

RESEARCH ARTICLE

Efficient Sensor Processing Technique Using Kalman Filter-Based Velocity Prediction in Large-Scale Vehicle IoT Application

JISU KWON¹ AND DAEJIN PARK^{1,2}, (Member, IEEE)¹School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, South Korea²School of Electronics Engineering, Kyungpook National University, Daegu 41566, South Korea

Corresponding author: Daejin Park (boltanut@knu.ac.kr)


This work was supported in part by the BK21 FOUR Project through the Ministry of Education, South Korea, under Grant 4199990113966, 10%; in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2018R1A6A1A03025109, 10% and Grant NRF-2022R111A3069260, 10%; in part by the Ministry of Science and ICT (MSIT) under Grant 2020M3H2A1078119; and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean Government (MSIT), Metamorphic approach of unstructured validation/verification for analyzing binary code, 40%, under Grant 2021-0-00944; the OpenAPI-based hardware/software (hw/sw) platform for edge devices and cloud server, integrated with the on-demand code streaming engine powered by Artificial Intelligence (AI), 20%, under Grant 2022-0-00816, and the Processing-in-Memory (PIM) Semiconductor Design Research Center, 10%, under Grant 2022-0-01170.

ABSTRACT Sensor nodes that operate as edge devices in Internet-of-Things (IoT) networks have various limitations, such as insufficient power supply and small memory size. Therefore, the sensor node must be able to use resources efficiently to achieve the specified software behavior of the target application. The application in this study involves an IoT network in which the sensor node requests the position from a moving vehicle and estimates the velocity by using a Kalman filter. Using the same sensing cycle for all vehicles improves accuracy regardless of the predicted velocity of the sensor node but increases unnecessary computations. The proposed technique defines distance weight W_D which controls weight policy of the vehicle's speed change. The distance weight W_D determines a communication cycle between the sensor node and vehicle, therefore this approach enables the sensor node to adaptively determine the data request period based on the state of the vehicle. When a slow-moving vehicle intermittently communicates with a sensor node, the time required for the computation performed by the sensor node can be significantly reduced. To evaluate the proposed technique, we experimented with a traffic simulator that was implemented in MATLAB. Compared with the increment in the root mean square error of the reference velocity that sensed the position at every time step, the decrement in the processing time of the sensor node was considerable. Experiments with four manually determined distance weights and a number of spawned vehicles showed that the sensor node processing time was reduced by up to 72.91%.

INDEX TERMS Embedded system, Kalman filter, sensor node, traffic simulation, vehicle tracking.

I. INTRODUCTION

In recent years, with the rapid expansion of the Internet-of-Things (IoT), the operation of the edge device end of networks, which directly interact with the surrounding environment, has attracted attention [1], [2]. In an IoT network, edge devices can be considered as sensor nodes, which are mostly responsible for collecting and processing data from

The associate editor coordinating the review of this manuscript and approving it for publication was Tamas Tettamanti .

the surrounding environment. The purpose of the sensor node is to repeat the sensing operation for the target environment, rather than maintaining versatility and performance to quickly execute various functions. Therefore, the sensor node is designed as a microcontroller-based embedded system and iteratively performs only the static sensing actions programmed in the on-chip flash memory. Because the software (e.g., firmware) is embedded in the microcontroller's on-chip memory, designing software for sensor nodes requires consideration of the available resources and functional aspects.

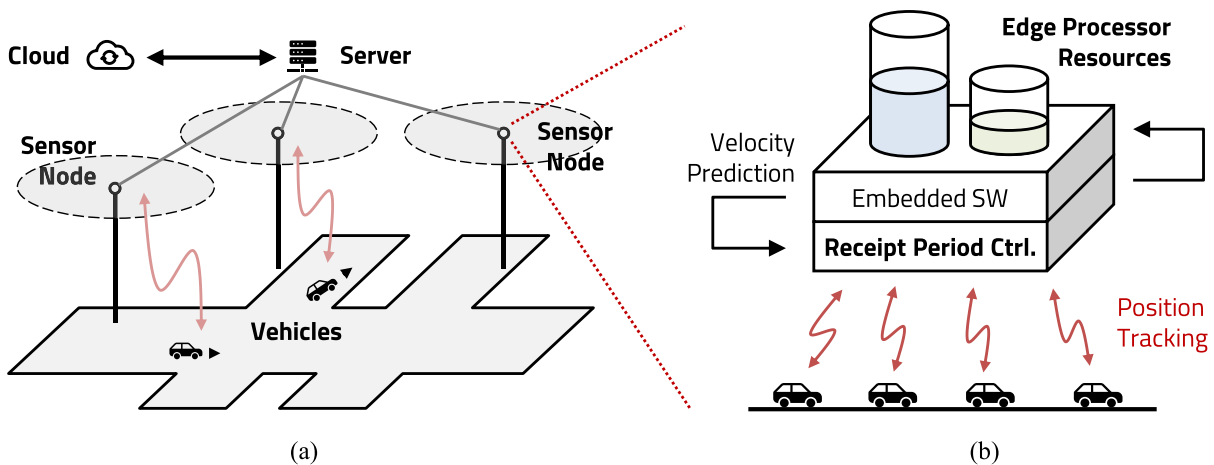


FIGURE 1. (a) Overview of smart-vehicle IoT network including sensor nodes that communicate with vehicles on the road. (b) Sensor node communication between vehicles consuming edge device resources.

To easily recognize the surrounding environment, which is the role of the sensor node, it is necessary to be close to the environment, which is the target of sensing. In other words, the sensor node should be at a physical distance from the server or gateway that gathers the data. Being away from servers with abundant available resources means that sensor nodes must be placed in harsh environments where sufficient resources are not available. The following limitations apply to sensor nodes.

- Low operation speed (clock frequency) [3];
- Small on-chip memory size and chip area [4], [5];
- Battery dependent power leading to insufficient power supply [6], [7];
- Static code in on-chip flash memory that is difficult to update [8], [9].

Starting with the firmware design stage, we can overcome the limitations of these sensor nodes and achieve optimal performance by designing efficient resource-usable software suitable for the application domain. The concept of sensor nodes is applied in diverse applications, such as agriculture, industry, and smart cities. In this study, we reduce the computation time by introducing a software design that considers the embedded characteristics of the sensor node for the efficient use of resources in smart vehicle applications. The role of the sensor node in smart-vehicle applications can be summarized as follows: communicating with the vehicle traveling on the road, collecting and processing data, and transmitting data to the server, as shown in Fig. 1(a). Because the sensor node is often installed at a position where power supply is deficient compared with a vehicle or server, the computation time in the operation process needs to be minimized. Therefore, to efficiently optimize the operations performed on the sensor node, the communication frequency must be adjusted with the vehicle or server to minimize the processing time overhead. This process requires an understanding of both the embedded system characteristics and the target application domain.

In this study, the sensor node communicates with the vehicle on the road to receive the current information on the

vehicle, as shown in Fig. 1(b). Based on this, we propose a technique to adaptively change the reception period of a sensor node based on the predicted vehicle condition. A moving object at a sensor node, such as a vehicle, can be tracked in diverse ways [10], [11]. In microcontroller-based embedded sensor node systems, poor performance can cause a large amount of noise in the measurement. Among several tracking methods, the Kalman filter is used for the vehicle-tracking operation of the sensor node [12], [13], [14], [14], [15]. The role of sensor nodes has expanded not only to sensing and transmission, but also to processing. If sensing value used directly, data can guarantee fidelity and accuracy. However, sensor nodes communicate with gateways or targets to collect data. Therefore, from the perspective of transmission, which is one of the roles of sensor nodes, overhead due to entire raw data transfer can be increased. For overhead reduction, we assumed an application that performs on-device processing of sensing values at sensor nodes. The sensor node receives the coordinate position from the vehicle and predicts the velocity by using a Kalman filter. When the vehicle transmits the position information to the sensor node, the sensor node determines whether to receive based on the predicted vehicle speed and adjusts the communication period. In addition, we assumed that vehicle transmits only position toward sensor node to minimize data traffic. An on-device processing conducted on the sensor node is velocity prediction based on the Kalman filter algorithm. This computation is used to derive the unmeasured variables from the insufficient measurements. The proposed technique uses a distance-based weight of W_D for more sporadic communication when the vehicle velocity is low, and reconfigures the receive cycle according to the time required by the vehicle to travel the distance based on this weight. The proposed technique focused on the operations performed within a single sensor node. We assumed a network environment where a multiple vehicle is connected to a single sensor node. The sensor node receives the state of vehicle from the communication and processes data. When the number of slow vehicles on the road increases,

the computation time on the sensor node is further reduced during infrequent communication periods of the sensor node as the communication cycle readapts in real time depending on the vehicle velocity. In addition, human efforts to update the sensor node firmware and changing environments can be reduced. The vehicle speed predicted with a diminished communication frequency should minimize the difference in the accuracy between the actual and predicted velocities without using the proposed technique.

