The Pennsylvania State University
The Graduate School

RESOURCE-AWARE CROWDSOURCING IN WIRELESS NETWORKS

A Dissertation in
Computer Science and Engineering
by
Yibo Wu

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2018

The dissertation of Yibo Wu was reviewed and approved* by the following:

Guohong Cao
Professor of Computer Science and Engineering
Dissertation Adviser, Chair of Committee

Robert Collins
Associate Professor of Computer Science and Engineering

Sencun Zhu
Associate Professor of Computer Science and Engineering
Associate Professor of Information Sciences and Technology

Zhenhui Li
Associate Professor of Information Sciences and Technology

Bhuvan Urgaonkar
Associate Professor of Computer Science and Engineering
Graduate Program Chair of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

The ubiquity of mobile devices has opened up opportunities for a wide range of applications based on photo/video crowdsourcing, where the server collects a large number of photos/videos from the public to obtain desired information. However, transmitting large numbers of photos/videos in a wireless environment with bandwidth constraints is challenging, and it is hard to run computation-intensive image processing techniques on mobile devices with limited energy and computation power to identify the useful photos/videos and remove redundancy. To address these challenges, we propose a framework to quantify the quality of crowdsourced photos/videos based on the accessible geographical and geometrical information (called metadata) including the orientation, position, and all other related parameters of the built-in camera. From metadata, we can infer where and how the photo/video is taken, and then only transmit the most useful photos/videos.

The goal of this dissertation is to design and evaluate metadata-based techniques to support resource-aware crowdsourcing in wireless networks. First, we propose a metadata-based framework called SmartPhoto which can select the most useful photos to cover the specified points of interest. Optimization problems regarding the tradeoffs between photo coverage and resource utilization are formalized and efficient algorithms with provable performance bounds are designed and evaluated. Furthermore, SmartPhoto has been evaluated through real experiments with Android smartphones, and various techniques have been designed to improve the accuracy of the collected metadata. Second, for some applications, the target of interest is an area rather than a point. Thus, we extend SmartPhoto to select the most useful photos to cover an area of interest. Since there are infinite number of points in an area and each point can be covered differently, area coverage is hard to analyze. To address this problem, we study how to select photos with the best area coverage under resource constraints and propose an efficient algorithm with provable performance bound. Third, for applications such as disaster recovery and

battlefield, the cellular network may be partly damaged or severely overloaded, and thus photos have to be delivered through Disruption Tolerant Networks (DTNs). In DTNs, data can only be transferred when two devices are within the wireless transmission range of each other, and the storage of each device is also limited. Therefore, it is important to prioritize more valuable photos to use the limited resources. We quantify the value of the photos based on their metadata, and propose a distributed photo selection algorithm to maximize the value of the photos delivered to the server considering bandwidth and storage constraints. Finally, we propose VideoMec, a crowdsourcing system which organizes the metadata of all videos taken by mobile devices. VideoMec supports comprehensive queries for any application to find and fetch desired videos from the corresponding mobile devices. For time-critical applications, it may not be possible to upload all desired videos in time due to limited wireless bandwidth and large video files. Thus, we propose efficient algorithms to select the most important videos or video segments to upload under bandwidth constraints. VideoMec has been implemented and experimental results have demonstrated its effectiveness on organizing and retrieving mobile videos from mobile devices with resource constraints.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First of all, I would like to express my sincere gratitude to my adviser Prof. Guohong Cao, for the patient encouragement and advice he has provided throughout my time as his student. I have been extremely lucky to have an adviser who cares so much about my research, my professional development, and my life overall. I have benefited a lot from his multidisciplinary expertise and visionary thinking. I could not have completed my Ph.D. study without his guidance and support.

Besides my adviser, I would also like to thank the rest of my doctoral committee: Prof. Robert Collins, Prof. Sencun Zhu, and Prof. Zhenhui Li. It is my great privilege to have these wonderful professors in my doctoral committee. Without their generous help and insightful comments, the dissertation could not have reached the present form.

My sincere thanks also goes to my co-authors Dr. Yi Wang, Dr. Wenjie Hu, Dr. Xiaomei Zhang, and Dr. Xiao Sun in the Mobile Computing and Networking Lab, for their hard work, stimulating discussions, and valuable comments on our published papers. I would also like to thank all my fellow labmates, especially Yi Yang and Li Qiu, for their sincere friendship and warm encouragement throughout the past five years. They have made my years of study at Penn State an enjoyable and memorable journey.

Last but not least, I am deeply indebted to my parents and my wife for their unconditional love. They have brought me the passion for science, and have given me the strongest support under all circumstances. This dissertation is simply impossible without their love and support. For that, I dedicate this dissertation to them.

This work was supported in part by the National Science Foundation (NSF) under grants CNS-1421578 and CNS-1526425, and by the US Army Research Laboratory under grant W911NF-09-2-0053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or the Army Research Laboratory.

Dedication

To my beloved parents and wife.

Chapter 1

Introduction

In the past decade, taking photos/videos using smartphones and tablets has become a common practice in people's daily life. In the near future, wearable devices such as smartglasses will make photo/video taking more user-friendly. The proliferation of all these mobile devices will enable a wide range of applications based on photo/video crowdsourcing, where a central server collects a large amount of photos/videos from the public to obtain desired information. Such applications include grassroots journalism [1], law enforcement [2], disaster recovery [3, 4], street view service [5, 6], and many others.

Consider the following example. A city is under a state of emergency due to external attacks or natural disasters. To find out where the damage is and how severe it is, first-hand on-site videos taken by people using mobile devices become extremely useful. There may be lots of videos taken by mobile users, and the communication infrastructure may be severely damaged. Thus, the wireless transmission of videos from mobile users to the central service or central authority will be limited by the bandwidth constraints, and it is a challenge to efficiently utilize the limited bandwidth to find and collect the most useful videos.

Street view service can also benefit from crowdsourcing. Areas like university campuses, theme parks, and outlets are inaccessible to street view cars [7] and thus often do not have street view. Areas under development may have totally different views after one year and regularly updating these views can be very costly. With crowdsourcing, millions of photos taken by visitors can be collected, processed, and embedded into maps to provide street-view-like service for areas where traditional

street view is not available or out of date. Unfortunately, the sheer amount of photos poses big challenges for image processing and recognition. Fully understanding the semantic of each photo by resource-intensive image processing techniques [8, 9] would be a luxury if not impossible. Thus, it is a challenge to analyze photos quickly and select those that provide the best coverage over the area.

In the above two applications, the biggest challenge is the resource limitation in wireless environments. The limited bandwidth or computing power reduces the amount of data that can be crowdsourced from the public, and thus reduces the performance of crowdsourcing applications. Therefore, it is important to develop *resource-aware* crowdsourcing solutions for these applications.

Some existing techniques may be applied to address this challenge. For example, some content-based image retrieval techniques [9, 10] can be applied to push the computing to the user end. Users are asked to download a special piece of code or manually validate image content. Although this can address the computing limitation of the server, it may discourage user participation and may not be suitable for disaster recovery scenarios where mobile devices have limited resources (i.e., computing power and energy) to execute those programs. As another solution, description-based techniques can be applied to annotate photos by tags [11] or GPS locations [12]. However, tags may be inaccurate or insufficient for applications to determine the coverage/content of photos. Some tags require user inputs and then discourage user participation. Moreover, location itself is not enough to reveal the view of the camera. Even at the same location, cameras facing different directions have different views.

To address all the above issues, we propose to use various geographical and geometrical information, called *metadata*, to characterize photos/videos. For photos, metadata includes camera location, orientation, field of view, coverage range, and other parameters such as focal length, which are accessible from the built-in sensors of most off-the-shelf mobile devices. Whenever a photo is taken, its metadata is automatically generated and recorded. From metadata, we can determine where and how the photo is taken and infer which area is covered by the photo without looking at the image content. For videos, metadata (e.g., orientation) constantly changes during video recording and thus needs to be recorded periodically as the video is being taken.

With metadata, photo/video crowdsourcing becomes much more resource-friendly. When a crowdsourcing event happens, the server first asks users to upload metadata, and selects photos/videos based on the uploaded metadata and the application needs (e.g., resource constraints). Then users upload the selected ones and get paid [13, 14, 15] (how users are paid is out of the scope of this dissertation). During this process, the unselected photos/videos (except their metadata) will not be transmitted to the server, thus saving bandwidth and energy on mobile devices. All the photos/videos are analyzed based on their metadata rather than actual images, which significantly reduces computational cost and delay. Based on the idea of metadata, we design a series of models and algorithms to support resource-aware crowdsourcing in a variety of application scenarios.

1.1 Focus of This Dissertation

The goal of this dissertation is to design and evaluate metadata-based techniques to support resource-aware crowdsourcing in wireless networks. Specifically, we focus on four scenarios, i.e., resource-aware photo crowdsourcing for point coverage, resource-aware photo crowdsourcing for area coverage, resource-aware photo crowdsourcing through disruption tolerant networks, and a metadata-enhanced crowdsourcing system for mobile videos. We briefly explain them in the following four subsections.

1.1.1 Resource-Aware Photo Crowdsourcing for Point Coverage

In photo crowdsourcing applications, the targets to be covered can be considered as points on the map, which can be buildings, intersections, or other landmarks. Intuitively, a target is well covered if there are multiple photos providing multiple views of the target from different angles. For example, a building is well covered if there are four photos viewing it from north, south, east, and west. This is dramatically different from the sensor coverage model in traditional wireless sensor networks, where the target is well covered as long as it is within the sensing range of one sensor. Due to this unique property of photo coverage, the models and

algorithms for traditional sensor coverage cannot be applied, and new ones need to be developed specifically for photo coverage.

To this end, we define *photo utility* to measure how well a set of photos cover the targets based on how many different angles they cover, which can be calculated from photo metadata. Then, to address the challenges of resource constraints, three optimization problems regarding the tradeoffs between photo utility and resource usage, namely the max-utility problem, the min-selection problem, and the min-selection problem with k-coverage, are formulated and studied. There problems are proved to be NP-hard, and we propose efficient approximation algorithms to solve them. We show the theoretical performance bounds of the proposed algorithms, and demonstrate their effectiveness through both real-world experiments and extensive simulations.

1.1.2 Resource-Aware Photo Crowdsourcing for Area Coverage

In some crowdsourcing applications, the target of interest is an area rather than a point. For example, when crowdsourcing for street view photos, the target area should be covered as much as possible and for each point covered, the photos should be taken from multiple angles. Since there are infinite number of points in the area, and each point can be covered by different photos from different angles, area coverage is much harder to analyze compared to point coverage.

To address the issue, we first extend the definition of *photo utility* so that it works for area coverage. Then we study how to efficiently analyze area coverage and calculate photo utility based on metadata. Next, we study an optimization problem which maximizes the utility of selected photos under resource constraints. The problem is proved to be NP-hard and an efficient approximation algorithm with guaranteed performance bound is proposed. Finally, the proposed algorithm is evaluated with both real-world experiments and extensive simulations.

1.1.3 Resource-Aware Photo Crowdsourcing Through Disruption Tolerant Networks

Some crowdsourcing applications have bad network conditions. For example, in a natural disaster or battlefield, the communication between the command center and mobile users (e.g., rescuers, survivors, and soldiers) can be extremely constrained. The cellular network may be damaged or overloaded with extensive requests, and human-carried satellite radios are only available to a small portion of users due to the high cost. As a result, we have to use Disruption Tolerant Networks (DTNs) to transfer photos among mobile users, and once available, use the cellular network or satellite connections to upload photos to the command center.

With DTN, selecting useful photos based on metadata becomes more challenging. The usefulness of a photo depends not only on its coverage obtained from metadata, but also on the probability that it is delivered to the command center via opportunistic DTN links. To address this challenge, we use *expected coverage*, which combines photo coverage and delivery probability, to measure the usefulness of photos. Every time two users move near each other and establish a DTN link, they transmit photos between each other to maximize the expected coverage of their photos. This way, useful photos can be quickly transmitted through DTN and delivered to the command center. The proposed design is evaluated with a real-world demo and extensive simulations using two large-scale DTN contact traces.

1.1.4 A Metadata-Enhanced Crowdsourcing System for Mobile Videos

Our previous studies focus on selecting photos to best cover some interested targets or target areas. The targets or target areas are given in *one* crowdsourcing task and the crowdsourced photos are only used for that task. To make crowdsourcing even more efficient, we design a system that collects the information (metadata) of all videos taken by mobile devices and makes it accessible to *all* crowdsourcing tasks. More specifically, whenever a video is taken by mobile devices, its metadata is automatically uploaded to a central server and stored in a database. Then, any crowdsourcing application can query the database to find desired videos and

retrieve them from the corresponding mobile devices. We design the system for videos because they are more complex than photos and a system that can handle videos will also be able to handle photos.

To build such system, we first design efficient methods to index and query all the collected video metadata, which is expected to grow exponentially with the proliferation of mobile devices. Then, after the query, it is possible that mobile devices cannot upload all requested videos due to limited wireless bandwidth, large video files, and the urgent need of time-critical applications. Thus, we design efficient algorithms based on metadata to select the most important videos or video segments to upload first. Finally, we implement a prototype of the proposed crowdsourcing system, and demonstrate its effectiveness based on real-world experiments.

1.2 Organization

The remainder of the dissertation is organized as follows. In the next chapter, the related work is discussed. Chapter 3 introduces photo utility for point coverage and studies the optimization problems regarding photo utility and resource usage. Chapter 4 extends photo utility to area coverage, and then studies how to efficiently calculate the utility and use it to select photos under resource constraints. Chapter 5 presents techniques on resource-aware photo crowdsourcing through disruption tolerant networks. Chapter 6 presents the proposed metadata-enhanced crowdsourcing system for mobile videos. Finally, we conclude the dissertation and discuss the future work in Chapter 7.

Chapter 2

Related Work

2.1 Photo Crowdsourcing

The mass adoption of mobile devices makes photo taking and sharing via online social networks much easier and more enjoyable. It creates opportunities for many applications based on photo crowdsourcing, which have received much attention recently in research. For example, in [12], the authors proposed to build photo annotated world maps and create 3D models of the objects from 2D photos via online social networks. Zhai *et al.* investigated the efficient use of crowdsourced photos in post-earthquake recovery scenarios [4]. In [16], the authors envisioned the possibility of using crowdsourced photos and associated sensor data to improve outdoor localization. The idea was further extended by Jigsaw [17], IndoorCrowd2D [18, 19, 20], and iMoon [21, 22] to achieve indoor navigation and floor plan reconstruction.

In addition, several image related crowdsourcing frameworks have been proposed. CrowdSearch [10] asks human workers to manually look at images and validate their content to search for desired images. MediaScope [23] uses content-based features such as color and texture to search for images on mobile devices. These content-based approaches involve power-intensive computation at both user and server end, and the authors had to reduce image resolution to lower the computational cost at the expense of accuracy. The authors mentioned that their solution could also be used for videos. However, when applied to videos, such computational cost will be much higher because each video has a large number of frames

to be processed (even after sampling). Compared to content-based approaches, which are challenged by the content diversity and the resource constraints, our metadata-based approach can effectively characterize photos while using very little bandwidth, storage, and computing resources.

2.2 Coverage in Camera Sensor Networks

Camera sensor networks have been extensively studied in recent years. [24, 25, 26, 27, 28, 29, 30, 31, 32]. One basic problem is the coverage problem, e.g., whether some given camera sensors can cover the interested area or how to optimally place some camera sensors to do so. One coverage problem studied is called pan and scan [33], which is proposed to maximize the total coverage in a camera sensor networks. For camera sensor placement, various optimization models and heuristics are studied in [34]. However, their coverage model is relatively simple. An object is considered to be well covered as long as it is within the coverage area of one photo, which does not consider the uniqueness of photo coverage.

