Aggregate Location Monitoring for Wireless Sensor Networks: A Histogram-based Approach

Chi-Yin Chow Mohamed F. Mokbel Tian He
Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN
{cchow, mokbel, tianhe}@cs.umn.edu

Abstract

Location monitoring systems are used to detect human activities and provide monitoring services, e.g., aggregate queries. In this paper, we consider an aggregate location monitoring system where wireless sensor nodes are counting sensors that are only capable of detecting the number of objects within their sensing areas. As traditional query processors rely on the knowledge of users' exact locations, they cannot provide any monitoring services based on the readings reported from counting sensors. To this end, we propose an adaptive spatio-temporal histogram to enable monitoring services without the need of users' exact locations. The main idea of the histogram is to keep statistics about the distribution of moving objects. At the core of the histogram, we propose three techniques, memorization, locality awareness and packing, to improve monitoring accuracy and efficiency. Furthermore, the histogram is designed in a way that achieves a trade-off between the energy and bandwidth consumption of the sensor network and the accuracy of monitoring services. Experimental results show that the proposed histogram provides high-quality location monitoring services (i.e., 90% accuracy for both skewed and uniform mobility patterns) and outperforms a basic histogram and the state-of-the-art spatio-temporal histogram by two orders of magnitude in most cases.

1. Introduction

Advances in sensor devices and wireless communication technologies have resulted in many new applications for military and civilian purposes, in which aggregate location monitoring is one of the key applications. Aggregate location monitoring has a simple form of *"What is the number of objects in a certain area"*. In general, aggregate location monitoring systems provide several valuable services that include: (1) *Density queries*, e.g., "determine the number of moving objects within a specified query region", (2) *Safety control*, e.g., "send an alert if the number of persons in a certain area exceeds a predefined threshold", and (3) *Resource management*, e.g., "turn off some building facilities if the number of people in a prespecified area is below a certain threshold". Real-life applications of location monitoring include employee tracking in workplaces [1], patient tracking in hospitals [2], and surveillance networks [3]. Such location monitoring systems rely on deploying either *identity* or *counting* sensors. *Identity* sensors communicate with a small wireless transmitter attached to human bodies to determine human's exact locations and identities (e.g., Bat [4], [1], Active Badge [5], and Cricket [6]). On the other side, *counting* sensors are able to determine the number of objects or people within their sensing areas (e.g., photoelectric sensors [7], [8] and thermal sensors [9]). Thus, the *counting* sensor is able to report only *aggregate location information*, i.e., its sensing area along with the number of detected objects within the sensing area, to a server.

This research was supported in part by NSF grants IIS-0811998, IIS-0811935, CNS-0708604, CNS-0626609 and CNS-0720465.

Deploying the simple functionality of aggregate location monitoring services in a wireless sensor network is challenging, mainly for the following three reasons: (1) *Counting* sensors are only capable of reporting aggregate location information about the number of objects in their sensing areas. However, traditional aggregate query processors (e.g., see [10], [11], [12]) rely mainly on the knowledge of the exact object location and there is no direct extension to extend their functionality to support environments where the exact location is unknown, yet aggregate location information is available. (2) *Identity* sensors are known to have major privacy leakage as they can be used to reveal the personal privacy of tracked persons (e.g., see [13], [14], [6]). To avoid such privacy leakage, several location privacy techniques are deployed to turn the exact information from *identity* sensors to be aggregate location information which is similar to the information sent from the *counting* sensors (e.g., see [13]). In this case, we have a similar challenge as the one in *counting* sensors. (3) Any algorithm applied to the sensor network should consider its limitations in terms of limited energy and communication bandwidth. As existing location monitoring algorithms require a continuous stream of location updates (e.g., see [10], [15], [11], [12], [16], [17], [18]), applying these algorithms directly to the sensor network will easily consume system resources.

In this paper, we propose an *adaptive spatio-temporal histogram* for aggregate location monitoring in a wireless sensor network that overcomes the aforementioned challenges. The key objective of our proposed histogram is to enable efficient and accurate location monitoring services while: (a) relying on aggregate location information (i.e., not knowing the exact personal locations) and (b) saving sensor network resources (i.e., energy and bandwidth). At the core of our *location monitoring system*, we employ an *adaptive spatio-temporal histogram* that models the distribution of moving objects in the system area. The main idea of the histogram is to maintain a grid data structure in which each grid cell acts as an *estimator* of the number of objects within its area based on the aggregate location information reported from sensor nodes. Then, the answer of an aggregate query is approximately estimated based on the *estimators* of the enclosed grid cells. To improve the accuracy of the *estimators*, and hence the accuracy of aggregate query answers, we propose three techniques, namely, *memorization*, *locality awareness*, and *packing*, that aim to model the object distribution, model the object movement pattern, and reduce the computational overhead, respectively.

A distinct feature in our *adaptive spatio-temporal histogram* is that it is well aware and takes care of the limitations of the underlying wireless sensor networks [19], e.g., limited energy and communication bandwidth. In that sense, the proposed *histogram* does not require all sensor nodes to send their readings continuously. Instead, a *scheduler* takes place in our system so that we minimize the rate of reporting aggregate location data from sensor nodes, and thus increasing the network lifetime and reducing network bandwidth consumption. The rate of reporting aggregate location data from sensor nodes can be controlled through the *schedular* to tune a trade-off between the accuracy of the histogram and the energy and bandwidth consumption of the sensor network.

Experimental evaluation shows that our *aggregate query processor*, backed by the proposed *adaptive spatio-temporal histogram*, provides a highly accurate answer (at least 90% for both skewed and uniform mobility patterns) for aggregate queries even if: (1) Sensor nodes do not report actual user locations, and (2) Sensor nodes send their aggregate location data in a low rate, thus saving sensor energy, i.e., increasing the network lifetime, and network bandwidth. In the skewed mobility pattern environment, the accuracy of our *adaptive* histogram is at least two orders of magnitude better than a *basic* histogram and an existing spatio-temporal histogram [20]. Furthermore, the experimental results show that the update time of our *adaptive* histogram is at least two orders of magnitude better than the *basic* histogram. The rest of this paper is organized as follows. Section 2 surveys related works. Section 3 formally defines our problem, and delineates our underlying system model. Section 4 describes the *basic* histogram, the *adaptive* histogram employing the three proposed techniques, and the *aggregate query processor*. Section 5 depicts the experiment results. Finally, Section 6 concludes this paper.

2. Related Works

In this section, we highlight the related works in two areas, *query processing in wireless sensor networks* and *spatio-temporal histograms*.

Query processing in sensor networks. Previous related works can be classified by their underlying architecture as *centralized* and *distributed*. The *centralized* approach relies on a *centralized* database server that processes queries based on the readings collected from sensor nodes [21], [22], [23], [24], [25]. The *distributed* approach injects a query to the network via a sensor node that is responsible for collecting the readings from other nodes based on the query predicate and computing the answer [26], [27], [28]. Our work employs the *centralized* approach because the deployed sensor nodes are not assumed to have any query processing capacity. On another dimension, previous works can be generally classified by their supported query types, *aggregate*, *probabilistic* and *location-based queries*. *Aggregate queries* are used to summarize the readings of a set of sensor nodes into a single statistic using aggregate operators [22], [23], [24], [28], e.g., *average*, *sum*, and *top-k*. *Probabilistic* queries enable the user to specify their acceptable tolerance to the error in the answer [21], [22]. Finally, *location-based queries* aim to find target objects that satisfy the spatial predicates [26], [27], e.g., range and k-nearest neighbor queries. Spatio-temporal aggregation techniques employing *sketches* cannot be applied to a *counting* sensor environment because they need the identity of the monitored objects [29], [30]. The closest work to ours is the location-based query processing in wireless sensor networks. However, since all previous works in this area rely on users' exact locations and/or identities, none of these works consider query processing over aggregate location data.

