Int. J. Appl. Math. Comput. Sci., 2019, Vol. 29, No. 2, 245–259
DOI: 10.2478/amcs-2019-0018

IOT SENSING NETWORKS FOR GAIT VELOCITY MEASUREMENT

JYUN-JHE CHOU [a], CHI-SHENG SHIH [a,*], WEI-DEAN WANG [b], KUO-CHIN HUANG [c]

[a]Graduate Institute of Networking and Multimedia
National Taiwan University, No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan
e-mail: cshih@csie.ntu.edu.tw

[b]Department of Medical Education and Bioethics
National Taiwan University, No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan

[c]Department of Family Medicine
National Taiwan University Hospital, No. 1, Changde St., Zhongzheng Dist., Taipei 10048, Taiwan

Gait velocity has been considered the sixth vital sign. It can be used not only to estimate the survival rate of the elderly, but also to predict the tendency of falling. Unfortunately, gait velocity is usually measured on a specially designed walk path, which has to be done at clinics or health institutes. Wearable tracking services using an accelerometer or an inertial measurement unit can measure the velocity for a certain time interval, but not all the time, due to the lack of a sustainable energy source. To tackle the shortcomings of wearable sensors, this work develops a framework to measure gait velocity using distributed tracking services deployed indoors. Two major challenges are tackled in this paper. The first is to minimize the sensing errors caused by thermal noise and overlapping sensing regions. The second is to minimize the data volume to be stored or transmitted. Given numerous errors caused by remote sensing, the framework takes into account the temporal and spatial relationship among tracking services to calibrate the services systematically. Consequently, gait velocity can be measured without wearable sensors and with higher accuracy. The developed method is built on top of WuKong, which is an intelligent IoT middleware, to enable location and temporal-aware data collection. In this work, we present an iterative method to reduce the data volume collected by thermal sensors. The evaluation results show that the file size is up to 25% of that of the JPEG format when the RMSE is limited to 0.5°.

Keywords: Internet of things, middleware, compression, data fusion, data reduction.

1. Introduction

Walking exercises the nervous, cardiovascular, pulmonary, musculoskeletal and hematologic systems because it requires more oxygen to contract the muscles. Hence, *gait velocity*, called *walking speed* (Middleton *et al*., 2015), has become a valid and important metric for senior populations (Middleton *et al.*, 2015; Studenski *et al.*, 2011; 2003).

Studenski *et al*. (2011) published a study that tracked gait velocity of over 34,000 seniors for periods ranging from 6 years to 21 years in the US. The study found that the predicted survival rate based on the age, sex, and gait velocity was as that accurate as predicted based on the age, sex, chronic conditions, smoking history, blood pressure, body mass index, and hospitalization. Consequently, it motivated the industrial and academic communities to develop a methodology to track and assess the risk based on gait velocity. The following years led to many papers that point to the importance of gait velocity as a predictor of degradation and exacerbation events associated with various chronic diseases including heart failure, COPD, kidney failure, stroke, etc. (Studenski *et al*., 2003; Pulignano *et al*., 2016; Kon *et al*., 2015; Kutner *et al*., 2015). In the US, there are 13 million seniors who live alone at home (Administration on Aging, 2015). Gait velocity and stride length are particularly important in this case since they provide an assessment of fall risk, the ability to perform daily activities such as bathing and eating, and hence the potential for being independent. The assessment of gait velocity is recommended to instruct

*Corresponding author

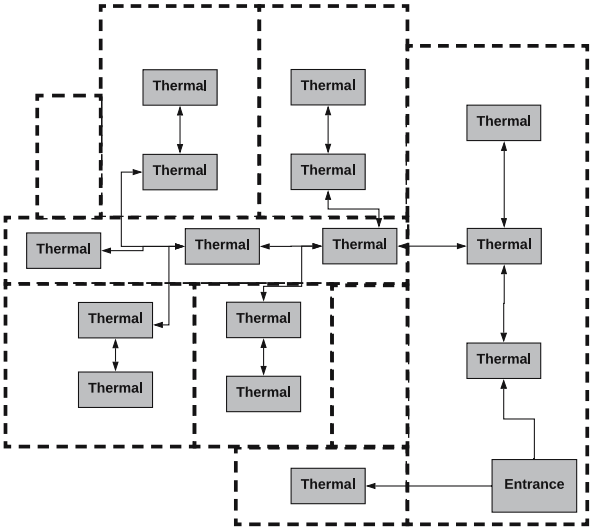


Fig. 1. Gait velocity measurement at smart homes.

the subjects to walk back and forth in a 5, 8 or 10 meter walkway. Similar results were found in a study comparing a 3 meter walk test to the GAITRite electronic walkway in individuals with a chronic stroke (Peters *et al*., 2014).

The above approaches are employed either at clinical institutes or designated locations. They are recommended by physicians but are required to be conducted in a limited time and location. Consequently, it is not trivial, if not difficult, to observe the change in long term. It is desirable for the elderly, their family members, and physicians to monitor gait velocity for the elderly all the time at any location. However, the assessment should take into account several factors, including accuracy, privacy, portability, robustness, and applicability.

Shih *et al*. (2017) proposed a sensing system to be installed at home or a nursing institute without revealing privacy and not using wearable devices. Given the proposed method, one may deploy several thermal sensors in his/her apartments, as shown in Fig. 1. In this example, numbers of thermal sensors are deployed to increase the coverage of the sensing signals. In large spaces such as a living room, there will be more than one sensor in one space; in small spaces such as a corridor, there can be only one sensor. One fundamental question is how many sensors should be deployed and how these sensors work together seamlessly to provide accurate gait velocity measurements.

To plan the number of sensors to be installed, one has to take into account if the sensing regions should overlap and if there is any interference. Our observations show that the above two questions are not trivial to answer. The first challenge is to overcome the errors caused by cone shape sensing signals. Figure 2 illustrates such errors.

When the person walks into the covered region shown on the left of the figure, the sensor will sense the leg first and show the heat sources at the edge of the cover space. Then, when the person continues to walk toward the center of the covered region, the sensor can sense the heat of their head, which is higher than the legs, but continuously shows the heat sources at the edge of the covered region. Consequently, on the heat map, they remain at the same locations; however, the person physically walks for 20 cm to 40 cm, depending on the height of the subject. This effect will lead to errors in gait velocity measurements on the edge of the covered region.

Another error is wrong location estimation, illustrated in Fig 3. (The dark and light grey boxes represent the heat sources whose temperature is greater than ambient temperature. Moreover, the heat sources marked in dark grey are hotter than the those marked in light grey.) When the person stands in overlapping regions of two adjacent sensors, both the sensors can

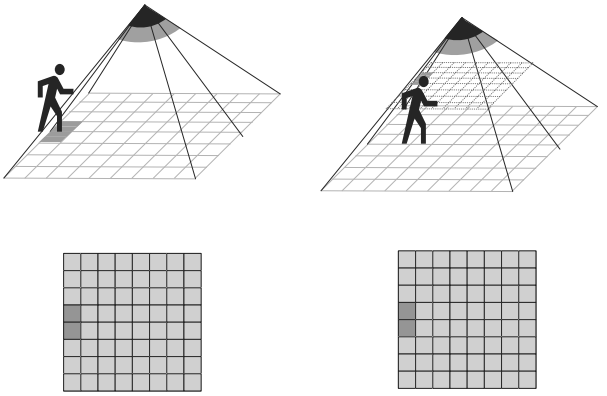


Fig. 2. Gait speed errors caused by cone-shape signals.

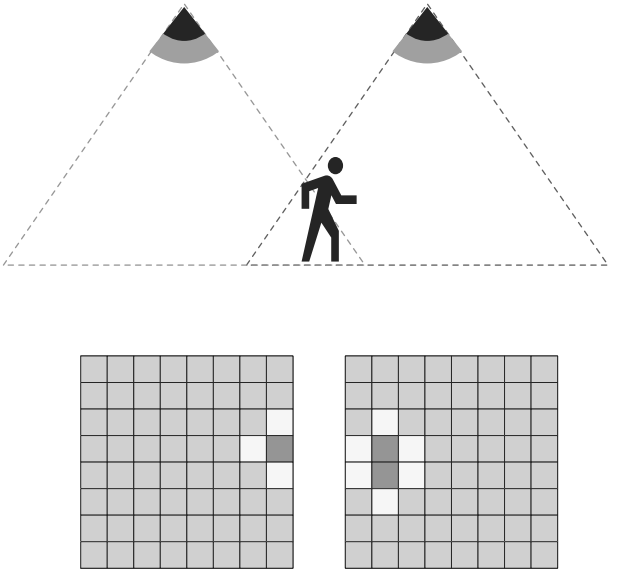


Fig. 3. Location errors caused by cone-shape signals.

detect him or her. When the sensor estimates the location of the person based on the highest temperature and assumes that it is the location of the head, the two sensors will mark the person at two different locations. Consequently, the heat source will be recorded twice in the system, which is not correct.