Our work is inspired by the full-view coverage model which was originally proposed in [35] and later extended in [36, 37]. An object is considered to be full-view covered if no matter which direction the object faces, there is always a camera capturing its front image, or more specifically, there is always a camera whose coverage range includes the object and whose viewing direction is sufficiently close to the object's facing direction. This means that multiple cameras are around the object and they provide the highest degree of coverage on the object. This idea inspires lots of follow-up work [38, 39] because the proposed model guarantees to capture the object's "face", which is especially useful in camera surveillance systems. However, in the full-view coverage model, an object is either full-view covered which implies the highest degree of coverage, or not. If an object is not full-view covered, it is not clear how well the coverage is. Compared to their model, the photo utility proposed in this dissertation is more general, because it measures the degree of coverage as a continuous function based on how many different directions are covered. More importantly, we conduct real-world experiments to demonstrate that our utility metric can truly reflect the coverage of real photos.

2.3 Routing in Disruption Tolerant Networks

Since cellular networks may not be available in applications such as disaster recovery and photos may need to be delivered through DTN, this research is also related to DTN routing. Early works in DTN routing assume that packets are equally important, and thus do not differentiate packets based on their content [40, 41, 42, 43]. Those works are not suitable for disaster recovery scenarios, where the value of each photo is different and must be considered in routing. More recent works consider the utility of packets and prioritize packets with high utility in routing decisions [44, 45, 46]. For example, [44] uses several variations of delay as the utility; [46] uses the average delay and the average delivery rate as the utility; and [45] routes prefetched webpages to DTN nodes, so it uses the probability that a link is actually clicked as the utility. A common property of the above utility metrics is that the utility of one packet is not explicitly affected by the presence of other packets. However, the utility metric in our work, photo coverage, is different. The coverage of a photo can change dramatically depending on the presence of other similar photos. Hence, existing utility-based approaches are not suitable for maximizing photo coverage.

PhotoNet [47] and CARE [48] are two closely related papers as they also study photo delivery in disruption tolerant networks. PhotoNet is a picture delivery service that prioritizes the transmission of photos by considering the location, time stamp, and color difference, with the goal of maximizing the "diversity" of the photos. Compared to their model, we consider direction and angle information, and develop techniques to obtain them from off-the-shelf mobile devices. These are very important and unique features for photos and enable us to develop much finer optimization models. Moreover, the solutions to our optimization problems are rigorously analyzed. CARE leverages image similarity detection algorithms to eliminate similar-looking photos, and thus increase the delivery capability of DTN. While those algorithms can detect similarity in terms of pixels, they do not necessarily detect similarity in terms of content or semantics. In comparison, our metadata-based model can detect a broader scope of similarity at a much lower computational cost, which is critical in resource constrained DTN scenarios.

2.4 Video Crowdsourcing

Existing works on video crowdsourcing require all video files to be uploaded to the cloud or cloudlets. For example, Chen *et al.* [49] presented a cloud leasing strategy where geo-distributed users watch live streaming videos crowdsourced from geo-distributed contributors. Simoens *et al.* [50] proposed a video crowdsourcing framework where video files are uploaded to the cloudlets (rather than the cloud). Based on the cloudlet framework, Bohez *et al.* [5] presented a live street view service. However, none of them uses fine-grained sensor data to characterize crowdsourced videos and reduce resource consumptions.

Applications based on mobile videos and sensor data have been proposed for various domains. For example, MoVi [51] uses various sensors to automatically detect interesting events in a social group and stitches video clips from multiple devices to produce event highlights. FOCUS [52] clusters mobile videos taken in a small area (e.g., a stadium) by camera orientation, and thus recognizes videos capturing the same event even when they have different viewing angles. Kim *et al.* [53] studied how to select frames from mobile videos, based on camera location and orientation, to generate high quality panoramic images. Lee *et al.* [54] studied the use of vehicular sensing platforms, i.e., vehicles equipped with cameras and sensors, in public safety and crime investigation applications.

Different from these existing works, we propose a general crowdsourcing system which organizes and indexes all videos taken by mobile devices based on metadata. The system supports comprehensive queries for a variety of applications to find and fetch desired videos while reducing resource consumption at both the server and the user side.

Chapter 3

Resource-Aware Photo Crowdsourcing for Point Coverage

3.1 Background

As mentioned in the introduction, metadata can be used to characterize photos and select photos that best cover interested targets. Intuitively, a target can be seen as a point on the map, and a good photo coverage should have multiple views of the target (point) and cover as many *aspects* as possible. Specifically, given a set of targets and photos, we consider an aspect of a target to be properly covered if it is within a proper range of a photo's viewing direction (defined in Section 3.2). Then we measure the quality of a photo by *utility*, which indicates how many aspects are covered. The utility is calculated based on the metadata, which can be practically obtained via various embedded sensors in most off-the-shelf mobile devices. They are independent of the image content, and hence the computation is very fast and resource-friendly compared to traditional content-based approaches.

With the above model, we address challenges brought by the resource constraint, which is referred to as the *Max-Utility problem*. Resource constraint of bandwidth, storage and processing capability limits the number of photos that can be uploaded to the server. Given the metadata of the candidate photos, how to find a given number of photos such that the total utility is maximized? Note that this is different from traditional maximization problems on sensor coverage

in which a target is covered as long as it is inside the sensing range. Here photos taken at different view points cover different aspects of the target. The total utility depends on how many aspects can be covered and how they are covered, which makes the problem unique and complicated.

Another challenge to be addressed is how to remove the redundancy and find the most representative photos. In general, the amount of candidate photos is significant and redundancy occurs if multiple photos are taken at similar locations and from similar angles. The less number of photos is selected, the less amount of bandwidth, storage and processing capability is needed. In the *Min-Selection problem*, given the coverage requirements of the targets, we want to find the minimum set of photos that satisfy the requirements. We also consider having certain level of redundancy in case better coverage is needed.

In this chapter, we propose *SmartPhoto*, a novel framework to evaluate and optimize the selection of crowdsourced photos, based on the collected metadata from mobile devices. We formulate the Max-Utility problem for bandwidth constrained networks, and then extend it into an online optimization problem. We study the Min-Selection problem for redundancy reduction, and also extend it to the case where better coverage (e.g., k-coverage) is desired. Moreover, we propose efficient solutions, and find the performance bounds in terms of approximation or competitive ratios for the proposed algorithms. We have implemented SmartPhoto in a testbed using Android based smartphones. We make use of multiple embedded sensors in off-the-shelf smartphones, and propose a series of methods to fuse data, correct errors, and filter out false information, to improve the accuracy of the collected metadata. Finally, the performance of the proposed algorithms are evaluated through real implementations and extensive simulations.

3.2 Models and Notations

Consider a post-disaster scenario in which a set of predefined targets are to be monitored by a group of people or reporters. They use mobile devices to take photos and transfer them back to the processing center. However, only a small number of photos can be transferred due to the limited bandwidth caused by damage to base stations or overwhelming cellular traffic. For this reason, reporters

first transmit the metadata of the photos, which is extremely lightweight compared to the original images. After that, the server runs optimization algorithms to determine what photos to be actually transferred and notifies the reporters to transmit the photos.

We first describe the models used in SmartPhoto to characterize targets and photos. Then the concept of utility is introduced. The idea is based on the observation that a good photo should cover as many aspects of the targets as possible. For an aspect to be properly covered, the target should be in a photo whose viewing direction is not too far away from the direction to which the aspect points. This is similar to the face recognition problem in computer vision: as the angle between the object's facing direction (the aspect) and the camera's viewing direction (the vector from the camera to the object) becomes wider, the detection rate of the recognition algorithm will drop dramatically [55, 56]. The utility defined here precisely indicates how many aspects of the target are properly covered.

3.2.1 Targets and Photos

At the beginning of each event, the application server distributes the information of the interested targets to the public users. The set of targets are denoted by $T = \{T_1, \ldots, T_m\}$. T_i also represents the location of the i-th target if there is no ambiguity. An *aspect* of the target, denoted by $\vec{v}$, is a vector that can be represented by an angle in $[0, 2\pi)$ with 0 degree indicating the one pointing to the right (east on the map). For ease of presentation, this angle is denoted by $arg(\vec{v})$ and is calculated by using arithmetic modulo 2π.

Given a set of photos: $P = \{P_1, \ldots, P_n\}$, each photo P_j is stored locally and it can be registered to the server with a tuple $(l_j, r_j, \varphi_j, \vec{d_j})$, called the *metadata* of the photo. Here l_j is the location where the photo is taken. To simplify notations, we also use P_j to represent the location if there is no ambiguity. r_j and φ_j are two internal parameters of the camera used to take the photo. r_j is the effective range of the camera, and φ_j is the field-of-view (FoV, represented in angle) of the camera lens. $\vec{d_j}$ is the orientation of the camera when the photo is taken. Note that $\vec{d_j}$ is the normal vector derived from the camera lens and vertical to the image plane. It can be acquired by using various sensors embedded in the mobile device. Details

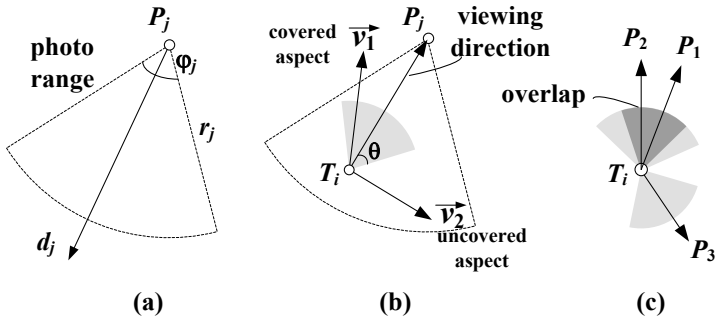


Figure 3.1. (a) Metadata defines the effective coverage range of a photo. (b) An aspect is covered if it is close to the viewing direction. (c) Coverage overlap shows the redundancy in the photos.

of obtaining these geographical information will be given in Section 3.6. As shown in Figure 3.1(a), the metadata defines the effective coverage range of the photo.

3.2.2 Photo Utility

For a target T_i and a photo P_j, T_i is said to be covered by P_j if P_j's range includes T_i. An aspect $\vec{v}$ of T_i is covered if the angle between $\vec{v}$ and $\overrightarrow{T_iP_j}$ is smaller or equal to a predefined angle θ called *effective angle*. Here $\overrightarrow{T_iP_j}$ is the viewing direction of the camera towards the target when the photo is taken[1]. Further, the utility of a photo P_j can be defined based on how many aspects of T_i are covered by this photo.

Definition 1. *[Utility] Given a target T_i and a photo P_j covering the target, the utility of P_j on T_i, denoted by $U_{P_j}(T_i)$, is the portion of aspect that is covered by P_j, i.e., $U_{P_j}(T_i) = \int_0^{2\pi} 1_{P_j}(v)dv$, where $1_{P_j}(v) = 1$ if $\vec{v}$ is covered by P_j, or 0 otherwise.*

Accordingly, the utility of a set of photos $P' = \{P_j : 1 \leq j \leq k\}$ regarding target T_i is the total portion of aspect that is covered by the photos of P', i.e., $U_{P'}(T_i) = \int_0^{2\pi} 1_{P'}(v)dv$, where $1_{P'}(v) = 1$ if $\vec{v}$ is covered by any P_j from P', or 0 otherwise.

Finally, the total utility of the photos regarding all targets $T = \{T_1, \ldots, T_m\}$ is the sum of the utility regarding each target. It is normalized by dividing the total number of targets, i.e., $U_{P'}(T) = \frac{1}{m}\sum_{i=1}^{m} U_{P'}(T_i)$.

For example in Figure 3.1(b), target T_i is covered by photo P_j. Its aspect

[1]Intuitively, it should be from P_j to T_i, but $\overrightarrow{T_iP_j}$ is used for ease of calculation.

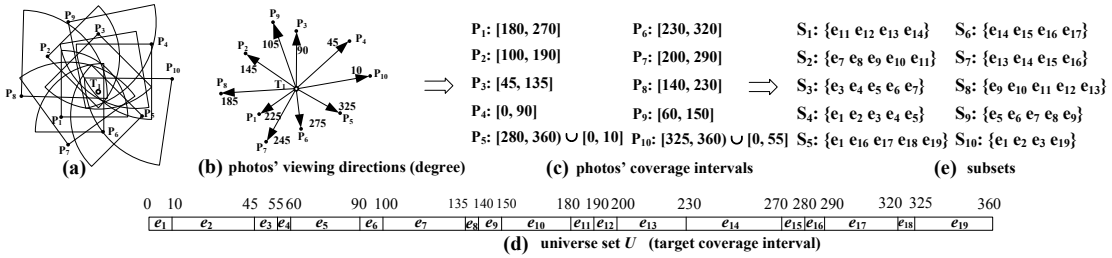


Figure 3.2. The conversion into a set system.

$\vec{v_1}$ is covered by P_j but aspect $\vec{v_2}$ is not. In fact, P_j covers all the aspects in $[arg(\overrightarrow{T_iP_j}) - \theta, arg(\overrightarrow{T_iP_j}) + \theta]$ (gray sector in Figure 3.1(b)). Thus, P_j's utility is 2θ. If there are multiple photos covering the same target, possible overlap (darker area in Figure 3.1(c)) among photos' coverage needs to be identified and removed. In that case, the overlap can only be counted once towards the total utility, which is reflected by the gray area in Figure 3.1(c).

3.3 Max-Utility with Bandwidth Constraint

In the scenario described in Section 3.2, the bandwidth constraint determines the number of photos that can be selected. The problem is defined as follows.

Definition 2. *[Max-Utility Problem] Given a set of m targets with known locations $T = \{T_1, \ldots, T_m\}$ and n photos $P = \{P_1, \ldots, P_n\}$ with known metadata, also given a predefined positive integer $B(\leq n)$, the problem asks for a selection of B photos P' out of the n candidates, such that the total utility of the selected photos $U_{P'}(T)$ is maximized.*

3.3.1 Conversion to Maximum Coverage

Without loss of generality, we first consider a single target T_i and use the coverage interval $I_i = [0, 2\pi)$ to indicate its aspect to be covered. Let $P = \{P_1, \ldots, P_n\}$ be the set of all photos covering T_i. Then for each P_j, if T_i is covered by P_j, the coverage of P_j on T_i (gray sector in Figure 3.1(b)) can be represented by a sub-interval of $[0, 2\pi)$, i.e.,

$$S_j \triangleq [x_j, y_j] = [arg(\overrightarrow{T_iP_j}) - \theta, arg(\overrightarrow{T_iP_j}) + \theta]. \tag{3.1}$$

Note that the angles are always calculated by using arithmetic modulo 2π. Here the two end points x_j and y_j are called dividing points, which divides I_i into two parts: one is S_j and the other is $I_i - S_j$. If there are more photos by which T_i is covered, there would be more dividing points.