Spatio-temporal histograms. Research efforts for spatio-temporal histograms aim to provide selectivity estimation for predictive spatio-temporal queries for one-dimensional [31] and multi-dimensional moving objects [32]. The main idea is to predict the effect of the moving trajectory on the query answer. Other works on selectivity estimation of spatio-temporal queries rely on duality transformation [33], the existence of a secondary index structure [33], clustering approaches [34], or Venn sampling [35]. However, none of these works can be immediately applied to our environment that includes sensor network and aggregate locations. It is important to note that it is always the case that the input to these previous works is user's exact locations, while the input to our query processor, that employs a spatio-temporal histogram, is a set of aggregate locations. The closest spatio-temporal histogram to ours is the one proposed in [20]. However, such histogram works only under the strict assumption of uniform distribution. We will extensively study the performance gain from applying our *adaptive* histogram with respect to [20].

3. System Model

Figure 1 depicts the system architecture of our aggregate location monitoring system that consists of two major components, *wireless sensor network* and *aggregate query processor*. A third major component that is not shown in the figure is a *resource-efficient sensor scheduler* that is responsible on tuning a trade-off between energy and bandwidth consumption of the *wireless sensor network* and the accuracy of the answers provided by the *aggregate query processor*. We first formally define the problem, and then describe each major component in detail.

Problem definition. We consider a set of sensor nodes $s_1, s_2, \ldots, s_n$ and a set of moving objects $o_1, o_2, \ldots, o_m$. The extents of two or more sensing areas may overlap. An ***aggregate location*** is defined as a reading from a sensor node s_i in a form of $(Area_i, N_i)$ where $Area_i$ is s_i's sensing area and N_i is the number of detected objects within $Area_i$, i.e., $N_i = |O_i|$, $O_i = \{o_j | o_j \in Area_i\}$. Given a system area $\mathcal{S}.Area$ and a continuous stream of *aggregate locations* ($Area$, N) reported from a set of sensor nodes, we build an ***adaptive spatio-temporal histogram*** embedded inside an ***aggregate query processor*** that has the ability to answer aggregate query Q that asks about the number of objects in a certain area $Q.Area \in \mathcal{S}.Area$.

Wireless sensor network. In our system, we consider stationary wireless sensor nodes. Each sensor node has only the capacity to report aggregate locations to a server that contains the *aggregate query processor*. The communication between the sensor nodes and the server is through a distributed tree [19]. To construct the distributed tree, the server broadcasts a message to the network. Then, each sensor node records the neighbor that is closer to the server. As long as there is a communication link from sensor nodes to the server, we do not have any assumption about the network topology, i.e., our system could be deployed indoor for building monitoring or outdoor for field monitoring. The assumption that each sensor node is aware of its sensing area is realistic as some techniques have been proposed for sensing area modelling (e.g., [36]).

Aggregate query processor. The *aggregate query processor* is embedded in the server and is responsible for: (1) collecting *aggregate locations* from sensor nodes, (2) maintaining the proposed *adaptive spatio-temporal histogram* that estimates the distribution of moving objects within the system area, and (3) answering location-based aggregate queries based on the maintained histogram. Furthermore, the user can issue queries via either the sensor node or the server. The detail of the *aggregate query processor* and the histogram will be discussed in Section 4.

Resource-efficient sensor scheduler. We employ a *resource-efficient sensor scheduler* that aims to reduce the rate of aggregate location information sent from each sensor node to the server. The main idea is that instead of having all sensor nodes send their information to the server at each single time unit, we alternate among the sensor nodes in a round robin fashion. In this case, at each time unit, only a few of the sensor nodes report their aggregate location information to the query processor, so our *sensor scheduler* can save the sensor energy and network bandwidth.

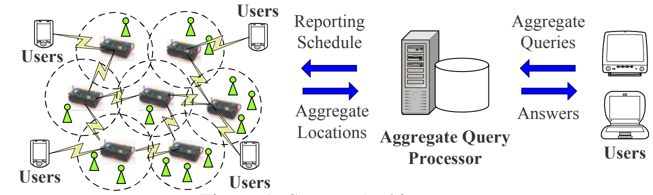


Figure 1: System Architecture

In particular, our *sensor scheduler* works as follows: Given a set of n sensor nodes deployed in the system area, we divide these n sensor nodes into p partitions where each partition includes n/p nodes. At every time unit t, one sensor node from each partition is selected to report its *aggregate location*. Within each partition, sensor nodes that send their aggregate location information to the server are selected in a round robin fashion such that a sensor node s_i would not send to the server two times before another sensor node s_j sends its information once. Thus, each sensor node needs to report its information every $(n/p) \times t$ time units. In this case, the system parameters p and t can be tuned to achieve a trade-off between query accuracy and the energy and bandwidth consumption of the network. A higher value of p or a lower value of t indicates a higher rate of information sent from the sensor nodes to the server, i.e., our *aggregate query processor*, and thus a better query accuracy, yet higher energy and bandwidth consumption.

It is important to note that the *resource-efficient sensor scheduler* is one of the main motivations behind developing our *adaptive spatio-temporal histogram*, which is the core of the *aggregate query processor*. If the scheduler is not there and all sensor nodes report their information at each time unit, then it will be trivial for the query processor to estimate the answer for aggregate queries without maintaining a histogram. The main idea is that the readings from all sensor nodes, which probably cover the entire system area, will be enough to give the actual distribution of moving objects in the system. However, such scenario is not friendly to a sensor network due to its extensive power and bandwidth consumption. Thus, upon deploying the *sensor scheduler*, we have to employ the *spatio-temporal histogram* to be able to model the distribution of moving objects even with less information sent from the sensor nodes.

4. Aggregate Location Monitoring

This section presents the proposed *aggregate location monitoring system* in wireless sensor networks. As has been outlined in Section 3, an *aggregate query processor* is embedded inside the monitoring server receiving a continuous stream of aggregate locations from sensor nodes in a form of ($Area$, N), where $Area$ is a monitored area (i.e., sensing area) and N is the number of detected objects within $Area$. At the core of the *aggregate query processor*, we propose an *adaptive spatio-temporal histogram* that estimates the distribution of moving objects in the system area. In this section, we start by proposing a *basic* spatio-temporal histogram that keeps track of the spatial and temporal features of the aggregate locations from the sensor nodes. As it is a *basic* one, it suffers from various drawbacks in terms of efficiency (i.e., overhead of maintaining the histogram) and accuracy (i.e., the ability to model the actual objet distribution). To overcome these drawbacks, we propose the *adaptive spatio-temporal histogram* that employs three techniques, namely, *memorization*, *locality awareness*, and *packing*, in which the *memorization* and *locality awareness* techniques mainly aim to enhance the histogram accuracy while the *packing* technique aims to improve the efficiency of maintaining the histogram. The rest of this section is organized as follows: Section 4.1 outlines the main data structure for histogram maintenance. The *basic* histogram is presented in Section 4.2. Section 4.3 discusses the three techniques applied to our *adaptive* histogram. Finally, Section 4.4 discusses the *aggregate query processor*.

