

ETX is a metric index using link quality, and its value is defined as the reciprocal of the frame arrival rate. Specific ally, it can be found using the following equation (1) where E_{pt} is the frame loss rate.

Table 1: Probability of the resynchronization process occurrence according to the truncated Nonce length and frame loss rate

Frame Loss Rate / Truncated Nonce Length	80%	60%	40%	20%	E_{pt}
1	64%	36%	16%	4%	$E_{pt}^{2^1}$
2	40%	13%	2.5%	0%	$E_{pt}^{2^2}$
4	2.8%	0%	0%	0%	$E_{pt}^{2^4}$
x	$80\%^{2^x}$	$60\%^{2^x}$	$40\%^{2^x}$	$20\%^{2^x}$	$E_{pt}^{2^x}$

$$ETX = \frac{1}{1 - E_{pt}} \qquad (1)$$

By solving this equation (1) for E_{pt}, the packet loss rate can be obtained. Therefore, in the sensor devices having information corresponding to Table 1, it is possible to select the truncated Nonce length dynamically to minimize the occurrence probability of the resynchronization process to any value or less.

4 EVALUATIONS

About the proposed method and previous methods in this research, we performed evaluation experiments after implementing these on the network simulator. Hereinafter, we describe the experimental environment, evaluation method, experiment method and the detail of these.

4.1 Experiment Environment

We implemented the proposed method and (b)(c) in Fig.3 as the previous methods on ContikiOS, which is a built-in OS for sensor networks, and operated on the network simulator Cooja [13] attached to ContikiOS.

Specifically, as shown in Fig.6, we created a simple small-scale model that established secure communication between two sensor devices such as Sender and Receiver, operated each method in this model. At this time, we emulated all sensor devices as Zolertia Z1 hardware [14].

For simplicity, unidirectional communication is performed from the Sender to the Receiver, and encrypted frames are transmitted and received in this scenario.

Detailed simulation parameters in the experimental environment are shown in Table 2. The communication standard conforms to (b) in Fig.3 as the general standard. In addition, only the length of Nonce is selected from among 0 to 8 bits or 8 bytes different according to the frame loss rate. Furthermore, the 0 bit corresponds to SNEP as the previous method and the 8 bytes without special handling to Nonce corresponds to the general method. In the block cipher mode of operation, we used AES-CCM* mode standardized on Zigbee which extended the CCM mode. Also, a frame loss rate is used as an index representing communication quality in LLNs. Moreover, considering the resynchronization process due to frame loss, experiments were performed until all data arrives at Receiver and completely decrypted after establishing secure communication.

4.2 Evaluation Method

In each experimental method, we measured the lifetime of the sensor device from the power consumption of the Sender emulated as Zolertia Z1 hardware on Cooja. We evaluate the effectiveness by calculating and comparing the lifetime ratio of each method where general method (b) in Fig.3 as 1 value.

4.3 Experimental Method

One experiment for each combination of frame loss rate and truncated Nonce length and the other experiment in the case of continuing to select the optimal truncated Nonce length to minimize the number of the resynchronization process. We describe the details of each experiment method below.

4.3.1 Experiment for Each Combination of Frame Loss Rate and Truncated Nonce Length

In this experiment, we evaluate whether the length of each Nonce can correspond to any communication quality assuming LLNs environment as the proposed method and the previous method. First, about the lifetime ratio of each sensor devices, we calculated from the power consumption until the Receiver took 1,000 KB of data from the Sender 10 times and decrypted all the data. At this time, to evaluate the performance for each length of Nonce in accordance with the frame loss rate, the experiment was proposed at intervals of 10% to 20% in the frame loss rate in Table 2.

4.3.2 Experiment for Continuing to Select the Optimal Truncated Nonce Length

In this experiment, we evaluate whether the effectiveness can be shown compared with the previous method when dynamically selecting optimum Nonce length using the proposed method. The basic simulation parameters were as shown in Table 2, but the frame loss rate was changed randomly between 20% and 80%, and the occurrence probability of resynchronization process was always 5% or less using ETX. Also, it was assumed that 1000 KB of data was transmitted ten times a day. In such an environment, we calculated the average lifetime ratio of sensor devices in each method.

5 RESULTS AND DISCUSSION

5.1 Results

The results obtained in each experimental method are shown in the following section.

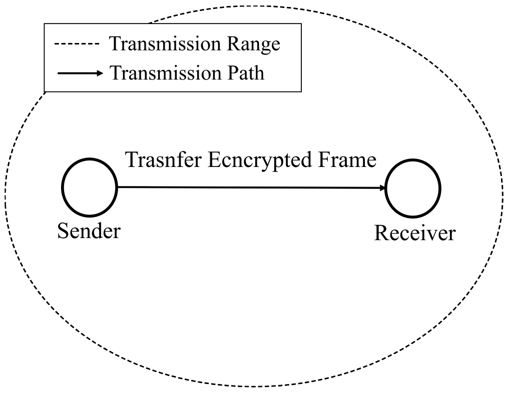


Figure 6: Experiment environment

Table 2: Simulation parameters

Parameter	Value
Data Link Protocol	IEEE802.15.4
Network Protocol	6LoWPAN + RPL
Transport Protocol	Compressed UDP
Frame Size	127[bytes]
Transfer Data	1000[Kbytes] * 10
Cipher Mode	AES-CCM*
Key Length	128[bits]
Block Length	128[bits]
MAC Length	8[bytes]
Frame Loss Rate	0%~90%
Nonce Length	0, 1, 2, 4, 8[bits], 8[bytes]

5.1.1 Experimental Results for Each Combination of Frame Loss Rate and Truncated Nonce Length

The result obtained by the experiment according to the combination of frame loss rate and the truncated Nonce length is shown below.

Figure 7 shows the Sender's lifetime ratio measured for each frame loss rate and truncated Nonce length (hereinafter referred to as *x*) in the simulation parameters shown in Table 2. In the case where the frame loss rate was 20% or less, all the proposed method and the previous method had improved the lifetime compared with the general method of transmitting 8 bytes of Nonce. On the other hand, when the frame loss rate exceeded 20%, the lifetime sharply decreased according to the length of Nonce. In particular, the rate of decrease was remarkable when the truncated Nonce length was 4 bits, but in the case of 8 bits, any frame loss rate was improved. Also, it could be seen that the truncated Nonce length at which the lifetime improves most was different depending on the frame loss rate except 0%.

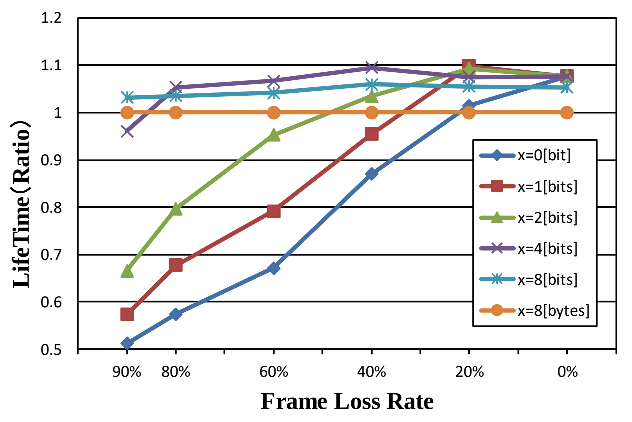


Figure 7: Lifetime ratio according to truncated Nonce length and frame loss rate by simulation

Table 3: Lifetime ratio obtained for each method by simulation

Method	Lifetime Ratio
General(Nonce Length: 8[bytes])	1
Previous(Nonce Length: 0[byte])	0.625
Proposal	1.058

5.1.2 Experimental Results for Continuing to Select the Optimal Truncated Nonce Length

The result obtained by the experiment for continuing to select the optimal truncated Nonce length so that the occurrence probability of the resynchronization process within 5% is shown in following Table 3. The effectiveness of the proposed method is clear because the proposed method was improved the lifetime by about 6%, while SNEP that previous method dropped the lifetime about 37% when compared with the lifetime of the general method that transmitted 8 bytes of Nonce.

5.2 Discussion

From the results shown in Fig.7 and Table 3, the effectiveness of the proposed method is clarified because previous methods cannot deal with unstable communication quality such as LLNs, whereas the proposed method improves the lifetime. This is considered because the number of resynchronization process has decreased, and the number of fragmentation data has also decreased because the ratio in the encrypted frame occupied by Nonce is reduced. Also, if the truncated Nonce length is about 4 to 8 bits, the lifetime is roughly improved in any frame loss rate, except when the loss rate is extremely high. This means that truncated Nonce length is enough size to operate in the LLNs environment. On the other hand, depending on the select of the truncated Nonce length, it is also clear that the possibility of greatly decreasing the lifetime also remains, and as also shown from the results in Table 3. So, it is effective to select continually the optimum truncated Nonce length.

However, in the experimental environment, since evaluation is limited to a simple secure communication model between two devices, in the future it is necessary to verify the effectiveness from many aspects according to the real environment. Particularly, there are many problems such as dealing with frame delay, handling burst loss caused by network congestion problem. It is also necessary to consider approaches to deal with these problems. Moreover, in order to further improve the proposed method, we will adjust the number of times to estimate the entire Nonce value according to the truncated Nonce length. Therefore, it is necessary to measure the processing load in the decryption process and the resynchronization process, and to measure how many times the decryption process can be increased.

6 CONCLUSION

In this paper, we discussed the unstable communication quality and the resource constraints of sensor devices which are the features of LLNs. Then, we designed the secure communication method that could deal with these features. To this purpose, we focused on Nonce which is one of the security elements and proposed the method to truncate this. As a result, we prevent the frequent occurrence of Nonce resynchronization processing at the time of frame loss, which was a problem of the conventional method, and minimize the number of times of processing. In consequence, we showed the effectiveness of the proposed method as a lightweight secure communication method that deals with unstable communication quality, and without excessive secure communication overhead to sensor devices by reducing encrypted frame size. As the evaluation method, we implemented the proposed method and the conventional method on sensor terminal which emulated, measured its lifetime ratio and compared it. Specifically, we first measured the effect of each method on the lifetime for each frame loss rate. After that, we assumed an environment in which the frame loss rate varies randomly, and compared the influence of each method on the lifetime.

As future prospects, there are we should address examine experiments and evaluation methods considering various more real environments. In particular, we consider that the high frame loss rate in the assumed environment is not practical in the real environment, which is limited to the simple performance evaluation of the methods. Therefore, first of all, it is important to focus on this situation and strictly evaluate the usefulness of the proposed method. And, it is also necessary to deal with the response to burst loss of encrypted frames and the delay problem in real connectionless network. Furthermore, in order to improve the performance of the proposed method, we will adjust the number of times to estimate the entire Nonce value according to the truncated Nonce length by measuring and comparing the processing load in the decryption process and resynchronization process.

REFERENCES

[1] D. Evans (Cisco Internet Business Solutions Group), "The Internet of Things - How the Next Evolution of the Internet Is Changing Everything," <https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf > [Accessed May 20, 2018].

[2] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," Internet Engineering Task Force RFC7228, (2014).

[3] ZigBee Alliance., "Zigbee specification. Technical Report Document 053474r20," Zigbee Alliance, (2014).

[4] IEEE Std 802.15.4-2017, "IEEE Standard for Low-Rate Wireless Networks," IEEE Standard, (2017).

[5] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Internet Engineering Task Force RFC6550, (2012).

[6] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," Internet Engineering Task Force RFC4919, (2007).

[7] D. Airehrour, J. Gutierrez, and S. K. Ray, "Secure Routing for Internet of Things: A survey," Journal of Network and Computer Applications, Vol.66, pp.404-412, (2016).