If there are multiple targets, every target corresponds to a coverage interval $I_i = [0, 2\pi)$ and each I_i is divided into sub-intervals by the corresponding dividing points. Let $U = \{e_1, \ldots, e_w\}$ be a universe set with each element representing a sub-interval and w being the total number of them. The weight of the element is the length of the sub-interval. For each photo P_j, a subset of U can be generated based on what sub-intervals are covered by it. Let S_j denote this subset. Then we have proved the following lemma:

Lemma 1. *A solution to the Max-Utility problem can be obtained by solving the following problem: given a universe set U of (non-negative) weighted elements, an integer B and a collection of subsets $S = \{S_1, \ldots, S_n\}$, find B subsets such that the total weight of the elements covered by the selected subsets is maximized.*

3.3.2 Greedy Selection Algorithm

The general maximum coverage problem is proved to be NP-hard [57]. A greedy algorithm can be used to find a solution. It works as a multi-round selection process. In each round, the weighted contribution (utility) of every unselected photos is calculated. The photo with the most contribution to the total utility is selected. If there are more than one photos with the most contribution, the one with the lowest index is selected. Once a photo is selected, it will be removed from the selection. The elements (sub-intervals) covered by the selected photo will be removed from future consideration. The selection process runs until B photos have been selected or every aspect of all targets has been covered, whichever comes first.

Theorem 1. *Let U_{opt} be the optimal value of the total utility that can be achieved by any B photos from P. Let U_{greedy} be the total utility achieved by the greedy selection algorithm. Then*

$$U_{greedy} \geq [1 - (1 - \frac{1}{B})^B] \cdot U_{opt} > (1 - \frac{1}{e})U_{opt}.$$

Proof. From Lemma 1, a selection of B subsets implies a valid selection of B photos. Moreover, the total utility of the photos is maximized if and only if the corresponding subsets has the maximum total weight. On the other hand, the subsets selected by the greedy selection can yield a total weight that is at least $(1 - 1/e)$ times the optimal value [57]. Therefore, the total utility of the selected photos is also lower bounded by $(1 - 1/e)$ times the maximum total utility. □

An Example: Figure 3.2 shows an example of one target and 10 photos. Suppose $\theta = 45°, B = 3$. Each photo's position is shown in Figure 3.2(a). The arrows in Figure 3.2(b) indicate photos' viewing directions, and the number beside the arrow (e.g., 10 beside $\overrightarrow{T_1P_{10}}$) indicates the angle of the viewing direction of the photo (e.g., P_{10}), which has been defined in Section 3.2. Based on this, each photo's coverage interval is calculated and shown in Figure 3.2(c) according to Equation (3.1). Then target T_1's coverage interval $I_1 = [0, 2\pi)$ is divided into sub-intervals by the endpoints of all photos' coverage intervals (Figure 3.2(d)). This is the universe set U which is composed of weighted elements from e_1 to e_{19}, and the weight of each element is reflected by its length. Finally, each photo's coverage interval is converted into a subset S_i of elements (Figure 3.2(e)).

We select 3 photos to maximize the total utility. Initially, each S_i has a weight of $2\theta = 90°$, and hence S_1 is selected due to the smallest index. Elements e_{11}, e_{12}, e_{13}, e_{14} are removed from U. Second, the weight of each of S_3, S_4, S_5, S_9 and S_{10} is still 90°, but for the others the weights become: S_2 is 80°; S_6 is 50°; S_7 is 20° and S_8 is 40°. Obviously, S_3 is selected. Then elements e_3, e_4, e_5, e_6, e_7 are removed from U. Finally, we consider the remaining subsets. The weights of S_5, S_6, S_7, S_8 are unchanged, but S_2 drops to 45°, S_9 drops to 15° and S_{10} drops to 80°. Therefore, the last selected photo is S_5. The final selection is S_1, S_3, S_5, corresponding to P_1, P_3, P_5, and the total achieved utility is 270°.

Theorem 2. *For the Max-Utility problem, the worst case time complexity of the greedy algorithm is* $O(Bn^2m)$.

Proof. Since there are n photos and m targets, calculating the coverage intervals of each photo on each target takes $O(mn)$ time. Then, a target can be covered by at most n photos, so there are $O(n)$ coverage intervals on a target. Sorting

the endpoints of those coverage intervals gives us the universe set, which takes $O(mn \log n)$ time for all m targets.

Next, photo selection is done in B steps. There are n candidate photos to consider in the first step, but the number decreases by 1 after each step. With assumption $B \ll n$, the number of candidate photos to consider is $\Theta(n)$ for any of the B steps. In total, we consider candidate photos for $\Theta(Bn)$ times. Now let's look at how much time it takes to handle one candidate photo. For each of the m targets, we need to add the weights of the elements covered by that photo. The number of those elements is proportional to the total number of elements on the target, which is in the order of $O(n)$. For example, given effective angle $\theta = 45°$, a photo always covers $1/4$ of the total aspects, so the elements it covers are $1/4$ of the total number of elements. This means calculating the utility of one candidate photo takes $O(mn)$. Getting these together, the selection process takes $O(Bn \cdot mn) = O(Bn^2m)$ time.

To summarize, the worse case time complexity is $O(mn + mn \log n + Bn^2m) = O(Bn^2m)$. □

3.4 Achieving Required Utility With Min-Selection

We consider a different scenario from the Max-Utility problem: the number of photos is minimized while the total utility is to be above a required level. In many practical applications such as virtual tours in map services, the major obstacle is to deal with the sheer amount of raw data (photos) obtained via crowdsourcing. Thus, it is desirable to remove redundancy and only keep the minimum selection of photos that satisfies the coverage requirement.

3.4.1 Problem Statement

Each target T_i is associated with a coverage requirement, represented by a coverage interval $I_i = [a_i, b_i], 0 \leq a_i, b_i < 2\pi$. The requirement is met if any aspect $\vec{v}$ chosen from I_i is covered. The problem is defined as follows.

Definition 3. *[Min-Selection Problem] Given a set of m targets with known locations $T = \{T_1, \ldots, T_m\}$ and n photos $P = \{P_1, \ldots, P_n\}$ with known metadata, also*

given the coverage requirements for the targets: $I = \{I_1, \ldots, I_m\}$*, the problem asks for a minimum selection of photos out of the* n *candidates, such that the coverage requirement for each target is met.*

Note that in this problem if the requirement can not be met due to the insufficiency of the original set of photos, the best achievable utility will be used as the criteria. Here the best achievable utility on a target is the utility of all photos on it, and the best achievable total utility on all targets is the sum on each targets normalized by the number of targets.

3.4.2 Min-Selection Algorithm

In the following description, it is assumed that the coverage requirement of each target can be satisfied by the whole set of photos. Then the following theorem shows the main result of our findings.

Theorem 3. *Suppose the targets' coverage requirements can be satisfied by all photos in the pool and let* N_{opt} *be the minimum number of photos to satisfy the requirement. There exists* N_{approx} *photos that can be found in polynomial time such that each target's requirement can be met by these photos and moreover,* $N_{approx} \leq O(\log mn)N_{opt}$.

Proof. We prove this by constructing the selection using a greedy algorithm.

First, we use a conversion process that is similar to Section 3.3.1. Here each target T_i's coverage requirement I_i is partitioned into sub-intervals by the dividing points, and the dividing points are the end points of the coverage intervals (sub-intervals) of the photos like before. After this preparation, all the sub-intervals are numbered, and can be represented by elements that altogether form an universe set $U = \{e_1, \ldots, e_w\}$, where w is the total number of sub-intervals. Then for each P_j, there is a subset $S_j \subset U$ which is comprised of the elements corresponding to the sub-intervals covered by P_j. Based on this, the problem of finding the minimum photo selection can be converted to the following problem:

Given a universe set U *and a collection of subsets of* U*:* $S = \{S_1, \ldots, S_n\}$*, and assume* $\cup_j^n S_j = U$*, how to find a subset* S' *of* S *such that* $\cup_{S_j \in S'} S_j = U$ *and* $|S'|$ *is minimum?*

This is an instance of the set cover problem, which has been proved NP-hard [57]. Thus for the Min-Selection problem, we can solve it by an approximation algorithm based on the greedy selection.

Specifically, the algorithm begins by selecting the photo (some S_j) that covers the most number of sub-intervals (elements). Once a photo is selected, it will not be removed. The sub-intervals covered will not be considered in the future. Photos are selected one by one based on how many new sub-intervals can be covered. Each time, the photo covering the most number of new sub-intervals is selected. Ties can be broken arbitrarily, e.g., by giving priority to the one with smaller index. The process stops if all sub-intervals is covered or no more photos can be selected (i.e., either photos are all selected or no more benefit can be achieved).

Once the photos are found, it is obvious all the elements in U is covered which implies the requirement of all targets are satisfied. By using similar argument from Theorem 3.1 in [57], it is easy to see the number of selected photos is upper bounded as shown in the theorem. □

The above discussion can be easily applied to the scenario of multiple targets. In that case, each target corresponds to a set U_i of elements (sub-intervals). Elements of all U_i will be considered to determine if a particular S_j can yield the most coverage. The algorithm stops if elements of all U_i are covered or no more progress can be made. A detailed example of the algorithm can be found in [58].

Theorem 4. *For the Min-Selection problem, the worst case time complexity of the greedy algorithm is* $O(n^3m)$.

Proof. The conversion process takes time $O(mn+mn\log n)$. For the selection process, we highlight two differences between the Max-Utility problem and the Min-Selection problem. First, in Max-Utility we select the photo that covers maximum weight of new elements, while in Min-Selection we select the photo that covers maximum number of new elements. Although weight and cardinality are different, they can be calculated by the same number of additions. Thus they are the same with regard to asymptotic running time. The second difference is that Max-Utility finishes with exactly B steps, while Min-Selection may finish in any steps between 1 and n. Considering the worst case, if Min-Selection finishes with n steps, the number of candidate photos it considers in those n steps is $n+(n-1)+\cdots+1 = O(n^2)$.

Since handling each candidate photo requires time $O(mn)$, the time complexity of the entire algorithm is $O(mn + mn \log n + n^2 mn) = O(n^3 m)$. □

3.5 Achieving k-Coverage With Min-Selection

Some crowdsourced photos may contain inaccurate information. This may happen in an emergency situation where photos have to be taken in a very short time, leaving not much time for users to contemplate. Even if metadata can help understand how and where the photo was taken, some real photos may still miss our expectations. The inaccuracy can be caused by various reasons such as image blur, occlusion, color aberration, or simply inaccurate metadata. To reduce the possibility that an important aspect of the object is missed, applications may require some degree of fault-tolerance, which can be achieved with k-coverage.

3.5.1 k-coverage

In this problem, an aspect is required to be covered k ($k \geq 2$) times. Each target T_i has a coverage requirement $I_i = [a_i, b_i]$, $0 \leq a_i, b_i < 2\pi$, but now I_i needs to be covered k times by the selected photos. The problem is formally defined as follows.

Definition 4. *[Min-Selection with k-Coverage] Given m targets with known locations $T = \{T_1, \ldots, T_m\}$ and n photos $P = \{P_1, \ldots, P_n\}$ with known metadata, also given the coverage requirements for the targets: $I = \{I_1, \ldots, I_m\}$ and an integer $k \geq 2$, the problem asks for a minimum number of photos out of the n candidates, such that the coverage requirement for each target is covered at least k times.*

As the original Min-Selection problem can be converted to the set cover problem, the k-coverage problem can be converted in the same way to the set multicover problem. The set multicover problem, as its name suggests, differs from the set cover problem that each element e in the universe set U must appear at least k_e times in the selected subsets of U, where k_e is a positive integer for element e. Note that in our case, $k_e = k$ for any element e.

The greedy algorithm for the original Min-Selection problem can be naturally extended to the case of k-coverage. Specifically, an element is *alive* until it is covered k times. In each step, we select the photo that covers as many alive

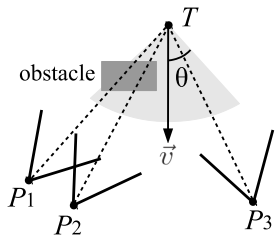


Figure 3.3. The idea of 2-coverage+: two photos covering an aspect should have fairly different viewing directions.

elements as possible, and then update the aliveness of the elements covered by this photo as appropriate. The selection proceeds until all elements are not alive or no more photos can be selected (either photos are all selected or no more benefit can be achieved).

Suppose the overall coverage requirements can be satisfied if all photos are selected. Dobson [59] proved that the above algorithm achieves an approximation ratio of $O(\log mn)$, which means the number of photos selected by the greedy algorithm does not exceed $O(\log mn)$ times the minimum possible number. Note that this approximation ratio is the same as the ratio achieved by the Min-Selection algorithm (Theorem 3).

3.5.2 2-coverage+

Although the k-coverage model looks good in theory, it may not work well in reality. Consider a scenario shown in Figure 3.3. An aspect $\vec{v}$ of the target T is covered by three photos, P_1, P_2 and P_3. There is an obstacle between the target and the photos. By the k-coverage model, if $k = 2$, any combination of two photos among P_1, P_2 and P_3 satisfies the coverage requirement on aspect $\vec{v}$. However, choosing P_1 and P_2 will actually leave aspect $\vec{v}$ uncovered because the target is blocked in photo P_1 and P_2.

The key observation here is that the existence of obstacles affects our choice of photos. Ideally, the server should avoid P_1 and P_2 since they are blocked by an object and do not cover aspect $\vec{v}$. Although we will propose techniques in Section 3.6 to detect occlusion and filter out photos with occlusion, these techniques are not guaranteed to detect all obstacles. Hence, when selecting photos, the server should take into consideration the potential existence of obstacles. Instead of choosing k photos arbitrarily, the photos with fairly different views should be

chosen, so that even one of them is blocked by an obstacle, the other still covers the aspect. Specifically, in the *2-coverage+* model, for each aspect covered by two photos, the angle between their viewing directions should be larger than a predefined threshold α.

Definition 5. *[Min-Selection with 2-Coverage+] Given m targets with known locations $T = \{T_1, \ldots, T_m\}$ and n photos $P = \{P_1, \ldots, P_n\}$ with known metadata, also given the coverage requirements for the targets: $I = \{I_1, \ldots, I_m\}$ and a threshold $\alpha \in [0, 2\theta]$, the problem asks for a minimum number of photos such that each aspect in the coverage requirements is covered by at least 2 photos, and the maximum angle between the viewing directions of those photos is greater than α.*

In this problem, the value of α determines the coverage resistance to obstacles. The larger α is, the farther the two photos are separated, and the less likely they are blocked by the same obstacle. However, α cannot be arbitrarily large. If a photo covers a given aspect $\vec{v}$, its viewing direction must be within a 2θ range, $[arg(\vec{v}) - \theta, arg(\vec{v}) + \theta]$ (light gray area in Figure 3.3). This means α has an upper bound 2θ. As α becomes closer to 2θ, it becomes harder to find two photos satisfying the requirement, because the two photos have to be positioned more precisely so as to both cover the aspect and have far enough viewing directions.

The 2-coverage+ problem is equivalent to the 2-coverage problem when $\alpha = 0$, so it is at least as hard as the k-coverage problem, and thus there is no polynomial time algorithm to find its optimal solution. Similar to the k-coverage problem, we can use a greedy algorithm to find a solution, but with some adjustment to the notion of aliveness. Specifically, the aliveness value of an element can be 2, 1 or 0. $alive = 2$ means that the element has never been covered; $alive = 1$ means that it has been covered at least once, and among the photos covering it, the maximum angle between their viewing directions is less than α; $alive = 0$ means that it has been covered at least twice, and among the photos covering it, the maximum angle between their viewing directions is no less than α.

In each step of the greedy algorithm, for each photo, we count the number of elements whose aliveness value would decrease if the photo were selected. Then we pick the photo with the largest count, and update the aliveness values accordingly. Here the decrease of an aliveness value means previously $alive = 2$ and after

selecting the photo $alive = 1$, or previously $alive = 1$ and after selecting the photo $alive = 0$. This selection process continues until all $alive = 0$ or no more photo can be selected (either photos are all selected or no more benefit can be achieved). The performance of the greedy algorithm will be evaluated in Section 3.7.2.

