國立清華大學電機資訊學院資訊工程研究所
碩士論文
Department of Computer Science
College of Electrical Engineering and Computer Science
National Tsing Hua University
Master Thesis

在智慧城市中遊戲化手機群眾外包系統
Efficient Mobile Crowdsourcing via Gamification for Smart City Applications

陳映亦
YingYi Chen
104062641

指導教授：徐正炘 博士
Advisor: Cheng-Hsin Hsu, Ph.D.

中華民國 106 年 05 月
May, 2017

國立清華大學
資訊工程研究所

碩士論文

在智慧城市中遊戲化手機群眾外包系統

陳映亦 撰

106
05

Acknowledgments

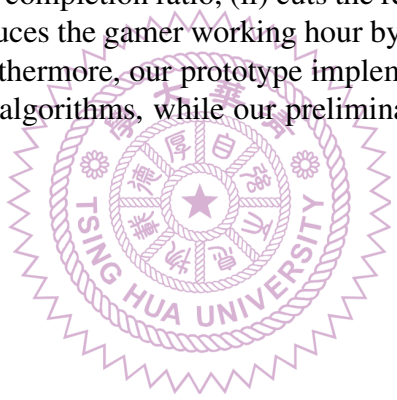
I would like to express my gratitude toward all the people who helped me in the past two years. I wouldn't be able to finish my thesis without their help along the way. I want to thank my parents specifically, for it is they who provided me with whole-hearted support over my decisions. I would also like to thank my labmates in Networking and Multimedia Systems Laboratory, especially Hua-Jun Hong, who helped me a great deal in the course of my research. Lastly, I would like to express my gratitude toward my adviser: Prof Cheng-Hsin Hsu. Without the guidance and the suggestion I received from him, I would not have been able to accomplish what I have done and learned today.

致謝

在此我要感謝在過去兩年中所有幫助過我的人， 如果沒有你們的幫助我一定沒有辦法順利完成我的論文。 在此我要特別感謝我的父母，他們提供我堅定不移的支持， 同時也支持我所作的每一個決定。我也要感謝網路與多媒體系統實驗室的同學們，特別是洪華駿在過去兩年的研究中幫助我非常的多。 最後，我要感謝我的指導教授：徐正炘教授。如果沒有他的給予我的指導以及建議， 在過去的兩年內我一定沒辦法完成如此多的事情以及學到如此多的東西。

Abstract

We present a gamified Smartphone Augmented Infrastructure Sensing (SAIS) platform for leveraging mobile gamers for applications such as smart cities. We develop a suite of algorithms to transparently guide the gamers to sensing task locations, in order to complete more tasks at shorter response time without incurring high workload on gamers. We evaluate our algorithms using extensive simulations and a real prototype implementation. The simulation results confirm that our algorithms achieve their design goals. For example, with 200 sensing tasks and 100 gamers, our algorithms on average: (i) achieve 63% higher completion ratio, (ii) cuts the response time by almost two-third, and (iii) reduces the gamer working hour by 81%, compared to the existing solutions. Furthermore, our prototype implementation demonstrates the practicality of our algorithms, while our preliminary user study receives positive feedback.



中文摘要

這篇論文基於一個結合手機與城市感測設施的資料蒐集平台，進一步地提出利用遊戲化的方式吸引更多玩家在智慧城市中進行感測資料的蒐集。 我們將這些蒐集感測資料的需求轉化為群眾外包任務，將這些任務發包給玩家去執行。為了讓整個系統可以快速地完成更多任務，並且不給玩家過多的負擔的情況下， 我們設計了一些演算法達成這些目的。我們透過模擬以及實作的方式去驗證我們提出的演算法和遊戲化的系統，模擬的結果證明我們的演算法在200個感測任務、100個玩家的情況下，比起現有的方法(1)有63%更高的任務完成率，(2)任務完成的速度將近3倍，(3)玩家花在感測的時間降低81%。另外，實作的系統經過玩家調查證明遊戲化的方式能確實達成我們的目的。

Contents

List of Figures

List of Tables


TSING HUA UNIVERSITY

Chapter 1

Introduction

Recently, the idea of smart city is becoming increasingly popular, thus, We develop a gamified crowdsourcing system to collect various sensory data for smart city. In this chapter, we describe the motivation, challenges, and contributions of this paper.

1.1 Motivation

Smart cities [42] require intelligent infrastructures to solve various resource management problems and large-scale social/economic challenges, such as traffic congestion, pollution monitoring, and disaster recovery. Market forecast predicts that the smart city market will grow at an annual rate of almost 20% and reach 1.45 trillion USD by 2020 [9]. Smart cities dictate city-wide sensing infrastructures, consisting of in-situ sensors and always-connected networks. Deploying, managing, maintaining, and upgrading the sensing infrastructures, however, are expensive, error-prone, and tedious. One way to cope with the issues due to missing in-situ sensors is to leverage sensor-rich smartphones for *mobile crowdsourcing* [41] or *mobile sensing* [24][1]. With mobile crowdsourcing, we instruct smartphone users to relocate to specific locations at certain time instances, to carry out the assigned sensing tasks, so as to fill up the gaps among in-situ sensors.

In our earlier work [26], we propose a hybrid sensing platform with in-situ sensors and smartphones, called *Smartphone Augmented Infrastructure Sensing* (SAIS) for smart city applications. In this paper, we propose to *gamify* [34] the SAIS platform using mobile games, similar to the popular Pokémon Go [8], to *transparently* assign sensing tasks to mobile gamers, in order to encourage user engagement, increase overall productivity, and retain smartphone users for a cost-effective and sustainable SAIS platform. Fig. 1.1 gives sample usage scenarios of our proposed SAIS platform, which consists of servers,

[1]The difference between mobile crowdsourcing and mobile sensing is insignificant in our discussions, and thus we use mobile crowdsourcing to refer both for brevity throughout this paper.

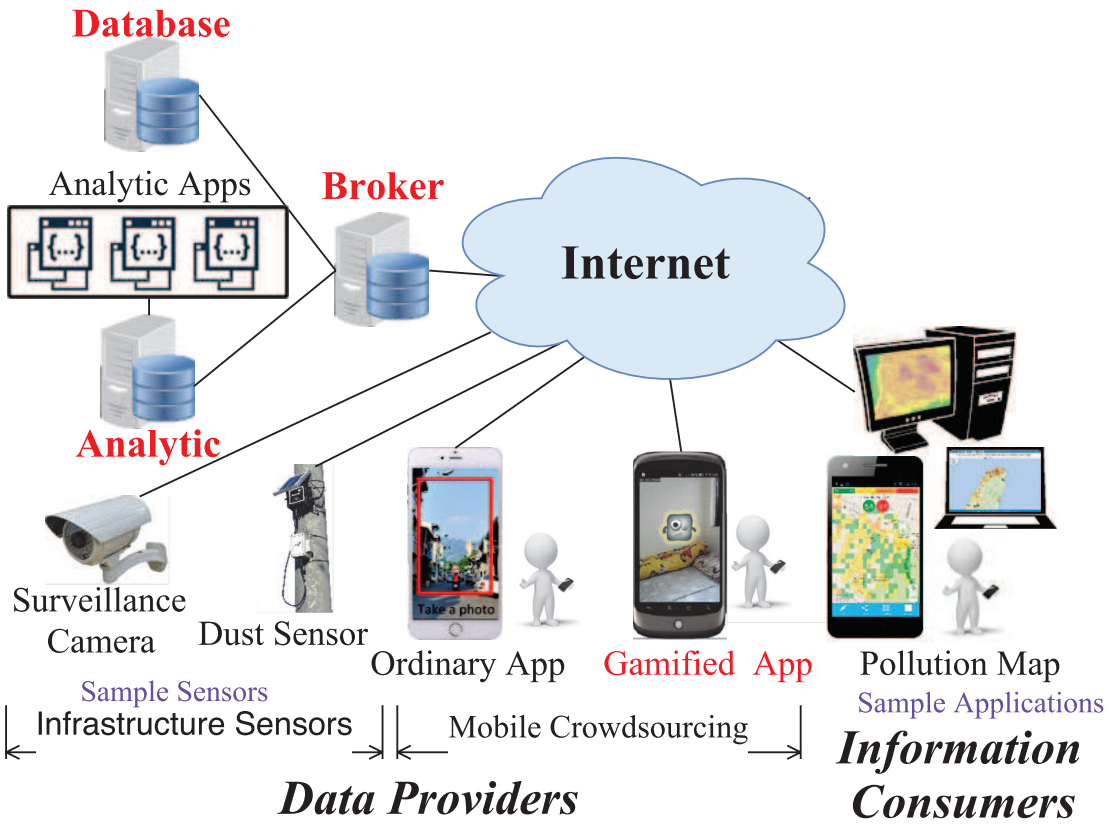


Figure 1.1: Sample usage scenarios of a gamified Smartphone Augmented Infrastructure Sensing platform.

data providers, and information consumers. There are three types of servers: (i) the broker, which runs management algorithms, (ii) the database servers, which store the sensory data, and (iii) the analytics servers, which execute analytic applications for smart cities. Both in-situ sensors (such as cameras and dust sensors) and mobile crowdsourcing users (with smartphone sensors or specialized sensor modules wirelessly connected to smartphones) act as data providers. Mobile crowdsourcing users contribute to the platform using: (i) the *ordinary* app that requires the users to manually fill in forms or (ii) the *gamified* app that transparently collects data in mobile games. The information consumers use computers to access the results from analytic applications, such as real-time pollution maps.

Gamifying the mobile crowdsourcing app is not an easy task, due to the significance of precision in terms of location and time. To overcome this challenge, we adopt Augmented Reality (AR) [33] to add *Non-Player Characters (NPCs)*, such as cartoon characters, to live camera feeds in the preview windows of smartphones. Then, by systematically moving the NPCs on the maps and in the preview windows, we guide the smartphone users to collect requested videos (or other sensory data) using smartphone cameras (or other sensors) in a transparent way. We notice that each task from smart city applications may *not*

be fulfilled by a single mobile gamer instantaneously, e.g., taking a photo of a historical monument from all 360 degrees demands for either: (i) collaboration among multiple mobile gamers or (ii) a longer time duration for a mobile gamer to take photos from multiple angles. Hence, to achieve efficient mobile crowdsourcing, we have to carefully design the gamified app on the smartphones and the management algorithms on the broker, to complete the sensing tasks in the shortest amount of time.

1.2 Challenges

1.2.1 Incentive Mechanism

The incentives mechanism for smartphone users to participate in mobile crowdsourcing is critical. The more smartphone users, the task can be completed more efficiently. Prior mobile crowdsourcing work often adopts monetary incentive [19, 26, 37]. For example, Feng et al. [19] consider the auction problem of spatial dependent mobile crowdsourcing. They present an auction framework for the crowdsourcing platform and smartphone users. Different from [19, 26, 37], the current paper considers *mobile games* as alternative incentives of mobile crowdsourcing. Such a concept, to our best knowledge, is first briefly touched upon in a user study [11], without being implemented nor evaluated. Talasila et al. [36] develop a mobile game for mobile crowdsourcing, in which aliens move on a map of mobile app, and mobile gamers have to follow the trails of aliens to gain points. Different from our work, their app does not adopt AR, and cannot guide mobile gamers for *directional* sensing, such as taking photos or shooting videos. Moreover, their work focuses on *area coverage* of *homogeneous* sensing tasks, while our SAIS platform is more general and comprehensive for heterogeneous sensing tasks.