Collaborating with neighboring thermal sensors, the walking speed and walking length of a person can be detected more precisely, and the detected area will become large to cover all rooms in a house. However, the increased number of sensors will cause a large workload to the network and storage. Using the deployment in Fig. 1 as an example, there are fifteen thermal sensors in an apartment. If Panasonic Grid-Eye sensors are used, each of them will employ 2 bytes per pixel to store the data. This will generate 1.7 GB data per day when each sensor samples at 10 Hz and there are 64 pixels per frame. The challenge is to reduce the amount of data transmitted over the network and stored in the systems.

Contribution. The contribution of this work is to enable gait velocity measurements at home and private spaces. The developed method uses temporal and location properties of the sensed data to eliminate the aforementioned errors. The location framework in the WuKong middleware simplifies the development and maintenance overhead, which are the foundations of the proposed methods.

The remainder of this paper is organized as follows. Section 2 presents related works and a background for developing the methods. Section 3 presents the system architecture, challenges, and mechanisms developed. Section 4 presents the evaluation results of the proposed mechanism, and Section 5 summarizes our works.

2. Background and related works

2.1. Related works. Activity recognition and monitoring are two fundamental metrics to understand the health condition of patients in health care. An active lifestyle leads to a healthy body condition and slows down the aging, both physically and mentally. Hence there have been a large number of works on developing methods and technology of activity recognition and monitoring.

Many related works use wearable sensors to collect data and recognize the type of activities. Lee and Chung (2009) use shirts for activity monitoring. A smart shirt which measures electrocardiogram (ECG) and acceleration signals for continuous and real time health monitoring is designed and developed. The shirt mainly consists of sensors for continuously monitoring health data and conductive fabric electrodes to get body signals. The measured physiological ECG data and physical activity data are transmitted in an ad-hoc network in the IEEE 802.15.4 communication standard to a base station and a server PC for remote monitoring. The wearable sensor devices are designed to fit well into shirt with a small size and low power consumption to reduce the battery size. An adaptive filtering method to cancel artifact noise from conductive fabric electrodes in a shirt is also designed and tested to get clear ECG signals even during the running or physical exercise of a person.

Milenković *et al*. (2006) use wearable devices to collect vital sign signals. Although wearable, it is not likely to wear these devices 24/7 and collect data. In addition, the devices require clock synchronization protocols among wearable devices, which imposes unnecessary requirements for hardware devices and leads to heavy power consumption. Alemdar and Ersoy (2010) study several systems designed for health care. Many of them rely on wearable devices to detect the location of the subject of interest or to collect vital sign information. The transceivers deployed in the environment forward the received data to a central database for further analysis. Using wearable devices to collect vital signs has its own merits. However, this approach only works when the subjects of interest are wearing these devices and cannot be used as the only way to collect health-care data. Our work proposes to deploy low-cost sensors in the environment to collect data, which can operate 24/7 and are not subject to users' preference.

Gravina *et al*. (2017) propose an activity-as-a-service framework to support activity recognition and monitoring in mobility. The aim of this work is to provide a service to collect data gathered by a body sensor network (BSN) and high performance computing for processing the collected sensor data. It uses mobile devices as the transmission gateway between wearable devices and cloud services. Fortino *et al*. (2015) propose a framework to collect data from multiple body sensor networks and process the data in a distributed manner.

Other works deploy sensors in the environment to collect information and recognize the type of activities. Lu and Fu (2009) design and implement a robust location-aware activity recognition system to recognize users' behaviors. The average accuracy of activities correctly detected is 92%. Although the method provides robust data collection, it requires a specially designed floor which can measure pressure but is extremely expensive.

Yi *et al*. (2013) present a work to preserve the privacy at wireless sensor networks. The aim of that work is to design a lightweight encryption algorithm to protect the communication between the sensor node and server. In our works, we will preserve the privacy while sensing. Gheid and Challal (2016) propose a protocol to analyze the sensed data without revealing privacy. The protocol consists of a cosine similarity protocol to assess the similarity between the sensed activities and external patterns. Bourke *et al*. (2016) propose to use video analysis and a tri-axial accelerometer to recognize the

activity pattern. The aim is to use minimal sensors and provide accurate detection results. However, privacy is not preserved in this work.

Khalajmehrabadi *et al*. (2016), Zhao *et al*. (2016), and Hsu *et al*. (2017) proposed to use radio signal patterns to monitor gait velocity in rooms. Some technology was developed to recognize the type of activities and to measure gait velocity for multiple persons in a room. The experimental results show that the proposed approaches provide very accurate measurements. The average error rates are 1.9% and 4.2% for gait velocity and the stride length, respectively. However, the proposed approaches require expensive signal receivers, each of which may cost more than 2,000 USD (USRP, 2019) to measure each subject.

2.2. Intelligent virtual runtime for IoT devices. WuKong (Reijers *et al*., 2013; Shih, 2016; WuKong, 2012) is an intelligent middleware for designing and managing large scale IoT services. It consists of two major components to fill the gap for developing and managing IoT applications: an *intelligent run-time environment* and a *flow-based developing framework*. The WuKong run-time environment is regarded as a virtual middle-ware (VMW). The reasons for this are two-fold. First, as IoT applications are deployed at different locations and evolve over time, it is very likely that the systems use devices (including sensors, actuators, and computing platforms) developed by different manufactures and communicating via different network protocols. Having virtual devices allows applications to run on heterogeneous devices and networks. Second, when the system needs to be reconfigured, the process of reprogramming devices will be less expensive when using a virtual machine design. In addition, the number of lines of code will be lower since virtual devices can offer higher level primitives specific for IoT applications.

The flow-based developing framework provides a high level development environment to design hardware independent IoT applications. The goal of this framework is to allow domain experts to design their IoT applications without complete knowledge of hardware devices and network protocols to conduct the services. In this framework, users compose services using a data-flow model and pre-defined services. The framework then generates executable code for selected devices using a distributed computing model.

Run-time environment of WuKong devices. Most of the IoT application development environments are hardware dependent and require the developers to specify hardware properties while designing IoT/Smart city applications. For example, in the operating system, the CPU type and the port number for sensors should be known. In order to reduce management overhead, the developed system may be required to use identical devices and platforms, or to store configuration files for each type of platforms. It is certainly impractical in large scale and evolving smart city systems. WuKong run-time virtualizes IoT devices by deploying virtual machines to these resource-limited devices. This model does not only provide a hardware independent runtime environment, but also enhances reliability and security for IoT devices.

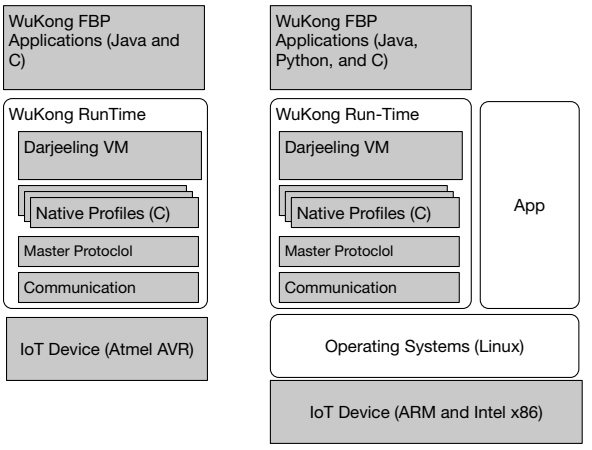


Fig. 4. WuKong run-time on IoT devices: on MCU-based devices (a), on uP-based devices (b).

