A Crowd-sourced Method for Real-time Detection and Localization of Unexpected Events

Taishi Yamamoto

Graduate School of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
Email: is0145rp@ed.ritsumei.ac.jp

Kenta Oku, Kyoji Kawagoe

College of Information Science and Engineering
Ritsumeikan University
Shiga, Japan
Email: {oku@fc, kawagoe@is}.ritsumei.ac.jp

***Abstract*—In this paper, a Real-Time Smart and Quick Unexpected-Event Detect (RT-SQUED) method is proposed to detect and localize unexpected events, such as traffic accidents, in real-time, assuming crowd sourcing with smartphone devices. The authors previously proposed the SQUED, a crowd-sourced system for detection and localization of unexpected events from smartphone-sensor data. When SQUED users find an event, they point their smartphones toward a direction of the event. SQUED can detect the location and the time of the event using the smartphone-sensor data. However, the SQUED method is difficult to scale because of its computational cost problem. Therefore, it becomes difficult to detect events in real-time. Our new method, called RT-SQUED, solves this problem using concurrent two-phase processing. RT-SQUED is composed of rough and concrete processing phases. In this paper, effectiveness of RT-SQUED is also discussed.**

Keywords–event detection; smart phones; crowd-sourced system; global positioning system; sensor data.

I. INTRODUCTION

Recently, event detection using sensors, such as cameras or microphones, has attracted much attention owing to the spreading of such sensors. Moreover, many event-detection methods using setting-type and special-purpose sensors have been developed [1][2]. The popularization of smartphones is changing how events are detected, because a smartphone is equipped with many kinds of sensor devices, e.g., a camera, a microphone, a GPS, and an accelerometer. A special sensor does not need to be installed in a specific location in advance, because almost everyone possesses a smartphone. Although a smartphone sensor can be used for event detection, a crucial problem remains. Sensor data contains many errors because of its low measurement accuracy and the human's irregular hand movements.

We previously proposed a Smart and Quick Unexpected-Event Detector, SQUED, to detect unexpected events using built-in smartphone devices [3]. SQUED gathers data from the smartphones of people near the location of an unexpected event, such as a traffic accident. Both a location from the GPS and the direction from a geomagnetic sensor are used to identify the actual event location. The main points of SQUED are 1) crowd sourcing and 2) a novel event-detection method, allowing it to estimate an accurate location even from inaccurate data. SQUED can detect an event even if only two people are near the event location. Moreover, the more people who are around the location, the more accurately SQUED can detect the location, using our event-detection method. Finally, it is available in blind spots, which are the biggest problem for pre-installed event-detection equipment.

However, SQUED method has the following two problems. First, it is difficult to expand the detection range. The wider the detection range is, the more the event detection interval decreases. Second, it is difficult to detect multiple events occurred simultaneously. These problems are caused by high computational cost in detecting events.

In this paper, we propose an improved SQUED method, RT-SQUED, to be able to detect the event efficiently in a wider range. RT-SQUED performs two-phase processing, concurrently. In the first phase, RT-SQUED roughly divides its detection area into grid units for the event detection. Second, RT-SQUED counts the number of users whose eye's-view triangles overlapped in each grid. By these phases, it can recognize regions not have to search. Therefore, the computational cost can be reduced when the scaling of the detection range is allowed. In the second phase, RT-SQUED apply the original SQUED method to the grid in which the number of users whose eye's-view exceeds the threshold.

This paper is organized as follows. Section II discusses our previous work. The proposed event detection method is described in Section III. Section IV presents the results of evaluation and discussion of the proposed method. The related works are also explained in Section V. Finally, Section VI concludes this papers.