1.2.2 Gamer Assignment

In order to complete the crowdsourcing task efficiently, we have to careful assign the task to the mobile gamers. The problem is that the human behavior is hard to predict, we can't assume that every mobile gamer is always available for all the tasks. There are two modes of Gamer Assignment: gamer select tasks and server assign tasks [23]. The gamer select tasks mode allows each gamer to choose the task that he or she prefers. However, some tasks that located in not crowded area may not be done. To make the high overall completion of the tasks, we only consider the server assign tasks mode. Jian et al. [12] proposed a crowdsourcing assignment model based on social relationship cognition and community detection. Unlike our work, their model select the gamers who have credible interaction with the information consumer of each task. Liao et al. [27] proposed an algorithm to

assign the crowdsourcing tasks close to the gamers' routine trajectories. Therefore, the gamers can complete the task without moving far away form their path.

1.3 Contribution

There are four contributions in this paper.

- Develop crowdsourcing platform. We implement a crowdsourcing platform, which contains (i)web page for information consumers, (ii) the server which handle gamer assignment, and (iii) clients with ordinary version and gamified version.
- Develop gamified crowdsourcing application. We implement a gamified app on Android which allow user to collect data for us while playing the game.
- Design three algorithms for our system.
  - *Optimal spot locator,* which calculates the minimal number of spots (locations) for each spatial-temporal task.
  - *Nearest gamer assigner,* which assigns the nearest gamer with sufficient capability to each spot.
  - *Nature NPC path generator,* which considers the locations of the optimal spots and mobile gamers, and generates natural NPC paths to transparently guide mobile gamers to relocate to the optimal spots and complete the task.

  These algorithms are the keys of the gamified app in our SAIS platform. The first two algorithms are implemented on the broker server, and the last one runs on smartphones. These optimization algorithms allow the SAIS platform to efficiently support smart city applications. This sets our platform quite different from Pokémon Go [8], which merely generates static monsters at arbitrary locations and addresses little, if any, city-scale issues.
- Evaluate our system with real user study. We find participants to use our applications. They give the gamified app higher score in our study.

1.4 Organization

The rest of this paper is organized as follows. We survey the related work in Ch. 2. Ch. 3 presents the optimization algorithms. This is followed by the simulations in Ch. 4. We implement the algorithms in a real SAIS platform and carry out a user study in Ch. 5. Ch. 6 concludes the paper.

Chapter 2

Background and Related Work

2.1 Smart City and Urban Computing

The rapid growth of population leads to many issues in urban cities, such as traffic congestion, energy consumption, and pollution. In order to solve these urban issues, the concept of smart city and urban computing becomes more popular. The smart city integrate multi dimensions that includes smart transportation, smart environment, smart health care, smart education, smart safety, and smart energy [31]. More detail, the smart city can divide into three layers encompassing the perception layer, the network layer, and the application layer [35]. The perception layer is responsible to collect information via sensor. The network layer makes accurate transmission and preprocessing of the information obtained in the perception layer. The application layer is to analyze the information and extract useful information from massive data. However, there are many research problems in all three layers. The preception layer faces sensor development. communication protocol, and energy consumption. Followed by the network layer with problems in information integration, traffic congestion, and real-time transmission. Lastly, the visual and analytical lowed applicaiotn layer is increasing the burden in regards to the massive data.

While the smart city has a very large concept, the urban computing is more focus on data analytics. There are lots of smart infrastructures which collects heterogeneous data in our smart city every seconds. The problem is how to acquisition, integration, and analysis big urban data. More precisely, the framework of urban computing includes urban sensing and data acquisition, urban data management, urban data analytics, and service providing [42].

In our system. our idea is closer to those of smart city as we are not focus on data analytics. We provide a platform allowing information consumer to send request, and get required data from crowdsourcing. The information consumer have to analyze data by

their own algorithms. Our research problem would be how to achieve a higher efficiency in crowdsourcing.

2.2 Crowdsourcing

Due to the explosive growth of the Internet, crowdsourcing has been used in various applications in the literature. These crowdsourcing applications can be grouped into several classes [41]: (i) voting systems, (ii) information sharing systems, (iii) social games, and (iv) creative systems. However, despite the systems being widely studied, only a few of them handle spatial-temporal dependent tasks, which are crucial to smart city applications. Since smartphones are widespread and equipped with various sensors, such as GPS readers, cameras, and digital compasses, they are ideal for spatial-temporal dependent crowdsourcing [11]. Kanhere [22] discusses several challenges in mobile crowdsourcing, including: (i) incomplete samples, (ii) context-awareness, (iii) user privacy, (iv) user credibility, and (v) energy conservation. These challenges, while important, are orthogonal to our work in this paper. In particular, the current paper considers a more difficult problem: *incentives for smartphone users to participate in mobile crowdsourcing.* Prior mobile crowdsourcing work often adopts monetary incentive [19, 25, 26, 37]. Different from their work, the current paper considers *mobile games* as alternative incentives of mobile crowdsourcing. Such a concept, to our best knowledge, is first briefly touched upon in a user study [11], without being implemented nor evaluated. Talasila et al. [36] develop a mobile game for mobile crowdsourcing, in which aliens move on a map of mobile app, and mobile gamers have to follow the trails of aliens to gain points. Similar to our work, smartphones in their system transparently collect sensory data, such as WiFi signals. Different from our work, their app does not adopt AR, and cannot guide mobile gamers for *directional* sensing, such as taking photos or shooting videos. Moreover, their work focuses on *area coverage* of *homogeneous* sensing tasks, while our SAIS platform is more general and comprehensive for heterogeneous sensing tasks.

2.3 Gamification

Gamification defines as using game element in no-gaming systems to improve user experience (UX) and user engagement [17]. Seaborn et al. [34] indicates the top fields for gamification research are education, health and wellness, online communities and social networks, crowdsourcing and sustainability. Hamari et al. [20] classify the motivational affordances for gamification into 10 categories: points, leaderboards, achievements and badges, levels, story and theme, clear goals, feedback, rewards, progress, and challenge.

Out of the ten, the points, leaderboards, and badges are most commonly used. De Luca et al. [28] propose a social power game that encourages people to save energy in a city. Their application visualize gamer's home energy consumption. Gamers can compare their energy consumption with their friends. They discuss the design challegnes in such applications. Such gamify application promotes the awareness of issues of energy use in the city.

may leads people more focus on energy problem in urban city.

2.4 Mobile Augmented Reality

Augmented Reality (AR), defined as "a form of virtual reality where the participant's head-mounted display is transparent, allowing a clear view of the real world" [30], is considered to be a promising technology with the progress of hi-tech products. There are three classes of displays for AR: (i) Head-worn displays (HWD), (ii) Hand-held displays and (iii) Projection displays. Head-worn displays, such as Google Glass and Microsoft HoloLens, allow users to look at a see-through displayer and get AR information. Hand-held displays provide AR video on the screens of mobile devices with camera. Projection displays is to project virtual information on the physical object directly. In this paper, we focus on hand-held displays which can be smartphone or tablet. Gamers can use their own smartphone to play our crowdsourcing game.

There are still challenging to run AR on smart phone. Some research about mobile AR is focus on recognizing the environment. AR is separated into two categories: in a prepared and in an unprepared environment. Running AR in a completely prepared environment demonstrate highly accuracy. Welch et al. [40] proposed a single-constraint-at-a-time algorithm (Scaat) to increase the performance of tracking in a completely prepared environment. Running AR in an unprepared environment is a difficult problem for hand-held devices. One of the solution is vision tracking on markers. It is used in the most of popular commercial applications, current AR games, such as Invizimals on PSP [38]. However, current research problem is focus on recognizing 3D environment. Most of the smart phones only equipped with one camera, it is hard to recognize the real world like the human eyes. To solve AR in a single camera mobile device, it becomes a monocular Simultaneous localization and mapping (monocular SLAM) problem. There are two classes for monocular SLAM. (i)Feature-Based Method, containing both filtering-based and keyframe-based approaches. With usually two steps. First, extract feature from the image. Second, compute camera position and construct 3D world. This method is faster, but less accurate and limited in featurable resource. (ii)Directed Method, it doesn't rely on features. Indeed, it directly computes camera position and rebuild 3D world with

total images. The method is accurate, robust, and able to use all information in the image. However, it needs to consume large computing resource. Some existing work like DTAM [32] and SLAM [18]. Although the advance of hardware and software makes mobile AR become real, there are still a lot challenges on mobile AR [33]. Primarily, the computation of a real-time and mark-less AR is still too high. So mobile phone can not run mark-less AR or run with a low frame per second. therefore, we implement an efficient AR game like Pok*é*mon Go [8].

Chapter 3

Problems and Solutions

Our proposed system contains three components: spot locator, gamer assigner, and NPC path generator. The spot locator is responsible for selecting the minimal number of spots for each task. Based on the selected spots, the gamer assigner assigns gamers to the spots. After assigning tasks, the NPC path generator moves NPCs in AR mobile app to attract and guide gamers to the right spots for finishing the tasks.

3.1 Notations

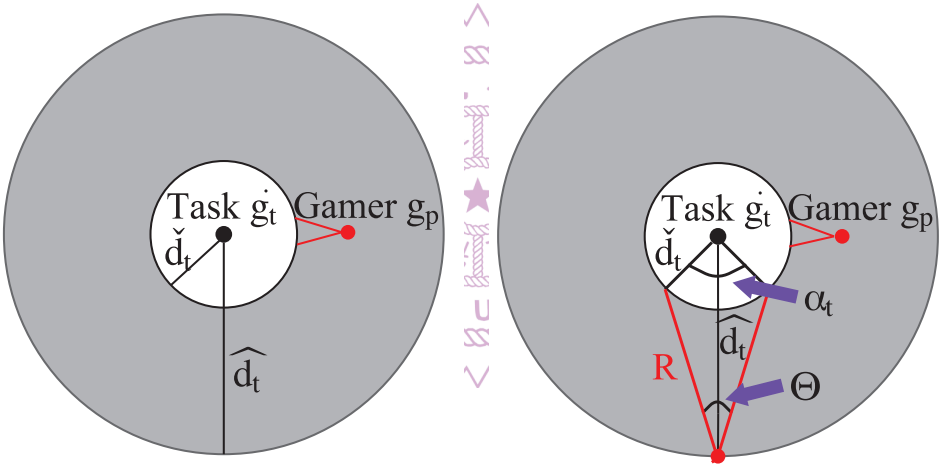
Table 3.1: Symbols Used in This Paper

Sym.	Description
T	Number of tasks
P	Number of gamers
$\mathbf{U}_t$	Requested angle set of task t
$\mathbf{Q}_t$	Assigned gamer set of task t
$\mathbf{S}_t$	Located spot set of task t
$\hat{d}_t$	Maximal effective distance of task t
$\check{d}_t$	Minimal effective distance of task t
$\dot{g}_t$	GPS location of task t
g_p	GPS location of gamer p
$\ddot{g}_s$	GPS location of spot s
Gn	GPS location of the NPC
α_t	Maximal covered angle of a single gamer of task t
R_t	The radius of FoV of the sensor requested by task t
Θ_t	The angle of FoV of the sensor requested by task t
$\hat{l}_t$	The beginning life time of task t
$\check{l}_t$	The ending life time of task t

Table 3.1 summarizes the symbols used throughout this paper. Let P be the number of active gamers, and g_p be the current location of gamer p, where $1 \leq p \leq P$. Let T be the number of sensing tasks from smart city applications. To be general, we consider *directional sensing* tasks, which require mobile gamers to perform each task from a specific

angle within a reasonable distance to the task location[1]. Examples of such tasks include: (i) capturing videos from all four corners of an intersection and (ii) measuring the noise levels at the eight gates of a busy train station. Each task t $(1 \leq t \leq T)$ is described by: (i) its GPS location $\dot{g}_t$, (ii) requested angle set $\mathbf{U}_t$, (iii) maximal effective distance $\hat{d}_t$, (iv) minimal effective distance $\check{d}_t$, (v) beginning life time $\hat{l}_t$, and (vi) end life time $\check{l}_t$. Fig. 3.1(a) illustrates a sample sensing task t at location $\dot{g}_t$ with $\mathbf{U}_t = \{[0°, 360°]\}$, and its effective distance between $\check{d}_t$ and $\hat{d}_t$. Mobile gamers have to perform each task by pointing their sensors, such as the cameras at the spot locations from the gray area. The spot location is represented by g_{ts} of task t and spot s. Such sensing tasks may be performed several times (by one or multiple gamers) in order to cover $\mathbf{U}_t$.