3.6 Testbed Implementation

A prototype of SmartPhoto has been implemented in a testbed using Samsung Nexus S running Android 2.3.6, Samsung Galaxy S III running Android 4.0.4, and Google (LG) Nexus 4 running Android 4.2.

In the testbed, the smartphones take photos with the metadata automatically recorded. The metadata is a tuple comprised of a GPS location, a range indicating how far the photo can cover, a field-of-view (FoV) angle of the camera taking the photo and an orientation vector indicating the facing direction of the camera lens. After the photo has been taken, the smartphone uploads the metadata of the photo to a centralized server, which is a PC in our lab running the photo selection algorithm. Then the server notifies the smartphones to upload the selected photos. In this section, we present the technical details on how to obtain the metadata, how to improve the accuracy of orientation measurement, and how to deal with occlusion and out-of-focus issues.

3.6.1 Metadata Acquisition

One critical issue is how to get the metadata from off-the-shelf smartphones. The location can be directly acquired via the built-in GPS receiver. The camera's field-of-view is accessible via the Android camera API [60]. The range is a little trickier as it depends on the resolution of the lens (and the image sensor), the zooming level (or focal length) and the requirement of the application. Applications requiring a survey of large scale buildings may find the photos useful even if they are taken a hundred meters away by a lower resolution camera, while others may require closer look at the object and hence may exclude photos taken more than a few meters away. In our experiment, as the subjects are buildings on campus, 50 meter is used as a reference range. We find that for our purpose, objects in photos taken within

this range are generally recognizable.

Orientation is a key factor that has not yet been fully taken advantage of in previous works. The way used to characterize the orientation in the Android system is to first define a local and a world coordinate system[2], and represent the orientation as a rotation matrix. The rotation matrix is used to transform a local coordinate tuple into a global one. Another way to represent the rotation is to use a three tuple called *azimuth, pitch,* and *roll*, which respectively indicate the phone's rotation around the Z, X and Y axes [61]. The two representations are equivalent and the orientation tuple (i.e., the angles) can be derived from the rotation matrix. In the following description, we use R to denote the rotation matrix.

In Android system, the rotation matrix can be directly obtained based on accelerometer and magnetic field sensor readings. The accelerometer measures the phone's proper acceleration along the three axes in the phone's local coordinate system, including the influence of the gravity. The magnetic field sensor provides the readings measuring the ambient magnetic field along the three axes in the local coordinate system. The coordinates of both the gravity and the ambient magnetic field are known in the world coordinate system. Thus, by combining the above readings and facts, the orientation of the phone can be obtained. Let us call this the "basic" method, and let the result be denoted by R_{basic}.

3.6.2 Techniques to Improve Accuracy

The rotation matrix R_{basic} is susceptible to noise and errors. It fluctuates quickly due to the vibration of accelerometer's reading. Also, the magnet field sensor's reading is easily affected by nearby magnet objects. Even worse, R_{basic} responses slowly to quick rotation of the phone. Thus, we propose several techniques to improve the accuracy of the orientation.

[2]In a world coordinate system, Z axis is perpendicular to ground and points to the sky; Y is tangential to the ground and points towards the magnetic north pole; X is the vector product of Y and Z. In the phone's local coordinate system, Z is perpendicular to the phone screen and points outward; the X axis is along the width of the phone and the Y axis is along the length [60, 61].

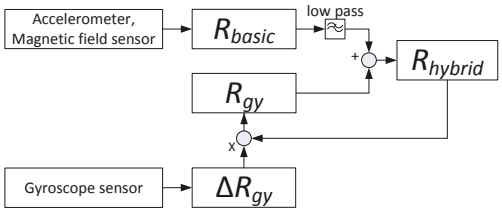


Figure 3.4. Hybrid method combines the results of the basic method with gyroscope readings.

3.6.2.1 Hybrid Method

Apart from the accelerometer and the magnetic field sensor, gyroscope is now available in most smartphones, and it can also be used to measure the rotation matrix.

Gyroscope measures the angular rotation speeds along all three axes in the phone's local coordinate system. By integrating (multiplying) the angular speed with the time interval between two consecutive sensor readings, we can obtain the rotation vector, which indicates the change of orientation in terms of rotation angles around the three axes. It can also be used to obtain the rotation matrix (denoted by ΔR_{gy}). Given an initial rotation matrix, which can be obtained from R_{basic}, we can get the new rotation matrix, denoted as R_{gy}, by $R_{gy} = R_{gy} \times \Delta R_{gy}$.

However, the cumulative error caused by the integration in R_{gy} can become greater and the result would drift as time goes by. In fact, the orientation derived from R_{gy} alone usually drifts over 10 degrees in about 20 seconds in our lab test.

Thus, we propose a hybrid method which combines the readings from the above sensors to improve the accuracy of orientation, as shown in Figure 3.4 and explained as below.

First, a simple Infinite Impulse Response (IIR) low pass filter is used on R_{basic} to remove the short term vibration, i.e.,

$$R'_{basic} = R_{basic} + \mu \cdot (R^{prev}_{basic} - R_{basic}),$$

where R_{basic} is the current reading and R^{prev}_{basic} is the previous reading from the basic method, and $\mu \in [0, 1]$ is an adjustable parameter balancing the current and previous values. In practice, we find $\mu = 0.3$ is good for our purpose.

Second, we combine R'_{basic} and R_{gy} to take advantage of both values; that is,

$$R_{hybrid} = \nu \times R_{gy} + (1 - \nu) \times R'_{basic}$$

We find $\nu = 0.9$ works well.

Third, R_{hybrid} is the output, and it will also be used as the initial matrix input for the computation of a new R_{gy}.

3.6.2.2 Enhancement by Orthonormalization

We exploit the orthonormal property of the rotation matrix to further improve the accuracy of orientation. In a valid rotation matrix, any pair of columns (or rows) of the rotation matrix are orthogonal, i.e., with unit length and vertical to each other. However, this property may be violated as errors occur. Thus, the rotation matrix R_{hybrid} obtained from the above method can be further calibrated by an orthonormalization process (e.g., the Gram-Schmidt process [62]) to get an enhanced rotation matrix $R_{enhanced}$.

Specifically, consider a 3×3 rotation matrix: $R_{hybrid} = [\alpha_1, \alpha_2, \alpha_3]$, with α_i being a column vector. Let the inner product of the two vectors α and β be $< \alpha, \beta >= \sum_{i=1}^{n} \alpha_i \beta_j$, where $n = 3$ is the dimension.
First, R_{hybrid} is orthogonalized by

$$\begin{aligned}
\xi_1 &= \alpha_1, \\
\xi_2 &= \alpha_2 - \frac{< \alpha_2, \xi_1 >}{< \xi_1, \xi_1 >} \xi_1, \\
\xi_3 &= \alpha_3 - \frac{< \alpha_3, \xi_1 >}{< \xi_1, \xi_1 >} \xi_1 - \frac{< \alpha_3, \xi_2 >}{< \xi_2, \xi_2 >} \xi_2.
\end{aligned}$$

Second, the above ξ_i's are normalized by

$$\beta_i = \frac{\xi_i}{\sqrt{< \xi_i, \xi_i >}}, i = 1, 2, 3.$$

Then, the final rotation matrix is

$$R_{enhanced} = [\beta_1, \beta_2, \beta_3].$$

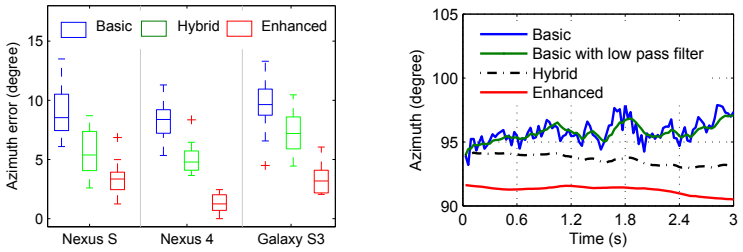
Table 3.1. Average error in azimuth (degree)

	Nexus S	Nexus 4	Galaxy S III
Basic	9.1(±2.0)	8.2(±1.5)	9.6(±2.4)
Hybrid	5.7(±1.9)	5.1(±1.3)	7.3(±1.7)
Enhanced	3.4(±1.4)	1.3(±0.7)	3.4(±1.3)

Comparisons: To verify the effectiveness of the optimization techniques, we measure the orientation using three different methods: the "basic" method, the "hybrid" method, and the "enhanced" method, and compare their results. We place the phone in a horizontal plane, so the orientation is reflected by the azimuth value. Then we rotate the phone 30 degrees and measure its azimuth reading against a commercial compass. Each measurement is repeated 50 times and the statistics are calculated. Figure 3.5(a) compares the measurement errors (in degree) by these three methods. The short bar in the middle of each box is the median value of the azimuth reading error, and the lower and upper side of the box are the first (25%) and third (75%) quartile, which is denoted by $Q1$ and $Q3$. Then the lower limit is calculated by $Q1 - 1.5 * (Q3 - Q1)$ and the upper limit is $Q3 + 1.5 * (Q3 - Q1)$. More details about the average error and standard variance of each method are listed in Table 3.1.

We find that the hybrid method can reduce the average measurement error by 37% compared to the basic method, and the enhanced method can further reduce the measurement error by more than 40% compared to the hybrid method. Also, new phones (e.g., Nexus 4), with more advanced hardware and OS, are more accurate with less variance. For all these phones, with our enhanced method, the average azimuth reading error is under 3.5 degrees, and the error can be reduced to 1.3 degree with Nexus 4.

To understand the effectiveness of these techniques clearly, we show the measurement results of these methods when the phone is turned to 90 degree, as illustrated in Figure 3.5(b). As can be seen, the basic method oscillates frequently. The hybrid method improves the accuracy compared to the basic method but still carries the reading errors. With orthonormalization, the enhanced method can significantly improve the accuracy of orientation.



(a) (b)

Figure 3.5. Orientation errors of the three proposed methods.

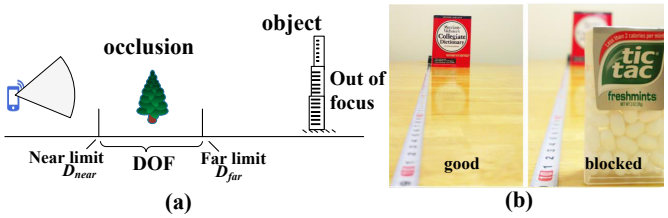


Figure 3.6. Using DOF to detect occlusion and out-of-focus.

3.6.3 Occlusion and Out-of-focus

After a photo is taken, we assume that the user will visually check if the object appears in the photo, as most people do. However, if the user does not check the photo, and the object is blocked by unexpected obstacles such as a moving vehicle, the photo will not be useful to the server. Even if the user checks the photo and the object is clear, it may be different from what the server is expecting. For example, the server may expect the photo to be about a building, but the user may be looking at a tree in front of the building. Although in the two scenarios, the smartphone may produce the same metadata (e.g., the same facing direction), the content could be very different, and the one focusing on (blocked by) the tree is useless for the server's task. Besides this problem, there are many other occasions that the interested targets are out-of-focus. Uploading these photos will waste lots of bandwidths and storage spaces.

We use a feature called *focus distance*, which is provided by many new smartphones with focusing capability, to solve the problem. The focus distance is the distance between the camera and the object perfectly focused in the photo. Note that the real distance between the camera and our interested target can be calculated by GPS locations. Thus in an ideal case, if the two distances do not match, the target is out-of-focus and the photo should be excluded from consideration.

The measurement of the focus distance is sensible to errors. A slight offset does not necessarily mean the target is out-of-focus. In fact, in photography the distance between the nearest and farthest objects that appear acceptable sharp in a photo is called the Depth-Of-Field (DOF). DOF is determined by four parameters: focal length (f), focus distance (v_o), lens aperture (A), and circle of confusion (CoC). Among these parameters, focal length and lens aperture are built-in and readable from the Android API. CoC (denoted by c) is a predefined number which determines the resolution limit for our application. Focus distance changes from case to case but obtainable from Android API. Therefore, we can calculate the near/far limit of DOF (Figure 3.6(a)) by $D_{near} = \frac{v_o(H-f)}{H+v_o-2f}$, $D_{far} = \frac{v_o(H-f)}{H-v_o}$, where $H = \frac{f^2}{Ac} + f$ is the hyperfocal distance [63].

After a photo is taken, the distance between the target and the camera (phone) is compared with the above two values. If the target falls into the DOF, the photo is considered valid; otherwise, it will be dropped. This filtering is done at the user side and the metadata of unqualified photos will not be sent to the server. As an example, consider the two photos in Figure 3.6(b). The dictionary is the interested target. In the left photo, the near and far limit of DOF is 85cm and 105cm respectively. In the right photo, the near and far limit of DOF is 5cm and 10cm respectively. The distance between the camera and the dictionary is 100cm. Based on these parameters, it is clear that the target falls into the DOF in the left photo. From the figure we can see, the dictionary is clear in the left photo but blocked by another object in the right photo. Note that this method can detect most obstacles but not all. If the obstacle is very close to the target (their distances to the camera are almost the same), it is possible that the target is blocked but still in the DOF.

Discussions: Energy is an important issue for mobile devices, especially in post-disaster scenarios. Although various built-in sensors are used to collect metadata, they do not consume too much energy due to the following two reasons. First, metadata are collected only when users are taking photos, and the sensors are inactive most of time. Second, crowdsourcing relies on large number of users to obtain information. A single user does not need to take many photos and thus does not spend too much energy.

Photos can be of low quality due to various reasons. Over-exposure or under-

exposure causes images to be too bright or too dark; camera movement and shutter speed affect how severe the image is blurred; the quality of lens and digital sensors is also important. These factors can only be analyzed by image processing. Thus, before photo selection, some efficient image processing techniques [64] may be applied at the user end to filter out low quality photos. However, existing image processing techniques are computationally expensive, and thus should be carefully adapted considering the resource limitations of mobile devices. Note that our approach is not meant to replace the role played by image processing algorithms, but to serve as an important complement to improve the utility of collected photos, especially when there are resource constraints.

3.7 Performance Evaluations

In this section, we first show a real world demo using the smartphone testbed, and then evaluate the performance of the photo selection algorithms by extensive simulations.

3.7.1 Demo in a Real-World Example

The testbed in Section 3.6 is used in a real-world example to demonstrate the effectiveness of the proposed photo selection algorithm. In this demonstration, a landmark (a bell tower) is the target. Photos are taken by using the reprogrammed smartphones around the target with the metadata automatically recorded. The metadata of all photos are later uploaded into a centralized desktop server. There are 30 photos in total. Although most of them are taken around the target, some are not facing the target, and some are blocked by trees or other objects. Also, the distribution is not uniform, due to the reality that people are likely to take pictures of the front (more attractive) side of the building.