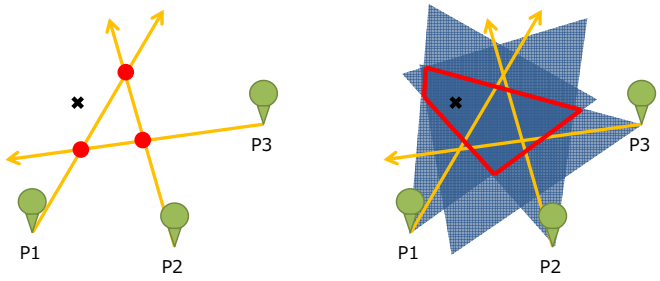
II. PREVIOUS WORK

A. Basic concept

In Figure 1(a), the "x" symbol indicates the actual location where an event occurred. Figure 1(a) shows three original directions obtained from the sensors of three pedestrians, P1, P2, and P3. Although each pedestrian tends to precisely point his/her smartphone toward the event location, an imprecise direction may be obtained from inaccurate sensor data, as shown in Figure 1(a). In Figure 1(a), the three direction lines do not intersect. Even for any pair of direction lines, the intersection is far from the actual event location.

The basic concept of our SQUED method is shown in Figure 1(b). In Figure 1(b), the eye's-view triangle introduced in our method is shown. The eye's-view triangle starts from the location of a pedestrian, with a pre-specified angle centering from the event direction estimated from the pedestrian's smartphone sensor. As in Figure 1(b), when an event occurs at location "x", a triangle for each pedestrian is constructed from

 ISBN: 978-1-61208-452-7



(a) Eyes direction (b) Eyes view

Figure 1. Eyes direction and eyes view

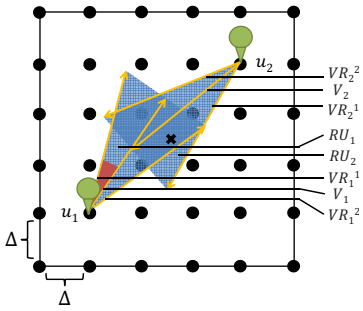


Figure 2. SQUED event detection method

the sensor data. Three triangles are checked in this case. The intersection of the three triangle regions is calculated. The red area, a parallelepiped represented by dashed lines, indicates the event location area, inside which the event likely occurred. With this introduction of an eye's-view triangle, the event-location area can be detected even when the observed sensor data contains noise, as in Figure 1(a). The more pedestrians who view an event and point a smartphone at it, the more precisely the location can be detected.

B. Definition

We define some symbols to describe our event detection method. R, a rectangular region, is pre-assigned as the event detection range. R is represented as a set of $\Delta \times n_x$ times $\Delta \times n_y$ grid points $\{P_{ij}\}$, $i = 1,..,n_x$, $j = 1,..,n_y$, which are disposed at equal intervals in both vertical and horizontal directions, as shown in Figure 2. Δ is the predefined interval between two adjacent grid points, in vertical and horizontal directions. The value of Δ is assumed to be having minimum accuracy on the detected event location. The left-bottom coordination point is (B_x,B_y).

Suppose there are users $U = \{u_k\}$, $k = 1,..,N_u$ pointing their smartphones towards an event. For each user u_k, his/her location and eye's-view direction are defined as L_k and V_k, respectively. We introduce the eye's-view triangle represented as RU_k for a user U_k. RU_k is calculated from the origin point L_k and two lines from L_k whose angles are VR_k^1 and VR_k^2, respectively, where VR_k^1=V_k-α and VR_k^2=V_k+α, α is a pre-defined parameter. The length of the two lines are fixed as H.

C. Method

For a given TU_k, we calculate the number of overlapped users, $DU_{i,j}$ for each grid point $P_{i,j}$ as follows: $DU_{i,j} = |U'_{i,j}|, where U'_{i,j} = \{u_l | P_{i,j} \in RU_l, u_l \in U\}$. We define the detected event region/location as E and the minimum number of overlapped users for event detection as MU. Then, the detected event region E is obtained as the following: $E = \{P_{i,j} | DU_{i,j} \geq MU\}$. The detected grid points of E are sorted with the value of $DU_{i,j}$ in descending order. Figure 2 shows an example of eye's-view triangles and an event-location region detected from the triangles. In this example, only two users found an event. This Figure has two eye's-view triangles. The intersection of the two regions is represented as a set of grid points, as shown in Figure 2. In our proposed method, we calculate the number of users whose eye's-view triangles overlap, rather than calculating a time-consuming intersection.

D. Problems of SQUED method

SQUED method has following problems: expansion of event detection range and detection of multiple events.