3.2 Optimal Spot Locator



(a) (b)

Figure 3.1: Illustrative examples of: (a) a sensing task from all 360 degrees, and (b) the FoV of a directional sensor and the located spots of a task.

Our proposed optimal spot locator algorithm locates the least number of spots to finish a task, i.e., covering $\mathbf{U}_t$. Different tasks require different sensors, which have diverse Field of Views (FoVs). The FoV is a fan-shaped effective sensing area described by radius R_t and angle Θ_t. Fig. 3.1(b) shows an example of FoV and the located spots. It is not hard to see that to maximize the coverage of each gamer standing on a spot, ideally the spot is located on the outer circle. We write the maximal covered angle as $\alpha_t = 2arccos((\hat{d}_t^2 + \check{d}_t^2 - R^2)/2\hat{d}_t\check{d}_t)$. We then locate the spots on the outer circle with an

[1]Omnidirectional sensing tasks, such as collecting WiFi fingerprints [36], are *degraded* versions of our general sensing tasks.

equal angle of α_t between any two adjacent spots. For example, there are six spots located in Fig. 3.1(b) because $\alpha_t = 60°$. Algorithm 1 shows the pseudo code of the optimal spot selector. In lines 2 and 3, we go over all the tasks and its requested angles $\in \mathbf{U}_t$. In lines 4 and 5, we locate N spots and compute specific locations of each spots. The computed spots are saved in $\mathbf{S_t}$.

For example, if we would like to cover $360°$ of the target and $\alpha_t = 30°$, we will have six spots equally distributed on the outer circle based on α_t. Hence, $\mathbf{S_t} = \{0°, 60°, 120°, 180°, 240°, 300°\}$.

Algorithm 1 Optimal Spot Locator

1: **function** SPOT_LOCATOR
2: **for** each task $t = 1, 2, \ldots, T$ **do**
3: **for** $\mathbf{e} \in \mathbf{U_t}$ **do**
4: $N = \lceil(\max(\mathbf{e}) - \min(\mathbf{e}))/\alpha_t\rceil$
5: **for** $n = 0, 1, \ldots, N-1$ **do**
6: $\mathbf{S_t} = \mathbf{S_t} \cup \{((\max(\mathbf{e}) - \min(\mathbf{e}))/N)n + \min(\mathbf{e})\}$

3.3 Nearest Gamer Assigner

The nearest gamer assigner assigns gamers to nearest spots, thus, the gamers have to report their locations while they are playing. We note that the assigned gamer must have the available sensors required by the task. The pseudo code of the nearest gamer assigner is shown in Algorithm 2. In lines 2 to 3, we find the most urgent task and then assign gamers to its spots. More specifically, in lines 4 to 7, we gradually assign the nearest gamers that are capable for the sensing task to the remaining spots. The algorithm ends upon all tasks have got enough gamers to cover the spots. Note that the nearest gamer assigner algorithm is periodically executed, such as once every 5 minutes. This is to adapt to system dynamics, like inactive gamers and overdue tasks.

Algorithm 2 Nearest Gamer Assigner

1: **function** GAMER_ASSIGNER
2: **while** there are remaining tasks **do**
3: find the most urgent task $\dot{t}$ by comparing $\check{l}_t$ and current time
4: **while** there are unsatisfied spots of $\dot{t}$ **do**
5: Find the nearest $\dot{p}$ by comparing $g_{\dot{t}}$ and g_p
6: **if** gamer $\dot{p}$ has required sensors for task $\dot{t}$ **then**
7: assign $\dot{p}$ to the nearest unsatisfied spot $\in \mathbf{S}_{\dot{t}}$ in $\mathbf{Q}_{\dot{t}}$

3.4 Nature NPC Path Generator

The nature NPC path generator creates the NPC paths to guide the gamers as illustrated in Fig. 3.2. An NPC is initially put on the intersection of the inner circle and the line between the spot and the task location. We show the NPC on the gamer's map and expect that the gamer goes towards the NPC. The gamer sees the NPC when their distance is less than $\hat{d}_t - \check{d}_t$. However, the gamer may look at the NPC in a direction without the task location shown in the gamer's preview window (Fig. 4.1(a)). The NPC path generator performs three steps to guide the gamer. First, the path generator moves the NPC to a point that is $\hat{d}_t - \check{d}_t$ away from the spot and on the line between the gamer and the spot (Fig. 3.2(b)). This step guides the gamer to move to the spot. Second, the path generator moves the NPC back to the intersection point (Fig. 3.2(c)). This step guides the gamer to turn towards the task location. Finally, the gamer reaches the spot and points to the task location (Fig. 3.2(d)), and then completes the task. Note that, although we propose the nature NPC path generator to guide the gamers, the gamers may not be able to precisely stand on the spot. In such case, we take out the sensed angles from the required angle set, and rerun the algorithms. For example, if a task t's requested angle $\mathbf{U}_t = \{[0°, 360°]\}$ and two gamers cover angles between $0°$ to $30°$ and $60°$ to $90°$, $\mathbf{U}_t$ becomes $\{[30°, 60°], [90°, 360°]\}$.

Algorithm 3 Nature NPC Path Generator

1: **function** PATH_GENERATOR
2: Let gi be the initial GPS location of NPC
3: $gi = \dot{g}_t + (\overrightarrow{\dot{g}_t g_{ts}})/\hat{d}_t\check{d}_t$
4: Put NPC at the initial point $Gn = g_i$
5: **while** g_p is not at spot location $\ddot{g}_s$ **do**
6: $Gn = g_p + (\overrightarrow{\dot{g}_t g_{ts}})/\hat{d}_t(\hat{d}_t - \check{d}_t)$
7: **while** Gn is not at gi **do**
8: $Gn = Gn + (\overrightarrow{Gngi})/(|\overrightarrow{Gngi}|)$

The pseudo code of the nature NPC path generator is shown in Algorithm 3. From lines 2 to 4, we calculate the initial GPS location of the NPC. From lines 5 to 6, we keep moving the NPC until the gamer is at the spot location. In lines 7 and 8, we move the NPC back to the initial location, so that the gamer rotates his/her smartphone towards the task location.

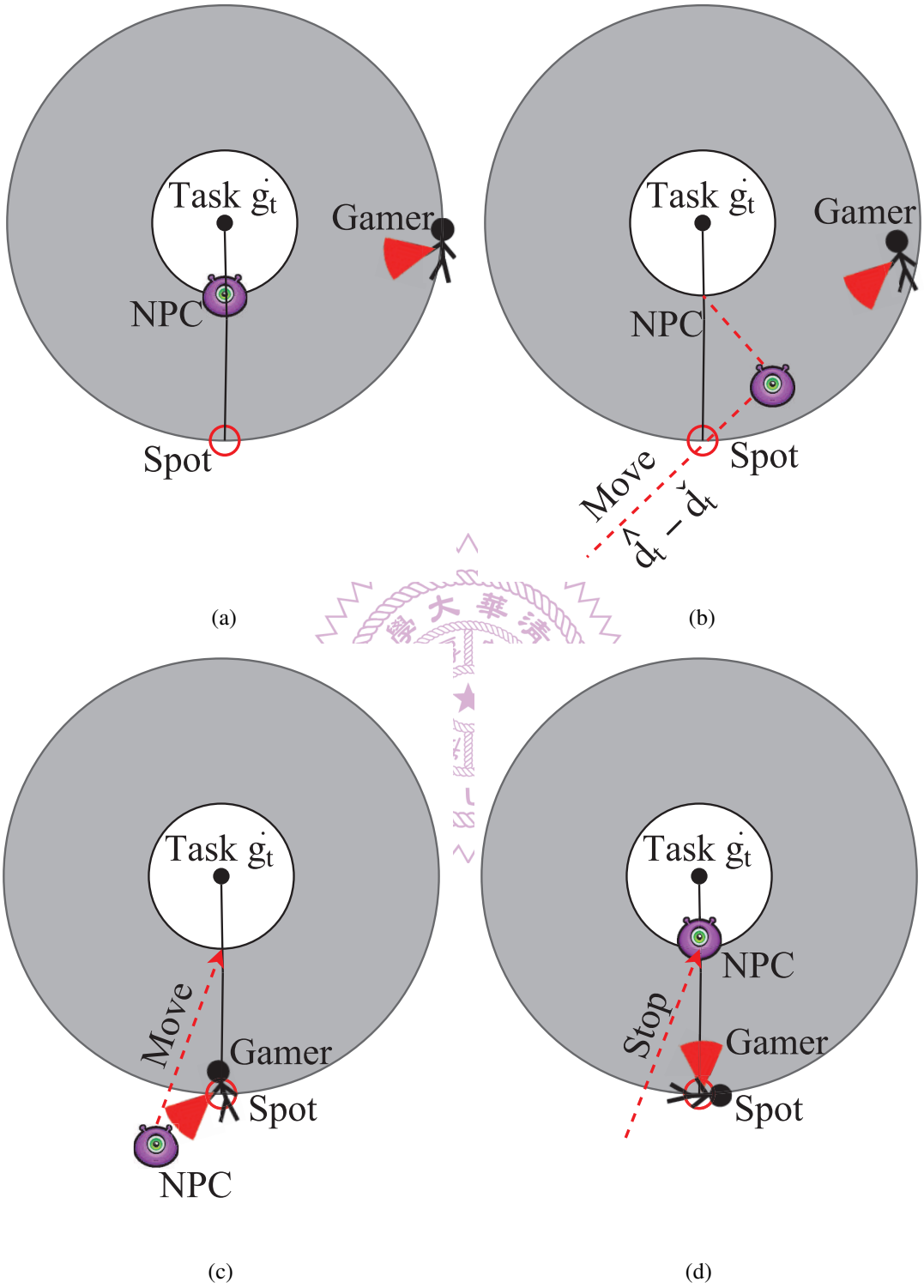


Figure 3.2: NPC path generator works in the following steps: (a) the gamer follows the NPC, (b) the NPC moves and guides the gamer to the spot, (c) the NPC moves and guides the gamer to rotate his/her smartphone, and (d) the gamer captures the NPC and performs the sensing task.

Chapter 4

Simulation

In this chapter, we conduct extensive simulations to evaluate our algorithms in larger scenarios. First, we introduce the baseline algorithms. Followed by our simulation settings and results.

4.1 Baseline Algorithms

We have also implemented two baseline algorithms in the simulator for comparisons. We denote the two baseline algorithms as VSN and Current in the figures and tables.