[8] Information technology Promotion Agency, "The Survey Regarding Block-cipher Modes of Operation Usable for Confidentiality, Message Authenticity, and Authenticated Encryption," <https://www.ipa.go.jp/security/enc/CRYPTREC/fy15/documents/mode_wg040607.pdf > [Accessed May 20, 2018].

[9] National Institute of Standards and Technology, "FIPS PUB 140-2 Security Requirements for Cryptographic Modules," (2002).

[10] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation," <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38a-add.pdf > [Accessed May 20, 2018].

[11] A. Perrig, R. Szewczyk, J. D. Tygar, V. Web, and D. E. Culler, "SPINS: security protocols for sensor networks," Wireless Networks Journal, Vol.8, pp.521-534, (2002).

[12] D.S.J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," Proceedings of ACM MobiCom, pp. 134-146, (2003).

[13] "Cooja Simulator," <http://anrg.usc.edu/contiki/index.php/Cooja_Simulator> [Accessed May 20, 2018].

[14] Zolertia, "Zolertia Z1 Datasheet," <http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf> [Accessed May 20, 2018].

(Received October 21, 2018)
(Revised June 24, 2019)

Shunya Koyama received B.E. degree in systems information science from Future University Hakodate, Japan in 2017. He is a graduate student of Future University Hakodate. His research interests include lightweight security in IPv6 wireless sensor network. He is a student member of IPSJ.

Yoshitaka Nakamura received B.E., M.S., and Ph.D. degrees from Osaka University in 2002, 2004 and 2007, respectively. He is currently an associate professor at the School of Systems Information Science, Future University Hakodate. His research interest includes information security and ubiquitous computing. He is a member of IEEE, IEICE, and IPSJ.

Hiroshi Inamura He is a professor of School of Systems Information Science, Future University Hakodate, since 2016. His current research interests include mobile computing, system software for smart devices, mobile/sensor network and their security. He was an executive research engineer in NTT docomo, Inc. He received B.E., M.E. and D.E. degree in Keio University, Japan. He is a member of IPSJ, IEICE, ACM and IEEE.

Regular Paper

Compression Algorithm Contribution for Infected-host Detection

Yasushi Okano[†], Kazunori Kamiya[†], Atsutoshi Kumagai[†], Taishi Nishiyama[†], Bo Hu[†], Masaki Tanikawa[†], and Kazuhiko Ohkubo[‡]

[†]NTT Secure Platform Laboratories, Japan
[‡]Kyowa Exeo Corporation, Japan
[†]{yasushi.okano.ye, kazunori.kamiya.ew, atsutoshi.kumagai.ht, taishi.nishiyama.pt, bo.hu.cf, masaki.tanikawa.ym}@hco.ntt.co.jp
[‡]k.ookubo@en2.exeo.co.jp

Abstract - Machine learning is being actively used to detect malware-infested hosts and their malicious communications. When applying machine learning, designing the right feature is the key for accurate detection. BoW (Bag of Words)-based feature extraction is widely applied in natural language processing and also utilized for malicious communication detection. However, BoW-based feature extraction does not always scale for handling network logs that often have new data sequences. By focusing on the fact that new data sequences in network logs are in many cases mostly similar but partly different, we propose a new detection method based on a data compression algorithm. Since the compression algorithm has a characteristic that data size after compressing is related to similarity of data, a compression algorithm based feature can be utilized for classification. According to our evaluation results with real-field proxy logs in an enterprise network, the proposed method has better at detection than a BoW-based detection method. In particular, its true positive rate (TPR) in a low false positive rate (FPR) area (0.5%) is over 30% higher than that for the BoW-based method. In addition, the results show that the proposed method effectively detects an infected host communicating with malicious URL that includes partially modified string from original malicious logs.

Keywords: malware, log analysis, data compression, machine learning

1 INTRODUCTION

Malware is becoming more sophisticated and has so many variants that anti-virus software does not detect all of them. In fact, it is reported that over 127.5 million pieces of malware were registered in 2016 [1]. To compliment the fact that detection at the endpoint is not always successful, network log analysis is one solution that monitors logs taken from network devices such as proxy and firewall and finds malicious communications derived from infected hosts.

In current log analysis, many monitoring rules have been deployed. Examples include network scans being detected once the number of different destination IP addresses from one source IP address exceeds a predefined threshold and a specific malware infection being determined if one host accesses a blacklisted URL. Many monitoring rules are based on operators' elaborations on creating rules, deciding thresholds, and maintaining blacklists. However, as malware has evolved to become able to change communication patterns easily, heuristic rule creation adds cost to operations and has difficulty catching up with malware modification. As a result, machine learning is gaining attention for automatically detecting evolving malware and for helping operations.

In applying machine learning for network log analysis, first, machine learning calculates feature values from network logs. For instance, feature values range from the length of a string, frequency of terms in device logs, and so on. Second, machine learning will classify the data into legitimate or malicious on the basis of feature values. In this process, infected hosts and malicious communications are detected.

There are many detection algorithms from LR (Logistic Regression), SVM (Support Vector Machine), RandomForest, and DNN (Deep Neural Network). However, the most critical factor for accurate detection is designing the right feature for the problem.

BoW (Bag of Words)-based features are widely applied in natural language processing and also utilized to detect malicious communications. However, BoW-based feature extraction does not always scale for handling network logs, which often have new data sequences.

By focusing on the fact that new data sequences in network logs are in many cases mostly similar but partly different, we propose a compression algorithm based feature and apply it to supervised learning for detecting malicious communications and infected hosts.

Simply put, a compression algorithm based feature is one form of the compression rate of data, which means how small the data becomes after being compressed. When the data sequence is similar to that in existing malicious data, the compression rate should be small because this data sequence is effectively compressed. On the other hand, if another data sequence is totally different from that in existing malicious data, the compression rate should be large. In this sense, the compression rate can be useful for finding if one data sequence is similar to that in malicious data. Consequently, the compression rate can contribute to detecting malicious communications.

We evaluated the proposed method with real proxy logs taken from an enterprise network. The results show that the proposed method is better at detection than the BoW-based method. In particular, the results show that the proposed method effectively detects an infected host communicating with a malicious URL that includes a partially modified string from original malicious logs.

Overall, our research makes three contributions.

1. We first apply the compression algorithm feature to supervised machine learning to detect malicious communications and infected hosts.

2. We evaluate the proposed method with real enterprise proxy logs and demonstrate that the proposed method performs better than a BoW-based classifier.

3. We analyze true positive and false negative use cases and clarify that the proposed method effectively detects partially modified strings from original malicious logs.

2 RELATED WORK

2.1 Classification Based on Compression Algorithm

Benedetto et al. [2] proposed relative entropy. Although patterns of the same consecutive code or similar repeated code are effectively compressed, patterns of different code are not. Relative information volume of data sequence x against data A is linked to how well data is compressed. Based on this observation, Benedetto et al. define relative entropy as how well new data x will be compressed with existing data A. Consequently, this is formulated as follows.

$$C_A(x) = Z(A\ cat\ x) - Z(A) \tag{1}$$

where Z is the function to output the data size after compression, and *cat* is the function to concatenate the first and second data sequences. Sometimes, normalized relative entropy is also used, which is defined to divide relative entropy by the size of data x.

Relative entropy has been applied to classification problems in several research areas [3] - [7]. To classify data x into group A and B, data x is normally classified into the more similar group. Relative entropy can be used as one index of expressing similarity; when relative entropy with group X is small, data x is similar to group X.

Bratko et al. [5] applied relative entropy to classify spam e-mails. They reported that it was more accurate than BoW based classification.

Nishida et al. [6] introduced a smoothing parameter and set the score in accordance with the following equation to classify malicious tweets from twitter logs.

$$Score = \frac{C_A(x) + \gamma}{C_B(x) + \gamma} \tag{2}$$

where γ is a smoothing parameter that should be set large to alleviate the impact of minor letters appearing a few times in a data string. Data x is classified as A if the score is small and B otherwise. This scoring technique enables us to apply a compression algorithm for a classification problem of comparably long data. Nishida et al. [6] also demonstrated that classification of twitter logs with this scoring mechanism has better accuracy than feature extraction with morphological analysis and classification with a CW(Confidence-Weighted linear classification) method [8].

Different compression algorithms are used depending on their purposes. It is reported that LZSS (gzip), LZW (compress), PMP (rar) are applicable for text data [3]-[6]. Adachi et al. [7] reported that bzip is applicable for music pieces.

2.2 Method of Extracting Feature from URL String

The BoW method is widely used to extract features from strings. BoW decomposes string text into words by separation of letters or morphological analysis and then generates each word as a one-dimensional feature. Since a URL is deemed as a one text string, BoW features can be extracted. Kumagai et al. [9] proposed BoW-based feature generation to apply LR supervised learning with L1 regularization and demonstrated that their method has better area under curve (AUC) than blacklist based detection.

Nelms et al. [10] proposed describing a URL attribute with a regular expression and applying unsupervised learning to generate a malware-specific URL access template. By comparing a target URL and the above template, the method successfully detects malware communication even when malware slightly modifies its access pattern.

In the security context, on the basis of knowledge on malware analysis, many kinds of statistical features have been proposed [11] such as the length of a URL and ratio of vowels in a URL.

3 PROPOSAL

We propose applying a compression algorithm based feature to apply supervised learning to detect malicious URLs and infected hosts. Since a large part of malware uses HTTP as a communication protocol with C2 servers, it can be mixed with normal Web access and is hard for operators to distinguish. Thus, in our research, we focus on analyzing HTTP proxy logs and detecting malicious URLs to find infected hosts.

An important observation on malware communication in HTTP is that they tend to access C2 servers with a slightly modified URL string in order to slip through blacklist-based detection with minimum engineering effort. In this case, simple blacklist matching does not

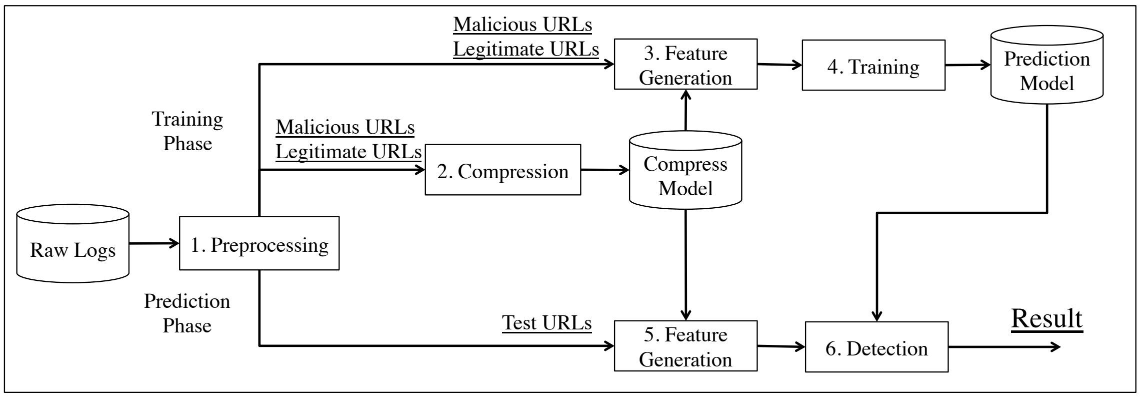


Figure 1: Overview of proposed method

Table 1: Dataset

	Number of hosts	Number of logs	Collection Period	Log Type
Malicious Logs	71,310	7,152,479	Feb. 2015 - Jul. 2015	Sandbox logs
Legitimate Logs	1,940	36,581,398	Feb. 2014 - Mar. 2014	Proxy logs in enterprise

catch up with malicious URLs since malware may have various URL access patterns even if its modifications are small. In this sense, we expect the compression algorithm based feature to correctly describe the similarity between a slightly modified malicious URL and a known malicious URL.