Figure 4 shows the run-time environment on WuKong-enabled IoT devices. Figure 4(a) presents the runtime for micro-controller-based IoT devices such as Arduino MEGA2560, which is powered by an Atmel AVR ATmega2560 micro-controller. On this type of devices, there is no operating system and the WuKong run-time starts when the system is turned on. WuKong runtime consists of a communication component, a master protocol component, native profiles, and Darjeeling JVM. The communication component is responsible for communicating with radio interface on the device so as to send and receive messages to/from other devices; the master protocol component is responsible for communicating with the WuKong master, which will be discussed later, so as to provide device properties and download applications from the WuKong master. Native profiles refer to the service adapter (or the device driver) of hardware components such as sensors and actuators. Last, Darjeeling JVM is a Java virtual machine to execute Java applications. Darjeeling VM supports limited Java APIs for embedded devices and is a stack-based VM. Different from traditional JVM, the Darjeeling VM uses the AOT (Ahead-of-Time) compiler rather than the JIT, to reduce the memory and storage usage requirements. The memory footprint of DarjeelingVM is less than 80 kB and

the optimized Java executable code is only 86% slower than the optimized native C executable code (Reijers and Shih, 2017). (Other Java JIT compilers are from 30 to 200 times slower compared with the optimized C implementation.)

WuKong development environment. The WuKong development environment is a graphical programming environment for flow-based programming. In the WuKong development environment, users select an appropriate pre-defined service class, called *WuClass*, to compose their applications. One WuClass can represent primitive sensing services such as temperature sensing and motion sensing, primitive actuation services such as buzz and display, or programmable decision services using Python, Java, or C.

Figure 5 shows the flow-based development environment in WuKong. In the environment, developers drag and drop predefined service components, named *WuClass* in WuKong, and data links to FBP programming canvas presented on the right. The example shows a smoke detector and evacuation sign application, which detects a smoke event using 'Smoke Sensing Services' shown on the left and displays an evacuation route using 'Display Services' shown on the right. The service in the center represents a fire agent to intelligently find a safe evacuation route. Each WuClass has predefined properties, which can be read-only, write-only, or read/write. In this example, the alarm property in the Smoke Sensing Service WuClass is a read-only property; the content property in Display Service WuClass is a write-only property. The directed lines between WuClasses represent directed data flows from one WuClass to another one.

Applications developed in the WuKong FBP environment only define the services and logical flows for the application. The service class can specify the minimum requirements for hardware devices to conduct the service. The application shown in Fig. 5

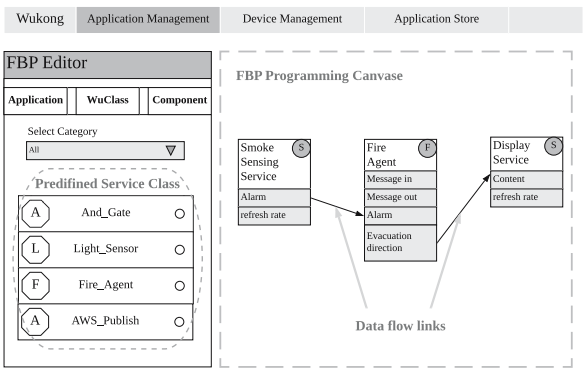


Fig. 5. Example application in the WuKong FBP environment.

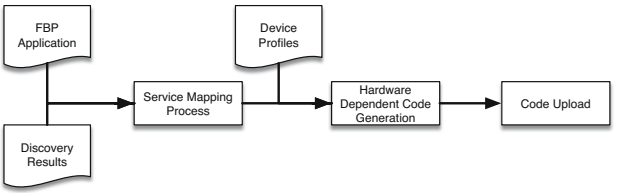


Fig. 6. Flow for service mapping and application deployment on the WuKong master.

can be deployed to one edge device or multiple devices connected by computer networks or wires. We discuss the deployment process in the next subsection.

Deployment-time service mapping. In the WuKong middleware, WuMaster is responsible for managing the services and devices in an area, similarly to a wireless access point for a wireless network. When a new device starts, it looks for WuMaster in the network and registers itself to WuMaster. WuMaster then starts the discovery process to collect hardware properties from the device. These properties will be stored on WuMaster for service management.

The WuKong middleware explicitly separates the deployment phase from the development phase. Figure 6 shows the process of mapping an FBP application to hardware devices and that of deploying applications to the devices. When the application is ready to be deployed, it is downloaded to WuMaster from an application store on the cloud. The first step for application deployment is to map logical services, i.e., WuClass, to physical devices. The service installer or developers can choose different mapping algorithms to map WuClass in FBP applications to meet different QoS requirements. For example, one may ask to use a minimal number of devices; another may ask to minimize the network traffic in the system. WuMaster will map services to physical devices based on the selected mapping policies. The discovery results are used to search for capable hardware devices to conduct WuClass. Moreover, the mapping service also creates messaging links from the sending device to the receiving one for each data link defined in the FBP application.

The second step generates executable code for the devices. (Many of the edge devices are not able to generate executable code from the source codes of high-level programming languages.) Each IoT device may have different physical sensing and actuation devices, and supports different software-enabled services. These capabilities are specified in device profiles. WuMaster generates the code based on the device profiles. The last step is to upload the code to the devices using computer networks. WuMaster communicates with the master protocol component (cf. Fig. 4) to upload the code. The uploaded code will be executed on top of the WuKong

run-time environment. Note that WuClass implemented in Python can only be uploaded to microprocessor-based devices.

The WuKong framework provides the development framework to integrate a number of sensing and computation services into a coherent IoT service. Specifically, it simplifies the development process by reusing pre-designed services and integrating the services by a data-flow model. Consequently, developers do not need to duplicate the pre-designed service onto the application. The pre-designed service has only one version of the code and can be deployed in a number of IoT devices. The WuKong framework also provides location-aware services so as to configure the services according to their location.

3. Location and temporal-aware sensing services

Gait velocity and activity recognition require the system to monitor the subject within a spacious area for a certain period of time. When the monitored area is not well controlled, there will be lots of noise in the environment, which leads to errors in the collected data and, hence, may reduce the accuracy and usability of the system. This section presents observations on the causes of the errors, and the proposed method to eliminate them.

3.1. Observations on the cause of sensing errors. In order to study the cause of the errors and impact on the accuracy, we conduct two sets of preliminary studies. Figures 7 and 8 show the settings of the experiments and their results on the errors caused by cone-shape sensing signals. The first experiment evaluates the errors caused by walking toward lines of sensors. The figure shows the detected walking distance relative to the edge of sensor S_1 and the path of the ground truth. The errors were caused by the height of the person and the cone-shape of sensing signals. Hence, when a person walks along a straight line at a constant speed, the measured speed will not be constant.

The most significant errors occur at the edge of the sensing regions, which are presented at the start and the end of the heat sources detected by each sensor. Because there are only 8 data points on both the horizontal and vertical axes, the walking path detected by the sensors is not linear and is a step function. Moreover, the length of the horizontal lines at the start and the end of the path is much longer than other segments. The prolonged segments are the results of cone-shaped signals. Thus, the location remains the same and the moving distance will be underestimated, and so is gait velocity. Fortunately, the results also show that overlapping the sensing regions may allow the system to eliminate the sensing errors on the edge.

The second experiment is designed to understand the error on location estimation on overlapping regions. In these experiment, the person walks on a line perpendicular to the line of the sensors. In this experiments, we set up three sensors: S_1, S_2, and S_3. Sensor S_1, on the left of sensor S_2, has 24 (3×8) data points overlapping with those of Sensor S_2, shown as the grey area in the figure. The person walks along three different paths, which are 15 cm apart and are represented as P_1, P_2, and P_3. Figure 8(b) shows the estimated walking paths by Sensor S_1 and S_2. The results show that sensor S_2 returns the correct walking path, but not Sensor S_1. Sensor S_1 is misled to believe that paths P_1 and P_2 are identical, which is not correct.