4.1. Histogram Data Structures

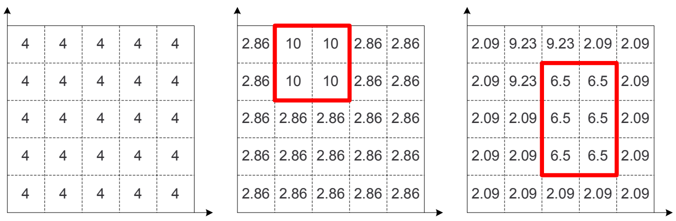
Our spatio-temporal histogram is represented by a two-dimensional array that models a grid structure $\mathcal{G}$ of r rows and c columns, i.e., a total of $r \times c$ disjoint grid cells. For each grid cell $\mathcal{G}[i, j]$, where $1 \leq i \leq r$ and $1 \leq j \leq c$, we maintain an *estimator* $\mathcal{H}[i, j]$ which is a float value representing the currently estimated number of objects lying in $\mathcal{G}[i, j]$. The grid structure $\mathcal{G}$ models the system space $\mathcal{S}$ in which there are $\mathcal{S}.N$ users in the system area $\mathcal{S}.Area$. It is important to note that $\mathcal{S}.N$ and $\mathcal{S}.Area$ are given to our query processor while r and c (i.e., the grid dimensions) are system tuning parameters. In practice, $\mathcal{S}.N$ can be computed online for both indoor and outdoor dynamic environments. For indoor environments, sensor nodes can be deployed at each entrance and exit to count the number of users entering or leaving the system [7], [9]. For outdoor environments, sensor nodes have been already used to count the number of people in a predefined area [8]. Thus, assuming the knowledge of $\mathcal{S}.N$ is a realistic assumption. In the rest of this paper, we assume that the sensor's sensing area (i.e., $R.Area$) aligns with the grid cell boundaries. The general case that the sensing area does not align with grid cells can be approximately reduced to our case by increasing the resolution of the grid structure (i.e., increasing the number of rows r and/or the number of columns c). The experimental evaluation of our system will consider this alignment error (Section 5).

4.2. Basic Histogram

Main idea. The *basic* histogram always assumes a uniform distribution of all objects within the sensor's monitored area and the system area $\mathcal{S}.Area$. Initially, the *basic* histogram assumes that the number of objects in the system area $\mathcal{S}.N$ is uniformly distributed over all grid cells. Once a sensor node reports an aggregate location, i.e., a monitored area $R.Area$ along with an object count of this area $R.N$, we update the *estimators* of all grid cells included in $R.Area$ by uniformly distributing $R.N$ over the included grid cells. The *estimators* of the grid cells that do not overlap with $R.Area$ is updated to reflect a uniform distribution of all objects that do not lie in the monitored area $R.Area$.
Histogram maintenance. Initially, the *estimator* $\mathcal{H}[i, j]$ of each grid cell is initialized uniformly as $\mathcal{H}[i, j] = \mathcal{S}.N/(r \times c)$ where $1 \leq i \leq r$ and $1 \leq j \leq c$. Once the *basic* histogram receives an aggregate location R=($R.Area, R.N$) from a senor node, it computes the sum of the current *estimators* within $R.Area$ as: $R.\widehat{N} = \sum_{\mathcal{G}(i,j) \in R.Area} \mathcal{H}[i, j]$. It is important to note that the difference between $R.N$ (the actual number of objects in $R.Area$) and $R.\widehat{N}$ (the estimated number of objects in $R.Area$) represents the current estimation error in the *basic* histogram. Then, the *estimators* $\mathcal{H}[i, j]$ of all grid cells within $R.Area$ is updated to be $R.N$ divided by the number of grid cells within $R.Area$, i.e., uniformly distribute the reported number of objects $R.N$ among all overlapped grid cells. For all other grid cells that do not overlap with $R.Area$, we increase or decrease their *estimators* by dividing the estimation error $R.\widehat{N} - R.N$ among all concerned grid cells. Formally, the *estimator* $\mathcal{H}[i, j]$ of each grid cell is computed as follows:

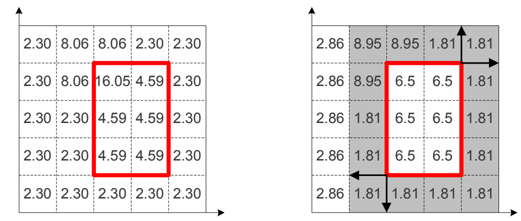
$$\mathcal{H}[i,j] = \begin{cases} \frac{R.N}{\#\text{ grid cells } \textit{inside } R.Area}, & \text{for } \mathcal{G}(i,j) \in R.Area \\ \mathcal{H}[i,j] + \frac{R.\widehat{N} - R.N}{\#\text{ grid cells } \textit{outside } R.Area}, & \text{for } \mathcal{G}(i,j) \notin R.Area \end{cases}$$

Example. Figure 2 gives an example of the initial *basic* histogram and the updated histogram for two aggregate locations R_1 and R_2. Figure 2a gives the initial *basic* histogram where the 100 objects are uniformly distributed over the 25 grid cells, i.e., each *estimator* is set to $100/25 = 4$. Figure 2b depicts the case where the *basic* histogram processes aggregate location R_1=($R_1.Area, R_1.N$=40), where $R_1.Area$ is depicted as a bold rectangle. Upon receiving R_1, we determine $R_1.\widehat{N} = 16$ as the sum of the current *estimators* in $R_1.Area$ in Figure 2a. Then, since there are only four grid cells within $R_1.Area$, the *estimators* of these cells are updated to $R_1.N/4 = 40/4 = 10$; i.e., a uniform distribution of $R_1.N$ among all grid cells in $R_1.Area$. Finally, the estimation error, i.e., $R_1.\widehat{N} - R_1.N = 16 - 40 = -24$, is uniformly absorbed by the grid cells outside $R_1.Area$ in which each *estimator* is added by $-24/21 = -1.14$, i.e., $4 + (-1.14) = 2.86$. Figure 2c gives the status of the *basic* histogram after processing R_2=($R_2.Area, R_2.N$=39).



(a) Initial histogram (b) $R_1.N = 40$ (c) $R_2.N = 39$

Figure 2: Initial histogram and *basic* histogram for R_1 and R_2



(a) Memorization technique for $R_2.N = 39$ (b) Locality-awareness technique for $R_2.N{=}39$

Figure 3: Memorization and locality-awareness techniques for R_2

$R_2.N$ is uniformly distributed to the six grid cells in $R_2.Area$, i.e., $39/6 = 6.5$, and the estimation error, i.e., $R_2.\widehat{N} - R_2.N = (2.86 \times 5 + 10) - 39 = -14.7$, is uniformly absorbed by the grid cells outside $R_2.Area$, i.e., the *estimator* of each grid cell outside $R_2.Area$ is added by $-14.7/19 = -0.77$.