To the best of our knowledge, our proposal is the first to apply a compression algorithm to detect malicious communication URLs and infected hosts. In addition, our method is different from existing compression algorithm based methods in that we use a compression algorithm-based score as a feature in supervised learning and the feature can be combined with other features. Furthermore, our research considers a URL structure that has many kinds of attributes such as FQDN, PATH, and QueryString to generate a multi-vector compression algorithm feature for each attribute.

Figure 1 shows an overview of the proposed method. The flow of our proposal is as follows.

1. Input raw logs and execute preprocessing to obtain malicious URLs, legitimate URLs and test URLs

2. Compress malicious URLs and legitimate URLs to generate compress model

3. Input malicious URLs and legitimate URLs with application of compress model to generate compression algorithm features, namely Z_{pos}(Malicious Compression Rate) and Z_{neg}(Legitimate Compression Rate). Z_{pos} and Z_{neg} are defined in equation (3) and (4) respectively. Features are calculated for each attribute of a URL.

4. Train classifier with compression algorithm features and generate prediction model

5. Generate compression algorithm features from test URLs with application of compress model

6. Detect malicious URLs and infected hosts with application of prediction model

As with preprocessing, suitable data must be selected in machine learning for correctly estimating a classifier's performance. We execute two-phase cleansing in this process. First, we delete duplicate URLs in legitimate and malicious logs. This is because hosts may access the same URLs repeatedly. To correctly estimate a classifier, we leave first-to-appear logs in a dataset and eliminate duplicate logs.

Second, we eliminate URLs included in both malicious and legitimate logs, since having the same logs in both datasets may degrade the classifier's performance. In fact, there are many cases in which the same URLs are included in both logs. For instance, some service URLs are automatically accessed from specific applications installed in many environments. Search engine URLs are also often accessed from infected hosts for connectivity checks and included in malicious logs.

As for the compression algorithm features, we define Z_{pos} and Z_{neg} as follows.

$$Z_{pos}(x) = \frac{C_{pos}(x) + \gamma}{L(x) + \gamma} \quad (3)$$

$$Z_{neg}(x) = \frac{C_{neg}(x) + \gamma}{L(x) + \gamma} \quad (4)$$

where C_{pos} and C_{neg} are relative entropy between data x and malicious log(pos) or legitimate log(neg), L is data size of x, and γ is a smoothing parameter.

4 EVALUATION METHOD

4.1 Dataset

The dataset used for all evaluations is shown in table 1.

Malicious logs are taken from an in-house sandbox [12] where we run over 70K malware downloaded on a daily basis from a malware-sharing site and collect pcaps to extract URL information. Legitimate logs are taken from real-environment proxy in an enterprise network.

4.2 Evaluation Indices

Evaluations are executed on the basis of a holdout test that uses previous data in time series as the training dataset and evaluates with later data in time. Evaluation indices are AUC, partial AUC ($pAUC$) [13], and true positive rate $(TPR)_{0.5\%}$ [14].

AUC is the area under the curve drawn on a 2D surface of a false positive rate (FPR) and TPR by changing the score threshold. $pAUC$ is the area under the curve of a limited range of a FPR $[p1, p2]$. Considering the TPR as a function having a FPR as a variable, AUC and $pAUC$ are defined as follows.

$$AUC = \int_0^1 TPR\, dFPR \tag{5}$$

$$pAUC = \int_{p1}^{p2} TPR\, dFPR \tag{6}$$

Through our evaluation, we set $[p1, p2] = [0, 0.1]$.

$TPR_{0.5\%}$ is the TPR value for a low FPR, specifically $FPR = 0.5\%$. In security operations, a low FPR is crucial since the final judgment is done by operators. $pAUC$ and $TPR_{0.5\%}$ are important indices to estimate detection capability with a low FPR.

4.3 Selection of Compression Algorithm

The first evaluation is aimed at selecting a suitable compression algorithm. Several kinds of compression algorithms are used in existing research, so through our evaluation, we can select one algorithm that performs with both good accuracy and the least CPU (central processing unit) time.

We tested major compression algorithms LZSS (zip, LZ77), LZT [15] (a variant of compression algorithms LZW and LZ78), bzip2, and LZMA. To limit CPU time, since some compression algorithms take a very long time, we sampled 10 K malicious logs from Dec. 2014 and Jan. 2015 Sandbox logs and 10 K legitimate logs from Feb. 2015 and Mar. 2015 Proxy logs. In addition, we take the URL as only one attribute to generate a feature vector and execute a simple scoring calculation as follows.

$$Score = \frac{Z_{neg}}{Z_{pos}} \tag{7}$$

The process of registering a training dataset to a compression algorithm feature generator differs depending on how the compression algorithm works. Now, we overview the compression algorithm and its characteristics.

LZSS utilizes a sliding dictionary, which compresses data by only recording relative position and data sizes when the target data matches the longest data in the previous sliding window. Thus, one characteristic of LZSS is that data outside of the sliding window are not considered for compression. In general, a 32 kB sliding window is widely utilized. However, we implemented LZSS with a 20 kB sliding window for lowering computational cost. The compression rate of target data x is calculated by combining x with each 20 kB window data and applying LZSS and then returning the minimum compression rate as the final score.

LZT is a variant of LZW, and both are utilized in gif format files and compression commands. The same as LZW, LZT compression is based on dynamic dictionary insertion where new data sequences are added in a dictionary and target data x is recognized as the dictionary ID whose data sequence has the longest match with the target data. In LZT, dictionaries are composed with a Trie tree. In our evaluation, the compression rate of target data x is calculated by looking up the dictionary. Unlike LZW, which discards new data sequences when the dictionary is full, LZT swaps the LRU (least recently used) data sequence for a new data sequence.

bzip2 and LZMA utilize block sort and Markov algorithm based compression, respectively. The same as LZSS, bzip2 needs a blocking area, and LMA needs a sliding dictionary area. However, these areas are much larger than those for LZSS. Hence, our implementation stores all data for the compression algorithm where the compression score is calculated by simply combining target data x with the training dataset and compressing it.

For implementation, we utilized an existing python library for LZSS (libz), bzip2 (libzip2), and LZMA(liblzma). LZT is implemented in-house with cython.

In this evaluation, we set the classifier as an SVM, and TPR/FPR as a per log-basis calculation. We set the smoothing parameter as $\gamma = 10$ for all evaluations.

4.4 Selection of URL Attributes and Classifier

A URL has several attributes such as the URL itself, FQDN, Path, and QueryString. (For simplicity, in this research, we use Path to mean both Path and QueryString.) Through our evaluation, we can select the best URL attributes for detecting malicious URLs.

We evaluate the accuracy of URL attributes. We take an URL itself, FQDN, Path, and combination of FQDN and Path for testing. In this evaluation, we set the classifier as an SVM, the compression algorithm as LZT, and TPR/FPR as a per host-basis calculation.

The classifier is another factor for selection. We evaluate detection capability between different classifiers: Ridge regression, linear SVM, LDA(Linear Discriminant Analysis), NB (Naive Baise), and Adaboost. To conduct fair

Table 2: Classifier Performance and Execution Time Comparison Between Compression Algorithm

Algo.	Configs	CPU Time	AUC	pAUC	$TPR_{0.5\%}$
LZSS	level 6	16490s	0.968	0.0797	60.90%
LZT	24bit dict	20s	0.973	0.0825	68.10%
bzip2	level 9	68690s	0.521	0.0057	0.40%
LZMA	—	98112s	0.975	0.0828	68.30%

comparison, we execute grid-search to select best hyper-parameters for each classifiers. Implementation is done by using a python scikit-learn library. In this evaluation, we set the URL attribute as the FQDN and Path combination, the compression algorithm as LZT, and TPR/FPR as a per host-basis calculation.

4.5 Comparative Evaluation

We evaluate the proposed method in comparison with the conventional BoW-based detection method. First, we compared detection capabilities of the proposed and conventional methods. We also measured detection accuracy over time to find out how fast trained models deteriorate. Computational efforts are another important factor for practical use, so we measure CPU time and memory usage of the proposed and BoW method. BoW of a URL is extracted by setting $/, ?, =, \&$ as a separator and splitting the URL. In this evaluation we set the classifier as SVM, the compression algorithm as LZT, and TPR/FPR as a per host-basis calculation.

5 EVALUATION RESULT

5.1 Selection of Compression Algorithm

Table 2 shows the evaluation results for different compression algorithms. LZMA gives the best AUC, $pAUC$ and $TPR_{0.5\%}$ but takes the longest to compute. In contrast, LZT gives similar AUC, $pAUC$, and $TPR_{0.5\%}$ to LZMA and computes very fast. Hence, we selected LZT as the default compression algorithm in the later evaluation. bzip2 and LZSS do not perform as well as LZMA and LZT.

5.2 Selection of URL Attributes and Classifier

Table 3 shows the evaluation results of different attributes of a URL. From these results, the combination of URL attributes FQDN and Path gives the best $TPR_{0.5\%}$ and $pAUC$. Hence, we select the FQDN and Path combination as the default URL attribute in the later evaluation.

To select a suitable classifier, first, we visualize malicious and legitimate features. We calculated the compression algorithm feature and mapping onto a 2D surface in Fig. 2 where the horizontal axis is Z_{pos} (i.e., the

Table 3: Evaluation between URL attributes

URL attributes	AUC	pAUC	$TPR_{0.5\%}$
URL	0.8965	0.0720	41.80%
FQDN	0.8956	0.0631	43.60%
Path	0.7990	0.0562	43.20%
FQDN, Path	0.9306	0.0825	65.30%

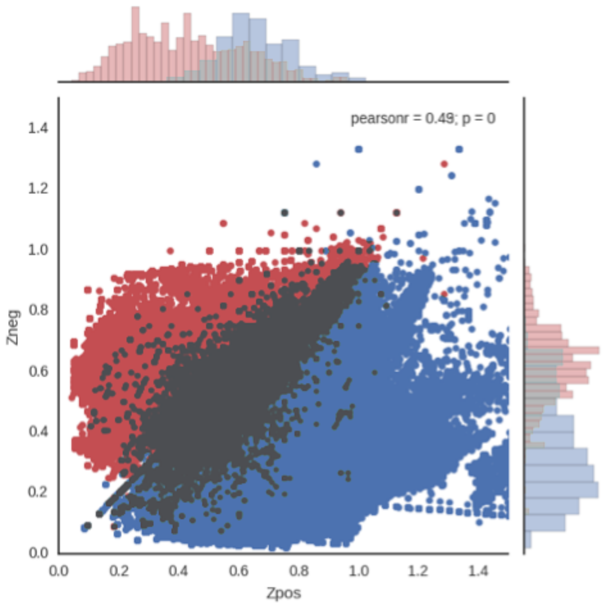


Figure 2: Compression algorithm feature mapping of malicious logs (red) and legitimate logs (blue)

malicious compression rate) and the vertical axis is Z_{neg} (i.e., the legitimate compression rate).

This visualization shows that although some overlapping areas exist, malicious logs (red legend) and legitimate logs (blue legend) are mapped in the upper-left and lower-right areas, respectively. Many of malicious and legitimate logs seems to be linearly separated on the basis of Z_{pos} and Z_{neg}. Hence, these visualized results suggest that linear classification works well.

Table 4 shows the detection capability of different classifiers. SVM gives the best $TPR_{0.5\%}$ and $pAUC$. Hence, we select SVM as default classifier in later evaluation.