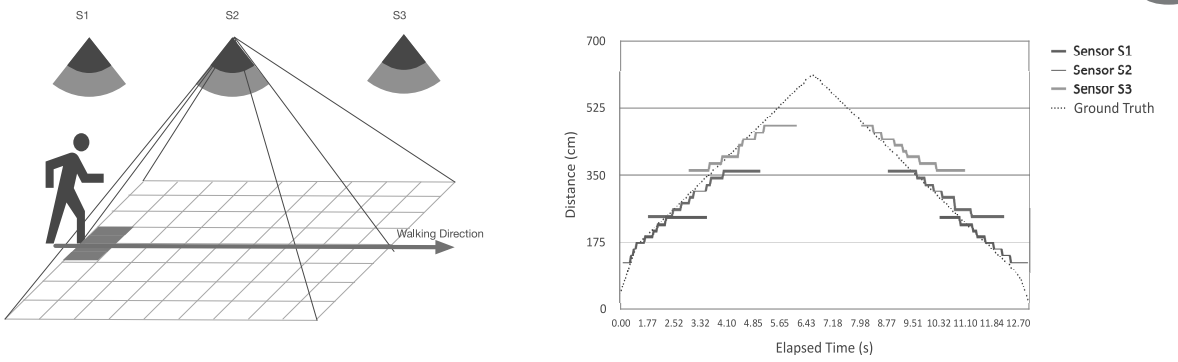
The above two experiments show that the error caused by cone-shape sensing signals can be eliminated by overlapping the sensing regions, as shown in Fig. 7(b). However, we have to carefully design the algorithm to fuse the sensing data on overlapping sensing regions.

3.2. Collaborative sensing between neighboring services. To remove the aforementioned errors caused by sensing signals and the deployment of sensors, the sensing devices in the system have to exchange the sensed data with their neighbors and calibrate themselves from time to time.

The first part of the proposed method is to take into account the location of the person and his/her height to eliminate the errors. Figure 7(b) shows that overlapping the sensing regions can eliminate the distortion on the edge of sensing regions; however, at the same time, Fig. 8(b) shows that it is not trivial to fuse the sensing data in the overlapping area.

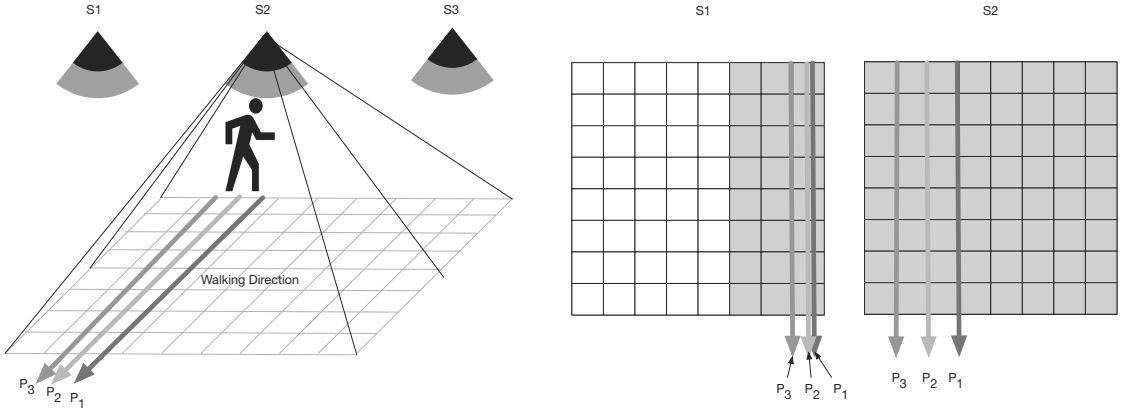
There are a number of methods to fuse the peak heat sources detected by different sensors. For the sake of simplicity, we assume that the real-time clock of sensing devices is well synchronized. (We will discuss how to synchronize the devices later.) The first step is to convert the locations sensed by different sensors into common coordinates. This step requires calibrating the sensed location onto a true coordinate, which is critical to the data points on the edge of the sensing regions.

The second step is to fuse the peak of heat sources from multiple sensors into one peak of heat sources. After the first step, the peak of heat sources from multiple sensors is located in the same coordinate system. The most intuitive way is to compute the mean of nearby peaks as the new peak of the heat source. This method is simple but ignores the fact that the sensing signals are of cone shape. The peak detected by different sensors represents different parts of the human body, as shown in Fig. 3. A better way is to use the weighted average based on the number of data points which are greater than the detection threshold. (We will discuss how to filter the data points using the threshold in a temporal



(a) (b)

Fig. 7. Errors and bias of parallel sensor deployment: walk in the direction parallel to the sensor deployment (a), location of peak values (b).



(a) (b)

Fig. 8. Errors at overlapping sensing regions: walk in the direction in perpendicular to the sensor deployment (a), location of beak values (b).

domain later.) The weight of data points covered by Sensor S_i, denoted by W_{S_i}, represents the number of pixels whose z-score is greater than the threshold in the region of Sensor S_x. The peak of data points covered by Sensor S_i, denoted by Z_{S_i} represents the position of the peak in the region of Sensor S_i. The new location of the peak in the overlapping regions of Sensor S_i, S_{i+1}, ..., denoted by Z_i, is computed by the weighted average of the peak covered by Sensor S_i, S_{i+1}, ... In other words,

$$Z_i = \frac{\sum_i Z_{S_i} \times W_{S_i}}{\sum_i W_{S_i}}. \tag{1}$$

Z-score filtering on temporal domain. To identify the heat source correctly, the developed approach consists of two methods. The first part is to remove the noise among the sensed data. To distinguish a spontaneous temperature increase from a stable temperature increase, Z-scores are computed for each data point, and only the z-scores of these data points that meet certain requirements, which will be discussed later, are selected as heat sources in the sensed space. The second part of the approach is designed to eliminate the impact of the ambient temperature caused by air condition, weather, and so on. Since z-score depend on the average readings and standard deviations, the impact can be eliminated by calibrating the average readings from time to time. The details of these two methods can be found in our earlier work (Shih *et al.*, 2017).

Walking path detection. After obtaining pixel of the peak, we need to transfer it to a real position. The horizontal and vertical angle of view of a sensor is 70°. We assume that the angles of view of every pixel are the same, and we know the distance between the object and the sensor. Figure 9 shows that the length of every pixels is different. The pixel closer to the edge of the sensing area has a wider sensing range. We can use the distance D and the angle between the sensor and the objects and

the vertical line of sensor θ to calculate the position of object P,

$$P = D\tan(\theta). \tag{2}$$

3.3. Location-aware data exchange in the WuKong middleware. The proposed method discussed above requires the sensing devices to exchange data with their neighboring devices. In the deployment scenario shown in Fig. 1, it will be labor-intensive and error-prone to program each device to only communicate with its neighboring devices. Moreover, it will be much difficult to maintain the services when either repair or replacement is needed.

WuKong middleware supports the flow-based program development environment and the pub/sub messaging exchange model. When the location of each device is provided, the messaging mechanism in the WuKong middleware will allow each device to communicate with its neighboring devices. Figure 10 shows the application to identify the person in the space and provide his/her height to calibrate the estimation.

Figure 10 shows the application for gait velocity measurement. In this application, WuClass `GridEye` is the service to sense thermal signals and process the data. While processing the sensed signals, it also receives signals from other sensors by subscribing to the data in the neighborhood, shown as the `AWSSub` service. At the same time, WuClass `GridEye` sends its estimates of heat sources to WuClass `GaitVelocityServer` to compute gait velocity, and publishes its sensed data to other devices in the neighborhood, shown as the `AWSPub` service. WuClass `GridEye` has a `UserHeight` property, which is optional. This property will allow the service to take into account the height of the person to better estimate the location. It can be provided via manual inputs, configurations, and run-time detection by other services.

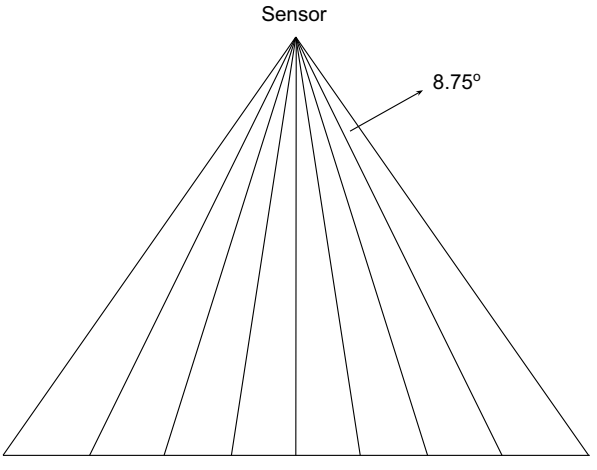


Fig. 9. Position of each pixel.