First, suppose that, we set Δ as 10 meters and the values of n_x and n_y as 1000. In this case, the number of grid points is 10000. For each of the grid points, SQUED calculate the number of overlapped users. As the number of users or $n_x \times n_y$ becomes large numbers, order of calculation will be larger. Therefore, it is difficult to expand the detection range.

Next, we assume that 10 events occur in a range of 5 km $\times$ 5 km at random. If the system can only search the range of 500 m $\times$ 500 m due to the above-mentioned problem, it is difficult to detect multiple events. In the case of expanding the event detection range, this problem becomes more serious.

Therefore, it is necessary to reduce the computational cost for event detection.

III. Proposed Event Detection Method

A. Basic concept

Figure 3 and Figure 4 show the rough processing phase, Figure 5 shows the concrete processing phase.

In the rough processing phase, it counts intersections between a rough grid dividing a detection range and a smallest rectangle surrounding the user triangle. In this example, the detection range is divided by four rough grids. Figure 3 shows the intersection detection in the left upper rough grid. Intersections in each rough grid are shown in Figure 4. Here, the rough grid which should detect in detail becomes only the left upper grid when the threshold is set to 2. Therefore, the concrete processing phase applies only to the left upper rough grid.

The concrete processing phase uses SQUED method which is mentioned above. In Figure 5, the part indicated as the red circle (the white circle in the gray scale) becomes the event detection point when the threshold is set to 2. Using this method, it can reduce the calculation cost.

B. Algorithms

Algorithm 1. EventDetection

```
1  EventDetection{
2  // RoughGridSize, ConcreteGridSize,
3  // DetectionRange, threshold are pre-defined.
4  // N-Interval, the number of concrete-phase
5  // processings in one rough-phase processing,
6  // is preseted.
7
```

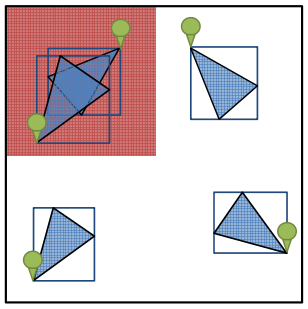


Figure 3. Rough processing phase

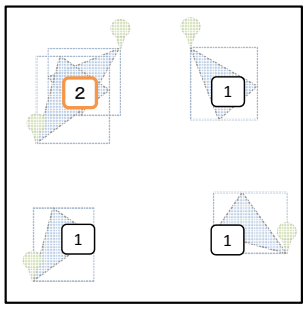


Figure 4. Rough processing phase 2

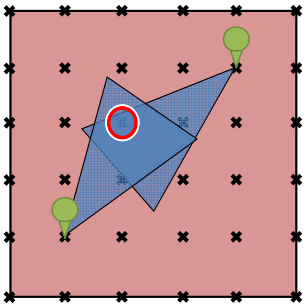


Figure 5. Concrete processing phase

```
8  //Initialization
9   RoughGrid <- CreateGrid(RoughGridSize,
10                 DetectionRange)
11  ConcreteGrid <- CreateGrid(ConcreteGridSize,
12                    DetectionRange)
13  //Main event detection process
14  While(){
15   If (Stop-condition is met) {Stop}
16   Obtain all USER data from Database and
17    set them to User.
18   // An element of User is a set of three
19   //  locations forming a triangle.
20   If (Counter exceeds N-Interval) {
21    DetectedRoughInfo <- RoughEventDetection
22                           (RoughGrid, User)
23    GridPointInfoUser <- GetConcreteGridPoint
24              (ConcreteGrid, DetectedRoughInfo)
25    Counter <- 0
26   }
27   DetectedResult <- ConcreteEventDetection
28                      (GridPointInfoUser)
29   Counter++
30   ShowEventsDetected(DetectedResult)
31  }
32 }
```