After the metadata is retrieved, the Max-Utility problem is solved by choosing 4 photos. The photos selected by our algorithm are shown in Figure 3.7(a). The positions and orientations of the photos are shown in Figure 3.7(d), where the original 30 photos are marked as dotted "V" shapes, and the selected photos are marked by bold lines. As a comparison, we randomly choose 4 photos as shown in

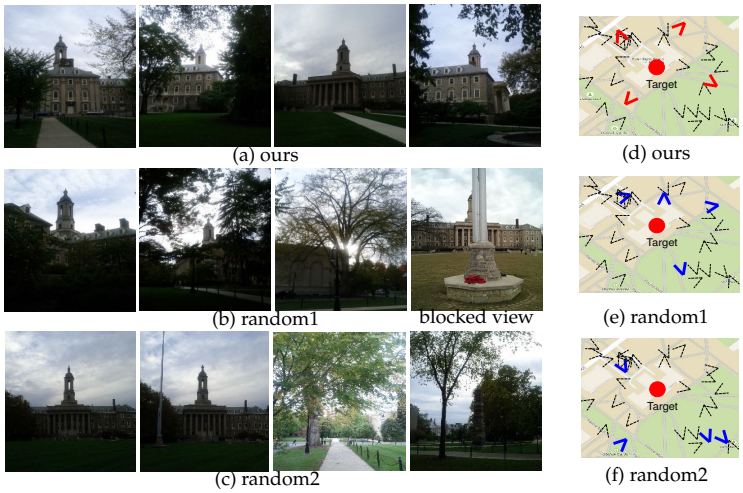


Figure 3.7. Demo results based on Max-Utility. (a) Photos selected by our algorithm. (b)(c) Photos selected randomly. (d)(e)(f) The locations and orientations of the photos are marked as "V" shapes on the map.

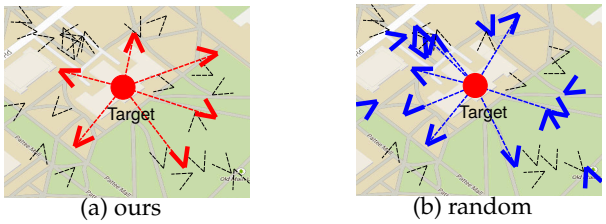


Figure 3.8. Demo results based on Min-Selection. To cover all aspects of the target, our algorithm uses 6 photos, and a random selection uses 15 photos.

Figure 3.7(b), and randomly choose another 4 photos as shown in Figure 3.7(c). It can be seen that the 4 photos chosen by our algorithm cover the target from 4 different locations well separated from each other, with each one from a totally different angle. The bell tower is viewable from all 4 photos. In contrast, only 2 photos in the first random selection cover the target. The third photo faces away from the target, which is because random selection does not consider the orientation. In the fourth photo, the target is blocked by a **flagpole**. This photo is not considered by our algorithm because the target is out of focus according to the DOF information. For the second randomly selected 4 photos, two of them cover the target but they are very similar and contain redundant information. The other two do not cover the target.

We also demonstrate the effectiveness of our algorithm for the Min-Selection problem where the coverage requirement is from 0° to 360°, i.e., all aspects of

the target. As shown in Figure 3.8(a), our algorithm selects 6 photos to meet the coverage requirement. The angle between any two adjacent viewing directions (dashed lines connecting the photos and the target) is less than 90°. Since the effective angle is set to 45°, all aspects are covered. We also compare the performance with a random selection approach, which randomly selects photos one by one until the coverage requirement is achieved. As shown in Figure 3.8(b), the random approach has to select 15 photos to meet the same coverage requirement. The experiment based on the random selection approach is repeated 100 times, and on average 21 photos are selected to meet the same coverage requirement. This demonstrates that our algorithm can significantly reduce the number of photos selected to achieve the required coverage.

3.7.2 Simulation Results

In this section, we evaluate the photo selection algorithms through simulations. Targets are randomly distributed in a 100m by 100m square area. We generate photo metadata to represent real photos taken by users as follows. The photo locations are uniformly distributed in a 200m by 200m square, with the target area in the center. The orientations are randomly distributed from 0 to 2π. The field-of-view is set to 120°, and the range is set to 50m as discussed in Section 3.6.1.

During the simulation, we compare our algorithms with a random selection algorithm that randomly selects photos at each step, until the bandwidth constraint is reached or the coverage requirement is satisfied. For a fair comparison, the random selection excludes any photos that have no target covered, but only consider photos that cover at least one target, i.e., relevant photos. Note that a more naive selection could be blindly selecting photos without considering this.

3.7.2.1 Results on Max-Utility Problem

In the first part, we evaluate the performance of our algorithm on addressing the Max-Utility problem. Intuitively, with more bandwidths, better coverage of the targets can be achieved. As shown in Figure 3.9(a), both our algorithm ("ours") and the random selection ("random") achieve more utility as the bandwidth B increases. The total utility achieved by all photos ("best achievable") is also shown

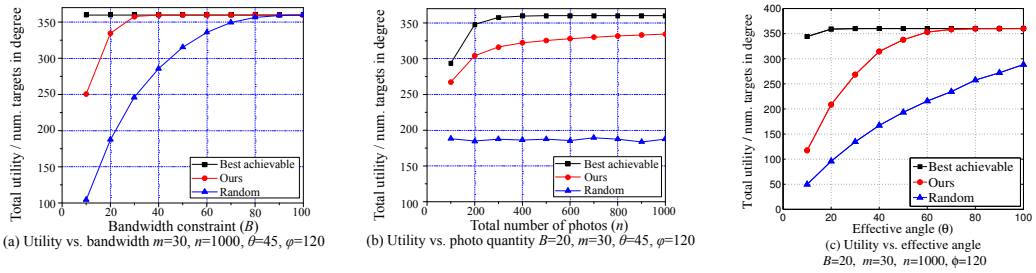


Figure 3.9. Simulation results on Max-Utility problem.

to provide an upper bound. The difference between our selection and the random selection is significant and the advantage of our algorithm is obvious especially when B is smaller, i.e., bandwidth is more constrained. Although the performance of both algorithms converges to the best-achievable utility as B becomes larger, the convergence of ours is much faster.

Figure 3.9(b) shows how the total utility changes as the number of candidate photos increases while other factors including bandwidth ($B = 20$) remain unchanged. The advantage of our algorithm is significant across the range. Considering the bandwidth limitation (only 20 photos can be selected to cover 30 targets), the difference between the utility achieved by ours and the best achievable level is small. Moreover, our algorithm can take advantage of the increasing density of photos, and improve its performance as the number of photos increases.

Figure 3.9(c) plots the total achieved utility against the effective angle θ, with all other factors unchanged. As the effective angle increases, the coverage intervals of photos grow accordingly, so the total utility of both our algorithm and random selection increases as expected. Moreover, when the effective angle is between 10 to 40 degrees, the utility achieved by ours increase at a faster rate (the red circle line has larger slope than the blue triangle line). This is because our algorithm tries its best to avoid coverage overlap and thus benefits more from the coverage growth of each single photo. Note that as the effective angle increases, photos have more coverage but also have more coverage overlap. When the effective angle is more than 40 degrees, the utility obtained by our algorithm approaches the best achievable, so its increasing rate cannot be as high as before.

3.7.2.2 Results on Min-Selection Problem

In this part the Min-Selection problem is studied. In reality, the given pool of photos can be very large and the number of relevant photos (i.e., photos covering at least one target) can increase very fast as the total number of randomly taken photos increases. Then, a careful selection of photos can greatly reduce the redundancy. Figure 3.10(a) shows the effectiveness of our selection algorithm on reducing the redundancy. There are 20 targets, and all the aspects from 0° to 360° are required to be covered. As the total number of photos varies from 500 to 2000, the number of related photos ("related") increases linearly. However, the number of photos selected by our algorithm ("selected by ours") to achieve the same coverage does not increase. It actually decreases slightly since our algorithm takes advantage of the increased density of the photos and improves its efficiency.

The algorithms are also evaluated under the situation that the number of targets (m) varies from 5 to 50, while the total number of photos is fixed to be 1000 and all other factors remain the same. As shown in Figure 3.10(b), the algorithms have to select more photos to cover the increased number of targets. However, the number of photos selected by our algorithm remains very low, and the increasing speed is much slower as the number of targets increases, which is much better than the random algorithm.

In Figure 3.10(c) we fix the number of targets as 30 and change the amount of aspects to be covered on each target. As expected, the number of photos needed to achieve the required coverage increases as more aspects are to be covered. Interestingly, the increasing rates of the lines rise on the left half but drop on the right half. The reason behind this can be explained as follows. Let us denote the amount of aspects to be covered on a target as x. The coverage requirement of target i is $I_i = [0, x]$. When x is small (e.g. 30°), the photos selected to cover $[0, x]$ also cover more than $[0, x]$, since by definition a photo always covers a 2θ range of aspects on a target (here $\theta = 45°$). Then if we increase x a little bit, not many new photos are needed to achieve the new coverage, since it is almost achieved by previous photos. As x keeps increasing, this advantage becomes weaker because the new coverage requirement $[0, x]$ is no longer within the 2θ range. Then many new photos have to be selected to complete the coverage. That is why the increasing rate rises for small x. On the other hand, when x is large, this advantage becomes

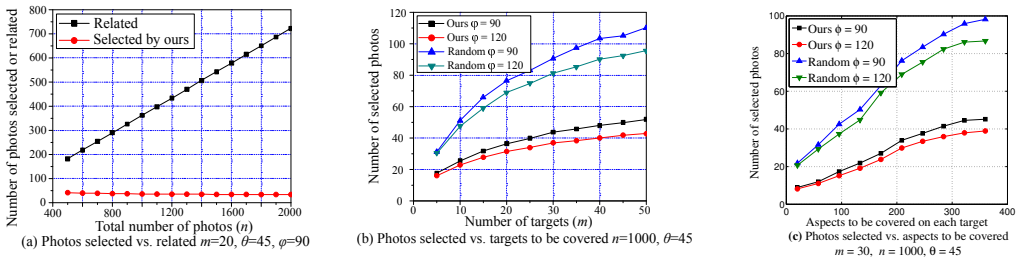


Figure 3.10. Simulation results on Min-Selection problem.

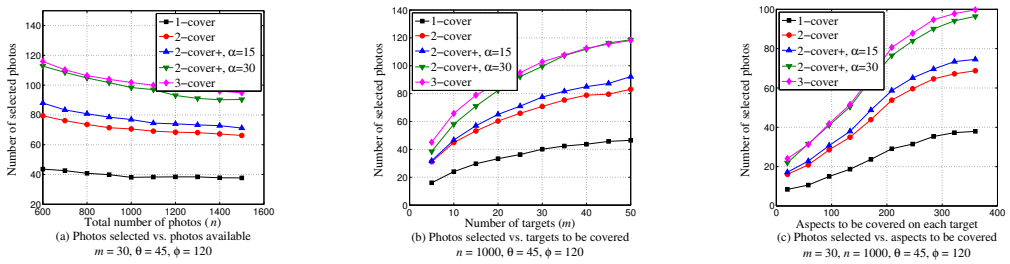


Figure 3.11. Simulation results on Min-Selection with k-Coverage.

stronger as x increases, because the remaining uncovered aspects are now within the 2θ range, and they are more likely to be covered by the selected photos. This explains the drop of the increasing rate for large x.

3.7.2.3 Results on Min-Selection with k-Coverage

In this part, we study the Min-Selection problem with k-coverage by comparing five different coverage settings. Three of them are based on the k-coverage model, with $k = 1, 2, 3$, respectively. The other two are based on the 2-coverage+ model, with $\alpha = 15$ and 30 degrees. Figure 3.11(a) plots the number of selected photos as a function of the total number of available photos. The other parameters are fixed and shown in the figure. First, it is clear that all algorithms are able to take advantage of the increased number of photos, make better choices, and reduce the number of selected photos. When comparing 1-cover, 2-cover and 3-cover, the number of selected photos is almost proportional to the degree (k) of coverage. This shows that k-coverage does not fundamentally differ from single coverage. It just repeats single coverage for k times. We also compare 2-cover (i.e. 2-cover+ with $\alpha = 0$), 2-cover+ with $\alpha = 15$, and 2-cover+ with $\alpha = 30$. The $\alpha = 15$ line is pretty close to the $\alpha = 0$ line. However, $\alpha = 30$ requires at least 25% more photos than $\alpha = 15$. This suggests that the difficulty of achieving 2-coverage+ and the value of α are not linearly related. Once α becomes large, it requires much more

photos to achieve the desired coverage.

In Figure 3.11(b), the number of targets varies from 5 to 50 while the number of available photos is fixed at 1000. For a target, all the aspects from 0° to 360° should satisfy the required level of coverage, either k-coverage or 2-coverage+. We have similar observations as previous figures. The relationships between the lines are similar to those in Figure 3.11(a), and the increasing trend of the lines is similar to that in Figure 3.10(b).

In Figure 3.11(c), we fix the number of targets but change the amount of aspects that should satisfy the required coverage. The relationships between the lines are similar to those in Figure 3.11(a) and (b), and the increasing trend of the lines is similar to that in Figure 3.10(c).

Chapter 4

Resource-Aware Photo Crowdsourcing for Area Coverage

4.1 Background

In the previous chapter, we consider the crowdsourcing targets as points on the map, and select photos based on their coverage on those points. However, in some applications, the target of interest is an area instead of a point. For example, to provide street view for an area, the collected photos should cover as many points in the area as possible. For each point covered, the photos should be taken from multiple angles to provide complete views. To achieve such coverage, it is important to understand how each point in the area is covered from all the different angles, and also how the photos overlap with each other, which results in redundant coverage. Since there are infinite number of points in the area, and there are many possible overlapping patterns between photos, it is a big challenge to efficiently analyze area coverage.

To address the challenge, we extend the definition of *photo utility* to make it suitable for area coverage. Specifically, the new photo utility quantifies how well a target area is covered by considering the coverage status of all points in the area based on integration. Then, we propose various techniques to analyze the overlapping patterns between photos and calculate photo utility efficiently and accurately. Note that although the coverage problem has been studied in

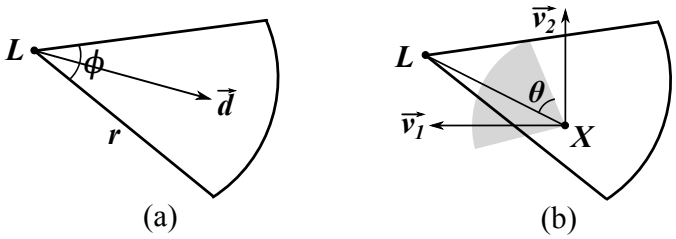


Figure 4.1. (a) Coverage area of a photo. (b) Illustration of aspect coverage. Aspect $\overrightarrow{v_1}$ is covered since $\angle(\overrightarrow{v_1}, \overrightarrow{XL}) \leq \theta$; aspect $\overrightarrow{v_2}$ is not covered since $\angle(\overrightarrow{v_2}, \overrightarrow{XL}) > \theta$. In fact, all the 2θ aspects in the gray area are covered.

wireless sensor networks, our application and model are different. Photo utility is determined not only by the area covered, but also by the angle or viewpoint from which the photo is taken. Inside the area, each point should be covered by photos all around it rather than by just one. These issues pose new challenges and will be addressed in this chapter.

With photo utility, we formulate the problem of selecting photos with the largest utility under a resource budget as the *budgeted max-utility problem*. Since the problem is NP-hard, an efficient approximation algorithm is proposed. Through rigorous theoretical analysis, we prove that the algorithm is near optimal, achieving constant approximation ratio compared to the optimal solution. Finally, the effectiveness of the proposed techniques is evaluated through both simulations and real-world experiments based on Android smartphones.

4.2 Models and Notations

We first review the definition of metadata from the previous chapter. Then, based on metadata, we extend photo utility to make it suitable for area coverage.

4.2.1 Metadata

The metadata of a photo consists of four parameters, $(L, \overrightarrow{d}, \phi, r)$. Location L is the geographic coordinates of the place where the photo is taken. Orientation $\overrightarrow{d}$ is the viewing direction of the camera when the photo is taken. It is a vector coming out from the camera aperture and perpendicular to the image plane. Field of view ϕ is an angle specifying how wide the camera can see. Objects outside the field of

view will not appear in the photo. Range r specifies how far the camera can see. It is the distance beyond which objects are no longer clearly recognizable in the photo.