The traffic simulator [16] in MATLAB is used to evaluate the proposed method implemented in vehicles traveling on lanes and intersections along with signaling systems on a two-dimensional coordinate plane. When a vehicle is generated based on a predefined vehicle spawn policy, it travels along a given road, encounters an intersection, and determines the route as per the signal rules. The direction and speed of the vehicles are set randomly within a predefined range. The sensor node located in the simulator environment receives the coordinate position data from the vehicle spawned on the road and predicts the vehicle speed by using the Kalman filter. The processing time of the sensor node is monitored and compared during the simulations while judiciously changing the communication cycle based on vehicle speed prediction.

The remainder of this paper is organized as follows. Section II presents the research on tracking moving objects and efficient operation of IoT sensors. Section III introduces background of the two-dimensional velocity estimation using the Kalman filter. Section IV describes the details of the proposed adaptive sensor node communication cycle adjustment technique. Section V presents the experiments for sensor node processing time overhead reduction as a case study. Finally, section VI concludes the paper.

II. RELATED WORK

The approach for improving the efficiency of wireless sensor networks is related to the network level and the sensor node processing level. A macroscopic study of wireless sensor networks is the perspective of connectivity and routing between sensor nodes. Authors in [17] and [18] proposed effective sensor routing scheme for large-area sensor networks. These studies focused on connections between sensor nodes rather than sensor node-target interactions. A study on the macroscopic resource consumption in a sensor network layer was conducted [19]. Research [20] focused on the network channel interface formation of a sensor node. They combined and optimized interfaces such as Zigbee, which is suitable for low bit-rate communication [21], to determine the optimal point between the sensor node communication coverage and energy consumption. The aim was to improve network efficiency from the perspective of communication and connection in wireless sensor networks [22], [23], [24]. These studies focused on the connections between sensor nodes and built an efficient wireless sensor network. The network level does not include any domain-specific internal functions designed at the sensor node.

Sensor nodes used as edge devices are primarily designed for microcontroller-based systems with embedded static software, i.e., firmware. Efficient resource consumption and functional updates of edge devices are important in IoT networks [25]. The operation of the sensor node can be divided into sensing, processing, and transmission [26]. In addition, transmission takes the largest proportion in the energy consumption of the sensor node. Authors in [27] reduce power consumption from image-based sensor node to processing and transmission balancing. This approach can be applied to general purpose environments that require image-based sensing.

A large number of methods exist for tracking object using the Kalman filter and Extended Kalman filter. Marchthaler et al. [28] tracked vehicle's position based on Extended Kalman filter. A stochastic covariance filter was used for preventing measurement noise reflected to estimation and increasing accuracy. Perumal et al. [29] performed vehicle and obstacle position estimation using Extended Kalman filter for path-planning of non-linear autonomous vehicle dynamics. Prévost et al. [30] used Extended Kalman filter to state estimation and trajectory prediction of moving unmanned aerial vehicle in 3D space. A dual Extended Kalman filter scheme has been used in [31] for vehicle state and parameter estimation, thus two Kalman filters are running in parallel for splitting the state and parameter estimation problems. Kim et al. [32] employed Extended Kalman filter for vehicle position tracking using radar and lidar, to minimize measurement noise from distance characteristic. Our approach differs from research on the performance improvement of Kalman filter itself in that we focus overall sensor node operation optimization for transmission and processing coupled application.

In this study, we focus on the efficient operation of sensor nodes by the application domain and not on the communication scheme of the sensor network. Because the behavior of edge devices in sensor networks depends on the target application, software design for efficient resource consumption of sensor nodes should also demonstrate the application characteristics. A wireless sensor network consisting of multiple sensor nodes requires consideration for network routing and communication interface layers between each node. Connections between sensor nodes were not included in the scope of the proposed technique because we set boundaries for operations performed within a single node. In addition, various interface layer factors for communication between sensor node and target object (e.g., distance, network latency, bitrate) are not considered at proposed technique. The proposed approach develops a microscopically efficient sensor network by eliminating the connectivity between sensor nodes and reducing the computational effort of a single sensor node by using target application-specific software.

In our previous work, we assumed a scenario in which multiple sensor nodes received coordinate data from a moving target on a random path [33]. The sensor node calculated the coordinate distance and quantized the velocity range to reduce the number of reception cycles from the target vehicle.

Consequently, a reasonable accuracy at the sensor node was maintained while reducing the operation costs of the sensor node. However, the number of sensing target vehicles was limited, and the moving paths and velocities were modeled without a real vehicle. This cannot be regarded as a movement of the vehicle traveling on the road if we assume that the object simply moves randomly over the two-dimensional coordinate plane. Therefore, a simulator with an actual vehicle driving characteristic model should be employed.

III. KALMAN FILTER TWO-DIMENSIONAL VELOCITY PREDICTION

A Kalman filter can predict unmeasured values in a linear system using measured values that include noise. Thus, a method was used to estimate the distribution of current state variables based on previously measured values. The linear system models that can be applied to the Kalman filter are represented based on the following general form of the linear state-space equations:

$$x_{k+1} = Ax_k + w_k, \tag{1}$$

$$z_k = Hx_k + m_k, \tag{2}$$

where x_k denotes the $n \times 1$ state vector; z_k is the $m \times 1$ measurement vector; A is the $n \times n$ system model matrix; H is the $m \times n$ measurement model matrix; w_k is the $n \times 1$ system noise vector; m_k is the $m \times 1$ measurement noise vector; and $k \in \mathbb{R} \geq 0$ is the time instant with m and n representing the sizes of the matrix and the vector, respectively. These are determined by the dimensions of the system model to which the Kalman filter is applied. The fundamental principle of the Kalman filter is to calculate the estimated value based on the measurement and determine the reliability of the predicted value using the error covariance. We assumed that the white noise, w_k , entering the system during the measurement of the Kalman filter and the white noise m_k generated by the sensor itself follow a Gaussian normal distribution with a mean of zero. System matrix A defines the behavior of the system model over time, and output matrix H defines the relationship between the observation state variables.

The Kalman filter operation begins with the selection of the initial values, P_0 and x_0 , of the error covariance, P_k , and the estimated values, x_k . The initial value is required only once for the Kalman filter operation. With the initial values of P_k and x_k selected in the previous step, the new error covariance of the current step, \overline{P}_k , and \overline{x}_k are estimated as follows:

$$\overline{x}_k = Ax^{k-1}, \tag{3}$$

$$\overline{P}_k = AP_{k-1}A^T + Q, \tag{4}$$

where the next time instant is $k > 0$ and Q is the $n \times n$ system noise covariance diagonal matrix. Using the error covariance, P_k , and the estimated \overline{x}_k obtained from (3)–(4), we can determine the Kalman gain, K_k , at the current step, k , as

$$K_k = \overline{P}_k H^T (H \overline{P}_k H^T + R)^{-1}, \tag{5}$$

where R is the $m \times m$ covariance diagonal matrix of the measurement noise m_k . This facilitates the estimation of the

\overline{x}_k and Kalman gain K_k predicted by applying (3)–(4). In addition, to consider value z_k measured in the current system, the measurement model matrix, H , which specifies the relationship between the measurement and estimation, is used. The estimated value, x_k , at current time step k is given by

$$x_k = \overline{x}_k + K_k (z_k - H \overline{x}_k). \tag{6}$$

Finally, the error covariance, P_k , which represents the reliability of the value estimated by using (6), is calculated as follows:

$$P_k = \overline{P}_k - K_k H \overline{P}_k. \tag{7}$$

Thus, whether to use the k th step estimation x_k is determined according to the error covariance P_k calculated in (7). The error covariance and estimations calculated in the k th step are reused in the $k + 1$ th step of the Kalman filter algorithm iteration. By repeating this procedure, the Kalman filter can predict the unmeasured values.