4.3. Adaptive Spatio-Temporal Histogram

Although being simple and easy to implement, the *basic* histogram suffers from two major drawbacks that significantly deteriorate its efficiency and accuracy: (1) The strict assumption of uniformity degrades the histogram accuracy, especially for skewed data distributions. (2) Processing sensor readings one by one results in exhaustive computation that degrades the histogram efficiency. To overcome these drawbacks, we design an *adaptive spatio-temporal histogram* where we propose three techniques, *memorization*, *locality awareness*, and *packing*, that make use of the *spatial* and *temporal* features of the received aggregate locations to improve the histogram accuracy and efficiency. The *memorization* technique uses the *temporal* feature in aggregate locations where the user distribution within a short time period is similar. The *locality awareness* technique uses the *spatial* feature in aggregate locations where the change posed by an aggregate location should affect only the nearby grid cells which are indicated by the user mobility pattern. The *packing* technique uses the *spatial* feature in aggregate locations where processing independent aggregate locations at the same time reduces computational overhead. In the remainder of this section, we first describe the *memorization* and *locality-awareness* techniques, and then represent an integrated solution combining these two techniques together along with the *packing* technique.

4.3.1. Memorization Technique. Main idea. The *memorization* technique aims to avoid the uniform distribution assumption in the *basic* histogram by utilizing the *temporal* feature in the aggregate locations received from the sensor nodes. The main idea is that the user distribution would be similar within a short time period. For instance, a dense area cannot suddenly become a sparse area without any transition phenomena. Thus, the *memorization* technique improves the histogram accuracy by utilizing current *estimators* to predict the next value of these *estimators*. With this idea of *memorization*, once a sensor node reports an aggregate location, i.e., a monitored area $R.Area$ along with an object count in this area $R.N$, we distribute $R.N$ over all grid cells in $R.Area$ based on the ratio of their current *estimators* to the sum of the current *estimators* within $R.Area$ rather than a uniform distribution as in the *basic* histogram. Similarly, grid cells that do not overlap with $R.Area$ are updated based on their previous distribution.

Histogram maintenance. Similar to the *basic* histogram, we initially set the *estimator* $\mathcal{H}[i,j]$ of each grid cell to be $\mathcal{H}[i,j] = \mathcal{S}.N/(r \times c)$ where $1 \leq i \leq r$ and $1 \leq j \leq c$. Once we receive an aggregate location $R{=}(R.Area, R.N)$ from a senor node, we compute the sum of the current *estimators* within $R.Area$, $R.\widehat{N}$, as in the *basic* histogram. However, the main difference here is that instead of uniformly dividing $R.N$ by the number of grid cells in $R.Area$, we distribute $R.N$ over each cell based on its contribution to $R.\widehat{N}$, i.e., for each grid cell $\mathcal{G}[i,j] \in R$, the *estimator* is set to $R.N$ multiplied by the ratio $\mathcal{H}[i,j]/R.\widehat{N}$. Similarly, the current estimation error, i.e., $R.\widehat{N} - R.N$, is distributed over all grid cells that are outside $R.Area$ based on their contribution to the total estimated number of objects outside $R.Area$, i.e., $\mathcal{S}.N - R.\widehat{N}$. Formally, the *estimator* $\mathcal{H}[i,j]$ of each grid cell is computed as follows:

$$\mathcal{H}[i,j] = \begin{cases} \frac{R.N \times \mathcal{H}[i,j]}{R.\widehat{N}}, & \text{for } \mathcal{G}(i,j) \in R.Area; \\ \mathcal{H}[i,j] + \frac{(R.\widehat{N} - R.N) \times \mathcal{H}[i,j]}{\mathcal{S}.N - R.\widehat{N}}, & \text{for } \mathcal{G}(i,j) \notin R.Area. \end{cases}$$

When $R.\widehat{N} = 0$, $R.N$ is uniformly distributed among the grid cells within $R.Area$. If some *estimators* within $R.Area$ are zero, we add a small ϵ, e.g., one, to every grid cell within $R.Area$ for calculating the first part of the equation. This is because we cannot update any *estimators* with a zero value, i.e., their weights are zero.

Example. Figures 2a and 2b give the initial histogram and the updated histogram after processing R_1, respectively. So far, the *memorization* technique has the same effect as that of the *basic* histogram. Figure 3a gives the updated histogram after processing $R_2{=}(R_2.Area, R_2.N{=}39)$. Similar to the *basic* histogram, we compute the sum of the *estimators* inside $R_2.Area$ as $R_2.\widehat{N} = 2.86 \times 5 + 10 = 24.3$. Since the top-right grid cell within $R_2.Area$ has a larger weight, i.e., $10/24.3$, we distribute a larger portion of $R_2.N$ to it, i.e., $39 \times 10/24.3 = 16.05$. The weight of other grid cells in $R_2.N$ is $2.86/24.3$, so their *estimators* are set to $39 \times 2.86/24.3 = 4.59$. The estimation error, i.e., $R_2.\widehat{N} - R_2.N = 24.3 - 39 = -14.7$, is absorbed by the grid cells outside $R_2.Area$ in proportional to their current *estimators*. The sum of the *estimators* outside $R_2.Area$ is the estimation error between $\mathcal{S}.N$ and $R_2.\widehat{N}$, i.e., $100 - 24.3 = 75.7$. The grid cell with a larger *estimator* absorbs a larger potion of the difference, so the *estimators* with a value of 10 is set to $10 + (-14.7 \times 10/75.7) = 8.06$. The *estimators* of other grid cells are set to $2.86 + (-14.7 \times 2.86/75.7) = 2.30$.

4.3.2. Locality Awareness Technique. Main idea. A distinguishing feature about *locality awareness* is that we do not update the *estimators* of all grid cells for each received aggregate location. Instead, only those cells that may be affected by an aggregate location are updated. We determine the *affected* cells based on discovering the *mobility pattern* of moving objects within the grid cells. Initially, we assume that the *affected area* of every aggregate location update is the entire system. The histogram will be in a steady state after it gets aggregate locations from all sensor nodes. Once the histogram becomes steady, it uses the *locality awareness* technique for forthcoming aggregate locations. For each forthcoming aggregate location R, we initially determine an *affected distance* (i.e., spatial locality) of R. Assuming the knowledge of the maximum movement speed of the moving objects, max_{speed}, we calculate the *affected distance* as $(t_{current} - s.t_{last}) \times max_{speed}$, where $t_{current}$ is current time and $s.t_{last}$ is the last update timestamp of sensor node s. Then, we expand $R.Area$ to the *affected distance*. The expanded area constitutes the *affected area* of R, $R.\mathcal{A}$. This means that any grid cell outside $R.\mathcal{A}$ should not be influenced by R.