5.3 Comparative Evaluation

Table 5 shows the evaluation results for the proposed and BoW-based classification methods. From these results, the proposed method has better $TPR_{0.5\%}$ and $pAUC$ than the conventional BoW-based classification method.

Figure 3 shows the $TPR_{0.5\%}$ deterioration over time where the vertical axis is $TPR_{0.5\%}$ and the horizontal axis is time in weeks. This figure shows that $TPR_{0.5\%}$ gradually decreases over time. However, the proposed method always achieves a higher $TPR_{0.5\%}$ than the BoW method until 14 weeks have past.

Table 6 and 7 show comparison with BoW method on CPU time and memory usage, respectively. Proposed

Table 4: Evaluation between classifiers

Classifier	AUC	pAUC	$TPR_{0.5\%}$
Ridge α=0.1	0.9268	0.0791	59.40%
SVM C=0.025	0.9306	0.0825	65.30%
LDA	0.9291	0.0784	57.30%
NB	0.8166	0.0602	13.50%
AdaBoost	0.9420	0.0785	58.30%

Table 5: Evaluation with conventional BoW-based classification method

Method	AUC	pAUC	$TPR_{0.5\%}$
Proposed	0.9306	0.0825	65.30%
BoW	0.9030	0.0657	32.00%

method consumes most of CPU time for compression process and its time is longer than any other process of BoW method. Still, once compression is completed, feature generation, training and detection are finished with less CPU time than BoW method. As for memory usage, proposed method consumes small memory for compression process and less memory for feature generation, training and detection compared with BoW method. Although BoW method generates one-hot vector for every single word appeared in URL so that memory usage tends to increase, proposed method generates compression algorithm feature vector in several dimensions so that memory usage does not steeply increase.

6 CONSIDERATION

We consider the reason the compression algorithm feature contributes to better classifying malicious and legitimate logs. Figure 4 shows the histogram of Z_{pos} of URL attributes for both malicious and legitimate logs, where the red and blue zone are histograms of malicious and legitimate logs, respectively. The histogram of malicious logs contains three peaks: A) the compression rate is very small, B) the compression rate is as high as that for legitimate logs, and C) the compression rate is very high.

A sample URL that belongs to pattern A is shown in table 8. For security reason, FQDN is masked with 'www.example.com' and QueryString values are masked with meta words. The first row shows the original URL string and its length, the second row shows the LZT compressed state and relative entropy with malicious logs, and the third row shows that with legitimate logs, where '|' shows that data sequences between '|' marks are expressed in 1 code. In compressing with malicious logs, the table shows that a 1,352-bit-long URL is compressed to 220 bits and many data sequences are expressed as 1 code. Especially in QueryString of URL, almost one key (e.g. "dstid=1") or one combination of a key and value (e.g. "countryid=...") is compressed as 1 code. This observation suggests that a QueryString key and

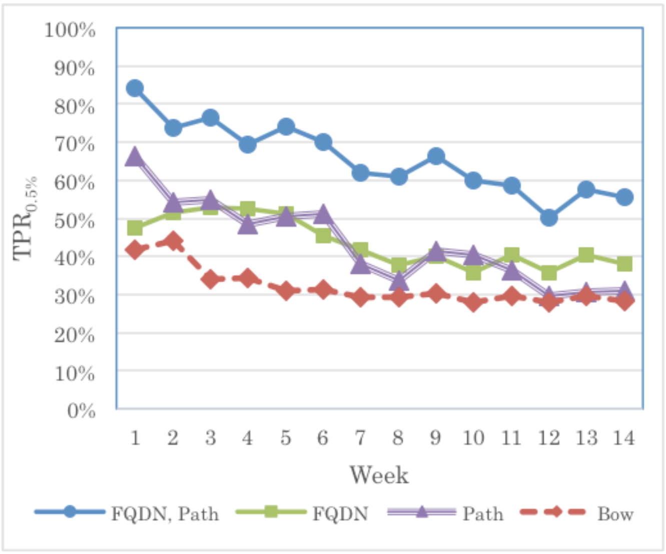


Figure 3: *TPR* deterioration over time

Table 6: CPU Time Comparison with Conventional Method (seconds)

Method	Compress	Generate Feature	Train	Detect
Proposed	13,809	3,752	22	438
BoW	-	7,145	950	408

value combination that exists in the training dataset is automatically recognized and compressed as 1 code. In contrast, a key and value combination that does not exist in the training dataset is automatically split. This is one use case that QueryString key exists but its value is modified in malware communication.

Other examples of pattern A for the FQDN attribute are FQDNs having sequential numbers in host names such as host1.example.com and host2.example.com. These FQDNs are recognized as totally different strings by exact matching, but in the compression algorithm that has the characteristic of longest matching, two FQDNs are recognized as similar strings. In fact, host2.example.com is compressed as |host|2.|example.com| after training data host1.example.com. This is another use case that FQDN is partially modified to similar FQDN.

URLs belongings to pattern B tend to have same FQDNs existing both in malicious URLs and legitimate URLs. Since Path of these URLs are different, compression rate does not get so small against both malicious logs and legitimate logs and makes detection difficult.

A typical URL belongings to pattern C is shown in table 9. This URL has an encoded or encrypted string. The second row shows compression results of the URL. It shows that the compression rate becomes large for both malicious and legitimate logs and makes classification difficult.

From this consideration, the proposed method is capable to detect malicious URL strings that are similar

Table 7: Memory Usage Comparison with Conventional Method (MB)

Method	Compress	Generate Feature	Train	Detect
Proposed	2,401.0	15,999.8	4.7	424.4
BoW	-	56,988.2	1,241.2	529.4

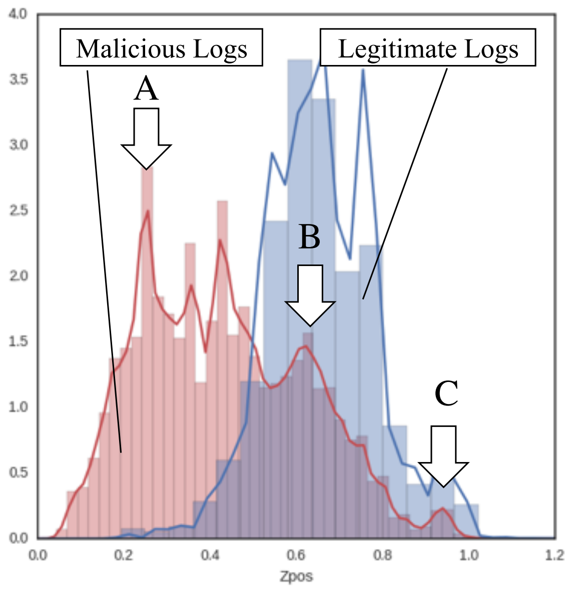


Figure 4: Histogram of Z_{pos}.

to but slightly different from existing malicious URLs. Of course, attacker can totally change URL strings from past attack vectors, in this case, proposed method does now work well. However, assuming that many attackers tend to use existing attackers' tool kit to set up their attack vectors and these tools are not so often drastically modified, proposed method should still be feasible.

7 CONCLUSION

We proposed a novel method for detecting malicious communication of infected hosts by generating a compression algorithm feature of URL attributes and classifying with supervised learning. Through evaluation, we demonstrated that the proposed method has higher detection capability than the conventional BoW-based detection method. In particular, its *TPR* in a low *FPR* area (0.5%) is over 30% higher than that of the BoW-based method. In addition, we clarified how the compression algorithm works in classification and demonstrated a real use case in which the proposed method detected malicious URL strings that are similar to but slightly different from existing malicious URLs.

REFERENCES

[1] https://www.av-test.org/fileadmin/pdf/security_report /AV-TEST_Security_Report_20162017.pdf, " ECURITY REPORT 2016/17 " , AV-TEST, (2017).

Table 8: Effectively classified URL and compression state

Uncompressed URL: 1352bit:
http://www.example.com//offers/DynamicOfferScreen?offerid=foo&distid=bar&leadp=baz&countryid=qux&sysbit=quux&dfb=corge&hb=grault&isagg=garply&version=waldo&external=fred&external=plugh&
Relative Entropy for Legitimate Logs: 900bit
|http://www.example.com/|/of|fers|/D|yna|mic|Off|erS|cre|en|?o|ffe|rid=foo|&dis|tid=b|ar|&le|adp|=ba|z&c|ountry|id=qu|x&s|ys|bit|=quux&|dfb|=corge&|hb=|grault&is|agg|=garply&ver|sion=|waldo|&ex|ternal|=fred&e|xter|nal|=plugh&|
Relative Entropy for Malicious Logs: 220bit
|http://www.example.com//offers/DynamicOfferScreen?offerid=foo|&distid=b|ar|&leadp=baz&|countryid=qux&sysbit=quux&df|b=corge&hb=grault|&isagg=garply&versio|n=wal|do&e|xternal=fred&e|xternal=plugh&|

Table 9: Poorly classified URL and compression state

Uncompressed URL: 4688 bits
http://www.example.com/api/vp/1?clk=gLg_PHWA9aSyioXkt-F4b3J9cI1ybf-t-x7VxWH5dmAWXwln-z ...(omit)
Relative Entropy for Legitimate Logs: 4420bit
|http://www.example.com/api/|vp|/1|?cl|k=|gLg|_PH|WA9|aS|yio|Xkt|-F4|b3J|9cI|1y|bf-|t-|x7|VxW|H5d|mAW|Xwl|n-|z···(omit)
Relative Entropy for Malicious Logs: 4980bit
A|http://www.example.com/api/v|p/1|?cl|k=|gL|g_|PH|WA|9a|Sy|io|Xk|t-|F4|b3|J9|cI|1yb|f-|t-x|7V|xWH|5d|mA|WX|wl|n-|z···(omit)

[2] D. Benedetto, E. Caglioti, and V. Loreto, " Language trees and zipping " , Phys. Rev. Lett., vol.88, (2002).
[3] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, " Towards Parameter-Free DataMining " , KDD, pp. 206-215, (2004).
[4] Y. Marton, N. Wu, and L. Hellerstein, " On compression-based text classification " , European Conference on Information Retrieval, pp. 300-314, (2005).
[5] A. Bratko, G. V. Cormack, B. Filipič, T. R. Lynam, and B. Zupan, " Spam filtering using statistical data compression " , Journal of Machine Learning Research, vol.7, pp.2673-2698, (2006).
[6] K. Nishida, R. Banno, K. Fujimura, and T. Hoshide, " Tweet-Topic Classification using Data Compression " , DBSJ Journal, Vol.10, No.1, pp.1-6, (2011).
[7] H. Adachi, M. Okabe, and K. Umemura, " Recognition of Music Composer with Compression-based

Dissimilarity Measure ” , DEIM2013.

[8] M. Dredze, K. Crammer, and F. Pereira, “ Confidence-weighted classification ” , Proceedings of 25th International Conference on Machine Learning, pp.264–271, (2008).

[9] A. Kumagai, Y. Okano, K., and M. Tanikawa, “ Supervised Classification for Detecting Malware Infected Host in HTTP Traffic and Long-time Evaluation for Detection Performance using Mixed Data ” , IEICE-ICSS, Vol.116, No.522, pp.43-48, (2016).

[10] T. Nelms, R. Perdisci, and M. Ahamad, “ ExecScent: mining for new C&C domains in live networks with adaptive control protocol templates ” , 22nd USENIX Conf., pp.589–604, Aug. (2013).

[11] K. Bartos, M. Sofka, and V. Franc, “ Optimized invariant representation of network traffic for detecting unseen malware variants ” , in USENIX Security Symposium, pp. 807–822, (2016).