In this study, we apply the Kalman filter to a vehicle velocity prediction application using the position on a two-dimensional road. To apply the Kalman filter, the state of an object at a particular time step must be linearly related to the state of the previous time step. If the position of the object can be determined from the periodic sampling of the sensor node, the distance information that has a linear relationship with the velocity can also be determined. Therefore, the relationship between the position and velocity can be represented by a linear system model. To use the Kalman filter algorithm, we need to derive a system model based on the relationship between position and velocity. Based on the assumption that the road on which the vehicle moves is two-dimensional, state variables x_k , defining position p and velocity v according to the x and y axial directions, are given by

$$x_k = \begin{bmatrix} P_k^x \\ v_k^x \\ P_k^y \\ v_k^y \end{bmatrix}. \tag{8}$$

The position and velocity are considered separately for x and y directions. The period during which the sensor node measurements are entered into the current k time point Kalman filter is denoted as Δt . The next time point $k + 1$ of vehicle position $p_{k+1}^{x,y}$ is determined by adding the distance traveled during Δt at a velocity $v_k^{x,y}$, which is predicted from the current position $p_k^{x,y}$, as a linear relationship. Furthermore, assuming that the velocity $v_k^{x,y}$ is affected only by the system noise w_k , the position and velocity states of the next time step $k + 1$ are represented as follows:

$$x_{k+1} = \begin{bmatrix} P_{k+1}^x \\ v_{k+1}^x \\ P_{k+1}^y \\ v_{k+1}^y \end{bmatrix} = \begin{bmatrix} P_k^x + v_k^x \times \Delta t \\ v_k^x \\ P_k^y + v_k^y \times \Delta t \\ v_k^y \end{bmatrix} + w_k. \tag{9}$$

Based on (1), we can obtain the state of the next $k + 1$ time point, x_{k+1} , by adding noise w_k to the system and the product of current state, x_k , and system model matrix, A . Based on the next time point $k + 1$ position, p , and velocity v in (9), we can derive the system matrix, A , to be multiplied by the current state, x_k , as follows:

$$Ax_k = A \begin{bmatrix} p_k^x \\ v_k^x \\ p_k^y \\ v_k^y \end{bmatrix} = x_{k+1} - w_k$$

$$= \begin{bmatrix} p_{k+1}^x \\ v_{k+1}^x \\ p_{k+1}^y \\ v_{k+1}^y \end{bmatrix} - w_k$$

$$= \begin{bmatrix} p_k^x + v_k^x \times \Delta t \\ v_k^x \\ p_k^y + v_k^y \times \Delta t \\ v_k^y \end{bmatrix}, \quad (10)$$

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (11)$$

Only position p is measured by the Kalman filter algorithm for the position and velocity variables constituting the system. The measurement model matrix H indicates that the relationship between the measured and estimated values uses only the information on position p . Value z_k measured by (2) is only for the vehicle coordinate positions p_k^x and p_k^y . Therefore, using only velocity p with z_k , we can obtain H as

$$z_k - m_k = Hx_k$$

$$= H \begin{bmatrix} p_k^x \\ v_k^x \\ p_k^y \\ v_k^y \end{bmatrix}$$

$$= \begin{bmatrix} p_k^x \\ p_k^y \end{bmatrix}, \quad (12)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (13)$$

We can predict the vehicle velocity, v_k , from the measured vehicle position, p_k , using the variables obtained from (8), (11), (13), and the Kalman filter algorithm. Finally, variables Q and R of (4)–(5) must be determined. The sizes of matrices A and H are 4×4 and 2×4 , respectively, that is, $n = 4$ and $m = 2$. Therefore, the sizes of matrices Q and R should be 4×4 and 2×2 , respectively.

$$Q = q \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R = r \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (14)$$

The diagonal matrices, Q and R , can be represented as the product of the identity matrix and integers q and r . The prediction result characteristics of the Kalman filter differ

depending on which value is selected for the q and r parameters in the application. The q and r values were heuristically selected to reflect the measurements, that is, $q = 200$ and $r = 50$.

IV. PROPOSED TECHNIQUE

A. ADAPTIVE RECEPTION CYCLE CONTROL

In the smart vehicle IoT network assumed in this study, the sensor node requests vehicle position data at regular intervals. The vehicle is driven on a two-dimensional plane road, and the position of the vehicle is represented by the coordinates of the x and y axes. When the moving vehicle transmits coordinate data to the sensor node, the sensor node predicts the velocity of the vehicle by applying the Kalman filter to the received position of the vehicle. The interface configuration procedure for connecting the communication channel between the vehicle and sensor node is omitted. Unlike vehicles, which can provide plenty of power, sensor nodes need to operate on battery-level power; therefore, an efficient firmware design is required to minimize resource consumption. However, simply increasing the data reception cycle in communicating with the vehicle can reduce process time, extending the life of the device but cannot guarantee the accuracy and fidelity of the sensor node operation with certainty. Therefore, the static operation of the sensor node should be removed and the firmware function needs to be adaptively changed based on the state of sensed vehicle.

The roles performed by sensor nodes consist of three parts: sensing, transmission, and processing. The fundamental approach of the proposed technique is to reduce resource consumption due to data transfer between sensor nodes and target objects. In other words, it should be considered that the proportion of resource consumption due to data transfer is large across sensor nodes. Therefore, we chose the Kalman filter as our comprehensive application where the sensor node communicates with the target and processes the received data. This approach is applied to the Kalman filter application and makes sensor node behave adaptively. Not only the vehicle's state sensing, but also the sensing value processing adaptively changes the static operation of the sensor node. The Kalman filter prediction can be used to determine the vehicle velocities, v_k^x and v_k^y , using the k th time point position, p_k^x and p_k^y received from the vehicle; v_k , which is the vector sum of the velocities along the x and y directions is calculated by using the following formula:

$$v_k = \sqrt{(v_k^x)^2 + (v_k^y)^2}. \quad (15)$$

The proposed technique configures a distance-based weight, W_D , for the position data reception cycle, allowing intermittent communication when the vehicle speed estimated by the sensor node is slow. As shown in Fig. 2, velocity v_k^i is estimated by using position p_k^i received from the i th vehicle and the Kalman filter operation. When the W_D factor is determined, the reception cycle can be run-time modified based on the vehicle velocity. The reception period, Δt_{rx}^i , from the k th to the time point of the next position data

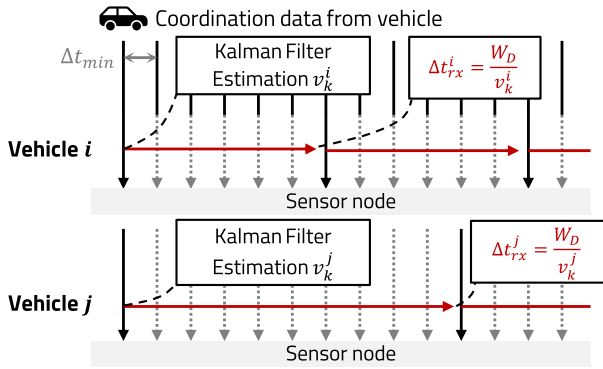


FIGURE 2. Proposed adaptive sensor node reception period according to vehicle velocity.

reception request is expressed as follows:

$$\Delta t_{rx}^i = \frac{W_D}{v_k^i}. \quad (16)$$

The sensor node does not immediately generate the next position request after velocity estimation but proceeds after hiatus of Δt_{rx}^i until the vehicle moves by the W_D . Because all vehicles move at the various speed, the sensor node requests data from different cycles based on the speed of the vehicle. The sensor node communication cycle with the j th vehicle, shown in Fig. 2, is determined by the same W_D as that used for the estimated speed v_k^j . W_D controls the reception cycle by the difference in the time consumption for vehicles of different velocities to move the same distance. The value of W_D and the velocity of the vehicle determine the weight added to the reception cycle. The communication cycle is not fixed for all vehicles, but fluid communication cycle is determined according to the target vehicle state.

$$v^1 \Delta t_{rx}^1 = v^2 \Delta t_{rx}^2 = \dots = v^j \Delta t_{rx}^j = W_D \quad (17)$$

The relationship between the sensor node communication period t_{rx} from 1st vehicle to the i th vehicle and the W_D can be described as in (17). Vehicles from 1st to i th all have different velocities when the receive cycle is changed using W_D . (i.e., v^1, \dots, v^i) The vehicle uses W_D to derive a unique t_{rx} . This calculation is iterated on the sensor node to determine the appropriate reception cycle for the vehicle's velocity as adaptive.