4.1.1 VSN

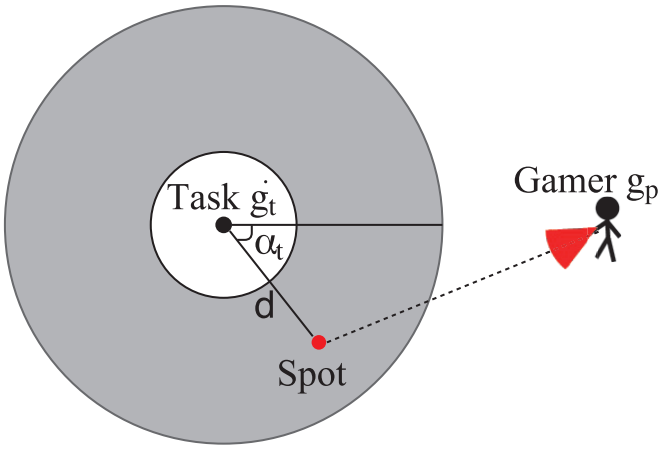
First, in Video Surveillance Networks (VSNs), an existing algorithm that randomly selects the spots within the effective distance until tasks are completed were proposed in Chen et al. [15]. In their scenario, the sensors equipped camera are randomly place in an area. The problem is how to find a minimum set of sensors to cover the whole area. Similar to our work, they consider FOV, rotation of each camera. The difference is that we use gamers to replace the sensors. We modify the VSN algorithm in order to use in our SAIS platform. From lines 2 to 3, we find the spot for each gamer who is assigned the task. Specifically, from lines 4 to 5, we calculate the distance and the angle to generate the new spot. In line 6, we add the new spot to the sets of $\mathbf{S_t}$.

4.1.2 Current Work

Second, a currently-used algorithm that mimics manual assignments and human behavior is considered. For each task, this algorithm locates the closest gamer, and places its NPC on the line between the gamer and the task location within the effective distance. From lines 2 to 3, we find the spot for each gamer who is assigned the task. Specifically, from

Algorithm 4 The VSN algorithm.

1: **function** VSN
2: **for** each task $t = 1, 2, \ldots, T$ **do**
3: **for** p who is assign to t **do**
4: let d = distance away from $\dot{g}_t$, which is randomly chosen between $\hat{d}_t$ and $\check{d}_t$ followed by gaussian distribution
5: let α_t = cover angle of task t, which is randomly chosen between 0 and 2π
6: $\mathbf{S_t} = \mathbf{S_t} \cup \{\dot{g}_t + d|\sin\alpha_t, \cos\alpha_t|\}$



(a)

Figure 4.1: The gamers' behavior of VSN.

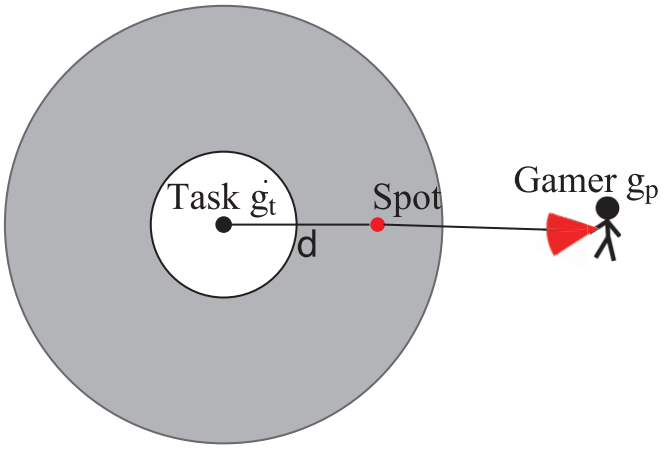
lines 4 to 5, we calculate the new spot location with random distance from task location to gamer location.

Algorithm 5 The Current Practice algorithm

1: **function** CURRENT_WORK
2: **for** each task $t = 1, 2, \ldots, T$ **do**
3: **for** p who is assign to t **do**
4: let d = distance away from $\dot{g}_t$, which is randomly chosen between $\hat{d}_t$ and $\check{d}_t$ followed by gaussian distribution
5: $\mathbf{S_t} = \mathbf{S_t} \cup \{(\dot{g}_t + (\overrightarrow{\dot{g}_t g_p})/|\overrightarrow{\dot{g}_t g_p}|d)\}$

4.2 Mobility Models

In order to simulate the gamers' movement in a large scale area, there are lots of research on mobility models [13]. The most frequently used mobility model is the Random Way Point model. Each gamer move independently to a randomly chosen destination. Specifically, the Random Way Point model randomly choose each gamer's destination, speed,



(a)

Figure 4.2: The gamers' behavior of current work.

and pause time. Another random model is the Random Walk Model, which randomly choose gamer's direction, not destination. There are some limitations of the Random Models, including temporal dependency, spatial dependency, and geographic restriction.

- **temporal dependency of velocity**. The velocity of gamer is a memoryless random process,i.e., the gamer in simulation may have strange behaviors like sudden stop, sudden acceleration and sharp turn. These behaviors is not going to happen in real life.
- **spatial dependency of velocity**. Each gamer move independently to other gamers in the Random Models. In real life, the gamers may follow other gamers, or interact with other gamers. Which is hard to simulate.
- **geographic restriction of Movement**. In the Random Models, the gamer move freely within simulation field without any restriction. However, in the real city, the movement of each gamers is bounded by obstacles, buildings, streets, of freeways.

In our simulation, we implement two mobility models in our simulation. We run the simulation with the Random Way Point Model and PathWay Mobility Model. The PathWay Mobility Model is to restrict the gamers' movement to the pathways in the map. We use the OpenStreetMap [6] to generate the map in simulations. Following is our simulation settings.

4.3 Settings

We have implemented a simulator in Java to evaluate our proposed algorithms. We use our campus map in the simulations, and its size is about 5×5 km^2. The simulator generates

P gamers at random locations following a uniform distribution. Each gamer has 1 to 3 hours available time, between 6 a.m. and 6 p.m. every day. The available time is generated using a Gaussian distribution, where the mean is 2 hours and the standard deviation is 1. Each simulation lasts for 1 week (simulation time). The gamers move around following the Random Waypoint model until they are assigned a task. We assume the travel speed of gamers is 5.4 km/hr. Upon being assigned a task, each gamer then attempts to complete the assigned task. The number of tasks is T and the life time of any task t is $L_T = \check{l}_t - \hat{l}_t$. The following parameters are used in our simulations: P = {25, 50, **100**, 200, 400}, T = {100, **200**, 400, 800}, and L_T = {1, 2, **3**, 4, 5} hours. The bold font indicates the default settings. We consider the following performance metrics:

- *Completion ratio.* For each task, the completion ratio is the percentage of the requested angles covered by the gamers. For all tasks, the completion ratio is the average ratio across all tasks.
- *Response time.* The time between the task received (from smart city applications) and completed (by gamers).
- *Working hour.* The average hours spent by gamers when carrying out the tasks.
- *Spots per task.* The number of resulting spots for each task.

4.4 Results

We run each setting 10 times and report the average results, along with 95% confidence intervals wherever applicable.

Our algorithm offers better quality of service. We report the performance results under the default settings in Figs. 4.3 and 4.4. We first observe from Fig. 4.3 that our algorithm achieves higher completion ratio and shorter response time. Fig. 4.3(a) presents the average completion ratio over time of a sample run. This figure shows that the completion ratios increase as time goes on. In the end, our algorithm achieves (64.38)% of the completion ratio, which outperforms the baseline algorithms by (64)% at most. Fig. 4.3(b) presents the overall completion ratios, which follow the same trend: on average our algorithm outperforms the baseline algorithms by up to 63%. Fig. 4.3(c) shows that the average response time over time. It is clear that our algorithm leads to much shorter response time. Fig. 4.3(d) gives the overall response time. On average, our algorithm achieves an average response time of 39.7 mins, while VSN and Current algorithms lead to average response times of 118.3 and 117.5 mins, almost 3 times compared to our solution.

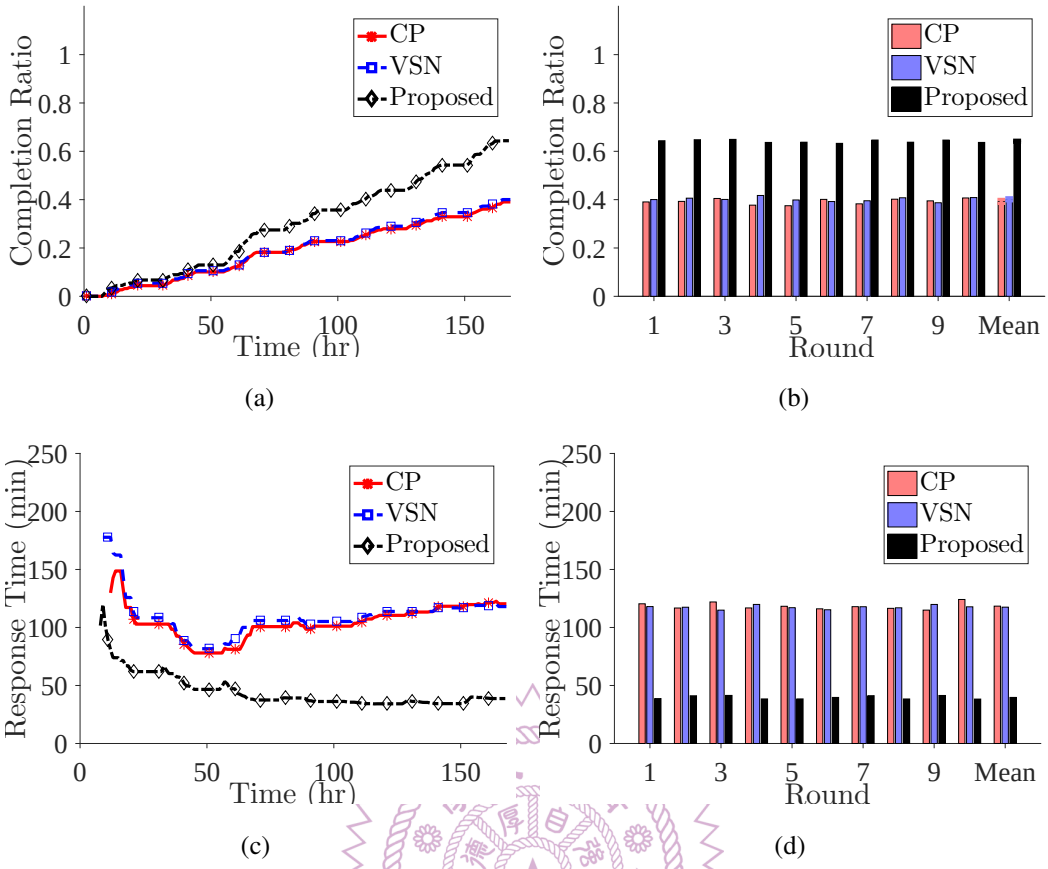


Figure 4.3: Our proposed solution results in better quality of service: (a) overall completion ratio and (b) overall response time.

Our algorithm incurs less workload on gamers and the system. We observe from Fig. 4.4 that our algorithm requires fewer working hours from gamers, which could be attributed to fewer task spots. In particular Fig. 4.4(b) shows that our algorithm results in an average working hour of 1.8 hrs, while the baseline algorithms lead to up to 9.8 hrs, a 81.63% reduction. This can be explained by Fig. 4.4(d), which indicates that the two baseline algorithms, compared to our algorithm, fail to effectively assign tasks to gamers: they end up with too many spots per task, as high as 15 times more than our algorithm. This is because VSN and Current algorithms are *random* and *greedy* algorithms, respectively. Therefore, they could not identify the best gamers for individual tasks, like our algorithm does. This behavior is the root cause of the inferior performance of the baseline algorithms.