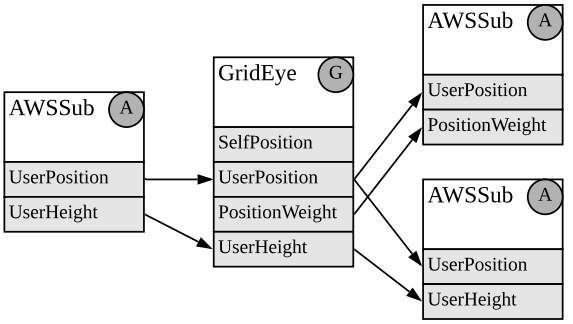


Fig. 10. FBP for gait velocity measurement.

3.4. Data compression/reduction. If we store a frame from Grid-Eye in a readable format, it will need 380 bytes and generate 328 MB data per day when it samples at 10 Hz. However, the indoor temperature environment ranges only from 5° C to 40° C and the resolution of Grid-Eye is 0.25° C. Hence, we can use one byte to present the temperature reading for each pixel without losing the accuracy and resolution. Consequently, we only need 64 bytes to store a frame, and the amount of sensing data will reduce to 55 MB per day. This approach is effective and straightforward, but limited to indoor temperature. Many compression and reduction ones, including lossless and lossy algorithms, have been proposed for general purposes and are listed below.

Huffman coding. Huffman coding is a lossless data compression coding method. The method analyzes input raw data, computes the frequency of the symbol in the raw data, and encodes the symbol using variable length bit strings. The rationale is to use short bit string to present high frequency symbols and long bit strings to present low frequency symbols so as shorten the raw data without ambiguity and loss.

When Huffman coding is used to encode the thermal image sampled by Grid-Eye, it can reduce the frame size from 64 bytes to 40.7 bytes, with a 6 byte standard deviation. When the Huffman coding is used to encode the thermal image sampled by FLIR ONE PRO, it can reduce the frame size from 307 kB to 257 kB, with a 8.1 kB standard deviation. In order to preserve all the data, the compression ratio of the Huffman coding ranges from 1 to 3 for most input data. In the meantime, its time complexity is $O(n \log n)$, where n is the size of the input data.

Z-score threshold. When it is acceptable to lose insignificant data, many compression algorithms for audio and image data are developed. To measure gait velocity of the elderly, only the movement of the heat sources should be stored. Hence, it is not required to keep all the details of the thermal images.

We can only send the locations of heat sources since the thermal sensors are mostly used to detect heat source. The z-score is defined as $z = (\chi - \mu)/\sigma$, where χ is the value of the temperature, μ is the average of the temperature in the covered area, and σ is the standard deviation of the temperature. The z-score provides a dynamic threshold to detect heat sources because the average temperatures change from one frame to another, and are not sensitive to changes in ambient temperature. In our earlier work (Shih *et al.*, 2017), we used the z- score instead of a static threshold to detect humans because the background temperature may have a 10° C difference between day and night, and when people walk through the sensing area covered by Grid-Eye, the temperature reading will only increase from 2° C to 3° C. Hence, it is impossible to use a static threshold to detect a human.

In the work of Shih *et al.* (2017), the pixels with significant data points only if the z-score is higher than two. Hence, we can reduce the frame size by dropping all pixels with low z-scores. The file size can be reduced from 64 B to 12.6 B with a 2.9 B standard deviation by the z-score threshold 2. Its compression ratio is greater than five.

Gaussian function fitting. Since the temperature readings of a human body in thermal data from Grid-Eye look like a signal cone, we may use a Gaussian function to fit the thermal data. A Gaussian function $y = Ae^{-(x-B)^2/2C^2}$ has three parameters: A, B and C. Parameter A refers to the height of the cone, B refers to the position of the cone's peak, and C refers to the width of the cone. We use the pixel with the highest temperature as the peak of the cone, so we only need to adjust A and C to fit the thermal data. Guo (2011) provides a fast way to fit a Gaussian function. In our testing, it will have a 0.5° C root-mean-square error on the average, and it only needs five bytes to store the position of the peak and two parameters. Its compression ratio is greater than ten.

3.5. Parameterized data reduction algorithm.

We propose a parameterized data reduction algorithm to eliminate insignificant data collected by thermal sensors. The proposed algorithm can be applied to meet the requirements of a file size or minimal sampling errors.

The input data are the thermal images, represented by 8×8 thermal arrays. The number of heat sources in each image is limited, usually less than 3, in the targeted applications. The goal is to reduce the size of thermal images subject to the mobility and shape of the human subjects. The reduction process will be conducted on IoT platforms with limited computation, storage and network capabilities. The first metric to evaluate the compressed file is the root-mean-square errors (RMSEs) between the input and output data. The second metric is the size of the compressed file.

Observations on data reduction. The observation shows that the nearby pixels usually have similar values, except at the edge of objects. Hence, we can divide an image into several regions, and the pixels in the same region have a similar value, so we can use the average value to represent them and this will not lead to too many errors. However, precisely dividing an image into some polygon regions needs a lot of computation power, and it is difficult to describe the edge of each region. Also, determining the number of regions is a challenge. Hence, to effectively describe regions, we assume that every region most be a rectangle, and every region can be divided into four regions by cutting in the middle along the horizontal and vertical lines . The image will start with only one region, and three regions will be added per round since we divide the region into four pieces.

Data structure and the region selection algorithm. To determine which region is to be divided, we assign every region a score, and put them into a heap. For each iteration, the algorithm picks the region with the highest score, divides it into four subregions, calculates the score of subregions, and puts them onto the heap. We use the sum of square errors of pixels in the region R as the score of this region:

$$\mu = E(R),$$
$$Score = \sum_{X \in R} (X - \mu)^2 = \sum_{X \in R} X^2 - |R|\mu^2.$$

By the equation above, we need the sum of squared errors and the sum of all data points in the region to calculate the score of the region. We use a four-dimensional segment tree as a container to store all possible regions and their scores. Since the segment tree is complete, the size of the tree is less than the doubled number of pixels. For each node of the segment tree, it records the range on both width and height it covered, sum $\sum_{X \in R} X$, and sum of square $\sum_{X \in R} X^2$ of pixels in the region. The root of the segment tree starts in node number 0, and each node i has four children from node numbers from $i \times 4 + 1$ to $i \times 4 + 4$. Hence, we only need to allocate a large array and recursively process from the root without the cost of generating the links of nodes. The algorithm of the *tree segmentation pre-process*, listed in Algorithm 1, shows how we generate the tree and calculate the sum of squared errors and the sum of all nodes.

Algorithm 1. Tree segmentation pre-process algorithm.

INPUT: Sensing data *Frame* and its width and height.
OUTPUT: Segment tree with squared errors of all regions.

1: Tree = new Array(max^2(Frame.Width, Frame.Height)$\times$ 2)
2: Function{setTreeNode}{x, left, right, top, bottom}
3: **if** left = right top = bottom **then**
4: Tree[x].Sum = Frame[left][top]
5: Tree[x].SquareSum = Frame[left][top]^2
6: **else**
7: setTreeNode($4x + 1$, left, (left+right)$/2$, top, (top+bottom)/2)
8: setTreeNode($4x + 2$, (left+right)$/2$, right, top, (top+bottom)/2)
9: setTreeNode($4x + 3$, left, (left+right)$/2$, (top+bottom)$/2$, bottom)
10: setTreeNode($4x + 4$, (left+right)$/2$, right, (top+bottom)/2, bottom)
11: Tree[x].Sum = $\sum_{i=4x+1}^{4x+4}$ Tree[i].sum
12: Tree[x].Average = $\frac{\text{Tree[x].Sum}}{\text{(right-left+1)}\times\text{(bottom-top+1)}}$
13: Tree[x].SquareSum = $\sum_{i=4x+1}^{4x+4}$ Tree[i].SquareSum
14: **end if**
15: Tree[x].SquaredError = Tree[x].SquareSum $-$ $\frac{\text{Tree[x].Sum}^2}{\text{(right-left+1)}\times\text{(bottom-top+1)}}$
16: EndFunction
17: setTreeNode(0, 0, Frame.Width, 0, Frame.Height)

In order to select the region properly, we use a priority queue to retrieve the region with the highest score. The priority queue is made by the heap, and starts with only the root of the segment tree. For each round, the priority queue pops the item with the highest score and pushes all its child in to the queue.

The size of compressed data depends on the numbers of iterations of region division. The compressed data size will be one plus the number of iterations times four bytes. The algorithm of data-driven data compression, listed in Algorithm 2, shows how we divide regions until a specified data size.