Determining the W_D value depends on the characteristics of the moving vehicle and communication frequency weighting policies. Let Δt_{min} be the cycle that requests the data from the fastest vehicle when considering the sensor node specification, and the vehicle maximum speed is v_{max} . The minimum value of W_D is calculated as $v_{max} \Delta t_{min}$. As W_D increases, the weight added to the communication cycle of the slower vehicle proportionally increases, which slows down the communication cycle with respect to the faster vehicle.

Sensor node internal total amount of computation P can be represented as follows:

$$P_{conv} = KF \times \frac{T}{\Delta t_{min}} + C \quad (18)$$

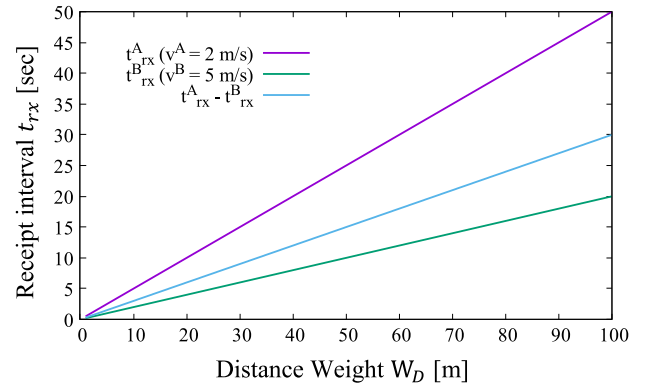


FIGURE 3. The difference in vehicle reception period obtained by subtracting vehicle velocities proportionally increases according to the weight W_D .

where P_{conv} is the total computation from conventional sensor node; KF is the computation from Kalman filter algorithm; C is the computation from other constant functions; T is the entire operating period; Δt_{min} is the data reception cycle from vehicle. In the conventional sensor node, the communication cycle between the target vehicle is fixed to constant with Δt_{min} . When the sensor node communicates with the vehicle at Δt_{min} fixed throughout T , the sensor node always consumes excess computing resources. Even though the status is different for each vehicle, if the same communication cycle is applied, overhead will occur due to unnecessary accuracy.

Applying the proposed technique (16) to (18) as follows:

$$P_{prop} = KF \times \frac{T \times v_{predict}}{W_D} + C \quad (19)$$

where P_{prop} is the total computation from proposing adaptive reception cycle control applied sensor node; $v_{predict}$ is the velocity prediction from Kalman filter; W_D is the aforementioned distance weight. Δt_{min} in (18) is replaced with $\frac{W_D}{v_{predict}}$.

Applying the proposed W_D , the sensor node's communication cycle is no longer constant. $v_{predict}$ can be defined as variable depending on the state of the vehicle. $v_{predict}$ is the vehicle velocity predicted by the Kalman filter when the new reception cycle t_{rx} is determined. In other words, the velocity of the vehicle at that time adaptively determines the reception cycle. Since the minimum value of W_D is $v_{max} \Delta t_{min}$, the minimum value of $\frac{W_D}{v_{predict}}$ is Δt_{min} .

$$\frac{W_D}{v_{predict}} \geq \Delta t_{min} \quad (20)$$

Therefore, we can derive $P_{conv} \geq P_{prop}$ by substituting (20) to (18, 19). Using W_D and variable cycles depending on velocity, rather than always communicating at maximum frequency, reduces sensor node computation.

As depicted in Fig. 3, we assume that the estimated speeds of the two vehicles, A and B, are $v^A = 2m/s$ and $v^B = 5m/s$, respectively. Fig. 3 shows the difference change between the two communication cycles $\Delta t_{rx}^A - \Delta t_{rx}^B$ of the sensor nodes $\Delta t_{rx}^A, \Delta t_{rx}^B$ while increasing weight W_D of the reference. As W_D increases, the sensor node reception cycle difference

Algorithm 1: Kalman Filter Algorithm for Prediction Velocity-Based Adaptive Reception Period Control

```

1 t_min : Minimum sensing interval
2 t_rx : Sensing interval of the vehicle
3 kal_vel : Velocity predicted by the Kalman filter
4 iter_rx : Iterations for t_rx comparison
5 pos_x,y : Received vehicle position
6 pre_t_rx, pre_kal_vel : Previous values of t_rx, kal_vel,
   respectively
7
8 % Parameter initialization
9 iter_rx = 1
10 pre_t_rx = t_min
11 pre_kal_vel = 0
12
13 while every t_min during simulation do
14   foreach ith vehicle do
15     if iter_rx == 1 then
16       Calculate t_rx^i using W_D and pre_kal_vel
17     else
18       t_rx^i ← pre_t_rx^i
19
20     if t_rx^i == iter_rx then
21       Position request sent to ith vehicle → pos_x,y^i
22       Predict kal_vel^i using the Kalman_Filter(
23         pos_x,y^i, t_rx^i )
24       iter_rx = 1
25     else
26       kal_vel^i ← pre_kal_vel
27       iter_rx = iter_rx + 1
28
29   pre_t_rx^i ← t_rx^i
   pre_kal_vel^i ← kal_vel^i

```

$\Delta t_{rx}^A - \Delta t_{rx}^B$ increases proportionally. When W_D increases, the ratio of the data reception cycles of the two vehicles is not affected, but the difference of the reception cycle increases. Thus, the weight is defined as an effect that increases the reception cycle t_{rx}^A of a relatively slow vehicle (for example, Vehicle A in Fig. 3). Furthermore, even if the absolute speed value of the vehicle is the same, a slower vehicle can be weighted to have a relatively sporadic communication cycle according to the W_D value.

Algorithm 1 describes a method that adaptively adjusts the vehicle communication cycle during run-time based on the Kalman filter operation. The estimated speed at the sensor node is described in Algorithm 1. Algorithm 1 is executed at every minimum sensing period t_{min} iteration, where the simulation updates the dynamic changes of the vehicle (Step 13). Because the speed is different for each vehicle, the Kalman filter operation is performed separately at the sensor node, which is described for the i th vehicle in Algorithm 1

(Step 14). If the i th period t_{rx}^i requesting the transmission of the vehicle position is not determined, the distance weight W_D and the Kalman filter estimated speed of the previous iteration pre_kal_vel are used to calculate t_{rx}^i . If $iter_rx$ is not equal to one, there is a wait for the modified request cycle, such that the previous $pre_t_{rx}^i$ is retained (Step 15-18). When $iter_rx$ satisfies t_{rx}^i , the Kalman filter estimates the velocity kal_vel^i , using the vehicle position $pos_{x,y}^i$. Then, $iter_rx$ is initialized to one so that the next t_{rx}^i can be determined (Step 21-23). Otherwise, the previous Kalman filter-predicted velocity $pre_kal_vel^i$ is retained and $iter_rx$ is incremented by one (Step 24-26).

In the proposed technique, the communication cycle of the sensor node does not operate in a fixed state and can be adjusted according to the dynamic velocity state of the target vehicle. General embedded systems have an overhead from fixed function using static program and inefficient resource utilization. The proposed technique recognizes the target environment (i.e., vehicle status) in a fixed internal function and operates adaptively. In addition, the proposed technique achieved efficient operation with an understand for the application domain without modifying the physical layer of the sensor node's connection architecture. Therefore, processing time on the device can be reduced. The reduction in the absolute overhead for a single sensor node may be small, but long-term operation in a large IoT sensor network provides significant and efficient results.