Algorithm 2. Rough & Concrete EventDetection

```
1  RoughEventDetection(RoughGrid, User) {
2   DetectedRoughInfo <- {}
3   For each grid in RoughGrid {
4    For each user in User {
5     MBR <- GetMinBoundingRect(user)
6     // user's MinimumBoundingRectangle is
7     //  calculated.
8     If (CheckIntersect(grid, MBR) is TRUE {
9      // it is checked if the MBR intersects
10     //  the grid or not.
11     Add (grid, user) to DetectedRoughInfo
12    }
13   }
14  }
15  Return(DetectedRoughInfo)
16 }
17 ConcreteEventDetection(GridPointInfoUser) {
18  DetectedResult <- {}
19  For each grid point as gridPoint
20   in GridPointInfoUser {
21   // UCounter is a counter to count the number
22   //  of users is included in the gridPoint.
23   UCounter <- 0
24   For each user in GridPointInfoUser {
25    If (CheckInclusion(gridPoint, user)) is TRUE {
26     // it is checked if the user is included
27     //  the gridPoint or not.
28     UCounter++
29    }
30   }
31   If(UCounter >= threshold) {
32    Add gridPoint to DetectedResult
33   }
34  }
35  Return(DetectedResult)
36 }
```

In the Algorithm 1, the main process, called EventDetection, is described.

Each user has the positional information of the three points forming a triangle. Three points consists of the location of the user and two points making up the eye's view is described in the previous section. A grid is a square composed of four coordinate grid points.

In EventDetection, first, two types of grid information are generated using RoughGrid and ConcreteGrid. CreateGrid is a function to create grids dividing from the DetectionRange at pre-defined size such as RoughGridSize. Then, RoughEventDetection, GetConcreteGridPoint and ConcreteEventDetection are iterated until the given stop condition is met, while RoughEventDetection and GetConcreteGridPoint are executed per preset N-interval. RoughEventDetection is a function to detect possible rough grids and users belonging to rough grids where a certain event may occur. GetConcreteGridPoint is a function to create a GridPointInfoUser, which is composed of a set of grid points and the users extracted for the grid point. The grid points are obtained by RoughEventDetection. The users for each grid point are extracted from all users by checking of intersection with each of grids and each of users. If a user intersects a grid, the user is extracted as belonging to the grid points, which are included in the grid. ConcreteEventDetection is a function to detect event concretely by using concrete grid points associated with users generated from the result of GetConcreteGridPoint. At the end of the loop, the function called ShowEventsDetected is executed to show the DetectedResult. In the ShowEventsDetected, detected result processed by the two functions of RoughEventDetection and ConcreteEventDetection. DetectedResult is composed of a set of grid points are presented to RT-SQUED system's users.

Algorithm 2 shows the details of our two phase event detection procedures.

RoughEventDetection is a function to perform in the first rough phase. In the RoughEventDetection, DetectedRoughInfo is first initialized to store the return value. Then, for each grid and each user the intersection between the grid and the user region is iteratively checked. In this check,

 ISBN: 978-1-61208-452-7

MinimumBoundingRectangle is created from the user triangle instead of the use of the user triangle due to rough checking. GetMinBoundingRect is a function to create a smallest rectangle, called MBR, surrounding the user triangle, which is described in the previous section. An MBR is created from the minimum and maximum values among three triangle-endpoints for each coordinate. After that, checking whether MBR intersected grid using CheckIntersect. CheckIntersect is a function to check if a MBR is intersected with a grid or not. This function uses the two coordinates of MBR and grid to check the intersection. If MBR is intersected with a grid, information on the corresponding the grid and the user is added to DetectedRoughInfo. Finally, RoughEventDetection return the DetectedRoughInfo.

ConcreteEventDetection is a function to perform the second concrete phase. In the ConcreteEventDetection, DetectedResult is first initialized to store the return value. Then, for each gridPoint and each user, it is checked whether the gridPoint is included in the user triangle, by using CheckInclusion. CheckInclusion is a function to check if the user triangle includes the gridPoint or not. This function uses the cross product vector for each of the sides of the user triangle. If a user triangle includes a gridPoint, UCounter is incremented. If the UCounter value is more than the given threshold, the corresponding gridPoints are added to DetectedResult. Finally, ConcreteEventDetection returns the DetectedResult.

IV. Evaluation & Discussions

We conducted an experiment to compare the accuracy and processing time with SQUED method and RT-SQUED. The data set that we used is one of the data set which we used in the case of the experiment of the previous paper.