Histogram maintenance. After we determine the *affected area* of an aggregate location R, $R.N$ is uniformly distributed among the grid cells within $R.Area$ as in the *basic* histogram. The estimation error, i.e., $R.\widehat{N} - R.N$, is uniformly absorbed by the grid cells within $R.\mathcal{A}$. All grid cells outside $R.\mathcal{A}$ are not affected. Formally, the *estimator* $\mathcal{H}[i, j]$ of each grid cell is computed as follows:

$$\mathcal{H}[i,j] = \begin{cases} \frac{R.N}{\text{\# grid cells within } R.Area}, & \text{for } \mathcal{G}(i,j) \in R.Area \\ \mathcal{H}[i,j] + \frac{R.\widehat{N} - R.N}{\text{\# grid cells in } R.\mathcal{A}}, & \text{for } \mathcal{G}(i,j) \notin R \wedge \in R.\mathcal{A} \\ \mathcal{H}[i,j], & \text{for } \mathcal{G}(i,j) \notin R \wedge \notin R.\mathcal{A} \end{cases}$$

Example. Figure 3b depicts the updated histogram after the *basic* histogram processes aggregate location $R_2{=}(R_2.Area, R_2.N{=}39)$ on the histogram depicted in Figure 2b. In this example, $R_2.Area$ is represented as a solid rectangle, the arrows represent the *affected distance* of R_2, and the *affected area* of R_2, $R_2.\mathcal{A}$, is represented as a shaded area. Similar to the *basic* histogram, $R_2.N$ is uniformly distributed among the grid cells within $R_2.Area$, so each *estimator* within $R_2.Area$ is set to $39/6 = 6.5$. The estimation error, i.e., $R_3.\widehat{N} - R_3.N = (2.86 \times 5 + 10) - 39 = -14.7$, is uniformly absorbed by the grid cells within $R_2.\mathcal{A}$. Each *estimator* of the 14 grid cells within $R_2.\mathcal{A}$ is added by $-14.7/14 = -1.05$. The *estimators* outside $R_2.\mathcal{A}$ are not affected by R_2.

4.3.3. Integrated Solution with Packing Technique. In this section, we first present the main idea of the *packing* technique, and then give the algorithm of the integrated solution that combines the three proposed techniques, *memorization*, *locality-awareness*, and *packing* techniques together for our *adaptive spatio-temporal histogram*.

Packing technique. This technique enables the histogram to process a set of spatially independent aggregate locations at the same time. A set of aggregate locations is spatially independent if their *affected areas*, computed by the *locality awareness* technique, do not overlap with each other. The key advantage of the *packing* technique is to reduce computational overhead, as we update the histogram once for a set of spatially independent aggregate locations.

Integrated Solution. Algorithm 1 depicts the pseudo code of the integrated solution that combines the three proposed techniques together to maintain the *adaptive* histogram. The input to the algorithm is the histogram and a set of of p aggregate locations $\mathcal{R} = \{R_1, \ldots, R_p\}$ during a scheduled time interval t, i.e., one sensor node reports an aggregate location from each partition. The algorithm employs one technique at each step.

Step 1: Locality-awareness step. This step finds the spatial independence among the aggregate locations in $\mathcal{R}$. For each aggregate location R in $\mathcal{R}$, we determine the *affected area* $R.\mathcal{A}$ of R (Line 3 in Algorithm 1). If the *affected areas* of a set of aggregate locations do not overlap, these aggregate locations are spatially independent.

Step 2: Packing step. The input of this step is the set of aggregate locations with their *affected areas* in $\mathcal{R}$. The output of this step is a set of groups $\mathcal{G} = \{G_1, \ldots, G_m\}$ in which each group G contains a set of spatially independent aggregate locations (Line 4 in Algorithm 1). For each group G_i in $\mathcal{G}$, we maintain a bit-vector V_{G_i} where all bits are initially set to zero. For each aggregate location R in $\mathcal{R}$, we create a bit-vector V_R where we set the bit corresponding to each grid cell in $R.Area$ or *affected area* $R.\mathcal{A}$. Then, we check if R can be added to one of exiting groups G_i in $\mathcal{G}$ by a bitwise AND operation between V_R and V_{G_i}. If the result is zero, R is spatially independent with other aggregate locations in G_i. Thus, R is added to that group and we update V_{G_i} by a bitwise OR operation between V_R and V_{G_i}. Otherwise, we repeat this checking with other groups. If no suitable group is found, we create a new group for R, and then add the new group to $\mathcal{G}$.

Step 3: Memorization step. The input of this step is a set of groups $\mathcal{G} = \{G_1, \ldots, G_m\}$ constructed by the previous step. For each group G_i of spatially independent

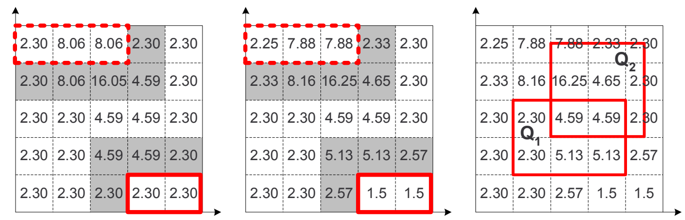
Algorithm 1 Adaptive Histogram Maintenance

1: AdaptiveHistogram ($\mathcal{H}$, $\mathcal{R}$)
2: $\mathcal{H}$ is the histogram, and $\mathcal{R} = \{R_1, \ldots, R_p\}$ is a set of aggregate locations;
3: Determine an affected area $R_i.\mathcal{A}$ for every $R_i \in \mathcal{R}$; **(Step 1)**
4: Partition the aggregate locations in $\mathcal{R}$ into a set of groups $\mathcal{G} = \{G_1, \ldots, G_m\}$ such that two readings R_i and R_j can be in the same group G_i only if their affected areas do not overlap, i.e., $R_i.\mathcal{A} \cap R_j.\mathcal{A} = \emptyset$, and $i \neq j$; **(Step 2)**
5: **for** Each group G_i in $\mathcal{G}$ **do** **(Step 3)**
6:     **for** Each aggregate location R_k in G_i **do**
7:         For $\mathcal{G}(i,j) \in R_k.Area$, $\mathcal{H}[i,j] \leftarrow \frac{R_k.N \times \mathcal{H}[i,j]}{R_k.\widehat{N}}$;
8:         For $\mathcal{G}(i,j) \in R_k.\mathcal{A}$, $\mathcal{H}[i,j] \leftarrow \mathcal{H}[i,j] + \frac{(R_k.\widehat{N} - R_k.N) \times \mathcal{H}[i,j]}{\sum_{\mathcal{G}(i,j) \in R_k.\mathcal{A}} \mathcal{H}[i,j]}$;
9:     **end for**
10: **end for**

aggregate locations R, we process each R=($R.Area, R.N$) by distributing $R.N$ among the grid cells within $R.Area$ in proportional to the value of their *estimators*. The new values of these *estimators* are computed by the equation given in Line 7 of Algorithm 1. On the other side, the estimation error, i.e., $R.\widehat{N} - R.N$, is distributed among the grid cells within its R's *affected area* $R.\mathcal{A}$ in proportional to the value of their *estimators*. These *estimators* are adjusted by the equation given in Line 8 of Algorithm 1.