[12] K. Aoki, T. Yagi, M. Iwamura, and M. Itoh, “ Controlling malware http communications in dynamic analysis system using search engine ” , Cyberspace Safety and Security (CSS), 2011 Third International Workshop onIEEE, pp.1–6 (2011).

[13] Y. Okano, A. Kumagai, M. Tanikawa, Y. Oshima, K. Aiko, K. Umehashi, and J. Murakami, “ Proposal of selection of training data using misdetected goodware for preventing misdetection of a static detector of malware”, IEICE-PRMU, Vol.115, No.224, pp.163-170, (2015).

[14] L. E. Dodd and M. S. Pepe. “ Partial AUC estimation and regression”, Biometrics, 59(3), pp.614–623, (2003).

[15] P. Tischer, “ A modified Lempel-Ziv-Welch data compression scheme ” , Aust. Comp. Sci. Commun. 9, 1, pp.262-272, (1987).

(Received November 28, 2019)
(Revised March 30, 2020)

Yasushi Okano He received a Master degree in Interdisciplinary Environmental Science from Kyoto University in 1995, and presently a senior research engineer at NTT Secure Platform Laboratories. He works on network and IoT security researches.

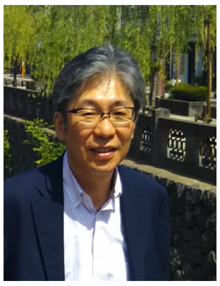
Kazunori Kamiya He received a Master Degree in Frontier Science from the University of Tokyo in 2004, and presently a senior research engineer at NTT Secure Platform Laboratories. He works on network security and network operation researches.

Atsutoshi Kumagai He received the B.S. degree in informatics and mathematical science form Kyoto University in 2010, and the M.S. degree in informatics from Kyoto University in 2012. In 2012, he joined NTT and is currently a researcher of NTT Software Innovation Center and NTT Secure Platform Laboratories, Tokyo, Japan. His research interests include machine learning, data mining, and cyber security.

Taishi Nishiyama He received his bachelor’s degree in Undergraduate Course Program of Aeronautics and Astronautics from Kyoto University in 2014, and master degree in Aerospace Engineering from University of Tokyo in 2016. He has worked in machine learning and network security at NTT Secure Platform Laboratories.

Bo Hu Bo Hu received an M.S. in wireless network engineering from Osaka University in 2010 and joined NTT the same year. He has mainly been engaged in researching network security technology, machine learning and inter-cloud technology. He developed a security orchestration architecture, a machine learning pipeline for large-scale traffic analysis, and inter-cloud protocols. He has also worked on cloud computing standardization activities in the Telecommunication Standardization Sector of the International Telecommunication Union and other standardization organizations.

Masaki Tanikawa He received a Master degree in Systems Science from Tokyo Institute of Technology in 1995, and presently a senior research engineer, supervisor at NTT Secure Platform Laboratories. He works on network security and network operation researches.



Kazuhiko Ohkubo Kazuhiko Ohkubo is the CISO of Kyowa Exeo Corporation. He received his M.S. in electrical engineering from the University of Tokyo in 1989. He received his Ph.D. in business administration and computer science from the Aichi Institute of Technology in 2019.He is a member of IEEE.

Regular Paper

Worker State Estimation Method with Reduced Manual Task for Teleworking Environment

Kazuyuki Iso[†], Takaya Yuizono[†], and Minoru Kobayashi[‡]

[†]Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, Ishikawa, Japan
[‡]Meiji University, 4-21-1 Nakano, Nakano-ku, Tokyo, Japan
{iso.kazuyuki, yuizono}@jaist.ac.jp
minoru@acm.org

Abstract - In a teleworking environment, sharing the states of workers is important to facilitate smooth communication; this requires a worker state estimation method to be adaptable to various workspaces. Previous methods have realized high estimation accuracy, but they have needed laborious manual work to suitably tag the learning data. This paper proposes a novel worker state estimation method by reducing the manual task of labeling using three automated processes: sensing, clustering, and selecting. A prototype system was developed and tested in two workspaces for evaluation. The system selected 26.1% and 22.5% of the obtained data for labeling in workspace 1 and workspace 2, respectively. As the data for labeling decreased, the observation time of a worker also decreased. Approximately 75% of the manual work could be reduced. The estimation accuracy was 93.2% in workspace 1 and 76.0% in workspace 2. This method was effective in reducing the manual labor involved in estimation. The estimation accuracy differed depending on the use conditions of the workspace. Methods to improve this metric are also discussed.

Keywords: Worker state estimation, Worker state sharing, Telework, Unsupervised clustering method

1 INTRODUCTION

A broadband network enables internet connectivity and allows the sharing of multimedia content. This allows a social infrastructure to be built, with which people can collaborate using multipoint video conferencing systems, chat systems, e-mails, and telephone conferences. In many companies, workers work remotely from various places, such as their homes. To work in cooperation with an organization, communication between the group members is important. Workers often need to communicate with remote workers; however, this also means that, while communicating, the individual work of the remote worker being contacted must be suspended. If remote workers are interrupted at inappropriate times, their individual work would suffer and, their productivity would decline [1][2]. When working in a common room, there is a shared awareness between the workers; a worker easily notices the states of the other workers. Therefore, a worker can initiate communication at an appropriate time. However, in teleworking, a worker is not able to easily discern the states of other workers, and therefore, abruptly sends them a message, either through a chat system or by a telephone call, thereby interrupting the work of those workers. For effective remote collaboration, recognition of the states of the remote workers is important. Related research [3]–[7] has reported on the significance of this factor.

This research aims to develop a worker state estimation method for use in teleworking. Figure1 shows the concept of the worker state sharing system. A terminal is placed in the room for teleworking. In the figure, co-worker B is a collaborator at another location. The role of this terminal is to estimate the states of the workers in the room and to share these states with other co-workers. The terminal is made aware of the state of a worker, which helps workers to time their communication opportunely. As shown in this figure, the system confirms the state of a worker and improves accuracy in estimation. At this time, the terminal classifies the state of the worker; the role of the worker is to teach the system how to share the current state. Following this process, the terminal does not disturb the worker when crafting, and the worker can appropriately communicate, or be communicated with, using a telephone or messenger, after the crafting activity is completed. The terminal can estimate the state adopted by the worker.

Many studies on human activity estimation have reported a high level of accuracy in estimation by using machine learning [8]. To build an estimator by the conventional method requires enormous amounts of learning data. The learning data are created by labeling supervised information onto data collected from numerous sensors. In many studies, creating this learning data is large and laborious a manual task.

Therefore, we propose a new method to construct an estimator that reduces the manual task. This method has four processes: sensing, clustering, selecting, and labeling. A comparison of the conventional method and the proposed method is shown in Fig. 2. The conventional method manually labels the sensor data and supervised information, and inputs them into the learning process. The target data for labeling include all the sensor data. The proposed method classifies the sensor data using a clustering method. The system selects the target data for labeling based on the results of the clustering process. The number of targets equals the number of clusters. As in the conventional method, the system initially creates an estimator with these four processes. In the proposed method, the process from sensing step to selecting step is automated

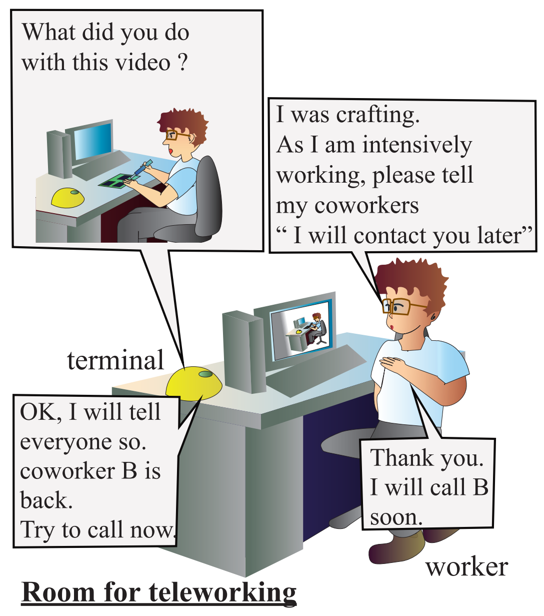

What did you do with this video ?
I was crafting. As I am intensively working, please tell my coworkers " I will contact you later"
terminal
OK, I will tell everyone so. coworker B is back. Try to call now.
Thank you. I will call B soon.
worker
Room for teleworking

Figure 1: Concept of the worker state sharing system. The terminal confirms the state of a worker and develops an estimator.

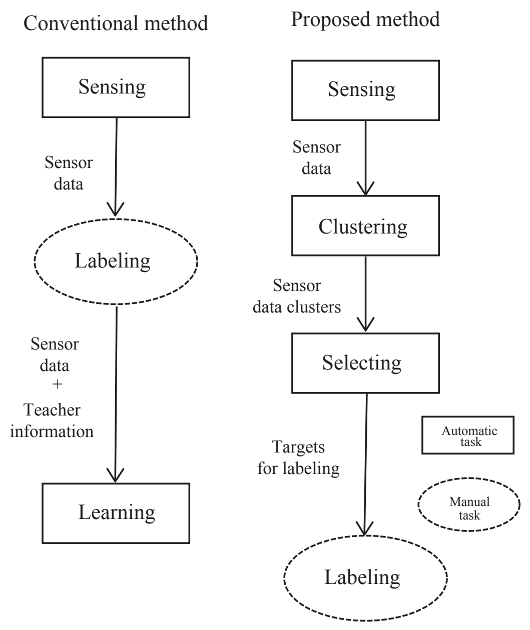


Figure 2: Comparison of the differences in the proposed method and conventional method

task. This process of selection decreases the manual tasks by reducing the number of targets for labeling. The labeling process is carried out only once after multiple targets are selected by the selection process. The worker observes the target scenes selected by the selection process and indicates the worker's state. An estimator is created by running these four processes only once. When estimating the worker's state from new sensing data, the system calculates the cluster containing the sensing data. At the time of collection of the sensing data, the worker's state is predicted by using the label of the cluster.

A prototype system was developed to evaluate the proposed method, and experiments were performed in two workspaces. The evaluation experiments confirmed the reduction in manual work. The rest of this paper is organized structure as follows. Chapter 2 describes related research on human activity estimation. Chapter 3 describes the proposed method in detail. Chapter 4 describes the experimental prototype system. Chapter 5 describes the experimental methods and their results. Chapter 6 provides a discussion of the results. Chapter 7 presents the conclusions.

2 RELATED RESEARCH

2.1 Human Activity Estimation

Numerous reports on human behavior estimation employ machine learning [8]. Avrahami et al. [9] reported on the estimation of the behavior of convenience store clerks and desk workers. They used a support vector machine (SVM) and the K-nearest neighbor (KNN) to learn the states of store clerks and office workers. A large amount of supervised data is required for learning using SVM- or KNN-based systems. In addition, Laput et al. [10] reported on a technological solution to estimate the activity in a house, and also used an SVM as the learning method. Both studies reported a high level of accuracy in estimation, but they required a large amount of learning data with supervised information to be generated.

Each telework environment is different in terms of the size and structure of the room, as well as the size and arrangement of the furniture. These conditions affect the data input from the sensor. Collecting data under all conditions is difficult, and learning data must be generated for each work environment. Reducing the magnitude of the task of generation of the learning data is important.

2.2 Sensor for Human Activity Estimation

Previous studies have reported that various sensors can be used to a estimate human activity. These sensors are classified into three types, as shown in Fig. 3.