TABLE I. SQUED method v.s. RT-SQUED

Method	Accuracy (F-measure)	Processing time (sec)
SQUED method	28.6%	31.67
RT-SQUED	28.6%	3.07

We set N-interval as 1 in this experiment. Table 1 shows the result of the experiment. The processing time of RT-SQUED became a one-tenth than that of SQUED method. In addition, the deterioration of the accuracy was not observed.

In our RT-SQUED, the proposed two phases, RoughEventDetection and ConcreteEventDetection, are concurrently executed. Through the experiment, we confirmed that the computational cost could be reduced and that event detection speed became faster. Moreover, the size of the detection range can be increased more. Therefore it can detect multiple events occurred simultaneously compared with SQUED method.

Furthermore, in RT-SQUED, N-interval in the first rough phase, the frequency of the RoughEventDetection execution, can be changed. This makes it possible to appropriately adjust the detection speed and detection accuracy, depending on the situation. With the ability of adjusting the frequency according to the event search range, real-time event detection can be realized.

V. Related Work

Wentao et al. proposed iSee, a detection system using the smartphone [4]. iSee uses two sensors, a GPS and a geomagnetic sensor. A user needs to swipe his/her smartphone screen toward the event location. iSee then acquires the user's location and the device direction to estimate the actual event direction using the swipe direction. Although the idea is similar to our SQUED, users need to swipe their smartphone screen. A SQUED user only needs to point his/her smartphone toward an event location. Moreover, the swipe direction contains much more noise than geomagnetic sensor data does.

Tran et al. proposed an algorithm that could recognize actions such as walking and running from videos [5]. It can detect events in a crowded video scene. In this approach, the point of intersection of the characteristic point of the object between each frame is extracted in order to recognize a movement trace. Wang et al. proposed an algorithm that could recognize other types of events, such as a parade and rock-climbing, from videos [6]. In their method, the local characteristic of the frame is replaced with a letter to detect an event more efficiently.

Tong Qin et al. proposed an crowdsourcing based event reporting system using smartphones with accurate localization and photo tamper detection [7]. In their system, it can identify the event summary and the event location by using event photographs, event descriptions and sensor data by smartphones of system users. Their system also supports the accuracy degradation due to tampering with photographs.

VI. Concludion

In this paper, we proposed a Real-Time Smart and Quick Unexpected-Event Detect (RT-SQUED) method to detect and localize unexpected events in real-time, assuming crowd sourcing with smartphone devices. We described how RT-SQUED solves SQUED's problem by using the proposed two-phase processing for the real-time event detection.

In the future, we will improve the detection method further and build a new system using RT-SQUED in order to perform the event detection in real-time.

Acknowledgment

This work was partially supported by JSPS 24300039.

References

[1] A. Harma, M. McKinney, and J. Skowronek, "Automatic surveillance of the acoustic activity in our living environment," in IEEE ICME 2005, July 2005, pp. 4 pp.6–8.

[2] A. F. Smeaton and M. McHugh, "Towards event detection in an audio-based sensor network," in ACM, ser. VSSN '05, 2005, pp. 87–94.

[3] T. Yamamoto, K. Oku, H.-H. Huang, and K. Kawagoe, "Squed: A novel crowd-sourced system for detection and localization of unexpected events from smartphone-sensor data," in IEEE/ACIS ICIS 2015, June 2015, pp. 383–386.

[4] R. W. Ouyang et al., "If you see something, swipe towards it: Crowdsourced event localization using smartphones," in ACM, ser. UbiComp '13, 2013, pp. 23–32.

[5] D. Tran, J. Yuan, and D. Forsyth, "Video event detection: From subvolume localization to spatiotemporal path search," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 36, no. 2, 2014, pp. 404–416.

[6] F. Wang, Z. Sun, Y.-G. Jiang, and C.-W. Ngo, "Video event detection using motion relativity and feature selection," Multimedia, IEEE Transactions on, vol. 16, no. 5, 2014, pp. 1303–1315.

[7] T. Qin, H. Ma, D. Zhao, T. Li, and J. Chen, "Crowdsourcing based event reporting system using smartphones with accurate localization and photo tamper detection," in Big Data Computing and Communications, ser. Lecture Notes in Computer Science, 2015, vol. 9196, pp. 141–151.

 ISBN: 978-1-61208-452-7