Example. Figures 4a and 4b depict an example of our *adaptive* histogram. Figure 4a gives the updated histogram after processing aggregate locations R_1 and R_2. Then, we receive two new aggregate locations R_3=($R_3.Area$,$R_3.N$=3) and R_4=($R_4.Area$,$R_4.N$=18), where $R_3.Area$ and $R_4.Area$ are represented as solid and dotted rectangles, respectively. First, the *locality-awareness* step computes the *affected area* for R_3 and R_4 where their *affected areas* are represented as shaded grid cells. Since the *affected areas* of R_3 and R_4 do not overlap, i.e., they are spatially independent, the *packing* step puts them into the same group, i.e., $G = \{R_3, R_4\}$. Finally, the *memorization* step processes G. For R_3, the sum of the *estimators* within $R_3.Area$ is $R_3.\widehat{N} = 2.3 + 2.3 = 4.6$. Since the values of the *estimators* within $R_3.Area$ are the same, these *estimators* are set to $3 \times 2.3/4.6 = 1.5$. The estimation error, i.e., $R_3.\widehat{N} - R_3.N = 4.6 - 3 = 1.6$, is added to the *estimators* within R_3's affected area $R_3.\mathcal{A}$ in proportional to the value of their *estimators*. The sum of the *estimators* within $R_3.\mathcal{A}$ is $2.3 \times 2 + 4.59 \times 2 = 13.78$. Hence, the *estimators* with a value of 2.3 are updated to $2.3 + 1.6 \times 2.3/13.78 = 2.57$ while the *estimators* with a value of 4.59 are updated to $4.59 + 1.6 \times 4.59/13.78 = 5.13$. Similarly, we update R_4 to the histogram accordingly. Figure 3b depicts the updated histogram after processing R_3 and R_4.

Cost Analysis. We analyze the update cost of both the *basic* and *adaptive* histograms for a scheduled time interval t during which p aggregate locations are reported from sensor nodes, i.e., one sensor node from each partition reports its aggregate location. The *basic* histogram updates the *estimator* of all grid cells for each aggregate location R, the update cost is $p \times r \times c$. Then, we consider the *adaptive* histogram. Let N_A be the average number of affected cells



(a) Locality-awareness and packing steps (b) Memorization step (c) Aggregate queries Q_1 and Q_2

Figure 4: Adaptive histogram and aggregate query processing

of R, i.e., R's affected cells are the grid cells within its monitored area $R.Area$ or *affected area* $R.\mathcal{A}$. For each R, the *locality-awareness* steps computes an *affected area*, the *packing* step computes a bit-vector and performs at most m bit-wise comparisons (m is the number of groups constructed by this step), and the *memorization* step updates the *estimator* of every affected cell. Thus, the total cost is $O(p(1 + N_A + m + N_A))$. The *adaptive* histogram performs better than the *basic* one when $r \times c > 2N_A + m + 1$, where in general, $r \times c$ is much larger than N_A and m. Experimental results show that the *adaptive* histogram outperforms the *basic* one by two orders of magnitude.

4.4. Aggregate Query Processing

Having our spatio-temporal histogram makes the job of the *aggregate query processor* trivial. Our focus is on aggregate monitoring queries where the query is concerned about the number of persons or objects in a certain region. Figure 4c depicts two aggregate monitoring queries Q_1 and Q_2. The query answer is the number of objects within the query region in the histogram. For queries aligning to grid cells, the query answer is simply the sum of all *estimators* within the query region. For example, the answer of Q_1 is $(2.3 + 4.59 + 5.13) \times 2 = 24.04$. On the other hand, if the query is not aligned to grid cells, we calculate the portion of each *estimator* that contributes to the query answer as the ratio of the grid cell area that overlaps with the query region to the grid cell area. For example, the answer of Q_2 is $4.59 \times 2 + 4.65 + 16.25 + (7.88 + 2.33 + 2.3 \times 2) \times 1/2 + 2.3 \times 1/4 = 38.06$. It is important to note that the quality of the answer of the aggregate query solely depends on the accuracy of the histogram. This issue will be discussed in the experiment section.

5. Experiment Results

In this section, we experimentally evaluate the performance of our *adaptive spatio-temporal histogram* (denoted as Adaptive) with respect to histogram accuracy (i.e., error in query answers), histogram efficiency (i.e., histogram update time), and the effectiveness of our *sensor scheduler*.

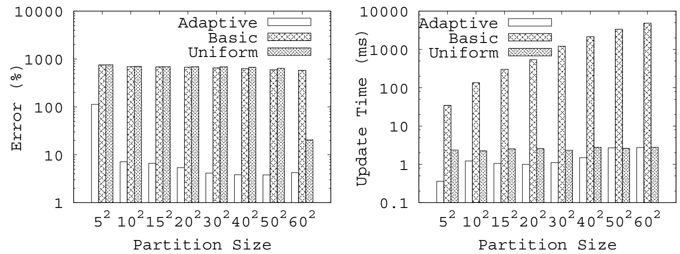
Baseline histograms. We compare the Adaptive histogram with two baseline histograms, *basic* histogram described in Section 4.2 (denoted as Basic) and an existing *spatio-temporal histogram* [20] (denoted as Uniform). We choose the Uniform histogram to compare with the proposed Adaptive histogram because it is the closest work to ours. Similar to Basic, Uniform also assumes a uniform distribution. However, unlike the Basic histogram, Uniform processes multiple aggregate locations at the same time. Given a set of aggregate locations, Uniform considers the grid cells covered by some aggregate locations as *light* grid cells, while the grid cells that are not covered by any aggregate location are considered as *dark* grid cells. For each aggregate location R, the object count $R.N$ is uniformly distributed among the grid cells within $R.Area$. The estimation error, i.e., $R.\widehat{N} - R.N$, where $R.\widehat{N}$ is the sum of the *estimators* within $R.Area$, of all aggregate locations is uniformly absorbed by the *dark* grid cells.

Mobility models. We consider two mobility models, *skewed* and *uniform*. In the *skewed* mobility model, we assign several non-overlapping circular hot spots (default is five), where each hot spot has an area of 10% of the system space. Moving objects are uniformly assigned to hot spots. Each hot spot is divided into four zones with different densities. These zones are defined by the distance from the center of the hot spot, i.e., 40%, 30%, 20% and 10% objects are moving within the 40%, 70%, 90%, and 100% distance of the radius of the hot spot. The moving objects are freely roaming among the zones. This *skewed* mobility pattern models real scenarios. For example, in a campus, classrooms, cafeteria and libraries are hot spots [37]. In the *uniform* mobility model (i.e., the number of hot spots is zero), we use the random way point model [38] in which the moving objects are freely roaming within the system space.

Parameter settings. In all experiments, the communication range of each sensor node is 100 units, and its sensing range is 50 units. Each sensor node has 20% sensing area overlapping with other sensor nodes. The network size constitutes the system space. All experiments run for 3,600 time units. Unless mentioned otherwise, the experiments consider 10,000 moving objects in a system space of 3,600 sensor nodes. The mobility speed is assigned uniformly within the range $[0, 20]$ space unit/time unit. The grid size of the histogram is set to 500×500 (i.e., $r = c = 500$). The whole network is divided into $p = 400$ partitions. At every t time units, one sensor node from each partition reports its aggregate location to the server. After the system is steady, i.e., each sensor node reports its aggregate location once, we issue 100 aggregate queries of a query region selected uniformly within the range $[1, 500]^2$ grid cells every time unit t. The accuracy of a histogram is measured in terms of the error in the query answer to the actual answer. Let M, E, and A be the total number of issued queries, the query answer, and actual answer, respectively. The error is measured as $\frac{1}{M}\sum_{i=1}^{M}\frac{|E_i - A_i|}{A_i}$. In the experiments, when $A_i = 0$, we consider the error as E_i.