Murao et al. [11] used wearable sensors to estimate the state of a remote worker. A wearable sensor can typically sense the worker to which it is attached. A wearable sensor is battery-powered, and therefore, requires regular charging. This charging is inconvenient for workers.

A personal computer (PC) is a well-known tool for teleworking. Hashimoto et al. [12] obtained useful information about remote workers by maintaining an operational log on the PCs of the workers. This method only estimates the work done using a PC.

Other methods for estimation have been proposed that use ambient sensors. Laput et al. [10] used a sensor module that combined multiple sensors in a room to estimate human activity. Avrahami et al. [9] installed a radio frequency (RF)-radar under a desk to sense human motion without interfering the work. However, these studies have a heavy manual tasks load of creating a large amount of learning data by labeling super-

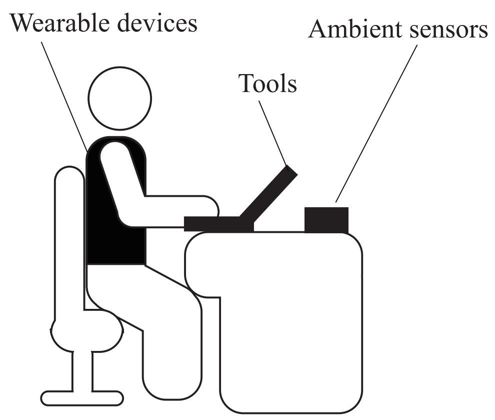


Figure 3: Classification of the sensors that collect the data for human activity estimation

vised information to sensor data.

Ambient sensors can be installed at locations that do not interfere with the actual work, and where a stable source of power is available. An ambient sensor can also collect data when the worker is not using the PC. Such ambient sensors are suitable for use in teleworking environments. The proposed method uses an ambient sensor installed in a telework environment. Our previous research [13][14] reported experiments to collect data using ambient sensors for classifying workers' states. The new proposed method creates an estimator using four processes including the selecting process and the labeling process.

3 PROPOSED METHOD

The proposed method consists of four processes: sensing, clustering, selecting, and labeling. The process from sensing to selecting is automated. Only labeling is the manual task. When developing an estimator with these four processes, the system records a video simultaneously along with the collection of sensor data. This video is used to observe the state of a worker in the labeling process. After clustering the sensor data, the system selects from the entire video the scene to be observed. This scene selection process shortens the observation time. The four processes are described in detail in the following paragraphs.

3.1 Sensing

The sensing process uses a microphone and distance sensor. The microphone detects signals based on the behavior of the worker (e.g., worker voice, keystroke sound, and door opening and closing sounds). The distance sensor detects the area where the worker is present. Our previous research [13][14] reported that the states of a worker can be clustered using vibration and distance sensors. In this research, we replaced the vibration sensor with a microphone, which can sense voices and sounds all around in a workspace.

The feature quantity, as shown in Fig. 4, is determined at fixed time intervals from each sensor data. The microphone data are Fourier transformed, and the distance sensor data are averaged.

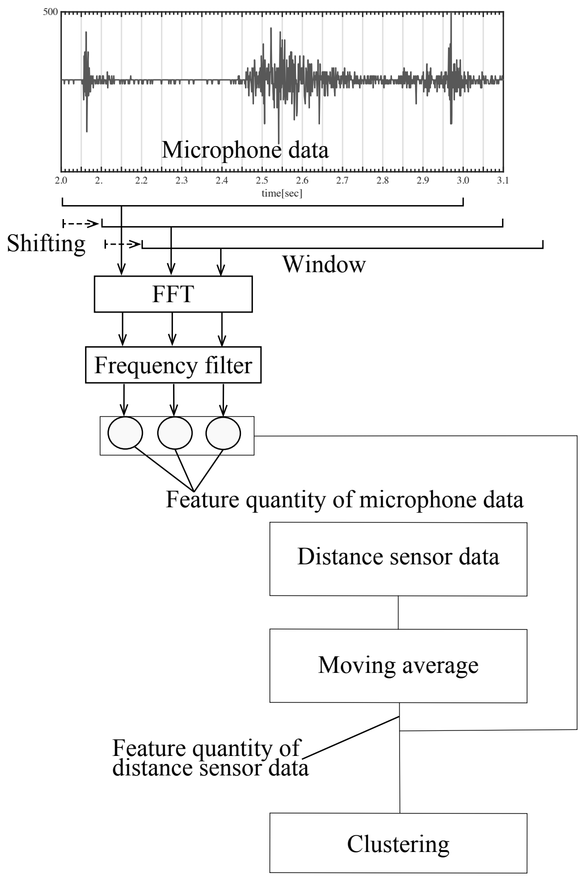


Figure 4: Sensing of the data from a microphone and distance sensor and calculation method of the feature quantity.

3.2 Clustering

This process uses an unsupervised clustering method. There are many clustering methods [15]. We compared the following four methods in terms of the accuracy and calculation time: k-means method, Gaussian mixture model (GMM) method, mean-shift method, and spectral clustering method (Table 1). For this evaluation, data were collected in the workspace for teleworking. The microphone and distance sensor data collected in the sensing process were used. When data collection was conducted, the worker's states were classified into the following four types: "Meeting via video conference.", "Using a PC on a desk", "Crafting on a desk", and "Leaving the seat". The data were collected for one day only. The calculation time ratio in Table 1 is a ratio for calculating time using the k-means method.

The proposed method uses the k-means method. The two methods, mean-shift and spectral clustering, slightly improve the accuracy compared to the k-means method, but the calculation time is more than 18 times, which is extremely long. There is nearly no difference between the accuracies of the GMM and k-means methods; however, the calculation time

Table 1: Accuracies and calculation times of the clustering methods

Method	Accuracy rate [%]	Calculation time ratio
k-means	89.4	1.0
GMM	89.6	1.3
Mean-shift	91.0	21.8
Spectral clustering	91.6	18.4

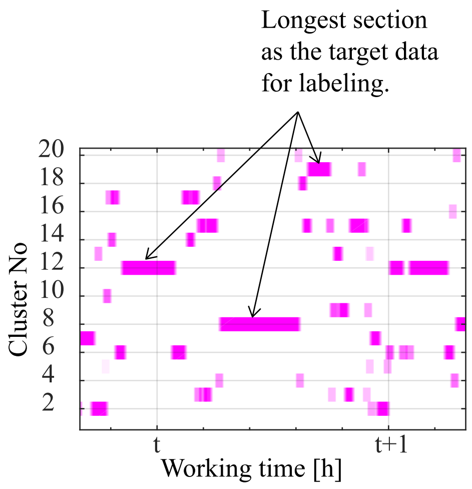


Figure 5: Example of arranging the sensor data in a time series and selecting the longest section as the target data

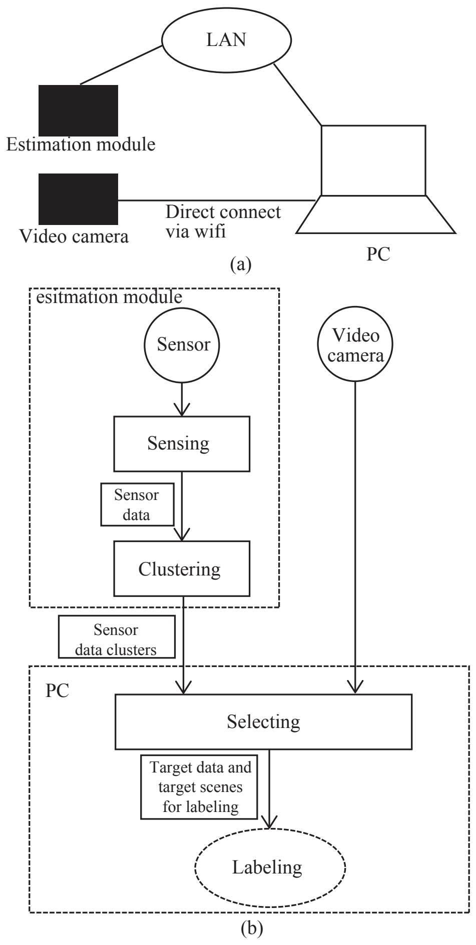


Figure 6: Prototype system. (a) Hardware configuration. (b) Flow of the four processes in the prototype system.

for the latter is short.

The k-means method can be used for clustering with sensors, even on small computers. Miniaturization of a terminal facilitates installation in the workspace.

3.3 Selecting

The system selects the target data so as to reduce the manual task of labeling. In the labeling process, a worker observes the video of the sensed time of the target data, determines the worker state, and labels the target data. The system selects the target data from the clustering result and cuts out the video scene captured when the selected data are sensed. The state of a worker may change frequently in a short time or remain the same for a long time. When a scene that changes frequently is selected, the length of the video to be observed is reduced, but the determination of the state becomes difficult. When a scene in a specific state continues to be selected, the video scene to be observed becomes longer, but the determination of the state is smooth. This process arranges the data into the cluster in a time series and selects the longest section as the target data for labeling (Fig. 5). The system cuts out the video scene from the time when the target data are sensed and presents the video scene to the worker. The system selects the data of one section from one cluster.

3.4 Labeling

The worker observes the video scene selected by the system and determines the worker state. The worker returns the determined state to the system. The number of video scenes observed in this process is the same as the number of clusters. The length of the video scene is shorter than that in the selecting process. The observation time for labeling is shorter than in the case of observing all the video scenes.

The system provides a complete estimator, with the workers labeling each cluster. When new sensing data is provided to this estimator, the distance between the data and the center of each cluster is calculated; the cluster was created by the k-means method. The closest cluster is selected, and the worker's state is predicted by using the label of the cluster.

4 Prototype system to develop estimator using four processes

As shown in Fig. 6, a prototype system is developed to evaluate this method. The prototype terminal consists of a sensor and a small computer, and executes the processes of sensing and clustering. The timing of the prototype terminal and video camera are synchronized. The prototype terminal is described in detail in the next section.

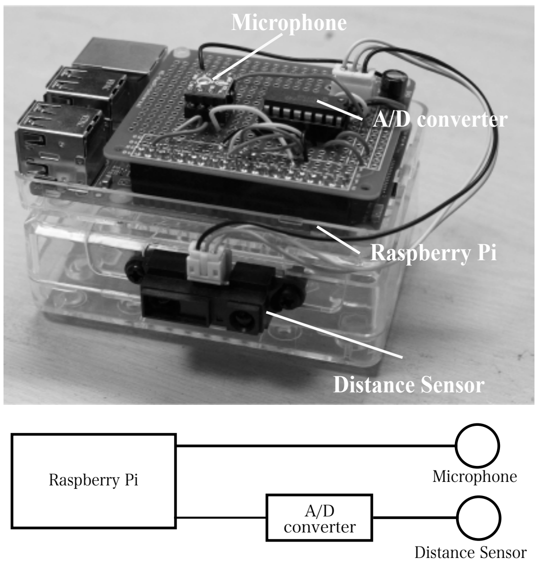


Figure 7: Prototype terminal with a combination of a microphone and distance sensors.

The video camera of this system is GoPro5. The prototype terminal and the PC are connected by the same LAN. Videos from the camera are transmitted directly to the PC. The clustering results are transmitted from the prototype terminal to the PC.

In the selecting process, the prototype program on the PC selects the target data, cuts out the video, and presents the video scenes to the worker. In the labeling process, the worker observes the video and labels the target data on the same PC.

The prototype terminal implements the sensing and clustering processes (Fig. 7). The terminal is used on a work desk. This terminal receives data from the microphone and distance sensors synchronously using a microcontroller. The microcontroller uses Raspberry Pi3 Model B. The microphone used is ADMP441, manufactured by Analog Devices. The distance sensor used is GP2Y0A710K, manufactured by SHARP. The direction of the distance sensor of the terminal is adjusted to the location where a worker sits. The sensing data from the two sensors are clustered by the k-means method in this terminal.