Fig. 4.1(a) shows how metadata determines the coverage area of a photo as a circular sector. Any point inside the coverage area is covered by the photo, and any point outside is not covered. Hence, the entire sector area is said to be covered by the photo. This sector model is widely used in previous studies on camera sensor networks [36, 35, 39].

4.2.2 Photo Utility

We extend photo utility to area coverage, which quantifies how well a set of photos can cover a target area. To clearly define utility, it is important to know that a camera normally captures the image of an object from a specific angle or direction. For example, when a photo shows a building, it shows either its front view, side view, or back view. These views may contain different information even though they are about the same building. For instance, the front side of a building may look good after a disaster but the back side can be on fire.

To capture the essence of camera, we use *aspects* to represent different sides of an object, and use *aspect coverage* to represent which sides of an object appear in a photo. Specifically, an aspect $\overrightarrow{v}$ of a point is a vector represented by an angle in $[0, 2\pi)$. Angle 0 represents the vector pointing to the right (east on the map), and it increases in the counter-clockwise direction. For ease of presentation, this angle is denoted as $arg(\overrightarrow{v})$. In Fig. 4.1(b), $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ are two aspects of point X with $arg(\overrightarrow{v_1}) = \pi$ and $arg(\overrightarrow{v_2}) = \frac{\pi}{2}$.

An aspect $\overrightarrow{v}$ of a point X is covered by a photo if X is inside the photo's coverage area and the angle between $\overrightarrow{v}$ and $\overrightarrow{XL}$ is less than a predefined threshold θ, called *effective angle*. Here $\overrightarrow{XL}$ is the photo's viewing direction on point X. In Fig. 4.1(b), the angle between $\overrightarrow{v_1}$ and $\overrightarrow{XL}$ is less than θ, which means $\overrightarrow{v_1}$ points towards the camera and thus appears in the photo (covered). In contrast, the angle between $\overrightarrow{v_2}$ and $\overrightarrow{XL}$ is more than θ, which means $\overrightarrow{v_2}$ points away from the camera and does not appear in the photo (not covered). As long as a point is covered by a photo, all its aspects from $arg(\overrightarrow{XL}) - \theta$ to $arg(\overrightarrow{XL}) + \theta$ are covered (gray area

in Fig. 4.1(b)). Therefore, the point has 2θ aspects covered by the photo.

Now we define the utility of a set of photos on a target area as follows.

Definition 6 (Utility)**.** *Let P be a set of photos, and $1_P(\overrightarrow{v})$ be an indicator function that equals 1 if aspect $\overrightarrow{v}$ is covered by at least one photo in P and 0 otherwise. The utility of P on a given point X is*

$$U_P(X) = \int_0^{2\pi} 1_P(\overrightarrow{v})\, d(arg(\overrightarrow{v})), \tag{4.1}$$

where $\overrightarrow{v}$ is an aspect of point X and $arg(\overrightarrow{v})$ is the variable of integration changing from 0 to 2π. The utility of P on a target area A is

$$U_P(A) = \int_A U_P(X)\, dX. \tag{4.2}$$

The above definition of photo utility reflects the requirement that the collected photos should cover as much area as possible and for each point covered, the photos should be taken from multiple angles and thus cover as many aspects as possible. If multiple photos in P cover the same aspect $\overrightarrow{v}$, it is counted only once towards the total utility since $1_P(\overrightarrow{v}) = 1$. In this case, these photos contain redundant information and their total utility is less than the sum of their individual utility.

4.3 Utility Calculation

Given a target area A and a set of photos P, the first problem is how to calculate utility $U_P(A)$ according to Definition 6. Since $U_P(A)$ is an integral of $U_P(X)$ on each point $X \in A$, we need to analyze how each point X is covered and calculate its $U_P(X)$ value. However, there are infinite number of points in the area, and different points can be covered by different photos from different angles. These issues make coverage analysis and utility calculation highly nontrivial. Therefore, we first study a simple case with two photos before looking into multi-photo cases.

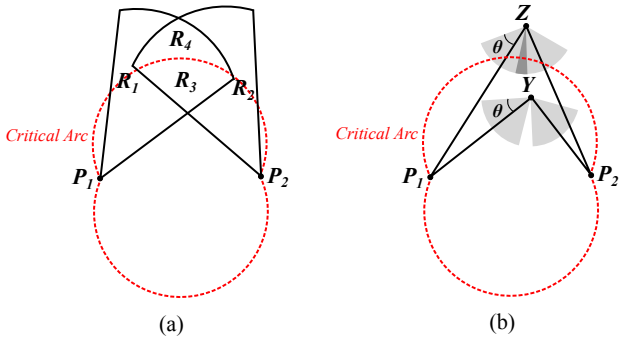


Figure 4.2. Two-photo case. (a) The area covered by P_1 and P_2 is partitioned into four regions. (b) Critical arcs determine whether photo coverage overlaps with each other.

4.3.1 Two-Photo Case

Consider two photos $P = \{P_1, P_2\}$ in Fig. 4.2(a). For simplicity, assume their coverage area is completely inside a much larger target area A (not shown in the figure). To calculate $U_P(A)$, we need to analyze how each point is covered by P and calculate its $U_P(X)$ value. To do this, we partition the area covered by P into four non-overlapping regions R_1, R_2, R_3, R_4. Let a_i denote the area of R_i $(i = 1, 2, 3, 4)$, and consider R_1 and R_2 first. Any point (say X) in R_1 is only covered by photo P_1, and thus it has 2θ aspects covered by P; i.e., $U_P(X) = 2\theta$. By Definition 6, the photo utility for region R_1 is $U_P(R_1) = \int_{R_1} U_P(X)\, dX = 2\theta a_1$. Similarly, the photo utility for region R_2 is $U_P(R_2) = 2\theta a_2$. Now consider regions R_3 and R_4, which are covered by both photos and separated by a *critical arc* $\overset{\frown}{P_1P_2}$.

Definition 7 (Critical Arc)**.** *Given a pair of photos P_1, P_2, let P_1, P_2 also denote the locations of the two photos if there is no ambiguity. A critical arc $\overset{\frown}{P_1P_2}$ is a circular arc between P_1 and P_2 such that any point X on it satisfies $\angle P_1XP_2 = 2\theta$. As shown in Fig. 4.2(a), there exist two critical arcs that are symmetric with respect to line $\overline{P_1P_2}$*[1]*.*

With critical arcs, we have the following properties established. Any point Y in region R_3 is on the inner side of the arcs, and thus satisfies $\angle P_1YP_2 > 2\theta$ (see Fig. 4.2(b)). This means that the viewing directions of P_1 and P_2 are far enough from each other such that the aspects covered by P_1 and P_2 do not overlap. Hence,

[1]From basic geometry, the center C of the critical arc must be on the perpendicular bisector of $\overline{P_1P_2}$ and satisfy $\angle P_1CP_2 = 4\theta$. This can be used to determine the position of the center and hence the critical arc.

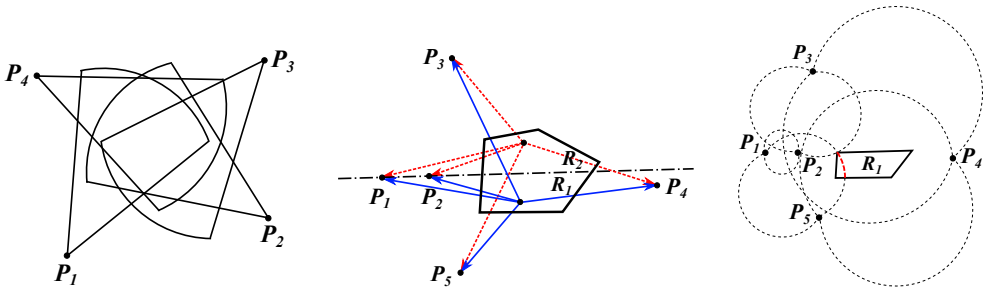
point Y has 4θ aspects covered by P; i.e., $U_P(Y) = 4\theta$. On the other hand, any point Z in region R_4 is on the outer side of the arcs, and thus has $\angle P_1ZP_2 < 2\theta$. This means that the viewing directions of P_1 and P_2 are close enough to each other such that the aspects covered by P_1 and P_2 have some overlap (dark gray area in Fig. 4.2(b)). Hence, point Z has $2\theta + \angle P_1ZP_2$ aspects covered by P; i.e., $U_P(Z) = 2\theta + \angle P_1ZP_2$. It follows that the photo utility for R_3 is $U_P(R_3) = 4\theta a_3$, and the photo utility for R_4 is $U_P(R_4) = 2\theta a_4 + \int_{R_4} \angle P_1ZP_2 \, dZ$. We will discuss how to calculate integral $\int_{R_4} \angle P_1ZP_2 \, dZ$ in Section 4.3.3. Once it is calculated, the total utility can be given as

$$\begin{aligned} U_P(A) &= U_P(R_1) + U_P(R_2) + U_P(R_3) + U_P(R_4) \\ &= 2\theta(a_1 + a_2 + 2a_3 + a_4) + \int_{R_4} \angle P_1ZP_2 \, dZ. \end{aligned}$$

Note that calculating $U_P(A)$ via the above formula requires the calculation of area a_1, a_2, a_3, a_4. Since regions R_1, R_2, R_3, R_4 can be arbitrary shapes formed by line segments and arcs, it is very complex to calculate the exact area value. Hence, we approximate each arc by a series of line segments (i.e., a polyline) and thus change all regions to polygons. The area of a polygon can be calcuated in $O(n)$ time, where n is the number of vertices [65]. By tuning the number of line segments that replace an arc, we can control the tradeoff between accuracy and computational complexity.

4.3.2 Multi-Photo Case

In this subsection, we extend the solution in the two-photo case to consider the general scenario. Again, we assume the coverage area of all the photos is completely inside a much larger target area. If not, only the portion inside the target area is the effective coverage area, and the portion outside is discarded. With multiple photos covering the same area, the overlap among photos in terms of aspects covered is hard to calculate. As in the two-photo case, the key to solve the problem is to partition the area into small regions such that in each region, the points are under the same coverage condition and the $U_P(X)$ value can be calculated using the same expression. In the following, we elaborate on the three steps of partitioning the



(a) Coverage area partition (b) Photo line partition (c) Critical arc partition

Figure 4.3. Multi-photo case. The target area is partitioned into small regions in three steps, so that all points in the same region have the same coverage condition. Then utility can be calculated for each small region separately.

target area, and describe how to calculate the utility of each small region obtained from the partition.

4.3.2.1 Coverage Area Partition

The coverage area of photos naturally partitions the target area into small, non-overlapping regions, as shown in Fig. 4.3(a). We have the following property after the partition.

Property 1. *Given a region obtained from coverage area partition, all points in it are covered by the same set of photos.*

4.3.2.2 Photo Line Partition

Consider a region R obtained from coverage area partition. Based on Property 1, there exists a set of photos $P = \{P_1, \ldots, P_n\}$ covering R. To analyze how these photos cover R, we need another partition step, called *photo line partition*. Specifically, we check line $\overline{P_iP_j}$ for all $i \neq j$. If it intersects with R, and P_i, P_j are on the same side of R, then R is divided into two smaller regions by line $\overline{P_iP_j}$; otherwise R is unchanged. After all lines are checked, R is divided into several smaller regions that satisfy the following property.

Property 2. *Given a region obtained from photo line partition, the photos covering the region can be ordered in clockwise (or counter-clockwise) direction according to their positions with regard to the region.*

As an example, consider the region in Fig. 4.3(b) covered by $\{P_1, \ldots, P_5\}$. After all lines are checked, the region is divided by $\overline{P_1P_2}$ into two smaller regions R_1, R_2. For R_1, the photos can be ordered in clockwise direction as P_1, P_2, P_3, P_4, P_5, and for R_2, that order is P_2, P_1, P_3, P_4, P_5. Note that the region is not divided by $\overline{P_1P_4}$ because P_1, P_4 are on different sides of the region and their order is the same for R_1 and R_2. The same applies to $\overline{P_2P_4}$.

4.3.2.3 Critical Arc Partition

Consider a region R' obtained from photo line partition. By Property 2, the photos covering R' can be ordered in clockwise direction, denoted as $P_1, P_2, \ldots, P_n$. The purpose of considering this order is to analyze the coverage overlap between two adjacent photos P_i and P_{i+1}, ($i = 1, 2, \ldots, n$ and $P_{n+1} = P_1$). For a point $X \in R'$, if $\angle P_iXP_{i+1} \leq 2\theta$, the viewing directions of P_i and P_{i+1} are close enough to each other such that the coverage of P_i and P_{i+1} overlaps. Thus, all the aspects within $\angle P_iXP_{i+1}$ are covered. If $\angle P_iXP_{i+1} > 2\theta$, the coverage of P_i and P_{i+1} does not overlap, which means only 2θ aspects within $\angle P_iXP_{i+1}$ are covered. Combining the above two cases, there are $\min\{2\theta, \angle P_iXP_{i+1}\}$ aspects covered within $\angle P_iXP_{i+1}$. Thus, the utility of photos $P = \{P_1, \ldots, P_n\}$ on point X is

$$U_P(X) = \sum_{i=1}^{n} \min\{2\theta, \angle P_iXP_{i+1}\}. \tag{4.3}$$

Then the tricky part is that for any given i, $\angle P_iXP_{i+1}$ could be either greater or smaller than 2θ depending on the position of X. Thus, we further divide R' by critical arcs $\widehat{P_1P_2}, \widehat{P_2P_3}, \ldots, \widehat{P_nP_1}$ to obtain the following property.

Property 3. *Given a region obtained from critical arc partition and any two adjacent photos P_i and P_{i+1}, either $\angle P_iXP_{i+1} < 2\theta$ holds for all points X in the region, or $\angle P_iXP_{i+1} \geq 2\theta$ holds for all points X in the region.*

As an example, consider one of the two regions, R_1, obtained from the photo line partition in Fig. 4.3(b). We perform critical arc partition on R_1 in Fig. 4.3(c). R_1 is completely outside the critical arcs $\widehat{P_1P_2}$ and $\widehat{P_2P_3}$, which means for $\forall X \in R_1$, $\angle P_1XP_2 < 2\theta$ and $\angle P_2XP_3 < 2\theta$. R_1 is completely inside the critical arcs $\widehat{P_3P_4}$ and $\widehat{P_4P_5}$, which means for $\forall X \in R_1$, $\angle P_3XP_4 \geq 2\theta$ and $\angle P_4XP_5 \geq 2\theta$. $\widehat{P_5P_1}$

divides R_1 into two regions. For the left smaller region, $\angle P_5XP_1 \geq 2\theta$, and for the right larger region, $\angle P_5XP_1 < 2\theta$. Hence, for the left smaller region, $U_P(X) = 6\theta + \angle P_1XP_3$; for the right larger region, $U_P(X) = 4\theta + \angle P_5XP_3$.