Table 1: Partition Size (Number of Bytes (K)/Time Unit)

Partition Size	5^2	10^2	15^2	20^2	30^2	40^2	50^2	60^2
Bytes (K)	16	66	148	264	594	1065	1672	2376



(a) Estimation Error (b) Histogram Update Tim

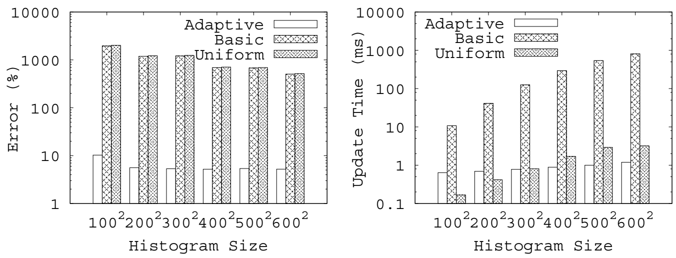
Figure 5: Partition Size

5.1. Effect of Partition Size (Sensor Scheduler)

Figure 5 depicts the performance of the histograms with respect to varying the number of partitions (p) from 25 to 3,600. Table 1 gives that the number of bytes (K) per time unit sent by all sensor nodes significantly increases when the partition size gets larger. This is because if the *sensor scheduler* increases the partition size, there are more sensor nodes reporting their aggregate locations every time unit t. Figure 5a shows that the accuracy of our Adaptive histogram improves when the partition size increases. Also, the histogram accuracy of Adaptive has two orders of magnitude better than Basic and Uniform in most cases. The update time of the Adaptive and Basic histograms increases when the partition size gets larger, while Uniform is not sensitive to the partition size (Figure 5b). This is because when there are more aggregate locations for each update, the Basic histogram updates the *estimator* of every grid cell once for each aggregate location, while Adaptive is likely to deal with more groups of spatially independent aggregate locations. Our Adaptive histogram effectively improves the update time of Basic by two orders of magnitude in most case or even three orders of magnitude when the partition size is larger than 30^2.

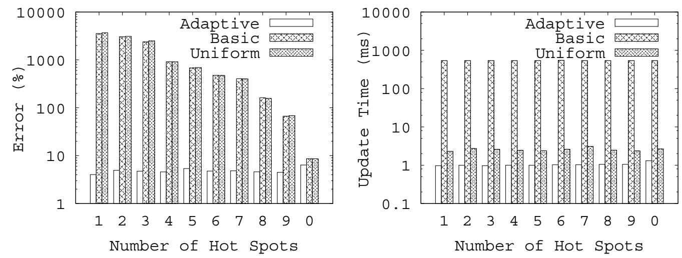
5.2. Effect of Histogram Size

Figure 6 gives the performance of the histograms with the increase of the number of grid cells from 100^2 to 600^2. Although increasing the histogram size results in better accuracy (Figure 6a), a histogram with more grid cells incurs higher update cost (Figure 6b). The Adaptive histogram improves the accuracy in query answers over Basic and Uniform by two orders of magnitude in most cases. Furthermore, the increase rate of the update time of the Adaptive histogram is slower than the Basic and Uniform histogram. This is because the spatial dependence among aggregate locations reduces (i.e., the number of groups constructed by the *packing* step reduces) as the histogram size gets larger. The update cost of our Adaptive histogram is better than Uniform when the histogram size is larger than 300^2. The



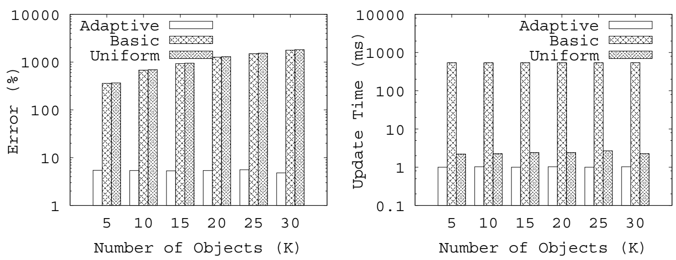
(a) Estimation Error (b) Histogram Update Time

Figure 6: Histogram Size



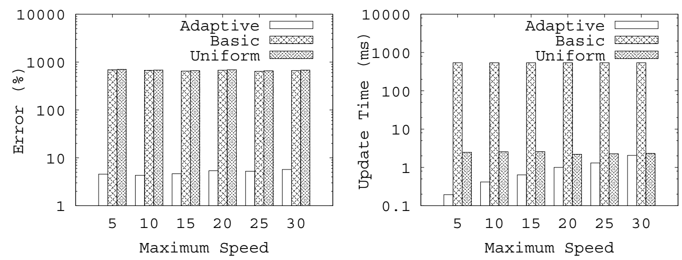
(a) Estimation Error (b) Histogram Update Tim

Figure 8: Mobility Patterns



(a) Estimation Error (b) Histogram Update Tim

Figure 7: Number of Moving Objects



(a) Estimation Error (b) Histogram Update Tim

Figure 9: Mobility Speeds

idea is that the Uniform histogram updates the *estimator* of every grid cell at least once for each update. On the other hand, our Adaptive histogram updates only the grid cells within the monitored area or *affected area* of aggregate locations. When the histogram size gets larger, our Adaptive histogram updates fewer grid cells for each update.

5.3. Effect of Number of Moving Objects

Figure 7 depicts the performance of the histograms with respect to varying number of moving objects from 5,000 to 30,000. Since increasing the number of mobile objects does not affect the *sensor scheduler*, the update time is not influenced by the number of objects (Figure 7b). The result shows that the performance of our Adaptive histogram is not sensitive to the number of objects (Figure 7a). However, the accuracy of Basic and Uniform degrades significantly when there are more objects. This deterioration is mainly posed by *unbalanced* aggregate locations which are reported by the sensor nodes whose sensing area across the grid cells with different object densities. Since both the Basic and Uniform histograms rely on a uniform distribution, these two histograms would overestimate and underestimate the *estimators* of the sparser and denser grid cells within the *unbalanced* aggregate locations, respectively. On the other hand, our Adaptive histogram considers historic data while updating the *unbalanced* aggregate locations. This means that a larger portion of the count object of the *unbalance* aggregate locations is absorbed by the denser grid cells. Thus, *unbalanced* aggregate locations have much less influence on our Adaptive histogram than Basic and Uniform.

5.4. Effect of Mobility Patterns

Figure 8 depicts the effect of mobility patterns on the histograms with various numbers of hot spots, i.e., from a very skewed mobility pattern (the number of hot spots is one) to a uniform one (the number of hot spots is zero). Similar to the previous experiment, the update cost is not affected by the mobility pattern (Figure 8b). The result shows that our Adaptive histogram adapts well to both the skewed and uniform mobility patterns. Since Basic and Uniform are designed with a uniform mobility pattern in mind, the accuracy of these two histograms improves as the mobility pattern becomes less skewed (Figure 8a). It is interesting to see that although Adaptive does not assume a uniform distribution of moving objects, it also performs better than Basic and Uniform in the uniform mobility pattern.