Our algorithm performs well under different numbers of tasks. Fig. 4.5, and Fig. 4.6 show the implications of more tasks on individual performance metrics. Fig. 4.5(a) shows that the gap of completion ratio is larger when the number of tasks increases. Particularly, the completion ratio of our algorithm outperforms the baseline algorithms by 4.8 times at most. Fig. 4.5(b) shows that the response time of our algorithm is less than

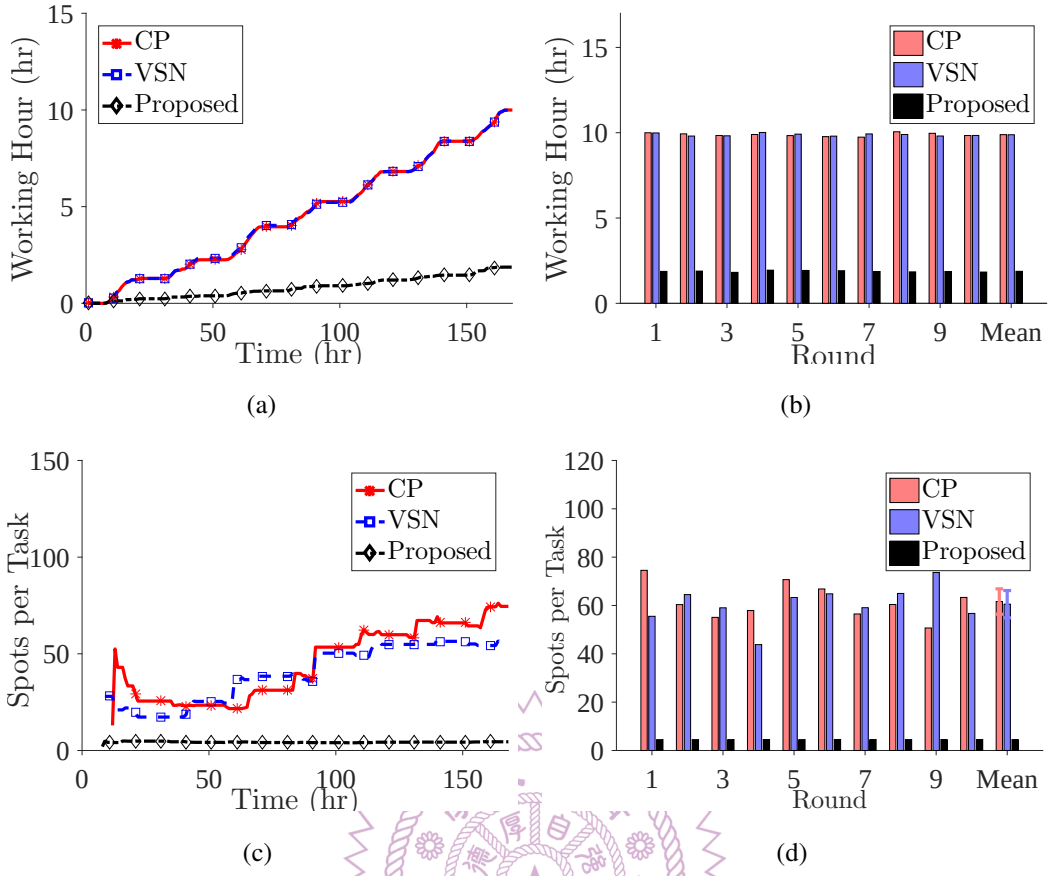


Figure 4.4: Our proposed solution incurs lower workload on gamers: (a) overall working hour and (b) overall spots per task.

half of that from the baseline algorithms. This is because our algorithm leads to fewer spots and assigns fewer gamers to the same task. Compared to the baseline algorithms, Fig. 4.5(c) shows that our algorithm reduces the average working hour by at most 87%. Fig. 4.5(d) reveals that our algorithms leads to only 5.3 spots per task, even when we have 800 tasks. In summary, Fig. 4.5, and Fig. 4.6 demonstrates that our algorithm constantly outperforms the baseline algorithms under different numbers of tasks.

Our algorithm performs well even with few gamers. Next, we study the implications of fewer gamers on individual performance metrics in Fig. 4.7, and Fig. 4.8. Fig. 4.7(a) shows that, when there are only 25 gamers, our algorithm achieves 60.82% completion ratio, which is almost 1.86 times higher than the baseline algorithms. Fig. 4.7(b) reveals that the response time of our algorithm is decreased from 58.84 to 35.26 mins when the number of gamers increases. Our response time is about one third of that from the baseline algorithms. Fig. 4.7(c) shows that the working hour of our algorithm declines from 7.7 to 0.4 hrs, when the number of gamers increases; while the same improvement is not observed on the two baseline algorithms. Figs. 4.7(b) and 4.7(c) illustrate that our algorithm can effectively leverage additional gamers, in contrast to the baseline algorithms.

The difference can be explained by Fig. 4.7(d). This figure shows that our algorithm produces stable numbers of spots per task, while the baseline algorithms assign many more gamers to each task, leading to waste of resources.

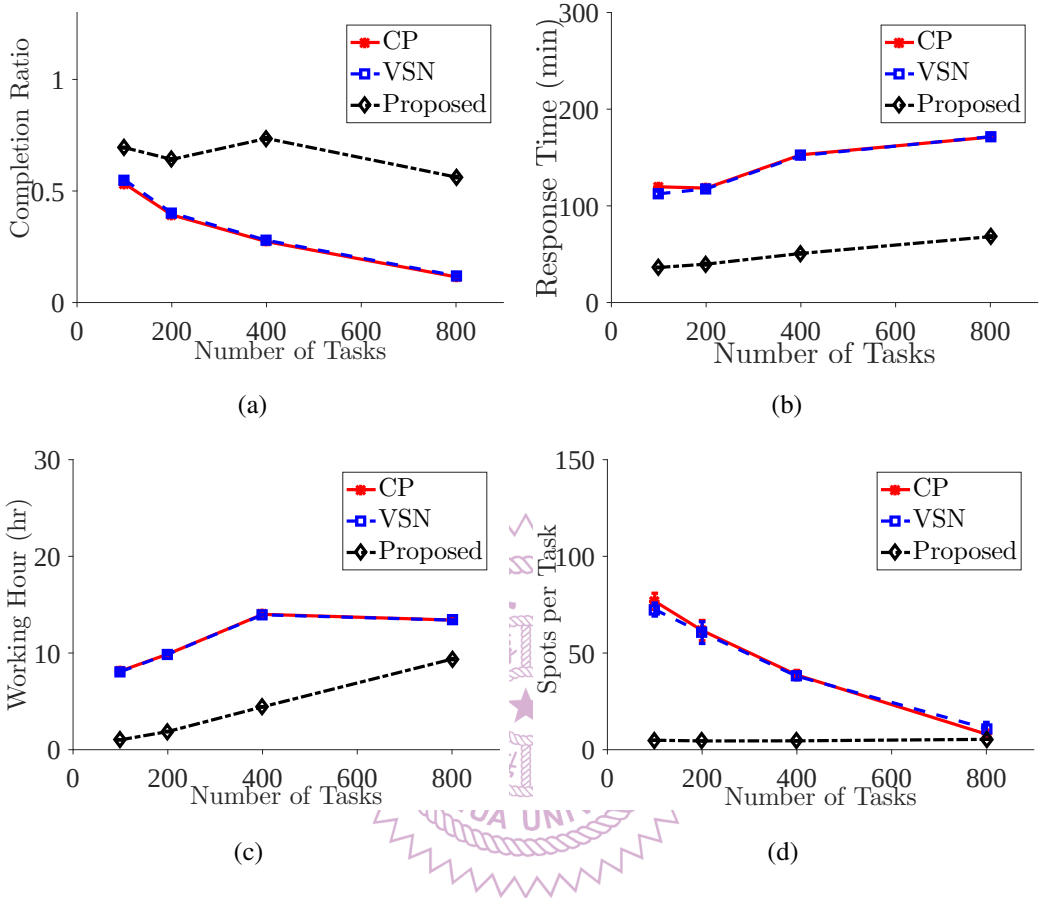


Figure 4.5: Performance comparisons under different numbers of tasks using Random Way Point: (a) the completion ratio, (b) the response time, (c) the working hour, and (d) the spots per task.

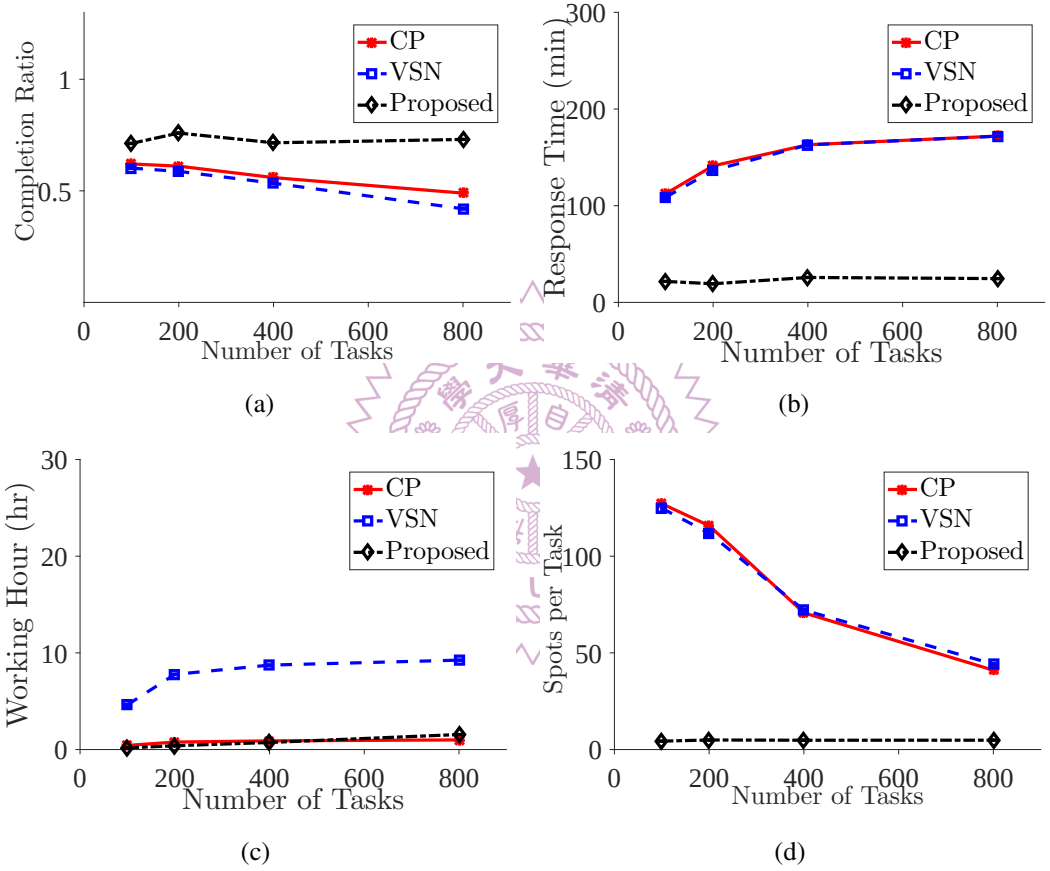


Figure 4.6: Performance comparisons under different numbers of tasks using Pathway Mobility Model: (a) the completion ratio, (b) the response time, (c) the working hour, and (d) the spots per task.