The error rate of the compressed data is the sum of the squared errors of regions in the priority queue. We can update the RMSE for each iteration by computing the difference of the squared error between the divided region and its sub-regions. The algorithm *error rate driven data compression*, listed in Algorithm 3, shows how we divide regions until specified RMSE requirements are met.

After the region dividing completes, we will generate the data string to be sent. The string is generated by

Algorithm 2. Data size driven data compression algorithm.

INPUT: Pre-processed segment tree.
OUTPUT: List of separated regions.

1: separatedRegions = new Array(sizeOf(Frame))
2: PriorityQueue = new Heap(sizeOf(Frame))
3: PriorityQueue.Push(Tree[0].SquaredError, 0)
4: CurrentDataSize = 1
5: **while** CurrentDataSize $<$ SpecifiedDataSize **do**
6: value, x = PriorityQueue.Pop()
7: separatedRegions.push(x)
8: **for** $j \leftarrow 1$ to # of subregions **do**
9: PriorityQueue.Push(Tree[$4x + j$].SquaredError, $4x + j$)
10: **end for**
11: CurrentDataSize $+=$ # of subregions
12: **end while**

performing a depth-first search on the segment tree. For each node, if it is in the $separatedRegions$ array, it returns a reminder β and visits its children. Otherwise, it returns the average temperature value and will not visit its children. The algorithm *data string generating*, listed in Algorithm 4, shows how we generate the data string after the region dividing completes. After the data string is generated, it will be compressed by *Huffman coding* as a final step.

The complexity of our algorithm can be divided into three parts. The first is to initialize the segment tree. The size of the segment depends on that of the frame. If the number of pixels in a frame is N, the height of the segment tree is $\mathcal{O}(N \log(N))$, and the number of nodes will be $\mathcal{O}(N)$. The time complexity of initialization is $\mathcal{O}(N)$. The second part is loading the thermal data. It will need to traverse the entire tree, from the leaf to the root. Since the segment tree can be stored in an array, it also takes $\mathcal{O}(N)$ time to load the thermal data. The third part is to divide regions. For each round, we pop an element from the heap and push four elements into a heap. If there are K iterations, the size of the heap will be $3K+1$. The time complexity of each pop and push operation is $\mathcal{O}(\log(K))$. Because there are K pops and $4K$ pushes, the total times will be $\mathcal{O}(K \log(K))$.

4. Performance evaluation

4.1. Location aware service. To evaluate the effectiveness of the proposed method, we repeat the experiments discussed earlier. In the experiments, three sensors are deployed on one straight line. The distance between two adjacent sensors is 2.4 m and the height of the deployed sensors is 2.7 m from the floor. A person walks along the line of the deployed sensors, shown in

Algorithm 3. Error rate driven data compression algorithm.

INPUT: Pre-processed segment tree.
OUTPUT: The list of separated regions.

```
1: separatedRegions = new Array(sizeOf(Frame))
2: PriorityQueue = new Heap(sizeOf(Frame))
3: PriorityQueue.Push(Tree[0].SquaredError, 0)
4: SquaredError = Tree[0].SquaredError
5: while sqrt(SquaredError/FrameSize) > SpecifiedRMSE do
6:    value, x = PriorityQueue.Pop()
7:    separatedRegions.push(x)
8:    SquaredError −= value
9:    for j ← 1 to # of subregions do
10:      PriorityQueue.Push( Tree[4x + j].SquaredError, 4x + j )
11:      SquaredError += Tree[4x + j].SquaredError
12:   end for
13: end while
```

Algorithm 4. Data string generating algorithm.

INPUT: The list of separated regions.
OUTPUT: Compressed data string.

```
1: DataString = new Vector()
2: Function{DfsSegmentTree}{NodeId}
3: if NodeId ∈ separatedRegions then
4:    DataString.append(β)
5:    DfsSegmentTree(4 × NodeId + 1)
6:    DfsSegmentTree(4 × NodeId + 2)
7:    DfsSegmentTree(4 × NodeId + 3)
8:    DfsSegmentTree(4 × NodeId + 4)
9: else
10:   DataString.append(Tree[NodeId].Average)
11: end if
12: EndFunction
13: DfsSegmentTree(0)
```

Fig. 7.

The experiments measure the difference between the fused walk path according to the sensing data and the walk path of the ground truth. Specifically, the difference is measured for overlapping sensing regions and non-overlapping sensing regions. We evaluate two algorithms: *average peak* and *weighted average peak*, to evaluate their accuracy on fusing the walk path. The root-mean-square errors are also computed to compare the accuracy of these two algorithms.

Figure 11 shows the fused walk paths using the average peak and weighted average peak algorithms when there are two non-overlapping sensors, which are S_1 and S_3, shown in Fig. 7. The true walking path is shown as the ground truth for comparison. According to the results, both the algorithms estimate the location of the peak even though there are no overlapping regions between the two sensors. The average peak algorithm estimates the location of the peak based on the discrete results from the non-overlapping sensors and can only give a low resolution estimate. On the other hand, the weighted average peak algorithm takes into account the number of pixels on each sensor to eliminate the errors on the location on the edge and, hence, can estimate the location of the peak at a higher resolution. Thus, the results produced by the weighted average peak algorithm (thin solid line) are much closer to the ground truth walk path (dotted line).

Figure 12 shows the walk path of the ground truth and fused walk path using average peak and weighted average peak algorithms when there are three sensors, which are S_1, S_2, and S_3, and the distance between sensors shortened to 1.2 m, and the sensing regions overlap. The fused walk path by the weighted average peak algorithm is shown as the thick light line and that by the average peak algorithm is shown as the thick dark line. When these two walk paths overlap, only that by the

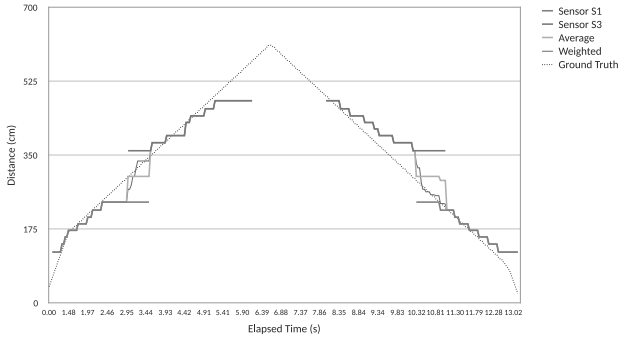


Fig. 11. Location estimated by two sensors and location-aware fusion.

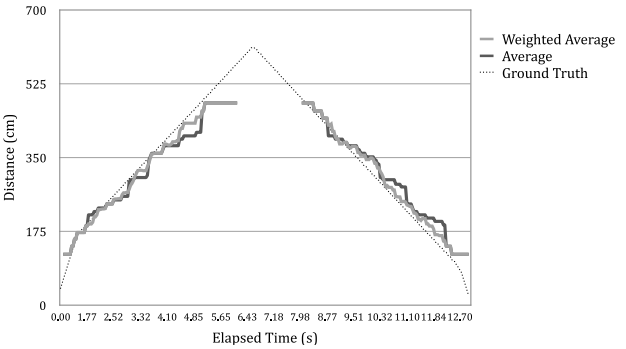


Fig. 12. Location estimated by three sensors and location-aware fusion.

weighted average peak algorithm is shown.

For the sake of presentation, the path plotted by the readings from individual sensors is not shown. Again, the weighted average peak algorithm estimates the walk path with a better accuracy by taking into account the number of data points having higher z-scores.

To quantify the accuracy for these two algorithms, we compute the root-mean-square errors of these two experiments. Figure 13 shows the root-mean-square errors of the path fused by the average peak and weighted average peak algorithms.

The results show that the weighted average peak algorithm causes much fewer errors by taking into account the number data points detected by the neighboring sensors in both experiments. When the sensing regions of the sensors overlap, the weighted average peak algorithm outperforms the average peak algorithm almost twice. When the sensing regions do not overlap, the weighted average peak algorithm also outperforms the average peak one more than twice. The average peak is very sensitive to whether the sensing regions overlap to provide sufficient sensing data points. However, the algorithm can tolerate the missing data points from the non-overlapping regions.