Figure 9 gives the performance of the histograms with respect to varying the maximum mobility speed from 5 to 30 space unit/time unit (the minimum mobility speed is zero). The result shows that the accuracy of all approaches is not sensitive to the mobility speed (Figure 9a). Also, the update time of Basic and Uniform is not sensitive to the mobility speed (Figure 9b). However, the update time of our Adaptive histogram increases when the mobility speed gets larger. The main reason is that the *affected area* of the aggregate locations becomes larger when the mobility speed increases. This means that the spatial dependence among the aggregate locations increases, so the *packing* step constructs more groups of aggregate locations. Since the update cost of the Adaptive histogram is affected by the number of grid cells within the *affected area* of the aggregate locations, its update cost increases when the *affected area* gets larger.

6. Conclusion

In this paper, we have proposed an aggregate location monitoring system in a wireless sensor network. The underlying environment consists of counting sensors that are only capable of reporting *aggregate locations*, i.e., their sensing areas along with the number of detected objects residing therein, to a query processor. At the core of the query processor, we propose an *adaptive spatio-temporal histogram* that models the distribution of moving objects and answers aggregate monitoring queries based on *aggregate locations*. Furthermore, we propose three techniques, *memorization*, *locality awareness*, and *packing*, that are combined together to enhance the histogram accuracy and efficiency by exploiting both the *spatial* and *temporal* features in *aggregate locations*. A distinct feature in the proposed histogram is that it is designed with the wireless sensor network in mind. Thus, it takes care of the limitations of wireless sensor networks that include limited power and communication bandwidth. Experimental results show that the *adaptive* histogram succeeds in giving highly accurate answers and clearly outperforms a *basic* histogram and the state-of-the-art spatio-temporal histogram [20] by orders of magnitudes, especially, for the cases of skewed data distributions.

References

[1] A. Ward, A. Jones, and A. Hopper, "A New Location Technique for the Active Office," *IEEE Personnel Communications*, vol. 4, no. 5, pp. 42–47, 1997.

[2] Sonitor Technologies Inc., "Healthcare Indoor Positioning Systems. http://www.sonitor.com/downloads/files/Health_Brochure_2007.pdf."

[3] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, and T. F. Abdelzaher, "VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance," *TOSN*, vol. 2, no. 1, pp. 1–38, 2006.

[4] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The Anatomy of a Context-Aware Application," *MOBICOM*, 1999.

[5] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The Active Badge Location System," *TOIS*, vol. 10, no. 1, pp. 91–102, 1992.

[6] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," *MOBICOM*, 2000.

[7] Onesystems Technologies, "Counting People in Buildings. http://www.onesystemstech.com.sg/index.php?option=com_content&task=view%&id=10."

[8] B. Son, S. Shin, J. Kim, and Y. Her, "Implementation of the Real-Time People Counting System using Wireless Sensor Networks," *IJMUE*, vol. 2, no. 2, pp. 63–80, 2007.

[9] Traf-Sys Inc., "People Counting Systems. http://www.trafsys.com/products/people-counters/thermal-sensor.aspx."

[10] Y. Cai, K. A. Hua, and G. Cao, "Processing Range-Monitoring Queries on Heterogeneous Mobile Objects," *MDM*, 2004.

[11] C. S. Jensen, D. Lin, B. C. Ooi, and R. Zhang, "Effective Density Queries of Continuously Moving Objects," *ICDE*, 2006.

[12] M. F. Mokbel, X. Xiong, and W. G. Aref, "SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases," *SIGMOD*, 2004.

[13] M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald, "Privacy-Aware Location Sensor Networks," *USENIX HOTOS*, 2003.

[14] G. Kaupins and R. Minch, "Legal and Ethical Implications of Employee Location Monitoring," *HICSS*, 2005.

[15] H. Hu, J. Xu, and D. L. Lee, "A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects," *SIGMOD*, 2005.

[16] K. Mouratidis, D. Papadias, and M. Hadjieleftheriou, "Conceptual Partitioning: An Efficient Method for Continuous Nearest Neighbor Monitoring," *SIGMOD*, 2005.

[17] Y. Tao, D. Papadias, and Q. Shen, "Continuous Nearest Neighbor Search," in *VLDB*, 2002.

[18] X. Xiong, M. F. Mokbel, and W. G. Aref, "SEA-CNN: Scalable Processing of Continuous K-Nearest Neighbor Queries in Spatio-temporal Databases," *ICDE*, 2005.

[19] D. Culler and M. S. Deborah Estrin, "Overview of Sensor Networks," *IEEE Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[20] H. G. Elmongui, M. F. Mokbel, and W. G. Aref, "Spatio-temporal Histograms," *SSTD*, 2005.

[21] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks using Probabilistic Models," *ICDE*, 2006.

[22] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," *VLDB*, 2004.

[23] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," *TODS*, vol. 30, no. 1, pp. 122–173, 2005.

[24] A. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang, "A Sampling-Based Approach to Optimizing Top-k Queries in Sensor Networks," *ICDE*, 2006.

[25] A. Silberstein, R. Braynard, and J. Yang, "Constraint Chaining: On Energy-Efficient Continuous Monitoring in Sensor Networks," *SIGMOD*, 2006.

[26] T.-Y. Fu, W.-C. Peng, and W.-C. Lee, "Optimizing Parallel Itineraries for KNN Query Processing in Wireless Sensor Networks," *CIKM*, 2007.

[27] S.-H. Wu, K.-T. Chuang, C.-M. Chen, and M.-S. Chen, "DIKNN: An Itinerary-based KNN Query Processing Algorithm for Mobile Sensor Networks," *ICDE*, 2007.

[28] X. Yang, H.-B. Lim, M. T. Ozsu, and K.-L. Tan, "In-Network Execution of Monitoring Queries in Sensor Networks," *SIGMOD*, 2007.

[29] P. Flajolet and G. N. Martin, "Probabilistic Counting Algorithms for Data Base Applications," *JCSS*, vol. 31, no. 2, pp. 182–209, 1985.

[30] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias, "Spatio-Temporal Aggregation Using Sketches," *ICDE*, 2004.

[31] Y.-J. Choi and C.-W. Chung, "Selectivity Estimation for Spatio-temporal Queries to Moving Objects," *SIGMOD*, 2002.

[32] Y. Tao, D. Papadias, and J. Sun, "The TPR*-Tree: An Optimized Spatio-temporal Access Method for Predictive Queries," *VLDB*, 2003.

[33] M. Hadjieleftheriou, G. Kollios, and V. J. Tsotras, "Performance Evaluation of Spatio-temporal Selectivity Estimation Techniques," *SSDBM*, 2003.

[34] Q. Zhang and X. Lin, "Clustering Moving Objects for Spatio-temporal Selectivity Estimation," *ACD*, 2004.

[35] Y. Tao, D. Papadias, J. Zhai, and Q. Li, "Venn Sampling: A Novel Prediction Technique for Moving Objects," *ICDE*, 2005.

[36] J. Hwang, T. He, and Y. Kim, "Exploring In-Situ Sensing Irregularity in Wireless Sensor Networks," *SENSYS*, 2007.

[37] W. jen Hsu, K. Merchant, H. wei Shu, and C. hsin Hsu, "Weighted Waypoint Mobility Model and its Impact on Ad Hoc Networks," *MC2R*, vol. 9, no. 1, pp. 59–63, 2005.

[38] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," *MOBICOM*, 1998.