5 EVALUATION TEST USING PROTOTYPE SYSTEM

5.1 Test Procedure

The terminals were positioned at two workspaces for the evaluation test. The first workspace (WS1) is a private room used by a worker at his home, that the worker used for teleworking once or twice in a week. Other co-workers never enter the room. When working in WS1, the worker in this room talks to other co-workers by videoconference. The estimator for WS1 was created by collecting data for one day at this workspace.

The second workspace (WS2) is a university office and is the room of an instructor. WS2 is primarily used by one worker, and other co-workers may enter it. In addition, the worker in this room talks to co-workers via a video conference. The estimator for WS2 was created from three-day sensor data, including in-room conferences and video conferences. The sensor data of the different workdays were collected and evaluated.

Table 2: Sensing result: collecting time and number of sensor data

Workspace	Total time	Number of data
WS1	12h 53min	44,654
WS2	16h 11min	59,105

Table 3: Selecting result: selecting time and number of sensor data. The last column is the number of video files created by the selecting process

Workspace	Total time	Number of selected data	Number of movie files
WS1	3h 21min	12,097	20
WS2	3h 38min	13,095	30

5.2 Test Result

Table 2 lists the collection time and number of sensor data determined using the prototype terminal.

The total time and number of data after the selecting process are listed in Table 3. The last column is the number of video files created by the selecting process. The system selects 26.1% of the data for labeling in WS1 and 22.5% data in WS2. The number of video scenes to observe is small: 20 scenes in WS1 and 30 scenes in WS2. As the target data for the labeling decreases, the observation time of a worker also decreases.

The four states of the workers in each workspace are observed in the video after the selecting process.

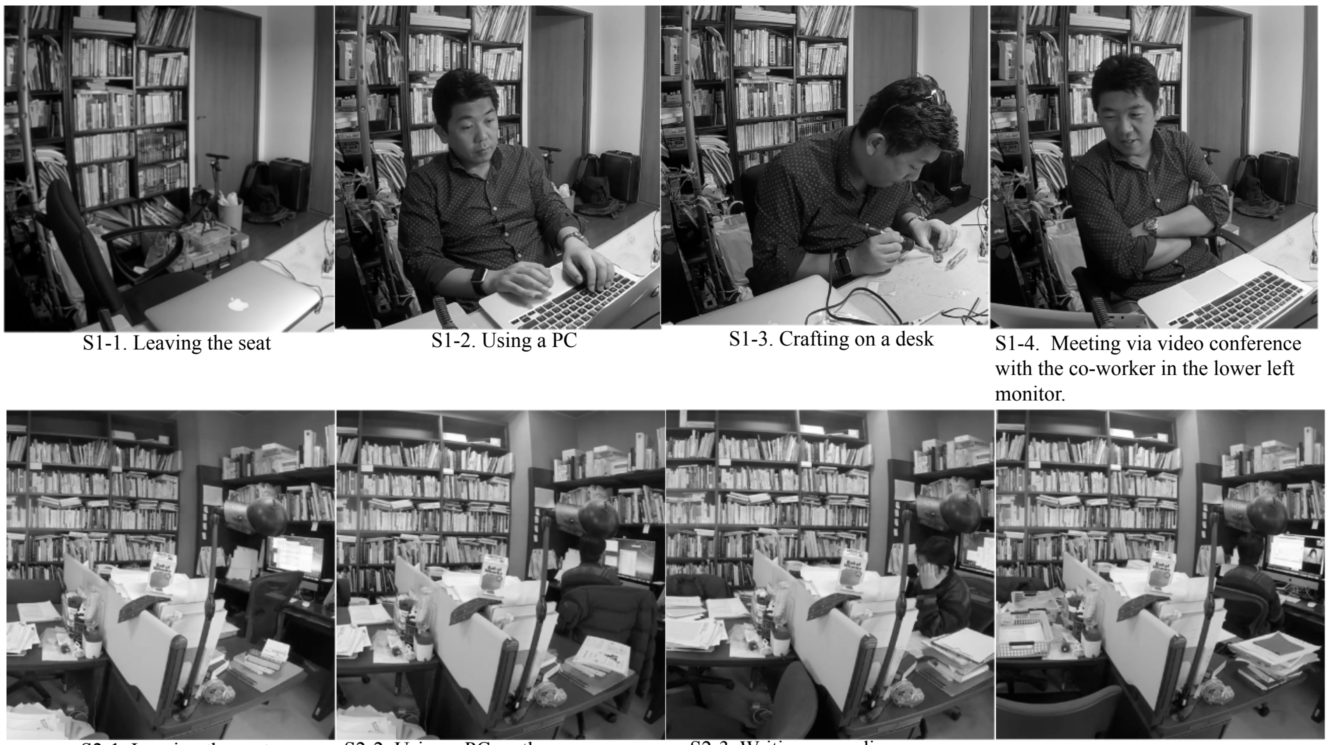
The states of a worker in WS1 are:

S1-1: Leaving the seat.
The worker leaves the seat and moves to the next room.

S1-2: Using a PC on a desk
For example, the worker creates documents using a PC on the desk or browses the web.

S1-3: Crafting on a desk
The worker works without using the PC. In this case, the worker crafts an electric circuit (soldering, cable making, and circuit assembly).

S1-4: Meeting via video conference.
Workers in both the workspaces use a PC to conduct a video conference. The workers talk to a remote worker. The worker in WS1 talks to other people in the same room.



S1-1. Leaving the seat S1-2. Using a PC S1-3. Crafting on a desk S1-4. Meeting via video conference with the co-worker in the lower left monitor.

S2-1. Leaving the seat S2-2. Using a PC on the wall side desk S2-3. Writing or reading on the central desk S2-4. Talking with other co-worker

Figure 8: States of a worker in workspace 1 and workspace 2

The states of a worker in WS2:

S2-1: Leaving the seat.
The worker leaves the seat and moves to the next room.

S2-2: Using a PC on the wall side desk
The worker uses a desktop PC on the wall desk. For example, the worker creates documents using a PC on the desk or browses the web.

S2-3: Writing or reading on the central desk
The worker works without using the PC. For example, the worker writes or reads paper documents. The worker works quietly and concentrates on the task.

S2-4: Talking with other co-workers
The worker talks with other co-workers in the same room. He also talks with a remote co-worker using a PC to conduct a video conference.

The estimator is created when a labeling result corresponds to a clustering result. The sensor data for the evaluation were collected on workdays different from the days when the estimator was made. The estimation accuracy at this time was 93.2% for workspace 1 and 76.0% for workspace 2. The estimation results for WS1 are shown in Fig. 9 and are shown in Fig. 10 for WS2.

The conventional method[9] using a single terminal could estimate the office workers' states with an accuracy of approximately 90%, and the same level of accuracy will be required in a telework environment. The result of successful estimation in WS1 was 93.2% and was very close to the workers' states observed in the video for several hours. The worker's states can be shared with an accuracy of 93.2%, and the effect of reducing unnecessary interruptions will be high. This result would be sufficient for meaningful sharing with remote co-workers.

In WS2, the estimation result and the workers' states showed a similarity of 76.0%. However, for the time segment T1 in Fig. 10, the estimation results for state "S2-4: Talking with other co-workers" were mostly incorrect for a duration of 25 minutes. The estimation accuracy was lower than that of WS1 and was insufficient. In the future, the effects of estimation errors need to be examined.

6 DISCUSSION

6.1 Reduction in Manual Tasks by Proposed Method

The video scene for labeling was reduced by more than 70%. At this time, each video contains the same work scene, and there is little change. The proposed method reduced the time required for observation and enabled the efficient determination of the state of a worker.

The labeling process was short and smooth, and estimators could be developed for each workspace. The worker state sharing system using these estimators could be constructed, and the sharing information was defined according to the task of each worker and the structure of the workspace. This method enabled co-workers to share each others state, and is good for creating communication opportunities according

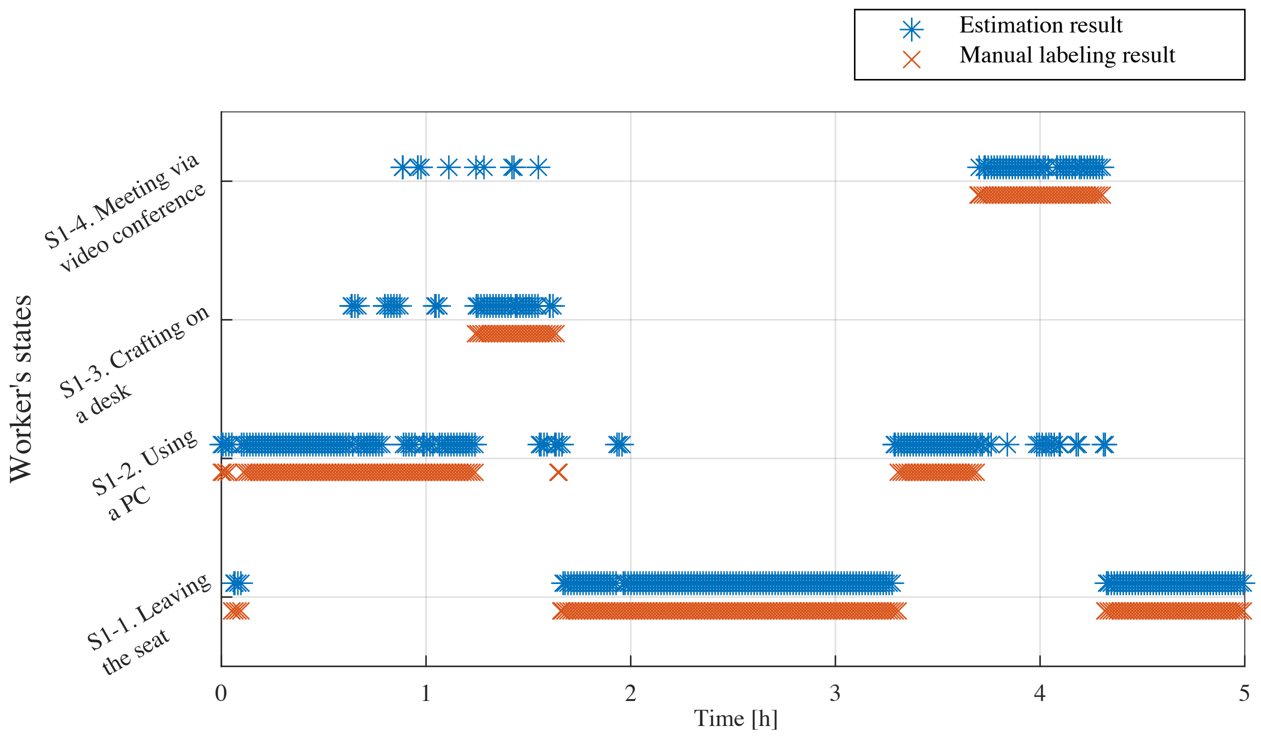


Figure 9: Estimation result for 5 h from the start of the work in workspace 1.