We can also use the same model to detect the walking path of a person with a different height. Figure 14 shows the detection result of a shorter user. The slope of data is lower and the tail of the curve is shorter since the distance between the user and the sensor is closer to the limited detection range of sensor, and the errors caused by the persons' height are less frequent. Figure 15 shows the walking path of the ground truth and the fused walk path of a 150 cm person by the weighted and unweighted algorithms. The segments with a difference between these two lines are on the edge of the sensing area of $S2$ where the unweighted algorithm cannot recognize the reliability of each sensor. Figure 16 shows that the line of the weighted algorithm follows the ground truth more smoothly. Figure 17 shows the root-mean-square errors of the paths overlapped by more than one sensor.

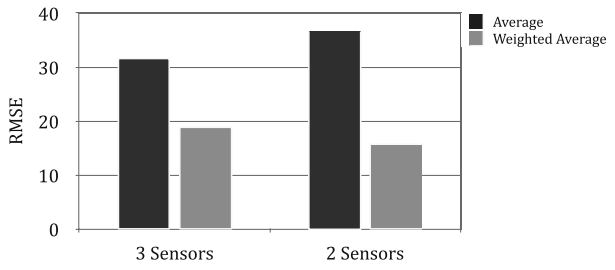


Fig. 13. Root-mean-square errors of the average and weighted average algorithms.

4.2. Parameterized data reduction. To evaluate the effectiveness of the parameterized data reduction algorithm, we measure two performance metrics, the RMSE and file size, of the compressed thermal images. These two metrics are also measured when a JPEG compression algorithm is used. It is the most popular lossy image compression algorithm. The output of the thermal sensor is a numerical matrix, but it is similar to an RGB image. Both of them are two-dimensional data, and most digital image processing algorithms can also be applied to the data from thermal sensors (e.g., edge detection, affine transformations). Hence, the data from the thermal sensor can be transformed to gray scale images. Thus, image compression algorithms can be applied to compress thermal images. We use `zlib` as the baseline of the compression algorithm. It is a portable and lossless data-compression library.

The JPEG images are generated by `OpenCV` 3.3.0, which uses `libjpeg` version 9, dated on 13-Jan-2013, and has image quality ranging from 1 to 99, where lower image qualities refer to a better compression ratio but a poor RMSE. The `zlib` file is generated by the `Python` built-in library using the slowest compression level.

Since JPEG is a general-purpose image format, it embeds image information in its header, which is the overhead of the compression algorithm. If we generate a JPEG image for each 8×8 pixel frame, the outcome of JPEG will be even worse than that of `zlib` on both the error and compression rate. To have a fair comparison, we combine 16 original 8×8 frames into one 32×32 pixel frame. The size of one frame becomes 1024 bytes.

Figure 18 shows a detailed comparison of the parameterized data reduction algorithm and JPEG. The former leads to lower RMSEs and better compression rates compared with JPEG compression. The size of a single 8×8 frame is 64 bytes. The average size of images compressed by `zlib` on compressing a single frame is about 50.5 bytes and that on compressing 16 frames together is 30 bytes. Hence, the compression ratio of `zlib` for a single frame is 0.8 and that for 16 frames is 0.5.

The file size of JPEG formats range from 22 bytes to 48 bytes. Hence, the compression ratio ranges from 0.34 to 0.75. However, the RMSE becomes more than 0.6° when the compression ratio is 0.34. The parameterized method outperforms the JPEG one in all tolerable RMSEs. The RMSEs of the files compressed by the proposed methods never exceed 0.6°. When the frames are compressed one by one, the compression ratio will be lowered due to the overhead of each frame, shown as `PDR-1` in the figure. When 16 frames are merged into one file and compressed, the compression ratio becomes higher, ranging from 30 bytes to almost zero bytes.

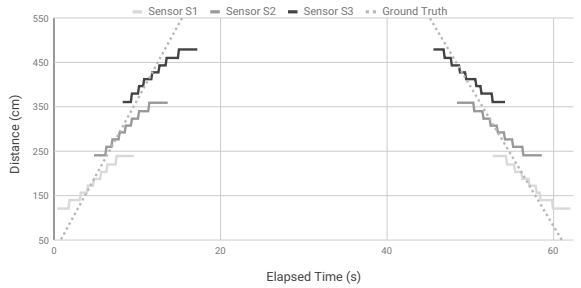


Fig. 14. Location of 150 cm person with a 170 cm model.

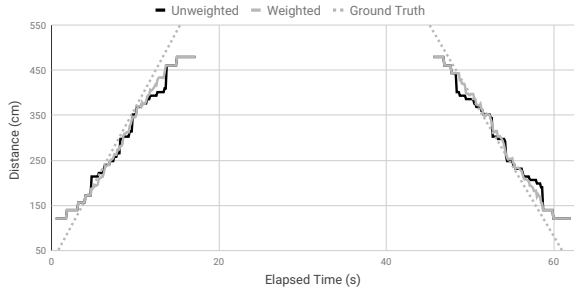


Fig. 15. Location estimated by three sensors of a 150 cm person and location-aware fusion.

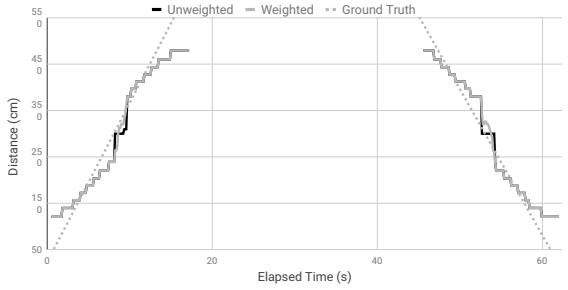


Fig. 16. Location estimated by two sensors of a 150 cm person and location-aware fusion.

5. Conclusion

This paper presented the design, implementation, and evaluation of distributed sensing services using temporal and location properties of sensed data points. The proposed approach was designed to eliminate the errors caused by cone shape sensing signals and the height of the person, which are common for the sensors deployed on the ceiling and to detect human mobility. The proposed approach takes advantage of the location framework in WuKong middleware to use one single application for all the sensing devices in smart homes. It not only measures gait velocity at a high accuracy, but also reduces the development and management overhead for distributed sensing services at smart homes.

Acknowledgment

This research was supported in part by the Ministry of Science and Technology of Taiwan (MOST 106-2633-E-002-001, MOST 106-2627-M-002-022-), National Taiwan University (NTU-107L104039), Intel Corporation, Delta Electronics, and Advantech.

References

Administration on Aging (2015). *A Profile of Older Americans: 2015*, US Department of Health and Human Services, `https://books.google.com.tw/books?id=B4hEnQAACAAJ`.

Alemdar, H. and Ersoy, C. (2010). Wireless sensor networks for healthcare: A survey, *Computer Networks* **54**(15): 2688–2710.

Bourke, A.K., Ihlen, E.A., Van de Ven, P., Nelson, J. and Helbostad, J.L. (2016). Video analysis validation of a real-time physical activity detection algorithm based on a single waist mounted tri-axial accelerometer sensor, *IEEE 38th Annual International Conference of the Engineering in Medicine and Biology Society (EMBC), Orlando, FL, USA,* pp. 4881–4884.

Fortino, G., Galzarano, S., Gravina, R. and Li, W. (2015). A framework for collaborative computing and multi-sensor data fusion in body sensor networks, *Information Fusion* **22**: 50–70.

Gheid, Z. and Challal, Y. (2016). Novel efficient and privacy-preserving protocols for sensor-based human activity recognition, *13th International Conference on Ubiquitous Intelligence and Computing (UIC 2016), Toulouse, France*.

Gravina, R., Ma, C., Pace, P., Aloi, G., Russo, W., Li, W. and Fortino, G. (2017). Cloud-based activity-aaservice cyber-physical framework for human activity monitoring in mobility, *Future Generation Computer Systems* **75**: 158–171.

Guo, H. (2011). A simple algorithm for fitting a Gaussian function [DSP tips and tricks], *IEEE Signal Processing Magazine* **28**(5): 134–137.

Hsu, C.-Y., Liu, Y., Kabelac, Z., Hristov, R., Katabi, D. and Liu, C. (2017). Extracting gait velocity and stride length from surrounding radio signals, *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI'17, Denver, CO, USA*, pp. 2116–2126, DOI: 10.1145/3025453.3025937.

Khalajmehrabadi, A., Gatsis, N., Pack, D. and Akopian, D. (2016). A joint indoor WLAN localization and outlier detection scheme using lasso and elastic-net optimization techniques, *IEEE Transactions on Mobile Computing* **PP**(99): 1–1.