4.3.2.4 Utility Calculation for Partitioned Regions

Denote a region obtained from the above three partition steps as R''. Based on Property 3, set $I_1 = \{i \,|\, \angle P_iXP_{i+1} \geq 2\theta\}$ and $I_2 = \{i \,|\, \angle P_iXP_{i+1} < 2\theta\}$ can be uniquely determined. Then equation (4.3) becomes $U_P(X) = 2\theta|I_1| + \sum_{i\in I_2} \angle P_iXP_{i+1}$. Thus, the photo utility for R'' can be given as

$$\begin{aligned} U_P(R'') &= \int_{R''} U_P(X)\, dX \\ &= 2\theta a|I_1| + \sum_{i\in I_2} \int_{R''} \angle P_iXP_{i+1}\, dX, \end{aligned}$$

where a is the area of R''. We will discuss how to calculate integral $\int_{R''} \angle P_iXP_{i+1}\, dX$ in Section 4.3.3. Once it is calculated, the total utility $U_P(A)$ can be obtained by adding up $U_P(R'')$ for all the regions obtained from the partition.

4.3.3 Integral over a Polygon

In this subsection, we discuss how to calculate integral $\int_{R''} \angle P_1XP_2\, dX$, where R'' is a polygon obtained from the above three partition steps and P_1, P_2 are two photos covering R''. Note that R'' is considered as a polygon since arcs are approximated by polylines.

To the best of our knowledge, the antiderivative of $\angle P_1XP_2$ cannot be expressed as elementary functions, which means it is mathematically impossible to obtain the exact integral value. Hence, the integral must be approximated by numerical integration such as [66]. Numerical integration picks certain number of sample points in the polygon and derives the weight of each sample point. It then evaluates the integrand on those points (i.e., calculates $\angle P_1XP_2$ for each sample point X), and sums up the weighted results to get the integral. The number of sample points serves as a knob to balance accuracy and computation time.

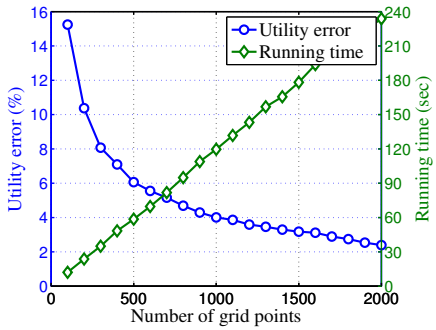


Figure 4.4. Utility error and running time of the grid point method.

4.3.4 Accuracy of Utility Calculation

To see the accuracy of utility calculation, we randomly generate the metadata of 50 photos in a 600m by 600m square area, and calculate their photo utility with regard to the area. The locations and orientations of the photos are uniformly random, and the field of view and range are fixed at 60° and 100m, respectively. The experiment is repeated 500 times on an Intel Core i5 3.30GHz machine, and both the average utility error and the total running time are obtained. To get the utility error, we calculate the exact utility value by setting the number of line segments that replace an arc to be 100 (i.e., sufficiently large), and setting the number of sample points in numerical integration to be 10000 (i.e., sufficiently many). In contrast, our method uses eight line segments to replace each arc and uses one sample point for each numerical integration. The average utility error of our method is 0.42%, and the total running time is 52 seconds.

As a comparison, we also calculate the utility using an intuitive method called *grid point method*, which simply samples the area with grid points. Specifically, the grid consists of $\sqrt{n} \times \sqrt{n}$ points evenly distributed over the square area (denoted as A), with the set of all grid points denoted as N ($|N| = n$). For every grid point $X \in N$, its $U_P(X)$ value is calculated by Definition 6. Then the total utility is estimated by $U_P(A) \approx \frac{a}{n} \sum_{X \in N} U_P(X)$, where a is the area of A and $\frac{a}{n}$ can be considered as the weight of each grid point. The average utility error and the total running time as a function of the number of grid points n is shown in Fig. 4.4. As can be seen, when the running time is 52 seconds, the grid point method uses about 400 points which has 7.2% utility error, compared to our method of 0.42% error. Therefore, with the same running time, our method is much more accurate

than the grid point method. Although the accuracy of the grid point method can be improved by increasing the number of grid points, its running time will be much longer compared to our method.

4.4 Budgeted Max-Utility Problem

With photo utility, we can measure how well photos cover a target area and select those that best serve the purpose of the application. Since different applications may have different photo selection problems, in this chapter, we study one fundamental problem called the budgeted max-utility problem.

4.4.1 Problem Statement

Consider a photo crowdsourcing application where the server is interested in the condition of a target area, e.g., how it is damaged after a disaster. The server issues a request to mobile users, who take photos and upload metadata to the server for evaluation. With the metadata of hundreds or thousands of photos, the server selects a subset of them such that the target area is best covered and the resource consumption of collecting or analyzing the selected photos is under a predefined budget. The budget can be any type of resource budget, such as the bandwidth limit for uploading the photos, the processing capability in terms of CPU cycle, or the total incentives that can be paid to mobile users. After selection, the server asks mobile users to upload the image files of the selected photos. The uploaded photos can be reviewed by humans or be fed into computer vision programs to extract useful information. With photo utility, the above problem can be formulated as follows.

Definition 8 (Budgeted Max-Utility Problem)**.** *Given a target area A, which is a bounded area of any shape*[2]*, also given a set of photos $P = \{P_1, \ldots, P_n\}$ and the resource cost associated with each photo $c_1, \ldots, c_n$, the goal of the problem is to select a subset of photos $P' \subseteq P$ such that the photo utility $U_{P'}(A)$ is maximized and the selected photos are subject to a knapsack constraint $\sum_{i:P_i \in P'} c_i \leq B$, where $B > 0$ is a predefined resource budget.*

[2]The problem and solution can be easily extended to multiple disjoint areas.

To see the difficulty of the problem, consider a special case of the problem where $\theta = \pi$ and $c_i = 1$ for all $i = 1, \ldots, n$. Since $\theta = \pi$, as long as a point is inside the coverage area of a photo, all of its 2π aspects are covered. Hence, any point X covered by P' has utility $U_{P'}(X) = 2\pi$, and the total utility is given by $U_{P'}(A) = \int_A U_{P'}(X)\, dX = 2\pi a$, where a is the total area covered by P'. Since $c_i = 1$ for all $i = 1, \ldots, n$, the knapsack constraint becomes a cardinality constraint $|P'| \leq B$. Then the problem is to select a subset P' such that $|P'| \leq B$ and the total coverage area a is maximized. This is known as the NP-hard maximum coverage problem [67].

4.4.2 Our Algorithm

Due to the hardness of the problem, we design efficient approximation algorithms. One natural idea is based on the greedy algorithm, which starts with the empty selection $P' = \emptyset$, and in each step, selects a photo P_i which maximizes the *per-cost marginal utility*, $(U_{P' \cup \{P_i\}}(A) - U_{P'}(A))/c_i$. The algorithm stops when the resource budget is met. Unfortunately, this cost-aware algorithm can perform arbitrarily badly compared to the optimal solution. Consider the example where we have photo P_1 with utility $U_{\{P_1\}}(A) = 2\epsilon$ and cost $c_1 = \epsilon$, and photo P_2 with utility $U_{\{P_2\}}(A) = B$ and cost $c_2 = B$. This cost-aware algorithm would select P_1 since $(U_{\{P_1\}}(A) - U_{\emptyset}(A))/c_1 = 2$ and $(U_{\{P_2\}}(A) - U_{\emptyset}(A))/c_2 = 1$. Then it cannot afford P_2 as the remaining budget is $B - \epsilon$, and the total utility it achieves is 2ϵ. In contrast, the optimal solution would select P_2, achieving B utility. As $\epsilon \to 0$, the algorithm becomes arbitrarily bad.

In the above example, we notice that simply ignoring the cost and selecting the photo with the largest utility (i.e., P_2) would yield the optimal solution. This idea inspires the cost-ignored algorithm, which in each step, selects the photo P_i that maximizes the *marginal utility*, $U_{P' \cup \{P_i\}}(A) - U_{P'}(A)$, without considering the cost. It is easy to see that the cost-ignored algorithm can also perform arbitrarily badly since it would select a photo with B utility and B cost even though there are many photos with $B - \epsilon$ utility and ϵ cost.

However, we have found that at least one of the two greedy algorithms, cost-aware algorithm or cost-ignored algorithm, cannot perform too badly. Thus, in

our algorithm, we use the cost-aware algorithm to get a result U_{ca}, and also use the cost-ignored algorithm to get a result U_{ci}. Then we use the better of the two results, $\max\{U_{ca}, U_{ci}\}$, as the final output. The following theorem shows the performance bound (i.e., approximation ratio) of our algorithm.

Theorem 5. *Let U_{ca} be the photo utility achieved by the cost-aware algorithm, and U_{ci} be the photo utility achieved by the cost-ignored algorithm. Also let U_{opt} be the maximal utility that can be achieved for the problem. We have*

$$\frac{\max\{U_{ca}, U_{ci}\}}{U_{opt}} > \frac{1 - \frac{1}{e}}{2} \approx 0.32.$$

Proof. According to Krause *et al.* [68], it suffices to show that the objective function, photo utility, is monotone and submodular.

Monotonicity: For any two sets of photos P, Q with $P \subseteq Q$ and every aspect $\overrightarrow{v}$, we have $1_P(\overrightarrow{v}) \leq 1_Q(\overrightarrow{v})$ since an aspect covered by P is also covered by Q. After integration, it follows that $U_P \leq U_Q$ (we use U_P to represent $U_P(A)$ for conciseness), which meets the definition of monotonicity.

Submodularity: By definition, photo utility is submodular if for any two sets of photos P, Q with $P \subseteq Q$ and every photo $P_i \notin Q$, we have

$$U_{P \cup \{P_i\}} - U_P \geq U_{Q \cup \{P_i\}} - U_Q.$$

The above formula can be rewritten as

$$\begin{aligned} &\int_A [U_{P \cup \{P_i\}}(X) - U_P(X)]\, dX \\ \geq &\int_A [U_{Q \cup \{P_i\}}(X) - U_Q(X)]\, dX, \end{aligned}$$

and further as

$$\begin{aligned} &\int_A \int_0^{2\pi} [1_{P \cup \{P_i\}}(\overrightarrow{v}) - 1_P(\overrightarrow{v})]\, d(arg(\overrightarrow{v}))\, dX \\ \geq &\int_A \int_0^{2\pi} [1_{Q \cup \{P_i\}}(\overrightarrow{v}) - 1_Q(\overrightarrow{v})]\, d(arg(\overrightarrow{v}))\, dX. \end{aligned}$$

Hence, it suffices to show that for every aspect $\overrightarrow{v}$,

$$1_{P\cup\{P_i\}}(\overrightarrow{v}) - 1_P(\overrightarrow{v}) \geq 1_{Q\cup\{P_i\}}(\overrightarrow{v}) - 1_Q(\overrightarrow{v}),$$

which can be done by discussing the following three cases.
Case 1: When both P and Q cover aspect $\overrightarrow{v}$, both sides of the above inequality become 0, and thus the inequality holds.
Case 2: When neither P nor Q covers aspect $\overrightarrow{v}$, both sides become $1_{P_i}(\overrightarrow{v})$, and thus the inequality holds.
Case 3: When Q covers aspect $\overrightarrow{v}$ but P does not, LHS becomes $1_{P_0}(\overrightarrow{v})$ and RHS becomes 0. It is true that $1_{P_0}(\overrightarrow{v}) \geq 0$.

These three cases cover all possibilities and hence the proof is complete. □

4.5 Performance Evaluations

In this section, we evaluate the performance of the proposed solutions. In particular, we are interested in answering the following questions:

1. Is our utility metric effective? Does high photo utility really imply good photo coverage?

2. Is our photo selection algorithm effective? How does it compare with other algorithms?

3. What is the impact of the number of candidate photos (n), the resource budget (B), and the effective angle (θ)?

We answer questions 1) and 2) by using real-world experiments, and answer question 3) with simulations.

4.5.1 Real-World Experiments

4.5.1.1 Experimental Setup

We use a Nexus 4 phone running Android 5.1.1 to take photos around a target area, with metadata automatically obtained and recorded in the phone (see Section 3.6.1

(a) (b) (c)

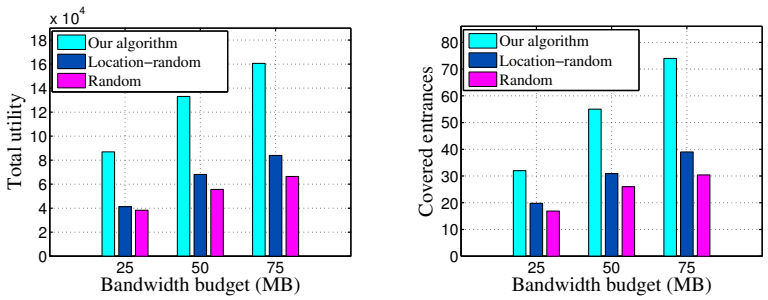
Figure 4.5. (a) Satellite image of the target area. (b) A photo covering two front entrances with white stairs. (c) A photo covering two back entrances with brown decks.

and our paper [69] for details). The target area (large pentagon in Fig. 4.5(a)) is a local townhouse community with 14 rows of houses (small rectangles), where each row consists of 6 or 8 units. The community spreads over a 264m by 183m area, and has a total of 106 units. Every two neighboring units, as shown in Fig. 4.5(b)(c), share a common front entrance with white stairs and a common back entrance with a brown deck. Hence, there are 53 front entrances and 53 back ones, for a total of 106 entrances.

The goal of this experiment is to select photos that cover the community as well as possible. Since how well the community is covered is often a subjective matter, we quantify it by counting the number of entrances appear in the photos. If all entrances appear in the selected photos, it is considered perfect coverage, which is expected to fully describe the condition of the community such as how it is damaged after a disaster.

We take 150 candidate photos around the community using the phone mentioned above. We use a bandwidth limit for uploading the selected photos as the resource budget, and the file size of each photo as its resource cost[3]. From the 150 candidates, which have a total size of 400MB, we select up to 75MB of photos using three selection algorithms: 1) **Our algorithm:** the proposed photo selection algorithm; 2) **Location-random:** select randomly from the photos whose locations are inside the target area; and 3) **Random:** select randomly from all candidate photos. For each set of selected photos, we record both the total utility and the total number of entrances covered (i.e., appear in the photos). Note that if an entrance is covered by multiple photos, it is counted only once towards the

[3]Bandwidth is only used as an example. It can be changed to any quantifiable resources in practice.



(a) (b)

Figure 4.6. Experiment results: (a) utility vs. bandwidth budget; (b) covered entrances vs. bandwidth budget.

total.

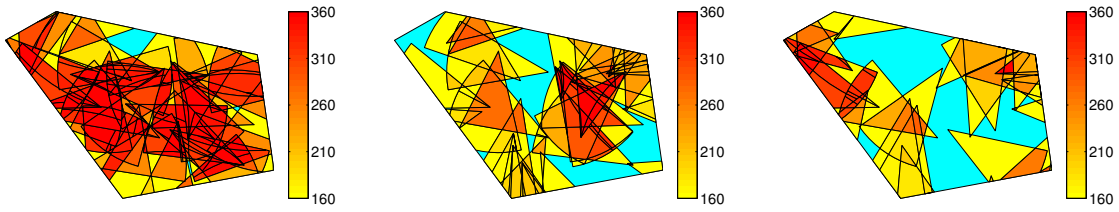
4.5.1.2 Results

Fig. 4.6 shows the results, where the results of Location-random and Random are the average of 20 repeated experiments. We have the following three observations.

First, photo utility is an effective metric of photo coverage. Generally speaking, the results for photo utility match the results for covered entrances. Photos with higher utility cover more entrances, and photos with lower utility cover fewer entrances. This indicates that our utility metric can truly reflect the coverage of real photos.