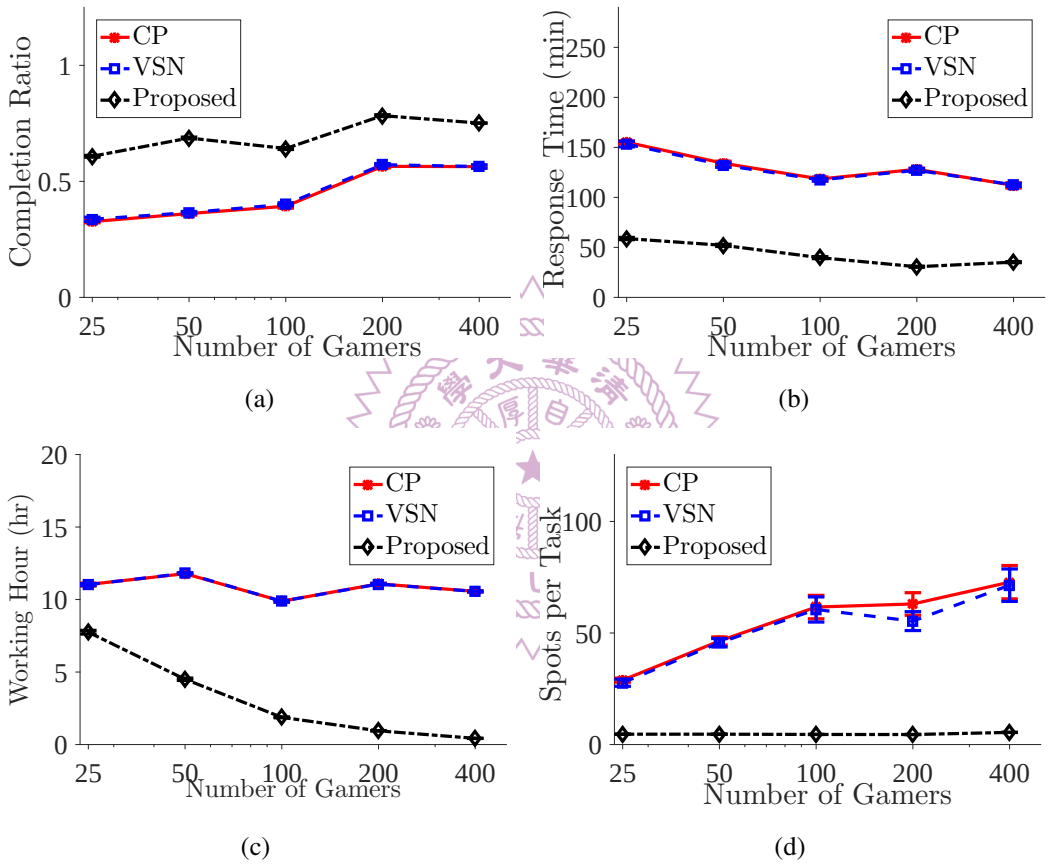


Figure 4.7: Performance comparisons under different numbers of gamers using Random Way Point: (a) the completion ratio, (b) the response time, (c) the working hour, and (d) the spots per task. The x-axes are in logarithmic scale.

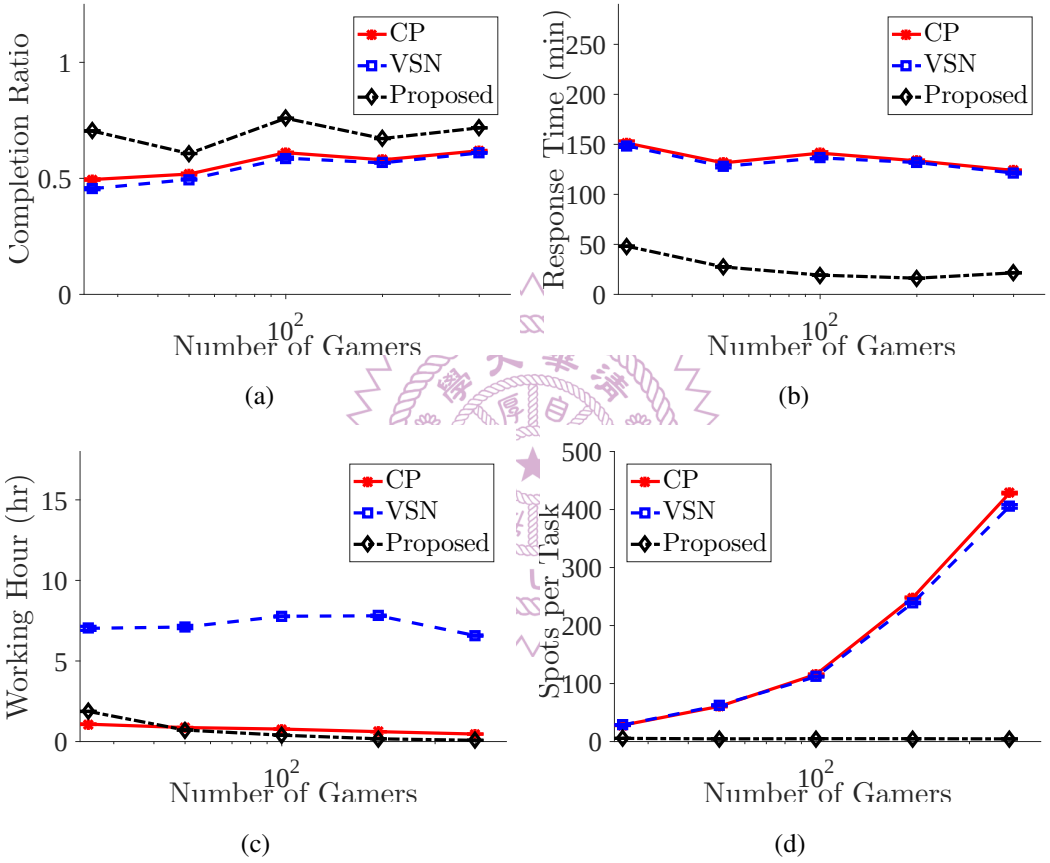


Figure 4.8: Performance comparisons under different numbers of gamers using Pathway Mobility Model: (a) the completion ratio, (b) the response time, (c) the working hour, and (d) the spots per task. The x-axes are in logarithmic scale.

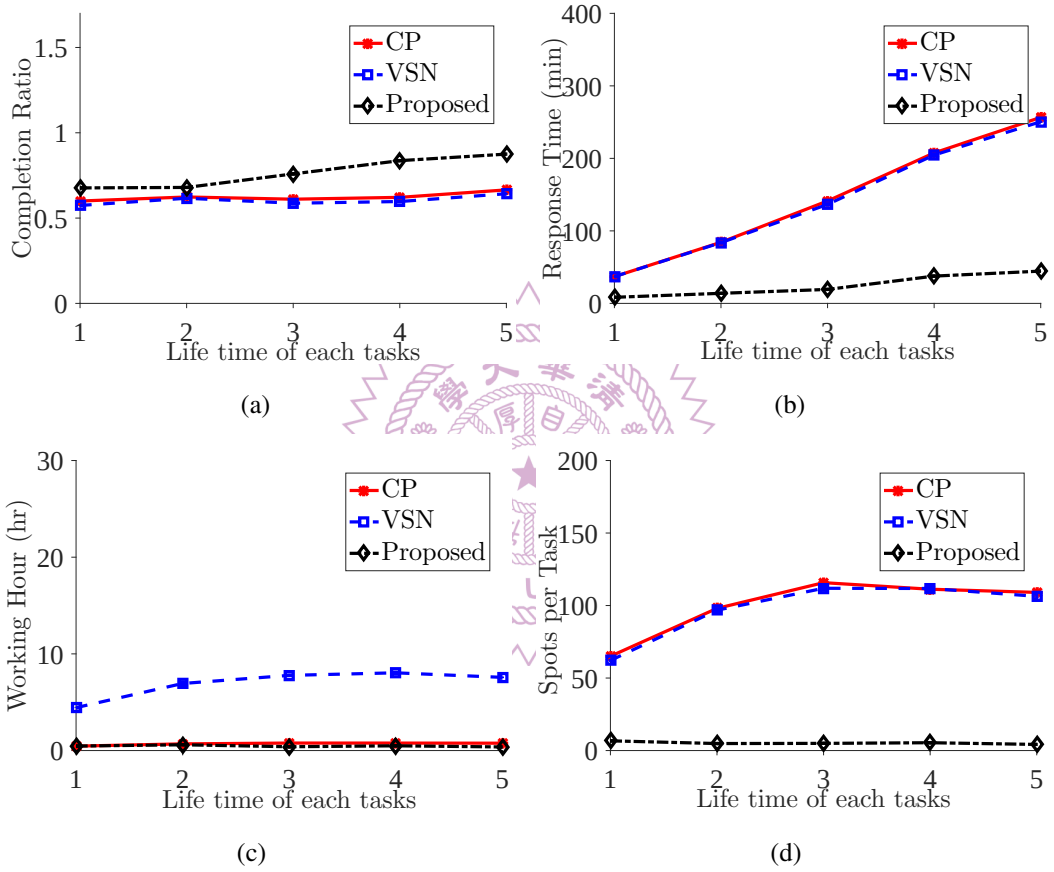


Figure 4.9: Performance comparisons under different life time of tasks using Random Way Point: (a) the completion ratio, (b) the response time, (c) the working hour, and (d) the spots per task.

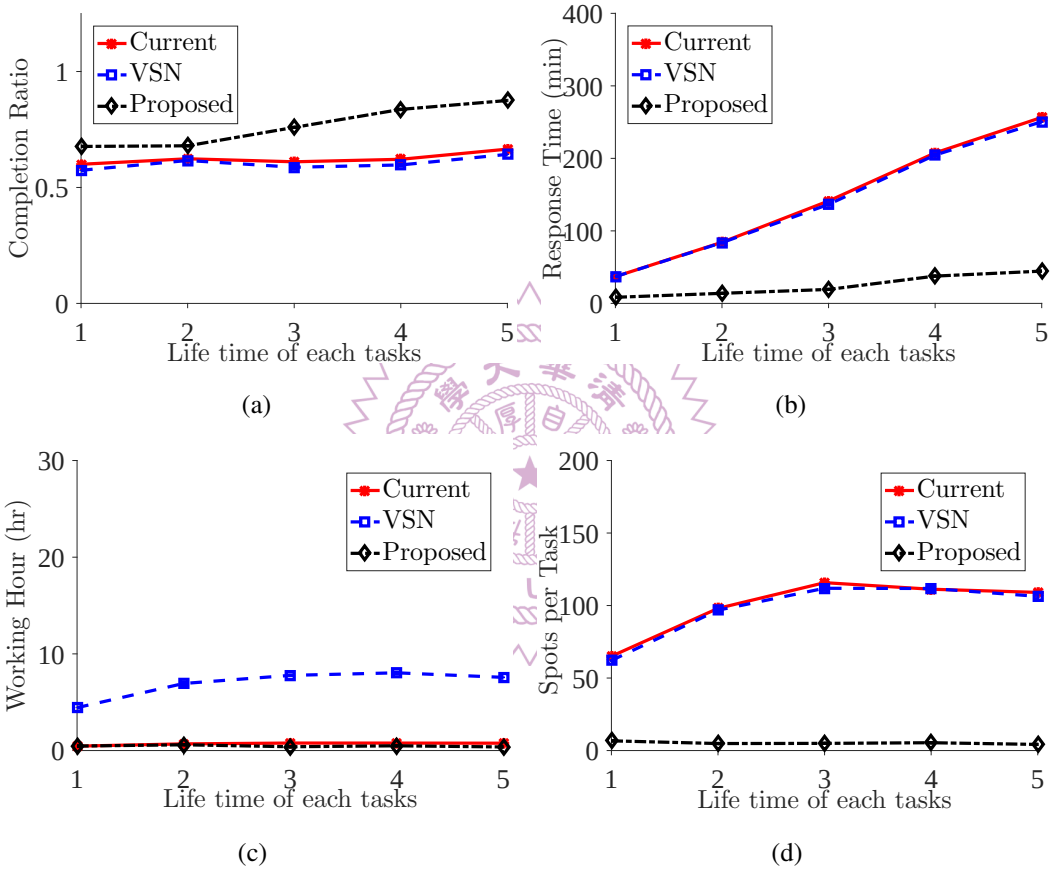


Figure 4.10: Performance comparisons under different life time of tasks using Pathway Mobility Model: (a) the completion ratio, (b) the response time, (c) the working hour, and (d) the spots per task.

Chapter 5

Implementation and User Study

In this section, we first describe the design and implementation of our prototype system, which runs on smartphones and a Linux server. Then, we leverage this prototype system for a user study.

5.1 System Design

Our prototype system contains: (i) mobile app, (ii) broker , and (iii) dashboard.