Kon, S.S.-C., Jones, S.E., Schofield, S.J., Banya, W., Dickson, M.J., Canavan, J.L., Nolan, C.M., Haselden, B.M., Polkey, M.I., Cullinan, P. and Man, W.D.-C. (2015). Gait speed and readmission following hospitalisation for acute exacerbations of COPD: A prospective study, *Thorax*

70(12): 1131–1137, http://thorax.bmj.com/content/early/2015/08/17/thoraxjnl-2015-207046.

Kutner, N.G., Zhang, R., Huang, Y. and Painter, P. (2015). Gait speed and mortality, hospitalization, and functional status change among hemodialysis patients: A US renal data system special study, *American Journal of Kidney Diseases* **66**(2): 297–304.

Lee, Y.-D. and Chung, W.-Y. (2009). Wireless sensor network based wearable smart shirt for ubiquitous health and activity monitoring, *Sensors and Actuators B: Chemical* **140**(2): 390–395.

Lu, C.H. and Fu, L.C. (2009). Robust location-aware activity recognition using wireless sensor network in an attentive home, *IEEE Transactions on Automation Science and Engineering* **6**(4): 598–609.

Middleton, A., Fritz, S.L. and Lusardi, M. (2015). Walking speed: The functional vital sign, *Journal of Aging and Physical Activity* **23**(2): 314–322.

Milenković, A., Otto, C. and Jovanov, E. (2006). Wireless sensor networks for personal health monitoring: Issues and an implementation, *Computer Communications* **29**(13–14): 2521–2533.

Peters, D.M., Middleton, A., Donley, J.W., Blanck, E.L. and Fritz, S.L. (2014). Concurrent validity of walking speed values calculated via the gaitrite electronic walkway and 3 meter walk test in the chronic stroke population, *Physiotherapy Theory and Practice* **30**(3): 183–188, DOI: 10.3109/09593985.2013.845805.

Pulignano, G., Del Sindaco, D., Di Lenarda, A., Alunni, G., Senni, M., Tarantini, L., Cioffi, G., Tinti, M., Barbati, G., Minardi, G. and Uguccioni, M. (2016). Incremental value of gait speed in predicting prognosis of older adults with heart failure: Insights from the IMAGE-HF study, *JACC Heart Failure* **4**(4): 289–298.

Reijers, N., Lin, K.-J., Wang, Y.-C., Shih, C.-S. and Hsu, J.Y. (2013). Design of an intelligent middleware for flexible sensor configuration in M2M systems, *Proceedings of the 2nd International Conference on Sensor Networks (SENSORNETS), Barcelona, Spain*, pp. 1–6.

Reijers, N. and Shih, C.-S. (2017). Ahead-of-time compilation of stack-based JVM bytecode on resource-constrained devices, *Proceedings of 2017 International Conference on Embedded Wireless Systems and Networks (EWSN), Uppsala, Sweden*, pp. 1–12.

Shih, C.-S. (2016). WuKong Release Document 0.4, https://www.gitbook.com/book/wukongsun/wukong-release-0-4/details.

Shih, C.-S., Chou, J.-J., Chuang, C.-C., Wang, T.-Y., Chuang, Z.-Y., Lin, K.-J., Wang, W.-D. and Huang, K.-C. (2017). Collaborative sensing for privacy preserving gait tracking using IoT middleware, *2017 International Conference on Research in Adaptive and Convergent Systems (RACS 2017), Krakow, Poland*, pp. 152–159.

Studenski, S., Perera, S., Patel, K., Rosano, C., Faulkner, K., Inzitari, M., Brach, J., Chandler, J., Cawthon, P., Connor, E.B., Nevitt, M., Visser, M., Kritchevsky, S., Badinelli, S., Harris, T., Newman, A.B., Cauley, J., Ferrucci, L. and Guralnik, J. (2011). Gait speed and survival in older adults, *Journal of the American Medical Association* **305**(1): 50–58.

Studenski, S., Perera, S., Wallace, D., Chandler, J.M., Duncan, P.W., Rooney, E., Fox, M. and Guralnik, J.M. (2003). Physical performance measures in the clinical setting, *Journal of the American Geriatrics Society* **51**(3): 314–322, DOI: 10.1046/j.1532-5415.2003.51104.x.

USRP N210 (2019). Software Defined Radio (SDR), Ettus Research, https://www.ettus.com/product/details/UN210-KIT.

WuKong (2012). Software repository, https://github.com/wukong-m2m/wukong-darjeeling.

Yi, X., Willemson, J. and Nait-Abdesselam, F. (2013). Privacy-preserving wireless medical sensor network, *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Melbourne, Australia*, pp. 118–125.

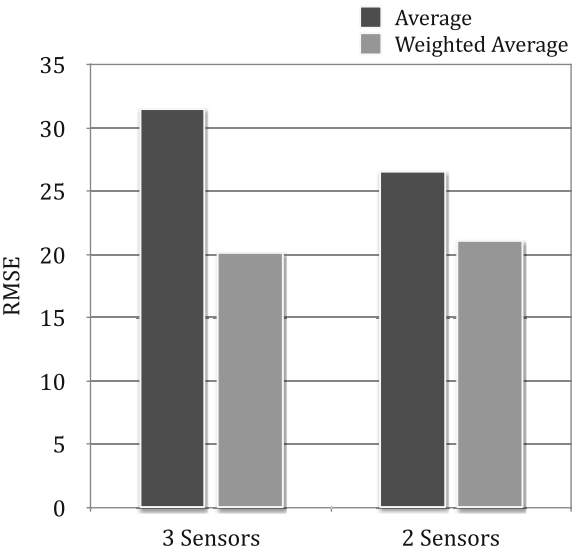


Fig. 17. Root mean square errors of average and weighted average algorithms on a 150 cm person.

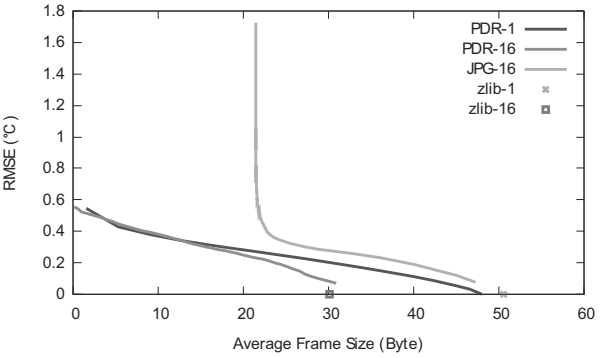


Fig. 18. Comparison of error rates and the compression ratio.

Zhao, M., Adib, F. and Katabi, D. (2016). Emotion recognition using wireless signals, *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, MobiCom'16, New York, NY, USA*, pp. 95–108, DOI: 10.1145/2973750.2973762.

Chi-Sheng Shih received his BSc in engineering science and his MSc in computer science from National Cheng Kung University in 1993 and 1995, respectively. He joined National Taiwan University in 2004 and has been a professor since 2018. His main research interests include embedded systems, hardware/software codesign, real-time systems, and database systems. Specifically, his main research interests focus on real-time operating systems, real-time scheduling theory, embedded software, and software/hardware co-design for system-on-a-chip. His research results have won several awards. He has also served on steering committees of several international conferences and on editorial boards of international journals.

Jyun-Jhe Chou received his Bachelor's degree in computer science and information engineering at National Taiwan University in 2014, and is now a PhD candidate. His research interests include the Internet of things, data quality, and cyber-physical systems.

Wei-Dean Wang received his MD degree from Taipei Medical University, Taiwan, in 1982 and his PhD degree in medical education from National Taiwan Normal University in 2003. He completed the residency training in family medicine at the Kansas University Medical Center in 1989, and has been the Diplomate of Family Medicine of the American Board of Family Medicine since 1989. His research interests are mainly in medical education, family medicine, elderly care, and international medicine. His current research covers various issues regarding medical and home care for dwelling elderly at home.

Kuo-Chin Huang has been the superintendent of the National Taiwan University Hospital, Bei-Hu Branch, Taipei, Taiwan, since 2015 and a professor with the Department of Family Medicine, College of Medicine, National Taiwan University, Taipei, since 2009. He holds MD and PhD degrees from National Taiwan University, and had conducted his post doc at the University of Sydney.

Received: 1 July 2018
Revised: 11 March 2019
Accepted: 21 March 2019