V. EXPERIMENTS

In this study, we used a traffic simulator environment implemented in MATLAB to evaluate the proposed technique [16]. Because is difficult to experiment with a scenario where the sensor node is communicating with multiple vehicles on an actual road, this situation was replicated using a MATLAB traffic simulator to analyze the behavior of the sensor node. The environment used for MATLAB simulation was a Windows 10 64-bit OS, Intel i7-9700K 3.6 GHz CPU, 48 GB RAM.

The open-source MATLAB traffic simulator used in this experiment is displayed in Fig. 4. The simulator implemented in MATLAB R2020b creates roads and intersections on a two-dimensional coordinate plane so that vehicles, moving at different speeds are on the road. Fig. 4(a) shows the placement of roads, intersections, and sensor nodes used in the simulator. The road has two intersections, and the position of the vehicle is constrained to the range of $(-155, 460)$ in the x-axis direction and $(-159, 159)$ in the y-axis direction. The position of the sensor node is $(159.6, 84.6)$. Fig. 4(b) represents the actual velocity of the vehicle spawned by the traffic simulator. The actual velocity is plotted when the number of spawned vehicles in the simulation is four. Each vehicle randomly moves within the maximum speed range and maintains a constant speed until it approaches an intersection. When the vehicle reaches the intersection and stops according to traffic lights, the speed decreases to zero. Recovering the set speed while slowing down indicates that the traffic light changes from red to green before the vehicle approaches the

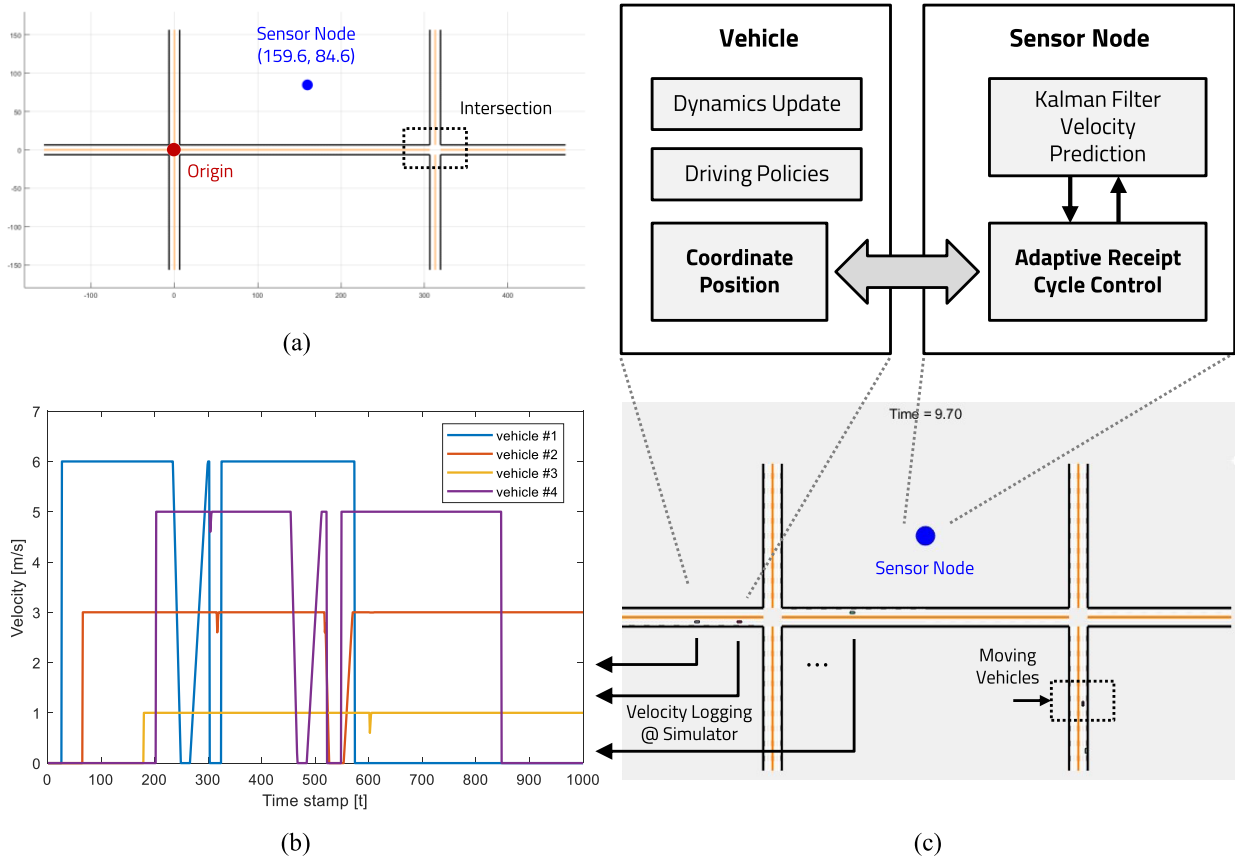


FIGURE 4. (a) MATLAB traffic simulator intersections and sensor node placement, (b) Moving vehicle real velocity comparison in MATLAB traffic simulator, (c) MATLAB traffic simulation window in progress and internal components.

intersection. In this experiment, we evaluate the proposed technique in a virtual environment using a traffic simulator instead of a real physical environment. The scope of the proposed technique focuses on reducing computation internal sensor nodes. To implement a real sensor node, the communication interface layer within sensor network is required. The general communication interface layer needs to consider various factors such as the distance between the sensor node and the object, the communication network latency. A proposed technique focused on the reduction of the computational load of the sensor node according to the target state, thus the communication interface was assumed to be out of boundary. The traffic simulator used in the simulation contains almost of real road components such as lanes, intersections, and traffic lights. In addition, the movement of vehicles spawned on the road is controlled by traffic light as well as its own velocity policy. These properties can minimize the difference between parameters in physical environment measurements and simulations. Furthermore, the simulation was configured considering the characteristic of sensor node on-device processing. Consequently, we were able to replicate the sensor node behavior in the physical environment in a similar way.

At the start of the simulation, Fig. 4(c) window will appear allowing the user to see the movement of the vehicle on the road in incremental time steps. The black dots indicate each vehicle in motion. The actual velocity shown in Fig. 4(b) was

logged from vehicles represented by the black dots. When a vehicle encounters a traffic light at an intersection, it decides whether to wait according to a predefined signal policy. The relationship between the sensor node and vehicle objects, which are internal components of the simulation, is shown in the simulator window. For vehicle objects, the dynamics are updated according to driving policies such as lanes, intersections, and traffic lights. The sensor node requests a position coordinate from the vehicle according to the reception cycle determined by the adaptive reception cycle control. The Kalman filter operation is executed using the position received through this request. Vehicles spawned on the road are determined by the random numbers from a Poisson distribution, as follows:

$$f(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{21}$$

where λ is the expected value of the number of incidents that occur within a specific time and k is the number of incidents that do occur. The number of vehicles on the road is constrained by the Poisson random number (21) when $\lambda = 1.0$ to determine whether to spawn a new vehicle. The initial speed of each vehicle is random within the maximum speed range and is subsequently adjusted by the intersection signal policy. The traffic light policy is determined by the duration of each color after the initial signal light condition.

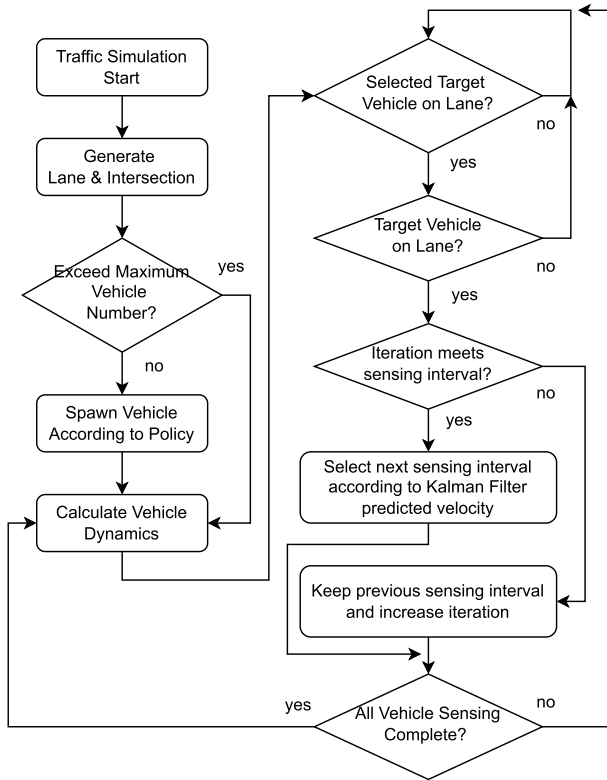


FIGURE 5. Sensor node and vehicle communication procedure flowchart for traffic simulation.