- *The mobile app* leverages a game engine to simulate and render a 3D world. It also realizes a sensory data collector to record and upload sensory data. Moreover, it implements our proposed NPC path generator.

- *The broker* consists of a database system for storing the tasks, sensory data, and gamer states. The broker also contains a task manager that keeps track of the gamers with assigned sensing tasks. Our two proposed algorithms, spot locator and gamer assigner, are implemented in the broker.

- *The dashboard* is essentially the Web interface for administrator (in our experiments) and smart city application developers (in real deployments) to: (i) submit and configure tasks and (ii) retrieve the sensory data.

5.2 Implementations

The broker is implemented as a daemon on a Linux server. We implement the dashboard on the same Linux server using NGINX [5]. The mobile app runs on Android OS, and connects to the broker using proprietary protocols over TCP connections. Existing AR algorithms [18, 39] are often too heavy for smartphones. Even for high-end smartphones,

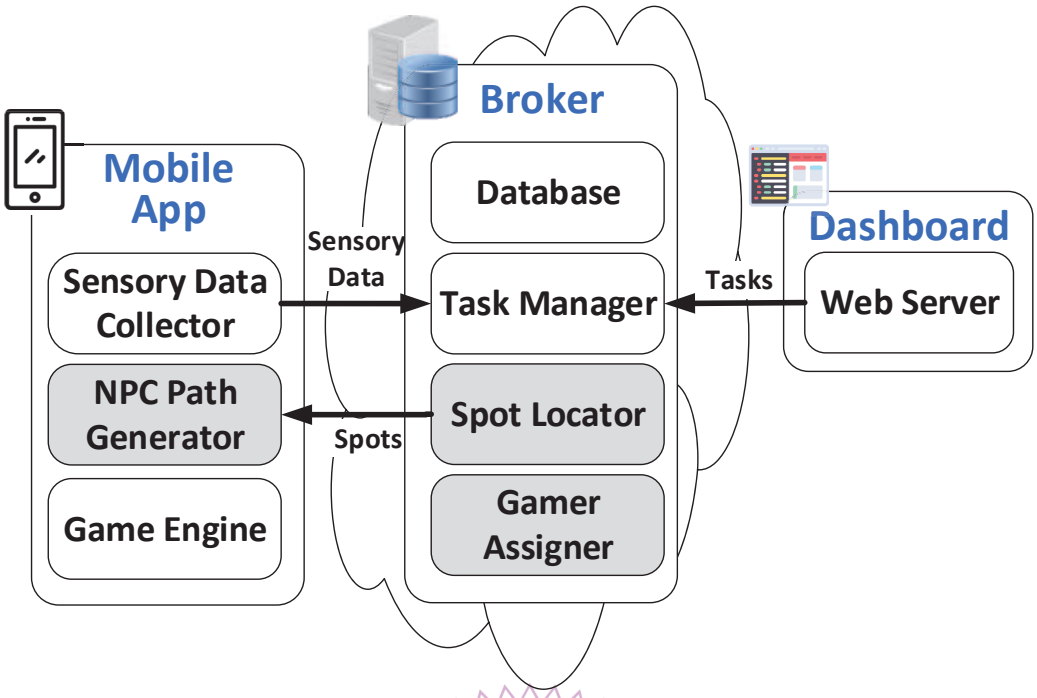


Figure 5.1: The design of our prototype system.

the excessive energy consumption due to intensive computations drains the batteries in no time. Therefore, we opt for light-weight AR heuristics based on smartphone sensors[1]. We benchmark our implementation for the running time of each algorithm. Table 5.1 gives the average running time among 10 runs with 100 gamers and 200 tasks. This table shows that even on a smartphone, the NPC path generator still runs in real time; while the spot locator and the gamer assigner scale to large systems. We adopt MySQL [4] as our database server on the broker. The ordinary app employs Google Map [2] to mark the task locations in Fig. 5.2(a). The gamified app, as shown in Fig. 5.3, is developed by a game engine, called Unity [10]. We adopts a 3D map library, called Go Map [1], for the map mode. Some additional code is added to facilitate the user study, which is detailed below.

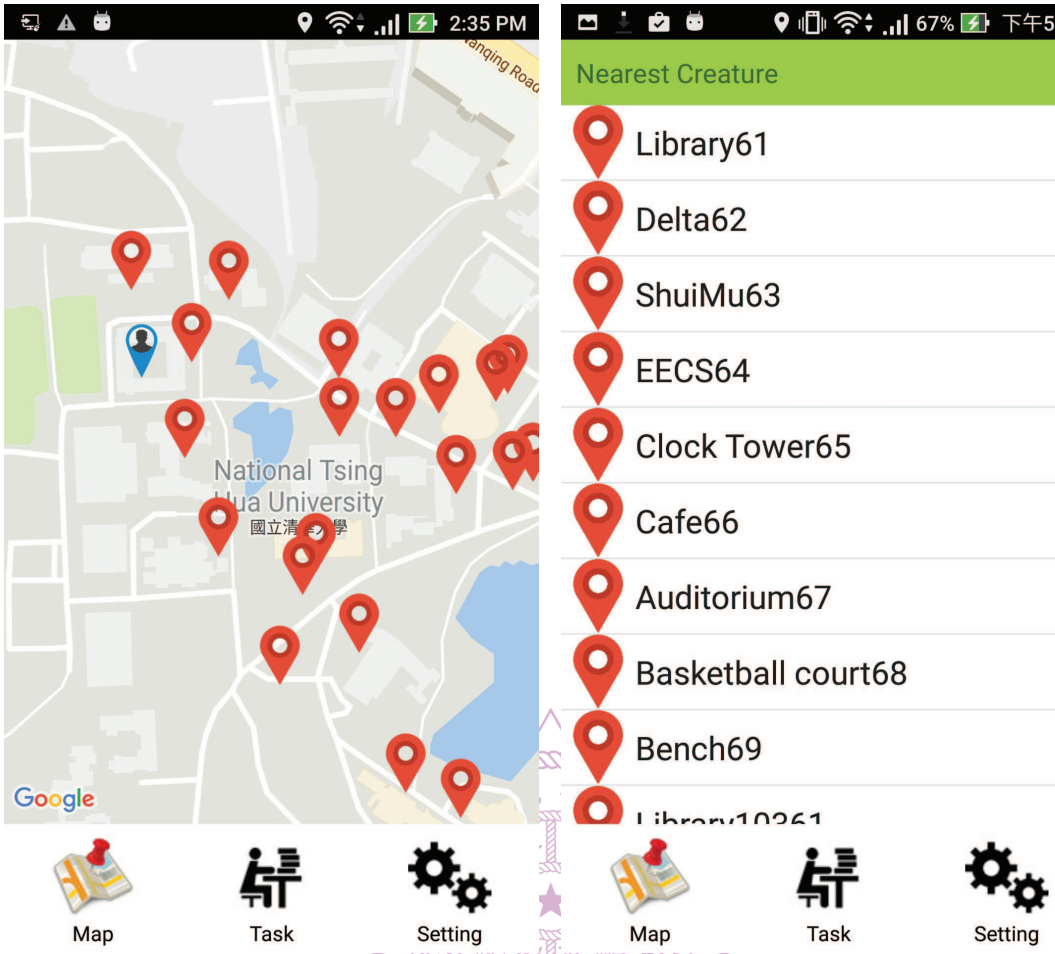
5.3 User Study

We conduct a user study on our campus to compare the two mobile apps of our platform. We recruit 4 gamers in their twenties (50% male) as our subjects and ask them to use both mobile apps for six days (three days each). We note that 3-day experiments may sound too short, but they do not deviate from real deployments too much: it is reported that most

[1]We notice that advanced hardware devices, such as Google Tango [3], may enable higher quality AR on smartphones. Implementing our solution on these devices is one of our future tasks.

users stop using an app after 3-7 days [14]. The gamified app uploads detailed logs to the broker. For example, we highlight sample gamer and NPC trajectories in Fig. 5.5 using dots. This figure shows that our NPC indeed guides the gamer moving forward the task location (at the bottom right corner of the map).

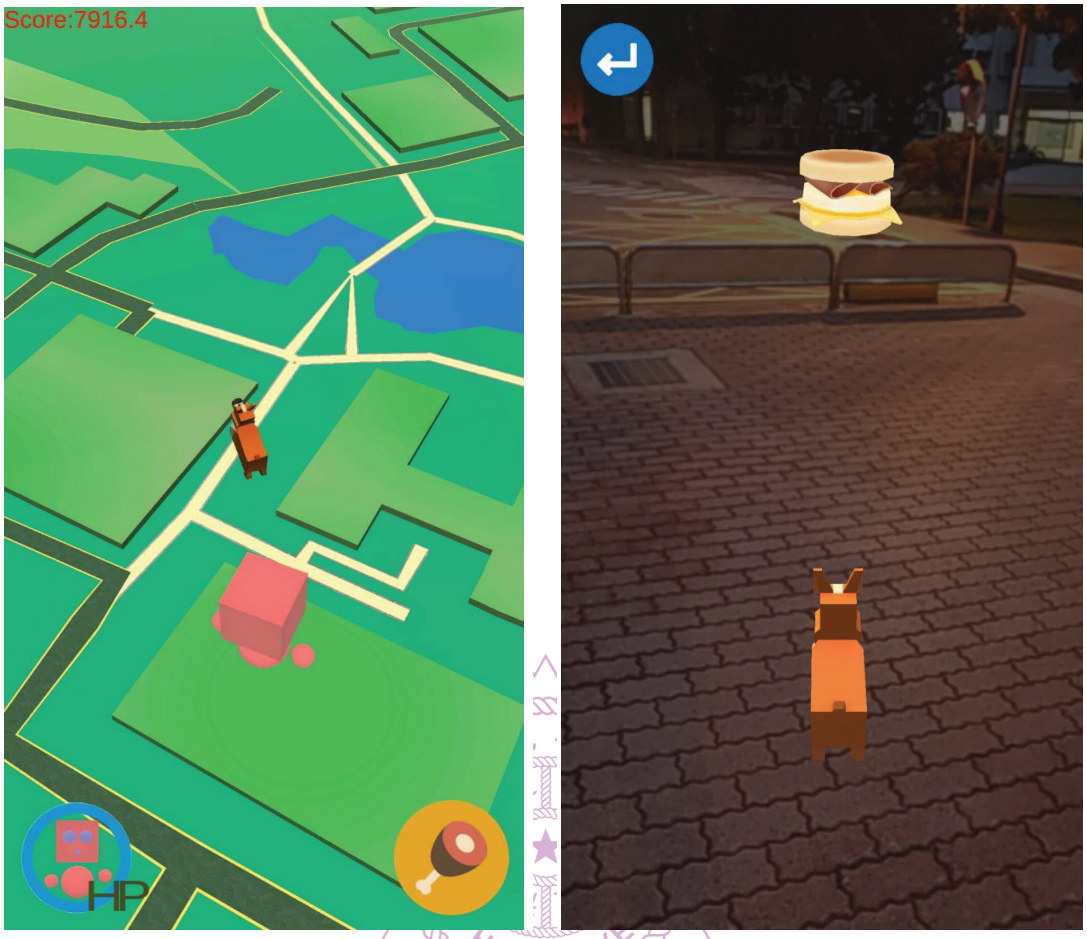
After each run, a gamer fills a questionnaire of 7 questions that are from the well-known Intrinsic Motivation Inventory (IMI) [16]. More specifically, the IMI interval questions are: (1) I enjoy this game, (2) the game is fun, (3) I think the game is boring, (4) playing the game doesn't hold my attention, (5) I would describe the game is interesting, (6) the game is enjoyable, and (7) when I play the game, I think about how much I enjoy it. We use five-point Likert scale (between 1 and 5) to assess the user study results. Fig. 5.4 summarizes the average scores. Notice that questions 3 and 4 are negative questions, i.e., lower scores are better. This figure clearly shows that the gamified app is more enjoyable and attractive.