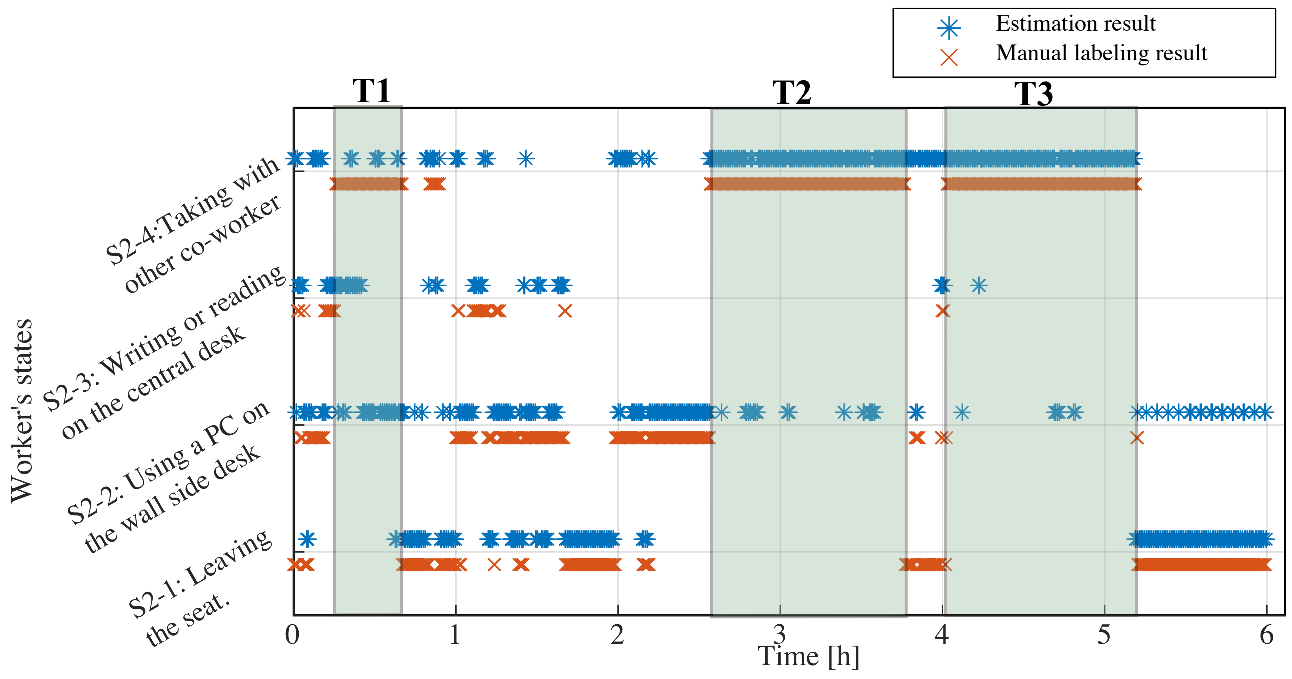


Figure 10: Estimation result for 6 h from the start of the work in workspace 2.

to the states of each worker.

Occasionally, even if a worker uses the same room, the features of the work may change, and the tools used may differ accordingly. In such cases, the estimator may have to be recreated to account for such changes. We expect the idea of the present method also contributes to reduce the amount of manual work to recreate the estimator. To develop a more robust method, we plan to examine the effectiveness of the present method to recreate the estimator based on a large amount of new sensor data generated when the features of the work has been modified.

6.2 Number of Workers in Workspace and Configuration of Sensors

The estimation accuracy of WS1 was high at approximately 90%, but the accuracy of WS2 was lower than that of WS1. One of the differences between the two workspaces was the number of workers. In time segment T1 in Fig. 10, another worker entered WS2, and the two workers talked. In addition, the two workers sometimes quietly browsed documents or searched for books on a bookshelf. In time segment T2 and T3 in Fig. 10, a worker in WS2 and a remote worker were talking to each other via video conference, and there was almost no quiet time. The estimation accuracies were different for these three segments because of the amount of voice data from the microphone. When designing this system, we assumed that more voice information could be obtained if there were two workers in the same room.

Only one distance sensor was attached to the terminal, and the system could not classify the differences in the states depending based on the number of workers. To classify the states of multiple workers, the terminal should have multiple distance sensors connected pointing in multiple directions, as the accuracy would then increase if the sensor data changed.

We will continue to improve the sensors on the terminals while investigating different conditions of the teleworking environment. Furthermore, we will install improved estimators in many telework environments and also evaluate the effect on remote collaboration work.

7 CONCLUSION

The sharing of the states of a worker with other workers is important to facilitate effective communication in teleworking. In this study, we developed a new method that required less manual work to develop a worker state estimator. We tested the new method for estimation in two workspaces. The system selected 26.1% of the data for labeling in workspace 1 and 22.5% data in workspace 2. As the target data for the labeling decreased, the observation time of a worker also decreased. Thus, 70% or more of the manual work could be reduced. The estimation accuracy at this time was 93.2% in workspace 1 and 76.0% in workspace 2. The proposed method significantly reduced the manual tasks. However, the estimation accuracy differed depending on the use conditions at the workspace. This difference may have been caused by presence of other co-workers. The accuracy could be improved by adding additional distance sensor(s) to consider other co-workers.

In future, the prototype system will be extended to remote collaboration to evaluate the effect of this method on smooth communication in this aspect also.

ACKNOWLEDGMENTS

We would like to thank Editage (www.editage.jp) for English language editing.

REFERENCES

[1] D. D. Salvucci, N. A. Taatgen, and J. P. Borst, "Toward a unified theory of the multitasking continuum: From concurrent performance to task switching, interruption, and resumption," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 1819–1828, ACM (2009).

[2] G. Mark, V. M. Gonzalez, and J. Harris, "No task left behind?: Examining the nature of fragmented work," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, (New York, NY, USA), pp. 321–330, ACM (2005).

[3] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work*, CSCW '92, pp. 107–114 (1992).

[4] T. Rodden, "Populating the application: A model of awareness for cooperative applications," in *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work*, CSCW '96, pp. 87–96 (1996).

[5] C. Gutwin and S. Greenberg, "Design for individuals, design for groups: Tradeoffs between power and workspace awareness," in *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, CSCW '98, pp. 207–216 (1998).

[6] C. Gutwin and S. Greenberg, "A descriptive framework of workspace awareness for real-time groupware," *Comput. Supported Coop. Work*, vol. 11, pp. 411–446 (2002).

[7] N. Romero, G. McEwan, and S. Greenberg, "A field study of community bar: (mis)-matches between theory and practice," in *Proceedings of the 2007 International ACM Conference on Supporting Group Work*, GROUP '07, (New York, NY, USA), pp. 89–98, ACM (2007).

[8] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Comput. Surv.*, vol. 46, pp. 33:1–33:33 (2014).

[9] D. Avrahami, M. Patel, Y. Yamaura, and S. Kratz, "Below the surface: Unobtrusive activity recognition for work surfaces using rf-radar sensing," in *23rd International Conference on Intelligent User Interfaces*, IUI '18, (New York, NY, USA), pp. 439–451, ACM (2018).

[10] G. Laput, Y. Zhang, and C. Harrison, "Synthetic sensors: Towards general-purpose sensing," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, (New York, NY, USA), pp. 3986–3999, ACM (2017).

[11] K. Murao and T. Terada, "A combined-activity recognition method with accelerometers," *Journal of Information Processing*, vol. 24, no. 3, pp. 512–521 (2016).
[12] S. Hashimoto, T. Tanaka, K. Aoki, and K. Fujita, "Improvement of interruptibility estimation during pc work by reflecting conversation status," *IEICE Transactions on Information and Systems*, vol. E97.D, no. 12, pp. 3171–3180 (2014).
[13] K. Iso, M. Kobayashi, and T. Yuizono, "A method for estimating worker states using a combination of ambient sensors for remote collaboration," in *Collaboration Technologies and Social Computing*, pp. 22–28, Springer International Publishing (2017).
[14] K. Iso, M. Kobayashi, and T. Yuizono, "A trial of ambient sensor method for worker states classifier," in *Proceedings of International Workshop on Informatics*, pp. 289–294 (2017).
[15] "scikit-learn Machine Learning in Python." `https://scikit-learn.org/`, accessed on May 10 (2019).

(Received June 10, 2019)
(Revised Jan. 14, 2020)

Kazuyuki Iso received a B.E. from Gunma National College of Technology,in 1998, and an M.S. from Japan Advanced Institute of Science and Technology, in 2000. He joined Nippon Telegraph and Telephone (NTT) Corporation, NTT Cyber Space Laboratories in 2000, where he has been engaged in research and development of communication systems using virtual reality technology. He is currently a Senior Research Engineer at NTT Media intelligence Laboratories, and a Doctor-course student at Japan Advanced Institute of Science and Technology. He is a member of IEEE, VRSJ, and IPSJ.

Takaya Yuizono received the B.E., M.E., and Dr. of Engineering from Kagoshima University, in 1994, 1996, 1999, respectively. He was a research associate in Kagoshima University, a lecturer and an assistant professor in Shimane University, respectively. He has been an associate professor in Japan Advanced Institute of Science and Technology since 2006. His research interests include in Groupware, CSCW, Creativity, and Education with interdisciplinary approach. He received best paper award in KES2005, NLP-KE2012, KICSS2016, and best paper award of journal of Japan Creativity Society in 2013, 2015, 2016, 2018. He is a member of ACM, IEEE, IPSJ, and IEICE.

Minoru Kobayashi is a professor at Meiji University, Tokyo Japan. His research interests include human-computer interaction, computer-supported cooperative work, and the design of IoT devices to enrich human-human communication. Kobayashi received a B.E. and an M.S. from Keio University, an M.S. from the Massachusetts Institute of Technology, and a Ph. D. in instrumentation engineering from Keio University. He is a member of ACM, IEEE, IPSJ, IEICE, and VRSJ.

Submission Guidance

About IJIS

International Journal of Informatics Society (ISSN 1883-4566) is published in one volume of three issues a year. One should be a member of Informatics Society for the submission of the article at least. A submission article is reviewed at least two reviewer. The online version of the journal is available at the following site: http://www.infsoc.org.

Aims and Scope of Informatics Society

The evolution of informatics heralds a new information society. It provides more convenience to our life. Informatics and technologies have been integrated by various fields. For example, mathematics, linguistics, logics, engineering, and new fields will join it. Especially, we are continuing to maintain an awareness of informatics and communication convergence. Informatics Society is the organization that tries to develop informatics and technologies with this convergence. International Journal of Informatics Society (IJIS) is the journal of Informatics Society.

Areas of interest include, but are not limited to:

Internet of Things (IoT)	Intelligent Transportation System
Smart Cities, Communities, and Spaces	Distributed Computing
Big Data, Artificial Intelligence, and Data Science	Multi-media communication
Network Systems and Protocols	Information systems
Computer Supported Cooperative Work and Groupware	Mobile computing
Security and Privacy in Information Systems	Ubiquitous computing

Instruction to Authors

For detailed instructions please refer to the Authors Corner on our Web site, http://www.infsoc.org/.

Submission of manuscripts: There is no limitation of page count as full papers, each of which will be subject to a full review process. An electronic, PDF-based submission of papers is mandatory. Download and use the LaTeX2e or Microsoft Word sample IJIS formats.

http://www.infsoc.org/IJIS-Format.pdf

LaTeX2e

LaTeX2e files (ZIP) http://www.infsoc.org/template_IJIS.zip

Microsoft Word$^{\text{TM}}$

Sample document   http://www.infsoc.org/sample_IJIS.doc

Please send the PDF file of your paper to secretariat@infsoc.org with the following information:

Title, Author: Name (Affiliation), Name (Affiliation), Corresponding Author. Address, Tel, Fax, E-mail:

Copyright

Publisher

Address: Informatics Laboratory, 3-41 Tsujimachi, Kitaku, Nagoya 462-0032, Japan

E-mail: secretariat@infsoc.org

CONTENTS