The execution of the traffic simulator and sensor node operation flow to evaluate the proposed technique is illustrated in Fig. 5. The sensor node receives the coordinate position of the moving vehicle to predict the velocity using the Kalman filter. We assumed that the minimum period for requesting the coordinate position from the vehicle at the sensor node is equal to the time step of the update of vehicle dynamics in simulation. The sensor node inputs the coordinates received for each vehicle into the Kalman filter to estimate the speed and determine the next position data reception period of the vehicle at run-time based on the distance weight W_D . When the coordinate is not received in that time step because of the modified t_{rx} , the predicted velocity is maintained from the previous t_{rx} . Subsequently, all the vehicles in the simulator are sensed, and the vehicle dynamics are updated in the next time step.

The velocity prediction result using the Kalman filter is compared with the actual vehicle velocity in Fig. 6. The actual velocity is calculated using the actual vehicle coordinates, and KF-Conventional is the sensor node Kalman filter velocity estimation result of requesting location data from the fastest cycle t_{min} . KF-Proposed is the velocity at which the vehicle position reception cycle is controlled by the sensor node using the proposed technique. If the position of the vehicle deviates from the coordinates set in the range of the road, all the speed values are logged to zero. In KF-Conventional, the position data are requested from the vehicle at all possible time steps of the sensor node, and the Kalman filter is applied

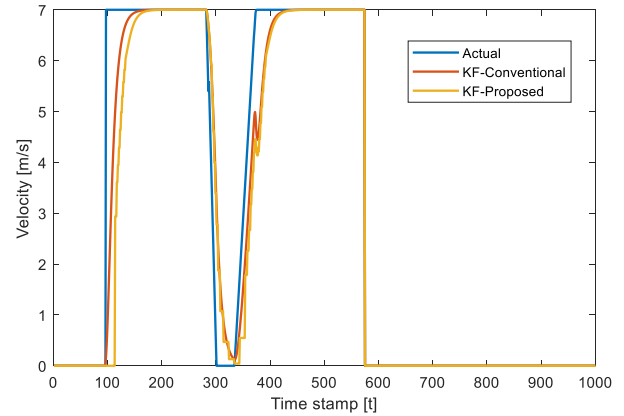


FIGURE 6. Comparison of the velocity in three cases, actual velocity (Actual); Kalman filter predicted velocity without the proposed technique (KF-Conventional); Kalman filter predicted velocity using the proposed technique (KF-Proposed).

for the speed estimation operation. Therefore, the difference from the actual speed is expected to be the smallest, but the calculation processing time is the largest. The discrepancies between the actual and predicted speeds were compared in terms of the root mean square error (RMSE).

$$RMSE = \sqrt{\frac{1}{n} \sum_{k=1}^n (vel_{act} - vel_{kalman})^2}, \quad (22)$$

where n is the number of total simulation time steps: k is the coordinate position sensing time step, vel_{act} is the actual vehicle velocity; and vel_{kalman} is the Kalman filter-predicted vehicle velocity.

To evaluate the proposed technique, we changed the number of vehicles spawned in the traffic simulation with the value of W_D and measured the RMSE without adjusting the reception cycle. In Fig. 7, Reference is the RMSE for actual velocity when performing Kalman filter velocity estimation in the simulation without using the proposed technique. In the proposed method, the weight assigned to the sensor node reception cycle differs depending on W_D . The minimum value of W_D is determined by the minimum available communication cycle and the maximum vehicle speed, depending on the sensor node specifications. In the experiment, the minimum case of W_D was selected as four considering the maximum speed of the vehicle spawned in the simulator and the sensor node minimum communication cycle. We increased W_D from four and measured sensor node computation and predictive accuracy in response to changes in W_D . Increasing W_D means focusing on reducing the amount of computation in the trade-off relationship between accuracy and computing efficiency. However, an excessively large W_D has too low accuracy and may not reach the acceptable threshold for the application. Thus, the result was measured using $W_D = \{4, 5, 6, 7\}$. We have decided that seven is the maximum case, which is an increase of 75 % from the minimum case for sufficient coverage of changes. The number of vehicles generated in the simulation was varied from 10 to 25. The results show that RMSE increases with respect to the Reference for

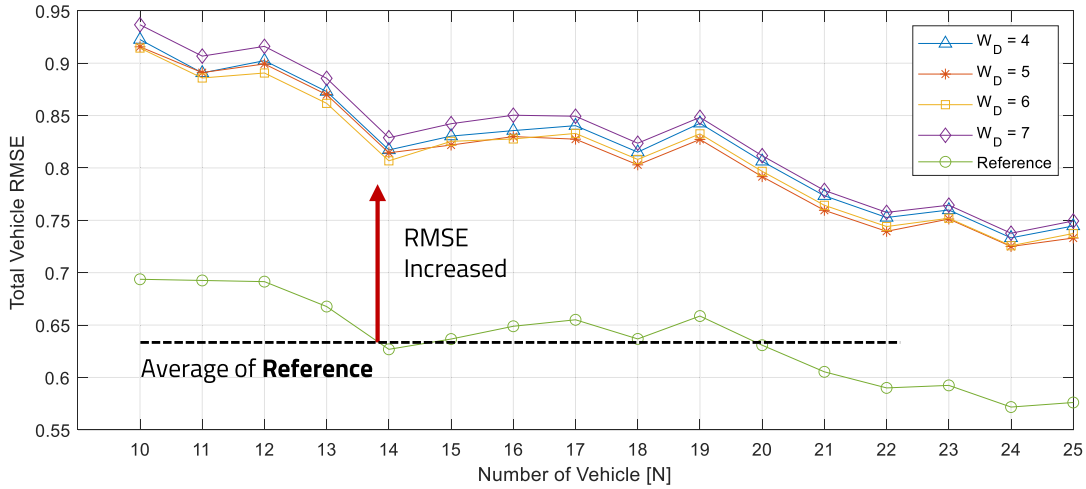


FIGURE 7. RMSE comparison between the actual velocity of the vehicle according to W_D and number of vehicles in the simulation.

TABLE 1. Percentage (%) of RMSE comparison between the Reference and W_D cases.

Distance weight W_D	RMSE difference according to the number of vehicles (%)							
	10	11	12	13	14	15	16	17
$W_D = 4$	32.99	28.61	30.51	30.74	30.37	30.43	28.79	28.29
$W_D = 5$	31.99	28.65	30.06	30.30	29.93	29.09	27.91	26.32
$W_D = 6$	31.82	27.91	28.81	29.06	28.71	29.64	27.56	27.16
$W_D = 7$	35.03	30.93	32.49	32.63	32.23	32.29	31.05	29.67
	18	19	20	21	22	23	24	25
$W_D = 4$	28.01	27.94	27.83	27.77	27.57	28.27	28.26	29.26
$W_D = 5$	26.10	25.60	25.51	25.47	25.33	26.81	26.81	27.26
$W_D = 6$	26.91	26.41	26.31	26.27	26.11	26.94	26.94	28.01
$W_D = 7$	29.34	28.80	28.68	28.61	28.40	29.04	29.02	30.08

TABLE 2. Percentage (%) of processing time comparison between Reference and W_D cases.