2:35 PM
National Tsing
Hua University
Google
Map
Task
Setting
67% 下午5:31
Nearest Creature
Library61
Delta62
ShuiMu63
EECS64
Clock Tower65
Cafe66
Auditorium67
Basketball court68
Bench69
Map
Task
Setting

(a) (b)


2:45 PM
CAPTURE
Map
Task
Setting

(c)



(a) (b)

Figure 5.3: Screenshots of the gamified app: (a) the map mode, used when task locations are still far, and (b) the AR mode, used when task locations are close.

Table 5.1: The Running Time (μs) of Our Algorithms

Algorithm	**Running Time**	
	Min	Max
Spot Locator	3161.85	3856.85
Gamer Assigner	55.12	57.93
NPC Path Generator	76.51	121.19

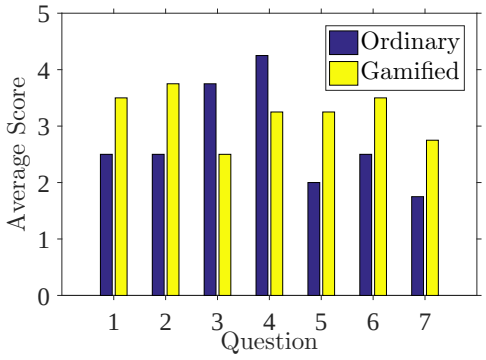


Figure 5.4: Scores from the IMI questions.

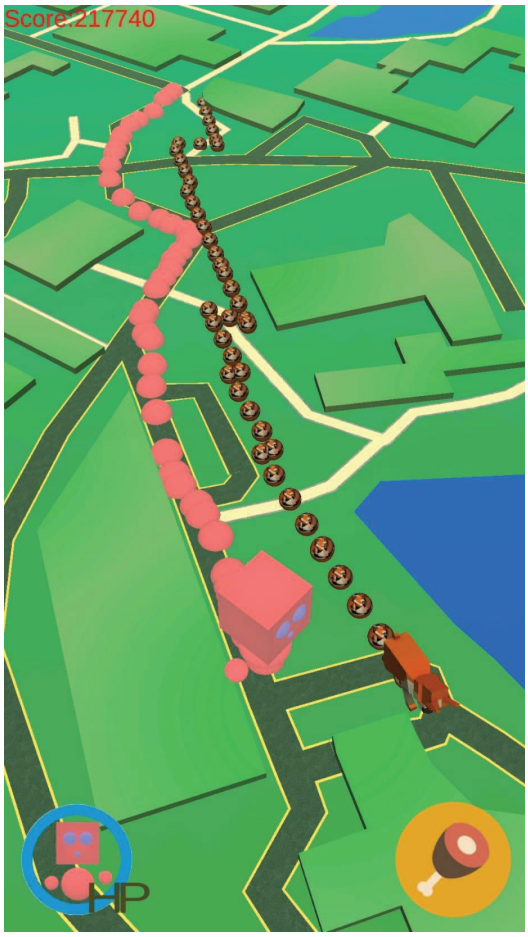


Figure 5.5: The map mode with the gamer's path and NPC's path.

Chapter 6

Conclusion

In this paper, we study the problem of gamifying mobile apps to transparently assign sensing tasks to gamers, so as to create a sustainable Smartphone Augmented Infrastructure Sensing (SAIS) platform. The crux of such a platform lies in three algorithms to: (i) locate the optimal spots for each task, (ii) assign each spot to the closest gamer, and (iii) produce natural NPC path to guide the gamer to the spot. We conduct extensive simulations to evaluate our proposed algorithms, which show that our algorithms provide better quality of service yet incur lower workload on gamers. For example, with 200 tasks and 100 gamers, our algorithms on average: (i) achieves 63% higher completion ratio, (ii) cuts the response time by almost two-third, and (iii) reduces the gamer working hour by 81%, compared to the existing solutions. Moreover, our algorithms scale to many more tasks and efficiently capitalize gamers for better quality of service. We also demonstrate the practicality of our algorithms through a real prototype implementation and a preliminary user study. The user study results are promising: gamers agree that the gamified app is more enjoyable and attractive.

Our work can be extended in several dimensions. In terms of algorithmic design, the proposed algorithms can be further enhanced. For example, our gamer assigner algorithm may take the gamers' routine trajectories into consideration, so that gamers can perform close-by tasks without deviating from their trajectories [25, 27]. We may also compare each algorithm individually in the simulation. In order to see the performance of each algorithm. In terms of systems implementation, several practical concerns need to be considered. For example, uploading videos via cellular networks is expensive and may drive the gamers away from our platform. Mechanisms for filtering out noisy sensory data and compressing sensory data, as well as transferring sensory data in delay-tolerate fashion [21] can be added to our gamified SAIS platform. Last, we note that our gamified app may not need to be a *standalone* mobile game, i.e., we can create an app similar to Poke Radar [7], to piggy back on the popular Pokémon Go game [8]. This will allow us

to focus on the overall platform design, rather than the mobile games themselves.

Bibliography

[1] GO map - 3D map for AR gaming. https://goo.gl/Sf9Y9j, January 2017.

[2] Google map Android API. https://goo.gl/FbBsh3, January 2017.

[3] Google Tango. https://developers.google.com/tango/, January 2017.

[4] MySQL. https://www.mysql.com/, January 2017.

[5] NGINX. https://www.nginx.com/, January 2017.

[6] Openstreetmap. https://www.openstreetmap.org/, January 2017.

[7] Poke radar. https://www.pokemonradargo.com/, January 2017.

[8] Pokemongo. http://www.pokemongo.com/, January 2017.

[9] Smart city market will grow tremendously at a CAGR of close to 20% until 2020, says Technavio. http://tinyurl.com/jcy6nqg, January 2017.

[10] Unity. https://unity3d.com/, January 2017.

[11] F. Alt, A. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz. Location-based crowdsourcing: Extending crowdsourcing to the real world. In *Proceedings of the Nordic Conference on Human-Computer Interaction: Extending Boundaries (NordiCHI'10)*, pages 13–22, Reykjavik, Iceland, October 2010.

[12] J. An, X. Gui, Z. Wang, J. Yang, and X. He. A crowdsourcing assignment model based on mobile crowd sensing in the internet of things. *IEEE Internet of Things Journal*, 2(5):358–369, 2015.

[13] F. Bai and A. Helmy. A survey of mobility models. *Wireless Adhoc Networks. University of Southern California, USA*, 206:147, 2004.

[14] A. Chen. Don't fret! losing 80% of your mobile app users is normal. https://goo.gl/b66zUz, January 2017.

[15] T. Chen, H. Tsai, C. Chen, and J. Peng. Object coverage with camera rotation in visual sensor networks. In *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC'10)*, pages 79–83, Caen, France, 2010.

[16] E. Deci, H. Eghrari, B. Patrick, and D. Leone. Facilitating internalization: The self-determination theory perspective. *Journal of Personality*, 62(1):119–142, 1994.

[17] S. Deterding, D. Dixon, R. Khaled, and L. Nacke. From game design elements to gamefulness: Defining gamification. In *Proceedings of the international academic MindTrek conference: Envisioning future media environments*, pages 9–15, 2011.

[18] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *Proceedings of the European Conference on Computer Vision (ECCV'14)*, pages 834–849, Zurich, Switzerland, 2014.

[19] Z. Feng, Y. Zhu, Q. Zhang, L. Ni, and A. Vasilakos. Trac: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM'14)*, pages 1231–1239, Toronto, Canada, May 2014.

[20] J. Hamari, J. Koivisto, and H. Sarsa. Does gamification work?–a literature review of empirical studies on gamification. In *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS'14)*, pages 3025–3034. IEEE, 2014.

[21] H. Hong, C. Fan, Y. Lin, and C. Hsu. Optimizing cloud-based video crowdsensing. *IEEE Internet of Things Journal*, 3(3):299–313, 2016.

[22] S. Kanhere. Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *Proceedings of IEEE International Conference on Mobile Data Management (MDM'11)*, pages 3–6, Luleå, Sweden, June 2011.

[23] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th international conference on advances in geographic information systems (SIGSPATIAL' 12)*, pages 189–198, New York, NY, USA, November 2012. ACM.

[24] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, September 2010.

[25] C. Liao. Detour planning problem on mobile crowdsensing systems. Master's thesis, Department of Computing Science, National Tsing Hua University, June 2015.

[26] C. Liao, T. Hou, T. Lin, Y. Cheng, A. Erbad, C. Hsu, and N. Venkatasubramania. SAIS: Smartphone augmented infrastructure sensing for public safety and sustainability in smart cities. In *Proceedings of the International Workshop on Emerging Multimedia Applications and Services for Smart Cities (EMASC'14)*, pages 3–8, Orlando, FL, November 2014.

[27] C. Liao and C. Hsu. A detour planning algorithm in crowdsourcing systems for multimedia content gathering. In *Proceedings of the 5th Workshop on Mobile Video (MoVid'13)*, pages 55–60, Oslo, Norway, 2013.

[28] V. D. Luca and R. Castri. The social power game: A smart application for sharing energy-saving behaviours in the city. *FSEA 2014*, 27:4, 2014.

[29] A. McAfee, E. Brynjolfsson, T. Davenport, D. Patil, and D. Barton. Big data. *The management revolution. Harvard Bus Rev*, 90(10):61–67, 2012.

[30] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. In *Proceedings of the Photonics for industrial applications*, pages 282–292, October 1995.

[31] T. Nam and T. Pardo. Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times (dg.o'11)*, pages 282–291, Maryland, USA, 2011.

[32] R. Newcombe, S. Lovegrove, and A. Davison. Dtam: Dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'11)*, pages 2320–2327. IEEE, 2011.

[33] G. Papagiannakis, G. Singh, and N. Magnenat-Thalmann. A survey of mobile and wireless technologies for augmented reality systems. *Computer Animation and Virtual Worlds*, 19(1):3–22, February 2008.

[34] K. Seaborn and D. Fels. Gamification in theory and action: A survey. *International Journal of Human-Computer Studies*, 74:14–31, February 2015.

[35] K. Su, J. Li, and H. Fu. Smart city and the applications. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 1028–1031. IEEE, 2011.

[36] M. Talasila, R. Curtmola, and C. Borcea. Alien vs. mobile user game: Fast and efficient area coverage in crowdsensing. In *Proceedings of International Conference*

on Mobile Computing, Applications and Services (MobiCASE'14), pages 65–74, Austin, TX, 2014.

[37] M. Talasila, R. Curtmola, and C. Borcea. Crowdsensing in the wild with aliens and micropayments. *IEEE Pervasive Computing*, 15(1):68–77, Jan 2016.

[38] C. Tan and D. Soh. Augmented reality games: A review. *Proceedings of Gameon-Arabia, Eurosis*, 2010.

[39] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg. Real-time panoramic mapping and tracking on mobile phones. In *Proceedings of the IEEE Virtual Reality Conference (VR'10)*, pages 211–218, Boston, MA, March 2010.

[40] G. Welch and G. Bishop. Scaat: Incremental tracking with incomplete information. In *Proceedings of the annual conference on Computer graphics and interactive techniques*, pages 333–344, 1997.

[41] M. Yuen, I. King, and K. Leung. A survey of crowdsourcing systems. In *Proceedings of the Privacy, Security, Risk and Trust (PASSAT'11) and IEEE Inernational Conference on Social Computing (SocialCom'11)*, pages 766–773, Boston, MA, October 2011.

[42] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: Concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology*, 5(3):38:1–38:55, September 2014.