Distance weight W_D	Processing time difference according to the number of vehicles (%)							
	10	11	12	13	14	15	16	17
$W_D = 4$	59.09	59.59	60.51	57.07	60.19	54.13	58.01	55.56
$W_D = 5$	59.11	57.55	62.28	58.86	63.64	61.86	63.38	57.24
$W_D = 6$	57.39	63.73	56.92	61.95	58.77	60.09	56.71	62.30
$W_D = 7$	57.89	62.95	60.90	65.55	63.32	67.57	63.38	64.14
	18	19	20	21	22	23	24	25
$W_D = 4$	69.32	65.28	67.18	63.41	67.77	62.39	65.37	63.89
$W_D = 5$	66.80	64.03	66.78	65.22	68.97	66.37	68.92	65.79
$W_D = 6$	63.64	69.95	67.69	67.80	69.19	70.64	68.83	72.22
$W_D = 7$	68.42	71.94	68.51	72.91	71.47	69.97	70.46	70.07

all W_D cases. The RMSE average for the Reference in all cases is 0.6359. Using W_D , we obtained the average RMSE for $W_D = 4, 5, 6, 7$ as 0.8213, 0.8125, 0.8129, and 0.8304, respectively.

In Table 1, the RMSE difference between the Reference and each W_D case is represented as a percentage. As W_D increased, the RMSE difference increased. Considering the role of the weight W_D , a larger W_D adds more weight to the reception cycle of slower vehicles. At the same vehicle speed, the increase in the relative reception cycle difference increases the RMSE with respect to the Reference.

As shown in Fig. 8, the change in the sensor node processing time is measured within the $W_D = \{4, 5, 6, 7\}$ range. The number of vehicles generated in the simulation is equal to the previous RMSE measurement, that is, within the range of 10 to 25. The proposed method has a lower processing time than the Reference, regardless of the W_D value. The

higher the value of W_D , the lower is the average processing time. The result can be analyzed by using the aforementioned correlation between weight W_D and the reception cycle. The average processing time for the Reference for all cases of the number of vehicles is 0.2547. Using W_D , we obtained the RMSE averages for $W_D = 4, 5, 6, 7$ as 0.1032, 0.097, 0.0862, and 0.0767, respectively. As the W_D value increases, the cost of the computation time on the sensor node decreases. This result indicates that a high W_D value increases the reception period weight assigned to slow vehicles.

The result of the processing time comparison for the change in W_D with respect to the Reference is presented in Table 2 as a percentage. For the $W_D = 4$ case, the average RMSE for all vehicle cases is 29.1% and the average processing time reduction is 59.2%. When the number of vehicles is 21 and $W_D = 7$, the reduction in processing time is the largest at 72.91%. In contrast, it is the lowest at 54.13% when

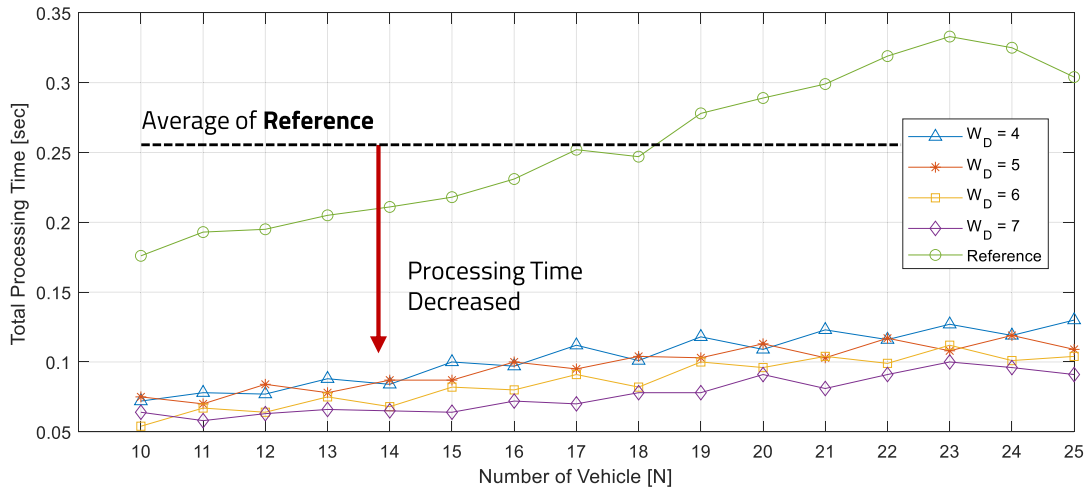


FIGURE 8. Sensor node processing time comparison according to W_D and the number of vehicles in the simulation.

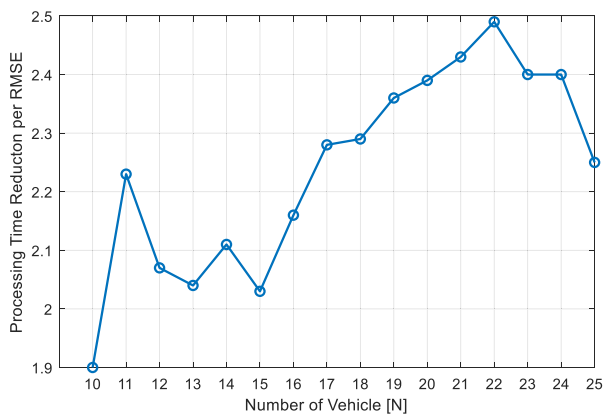


FIGURE 9. For all W_D cases, the average processing time reduction per RMSE increases with the number of vehicles in the simulation.

the number of vehicles is 15 and $W_D = 4$. Similarly, for the other W_D cases, the reduction in processing time is greater than the increment in the RMSE. The velocity prediction accuracy can decrease to a reasonable extent; however, the processing time at the sensor node is significantly reduced. For battery-powered sensor nodes, efficient resource usage is prioritized under various constraints. Therefore, reducing the process time using the proposed technique can be useful.

Fig. 9 shows the reduction in the processing time per RMSE as the number of vehicles in the simulation increases. Each plotted value was derived as the average of Table 1 and 2 W_D cases. A higher graph value indicates a larger process time improvement with the same RMSE fluctuation. The reduction in the processing time per RMSE tends to increase as the number of vehicles spawned in the traffic simulation changes from 10 to 25. In some sections (e.g., for the number of vehicles 10–13, and 22–23), the processing time reduction per RMSE appears to be reduced. However, considering the overall tendency of the graph, the processing time reduction per RMSE value increases with the number of vehicles. The reduction that occurs in the case of 22–25 section can be considered a temporary phenomenon.

Therefore, it can be expected that the process time reduction per RMSE value increases since the number of vehicles 25. The number of vehicles coverage applied in the experiment is from 10 to 25 increased by 150%. Therefore, the contribution of the proposed technique can be justified enough considering the achieved processing time overhead reduction of more than twice the RMSE fluctuation in all cases, only except 10. Consequently, the proposed technique is more suitable when the number of vehicles is large, and the sensor nodes of a large-scale IoT network managing a large number of vehicles are expected to have a massive effect.

VI. CONCLUSION

In this paper, we propose a technique for the efficient use of sensor node resources in large-scale IoT applications. In particular, to predict a vehicle’s velocity by using the Kalman filter, the reception cycle of the position data from the vehicle is adaptively controlled to reduce the processing time of the sensor node. Thus, the processing time at the IoT sensor node is significantly reduced by minimizing the speed estimation accuracy. In addition, effective results were achieved for large-scale IoT networks when the number of vehicles increased. The experimental results showed that the processing time decreased by up to 72.91% compared to the RMSE increase in the case of 21 vehicles and $W_D = 4$. The proposed technique made a practical contribution that allow efficient computation on sensor nodes designed in embedded systems. In future work, we would like to expand the proposed technique to wireless sensor network which consisted to cooperative multiple sensor nodes to achieve macro-view operation. Moreover, this research can be extended to physically embedded board environments to achieve efficient resource consumption in terms of sensor node energy.

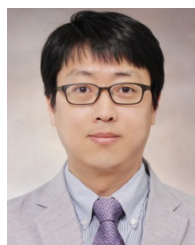
REFERENCES

- [1] T. Adegbija, A. Rogacs, C. Patel, and A. Gordon-Ross, “Microprocessor optimizations for the Internet of Things: A survey,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 7–20, Jan. 2018.
- [2] D. L. Dutta and S. Bharali, “TinyML meets IoT: A comprehensive survey,” *Internet Things*, vol. 16, Dec. 2021, Art. no. 100461.

- [3] X. Wang, M. Magno, L. Cavigelli, and L. Benini, "FANN-on-MCU: An open-source toolkit for energy-efficient neural network inference at the edge of the Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4403–4417, May 2020.
- [4] D. Blaauw, D. Sylvester, P. Dutta, Y. Lee, I. Lee, S. Bang, Y. Kim, G. Kim, P. Pannuto, Y. Kuo, D. Yoon, W. Jung, Z. Foo, Y. Chen, S. Oh, S. Jeong, and M. Choi, "IoT design space challenges: Circuits and systems," in *Symp. VLSI Technol. (VLSI-Techol.): Dig. Tech. Papers*, Jun. 2014, pp. 1–2.
- [5] I. Fedorov, R. P. Adams, M. Mattina, and P. Whatmough, "Sparse: Sparse architecture search for CNNs on resource-constrained microcontrollers," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 1–13.
- [6] S. Ciccia, G. Giordanengo, and G. Vecchi, "Energy efficiency in IoT networks: Integration of reconfigurable antennas in ultra low-power radio platforms based on system-on-chip," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6800–6810, Aug. 2019.
- [7] A. Pullini, D. Rossi, L. Loi, G. Tagliavini, and L. Benini, "Mr.Wolf: An energy-precision scalable parallel ultra low power SoC for IoT edge processing," *IEEE J. Solid-State Circuits*, vol. 54, no. 7, pp. 1970–1981, Jul. 2019.
- [8] J. Kwon, M. G. Seok, and D. Park, "Low-power fast partial firmware update technique of on-chip flash memory for reliable embedded IoT microcontroller," *IEICE Trans. Electron.*, vol. 104, no. 6, pp. 226–236, 2021.
- [9] J. Kwon, M. G. Seok, and D. Park, "User insensible sliding firmware update technique for flash-area/time-cost reduction toward low-power embedded software replacement," in *Proc. IEEE Symp. Low-Power High-Speed Chips (COOL CHIPS)*, Apr. 2020, pp. 1–3.
- [10] K. Ramya, K. P. Kumar, and V. S. Rao, "A survey on target tracking techniques in wireless sensor networks," *Int. J. Comput. Sci. Eng. Surv.*, vol. 3, no. 4, p. 93, 2012.
- [11] S. Kim, J. Cho, and D. Park, "Moving-target position estimation using GPU-based particle filter for IoT sensing applications," *Appl. Sci.*, vol. 7, no. 11, p. 1152, Nov. 2017. [Online]. Available: <https://www.mdpi.com/2076-3417/7/11/1152>
- [12] G. Welch and G. Bishop, *An Introduction to the Kalman Filter*. Chapel Hill, NC, USA: 1995.
- [13] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," in *Proc. 8th Int. Conf. Intell. New. Intell. Syst. (ICINIS)*, Nov. 2015, pp. 74–77.
- [14] K. Hirpara and K. Rana, "Energy-efficient constant gain Kalman filter based tracking in wireless sensor network," *Wireless Commun. Mobile Comput.*, vol. 2017, pp. 1–7, Mar. 2017.
- [15] M. Raitoharju and R. Piche, "On computational complexity reduction methods for Kalman filter extensions," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 34, no. 10, pp. 2–19, Oct. 2019.
- [16] E. Gravelle. (2016). *Traffic-Simulator*. [Online]. Available: <https://github.com/evangravelle/traffic-simulator>
- [17] H. Kim and S. W. Han, "An efficient sensor deployment scheme for large-scale wireless sensor networks," *IEEE Commun. Lett.*, vol. 19, no. 1, pp. 98–101, Jan. 2015.
- [18] C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy, "Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink," *IEEE Trans. Mobile Comput.*, vol. 14, no. 9, pp. 1947–1960, Sep. 2014.
- [19] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-efficiency of MIMO and cooperative MIMO techniques in sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 6, pp. 1089–1098, Aug. 2004.
- [20] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng, "Efficient in-network moving object tracking in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 5, no. 8, pp. 1044–1056, Aug. 2006.
- [21] H. Qin and W. Zhang, "ZigBee-assisted power saving management for mobile devices," *IEEE Trans. Mobile Comput.*, vol. 13, no. 12, pp. 2933–2947, Dec. 2014.
- [22] C. S. Abella, S. Bonina, A. Cucuccio, S. D'Angelo, G. Giustolisi, A. D. Grasso, A. Imbruglia, G. S. Mauro, G. A. M. Nastasi, G. Palumbo, S. Pennisi, G. Sorbello, and A. Scuderi, "Autonomous energy-efficient wireless sensor network platform for home/office automation," *IEEE Sensors J.*, vol. 19, no. 9, pp. 3501–3512, May 2019.
- [23] J.-H. Chang and L. Tassioulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 4, pp. 609–619, Aug. 2004.
- [24] P. Floreen, P. Kaski, J. Kohonen, and P. Orponen, "Lifetime maximization for multicasting in energy-constrained wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 117–126, Jan. 2005.
- [25] S. Khan, A.-S. K. Pathan, and N. A. Alrajeh, *Wireless Sensor Networks: Current Status and Future Trends*. Boca Raton, FL, USA: CRC Press, 2016.
- [26] F. Bouabdallah, N. Bouabdallah, and R. Boutaba, "On balancing energy consumption in wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 58, no. 6, pp. 2909–2924, Jul. 2009.
- [27] L. Ferrigno, S. Marano, V. Paciello, and A. Pietrosanto, "Balancing computational and transmission power consumption in wireless image sensor networks," in *Proc. IEEE Symp. Virtual Environments, Human-Comput. Inter. Meas. Syst.*, Jul. 2005, p. 6.
- [28] R. Marchthaler, "Adaptive extended Kalman filter (ROSE-filter) for positioning system," 2021, *arXiv:2108.11321*.
- [29] D. G. Perumal, B. Subathra, G. Saravanakumar, and S. Srinivasan, "Extended Kalman filter based path-planning algorithm for autonomous vehicles with I2V communication," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 652–657, 2016.
- [30] C. G. Prevost, A. Desbiens, and E. Gagnon, "Extended Kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle," in *Proc. Amer. Control Conf.*, Jul. 2007, pp. 1805–1810.
- [31] T. A. Wenzel, K. J. Burnham, M. V. Blundell, and R. A. Williams, "Dual extended Kalman filter for vehicle state and parameter estimation," *Vehicle Syst. Dyn.*, vol. 44, no. 2, pp. 153–171, 2006, doi: [10.1080/00423110500385949](https://doi.org/10.1080/00423110500385949).
- [32] T. Kim and T.-H. Park, "Extended Kalman filter (EKF) design for vehicle position tracking using reliability function of radar and lidar," *Sensors*, vol. 20, no. 15, p. 4126, Jul. 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/15/4126>
- [33] J. Kwon and D. Park, "Implementation of computation-efficient sensor network for Kalman filter-based intelligent position-aware application," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIIC)*, Feb. 2020, pp. 565–568.



JISU KWON received the B.S. degree from the School of Electronics Engineering, Kyungpook National University, Daegu, Republic of Korea, in 2019. He is currently pursuing the integrated Ph.D. degree with the AI-Embedded System Software-on-Chip Laboratory, School of Electronic and Electrical Engineering, Kyungpook National University. His research interest includes behavior-changeable neuromorphic deep learning processors based on partial software replacement and hardware reconfiguration-coupled architecture.



DAEJIN PARK (Member, IEEE) received the B.S. degree in electronics engineering from Kyungpook National University, Daegu, South Korea, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2003 and 2014, respectively. From 2003 to 2014, he was a Research Engineer at SK Hynix Semiconductor, Samsung Electronics, for over 12 years. He worked on designing low-power embedded processor architectures and implementing fully AI-integrated system-on-chip with intelligent embedded software on a custom-designed hardware accelerator, especially for hardware/software tightly coupled applications, such as smart mobile devices and industrial electronics. Since 2014, he has been a full-time Associate Professor with the School of Electronics and Electrical Engineering and the School of Electronics Engineering, Kyungpook National University. He has published over 200 technical articles and 40 patents. In 2014, he was nominated as one of the Presidential Research Fellows 21, Republic of Korea.

• • •