Second, our algorithm achieves about 100% more utility and covers about 85% more entrances than Location-random, which performs better than Random. Our algorithm is much better because it considers photo utility and resource cost across the selection process and it has a proved approximation ratio of 0.32. Location-random is better than Random because it limits candidate photos to those located inside the target area, and thus ensures that every photo it selects covers some part of the target area. However, location itself is not enough to determine the photo coverage, which is a major reason that Location-random is not comparable to our algorithm.

Third, when photos are selected properly, we can save a significant amount of bandwidth and still achieve excellent coverage on the target area. Note that uploading all the 150 candidate photos would consume 400MB bandwidth. They achieve 1.66×10^5 utility and cover 86 of the 106 entrances, while the remaining



(a) Our algorithm (b) Location-random (c) Random

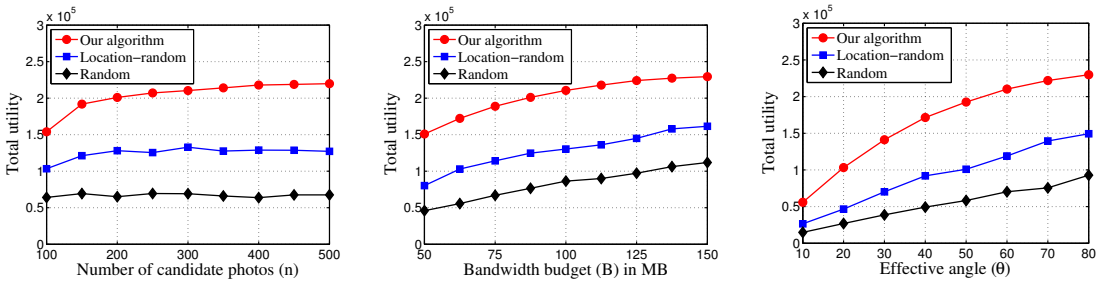
Figure 4.7. Heat maps of the photos selected by the three algorithms. This figure is better viewed in color.

20 are back entrances (decks) surrounded by trees and not "coverable" by any photos. In comparison, uploading the photos selected by our algorithm consumes only 75MB ($\frac{75}{400} \approx 19\%$) bandwidth, yet they achieve 1.60×10^5 ($\frac{1.60}{1.66} \approx 96\%$) utility and cover 74 ($\frac{74}{86} \approx 86\%$) entrances. We believe this is very good coverage and these photos can accurately reflect the condition of the community.

Fig. 4.7 further illustrates how well the selected photos cover the community. Subfigures (a)(b)(c) shows the 75MB of photos selected by our algorithm, Location-random, and Random, respectively. As can be seen, the target area (large pentagon) is partitioned into many small regions by the method described in Section 4.3. Each region is colored based on how many aspects are covered on its centroid, i.e., cyan if no aspects are covered, yellow if 160° aspects are covered (θ is set to 80° here so one photo covers 160°), and red if all 360° aspects are covered. These heat maps show that our algorithm achieves much better coverage than the other two algorithms. Many regions in subfigure (a) are fully or almost fully covered by the selected photos, which is why they can capture most of the entrances in the community.

4.5.2 Simulations

We generate photos with random locations and orientations in a 400m by 400m square, and set the 200m by 200m square in the center as the target area. The photos have 60° field of view and 100m range. The file size of each photo is generated randomly in $[2, 4]$MB. In each experiment, we select photos from n candidate photos under bandwidth budget B using the same selection algorithms as those in the real-world experiments, and then record the total utility achieved by the selected photos.



(a) (b) (c)

Figure 4.8. Simulation results: (a) utility vs. number of candidate photos; (b) utility vs. bandwidth budget; (c) utility vs. effective angle.

Fig. 4.8 shows the simulation results, where the results of Location-random and Random are the average of 20 repeated experiments. Our algorithm performs much better than the other two across all experiments. In what follows, we discuss the impact of the number of candidate photos (n), the bandwidth budget (B), and the effective angle (θ), respectively.

The impact of the number of candidate photos (n) is shown in Fig. 4.8(a), where $B = 100$MB and $\theta = 60°$. As can be seen, Location-random or Random does not benefit from the increase of n because they select photos randomly. On the other hand, the results of our algorithm exhibit diminishing returns as n increases. In particular, the total utility increases very little after $n > 400$. This means that there is a turning point n' (in this case $n' \approx 400$) beyond which crowdsourcing more candidate photos cannot bring enough benefit. In real applications, the server can find such a turning point by simulations and crowdsource no more than n' photos.

Fig. 4.8(b) shows the impact of the bandwidth budget (B) with $n = 300$ and $\theta = 60°$. Again, diminishing returns can be observed for our algorithm, and a turning point can be found roughly at $B' \approx 125$MB. Note that in real applications, the turning point is not determined subjectively. Instead, it is the point where the cost of collecting or selecting more photos exceeds the benefit brought by their utility. For the other two lines, they grow almost linearly as B increases. This is because for those two algorithms, the total utility is much less than the maximum possible utility (2.5×10^5, i.e., 2π times the size of the target area). Thus on average, every selected photo brings roughly the same increase in utility.

Fig. 4.8(c) shows the impact of effective angle (θ) with $n = 300$ and $B = 100$MB. As θ increases, the aspects covered by each photo increase linearly, and thus the total utility grows almost linearly when it is much less than the maximum

possible utility. The growth slows down when the total utility becomes larger, since there are more coverage overlaps between photos. In real applications, θ controls the number of photos needed to fully cover (i.e., cover all aspects of) an object. It should be set based on the level of details that the application demands for the covered objects.

4.6 Discussions

4.6.1 Occlusion

As mentioned in the previous chapter, occlusion means that the view of the target is blocked by obstacles, though the target is considered to be covered based on the metadata. One way to detect occlusion is to process the photo using resource-intensive computer vision algorithms, which is contrary to the purpose of resource-aware crowdsourcing. Another way to detect occlusion is to use a camera parameter called *Depth-Of-Field*. When the target is a point and it is outside the camera's Depth-Of-Field, then it must be occluded (see Section 3.6.3 for details). When the target is an area, the same technique can be used to detect whether any part of the area is occluded. However, this technique cannot detect the case where the target and the obstacle are close to each other. Another idea is to check a map and use the locations and shapes of buildings to detect occlusion caused by buildings. This idea needs further investigation and it cannot detect occlusion caused by other obstacles such as trees and vehicles. Due to its complexity, we leave this study as future work.

It is important to note that occlusion issue is orthogonal to the contributions of this work. Since everything behind the obstacle is not visible in the photo, occlusion effectively reduces the photo's coverage area. The coverage area may no longer be a circular sector, and it can be an arbitrary shape depending on the locations and shapes of obstacles. However, our design works for arbitrary shapes of photo coverage area. If the occlusion issue is solved and a more accurate coverage area is found, we can use it to improve the performance. Otherwise, we can still use the sector model as it represents the general situation and works well in our real-world experiments. Moreover, our utility metric encourages each point to be

covered from multiple directions, which effectively mitigates the occlusion issue because an object blocked in one direction may be visible from other directions.

4.6.2 Photo Quality Control

Crowdsourced photos can be of low quality due to problems like blurring, distortion, noise, and improper brightness. Collecting and analyzing such photos is a waste of resources. Thus, before metadata is sent to the server, image quality assessment (IQA) [70, 64] can be used in mobile devices to detect low-quality photos. The metadata of unqualified photos will not be sent to the server or be considered in photo selection. However, existing image processing techniques are computationally expensive, and thus should be carefully adapted considering the resource limitations of mobile devices.

Chapter 5

Resource-Aware Photo Crowdsourcing Through Disruption Tolerant Networks

5.1 Background

An important application of photo crowdsourcing is emergency management [3]. In a natural disaster or a battlefield, a command center may need information about some specific targets. Rescuers, survivors and soldiers in the field can use mobile devices to take photos and upload them to the command center. The information contained in the photos, such as the damage of a building or the extent of flooding, helps the command center make critical decisions on the assignment of manpower, equipment, and supplies.

Unfortunately, the communication between the command center and crowdsourcing participants can be extremely constrained in these scenarios. The cellular network may be partly damaged or overloaded with extensive requests, and hence not accessible to all participants. Human-carried satellite radios may be an option, but only a small portion of participants have satellite radios due to the high cost. As a result, it is better to use Disruption Tolerant Networks (DTNs) to transfer photos among participants, and once available, use the cellular network or satellite connections to upload photos to the command center. Although DTN cannot

guarantee prompt data delivery, it may be the last resort and its cost-effectiveness also makes it a feasible solution in such resource constrained environments.

Even with DTN, how to save resources such as storage and bandwidth poses many challenges. Participants can take many megapixel photos which consume a lot of storage space. The photos may contain redundant information, or may be irrelevant to the targets of interest. These redundant or irrelevant data exacerbate the contention for storage resources. Moreover, data can only be transferred when two participants are within the wireless transmission range of each other. Transmitting redundant or irrelevant data reduces the chance to transmit other useful data since the wireless bandwidth is limited and the participants may move out of the transmission range quickly. Thus, it is important to eliminate redundant or irrelevant photos without sacrificing useful information related to the targets of interest.

We take two steps to address these challenges. First, based on the proposed metadata, we define a *photo coverage model* to quantify the value of photos. Second, we study how to incorporate the value of photos into routing, so that the most useful photos are prioritized to use the limited storage and bandwidth resources. There are some existing utility-driven routing algorithms for DTN [44, 45, 46], and simply using photo coverage as the utility metric and running their algorithms could enable us to prioritize photos based on their individual coverage. However, this simple solution does not consider the redundancy between photos. If two similar photos both have high coverage, those algorithms will prioritize both photos in routing, without considering that it is only meaningful to deliver one of them. This unique feature, caused by the redundancy (or coverage overlap) between photos, motivates us to develop a *photo selection algorithm* different from previous utility-based approaches. Our algorithm considers both the coverage overlap between photos and the contact opportunities related to user mobility. It prioritizes the storage and transmission of the most useful photos, and thus significantly increases the value of the photos delivered to the command center.

In this chapter, we propose a resource-aware photo crowdsourcing framework based on DTNs, including a photo coverage model and a photo selection algorithm. The photo coverage model quantifies the value of photos using lightweight metadata, and the photo selection algorithm considers the coverage overlap between

photos, which is unique for photo crowdsourcing and has not been addressed in previous utility-based routing algorithms. We evaluate our design through experiments based on Android smartphones and through extensive trace-driven simulations. The results demonstrate that our resource-aware framework works well in practice and significantly increases the value of the crowdsourced photos.

5.2 Photo Coverage Model

In this section, we introduce the model used to quantify the value of photos (called *photo coverage*).

5.2.1 PoIs and Photos

The command center has some *Points of Interest (PoIs)* it wants to observe. It issues a PoI list containing the coordinates of the PoIs, and spreads it to as many participants as possible through DTN or other communication networks. The PoI list is denoted as $X = \{x_1, x_2, x_3, \cdots\}$, where each x_i is a PoI. Without ambiguity, x_i also means the location of the i-th PoI.

Each participant has some photos in the mobile device, called his/her *photo collection*. A photo collection is denoted as $F = \{f_1, f_2, f_3, \cdots\}$, where each f_j is a photo. A photo f is characterized by its metadata $(l, r, \phi, \overrightarrow{d})$, which is the same metadata as in the previous two chapters. Here l is the location where the photo is taken. r is the coverage range of the camera, beyond which people can hardly identify anything in the photo. ϕ is the field-of-view of the camera, which determines how wide the camera can see. $\overrightarrow{d}$ is the orientation of the camera when the photo is taken. It can be expressed as a vector coming out from the camera and vertical to the image plane. These four parameters jointly determine the coverage area of a photo (gray area in Fig. 5.1(a)).

5.2.2 Point Coverage and Aspect Coverage

The crowdsourced photos should cover the PoIs. In traditional sensor networks, a target is covered if it is inside the coverage area of a sensor. Similarly, a PoI is covered if it is inside the coverage area of a photo, referred to as *point coverage*.

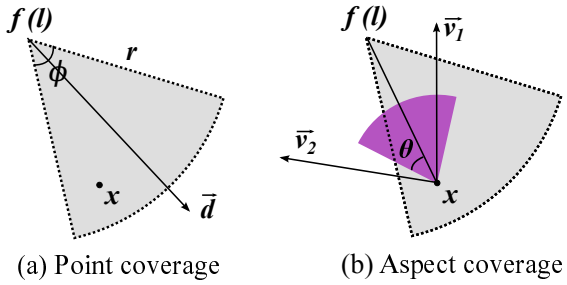


Figure 5.1. (a) For point coverage, PoI x is covered by photo f, so $C_{pt}(x, f) = 1$. (b) For aspect coverage, aspect $\overrightarrow{v_1}$ is covered by photo f since $\angle(\overrightarrow{v_1}, \overrightarrow{xl}) < \theta$. $\overrightarrow{v_2}$ is not covered since $\angle(\overrightarrow{v_2}, \overrightarrow{xl}) > \theta$. In fact, all the aspects in the darker area are covered by photo f, so $C_{as}(x, f) = 2\theta$.

For a PoI x and a photo collection $F = \{f_1, f_2, \cdots\}$, point coverage $C_{pt}(x, F) = 1$ if x is in the coverage area of any f_j from F; otherwise $C_{pt}(x, F) = 0$.

If $C_{pt}(x, F) = 1$, PoI x can be found in photo collection F. However, it is not clear how x appears in the photos. If the PoI is a building, we may see its front view, side view, or even back view. This is quite different from traditional sensor coverage, where the relative angle of sensors and targets does not matter. From the command center's perspective, only seeing the PoI is not good enough, and it is better to view the PoI from multiple directions so as to obtain omnibus information.

Therefore, we introduce *aspect coverage* to capture this unique property of photos. An aspect of a PoI, denoted by $\overrightarrow{v}$, is a vector that can be represented by an angle in $[0, 2\pi)$. Angle 0 represents the vector pointing to the right (east on the map), and it increases in the clockwise direction. For example, in Fig. 5.1(b), $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ are two aspects with angle 270° and 190°, respectively.

Aspect $\overrightarrow{v}$ of PoI x is covered by photo f if the following conditions hold: x is inside the coverage area of f, and the angle between $\overrightarrow{v}$ and $\overrightarrow{xl}$ is less than a predefined threshold θ (called *effective angle*). Here $\overrightarrow{xl}$ is the vector from the PoI to the camera, representing the viewing direction of the camera. Then for a PoI x and a photo collection F, we have aspect coverage $C_{as}(x, F) = \int_0^{2\pi} 1_F(v)dv$, where $1_F(v) = 1$ if $\overrightarrow{v}$ is covered by any f_j from F, or 0 otherwise.

Aspect coverage shows how many aspects of a PoI are covered in the photos. With larger aspect coverage, more different views of the PoI can be seen, and thus more information can be obtained about the PoI. However, only considering aspect

coverage may cause problems. A photo collection may have large aspect coverage on some PoIs but leave many other PoIs completely uncovered. Thus, we should first consider point coverage to ensure more PoIs are covered, and then maximize the aspect coverage of those PoIs.

5.2.3 Photo Coverage

Photo coverage combines point coverage and aspect coverage, and gives point coverage higher priority.

Definition 9 (photo coverage)**.** *Given a PoI x and a photo collection F, photo coverage C_{ph} is defined as*

$$C_{ph}(x, F) = (C_{pt}(x, F), C_{as}(x, F)),$$

where $C_{pt}(x, F)$ and $C_{as}(x, F)$ obey lexicographical order. In other words, $C_{ph1} > C_{ph2}$ if and only if 1) $C_{pt1} > C_{pt2}$ or 2) $C_{pt1} = C_{pt2}$ and $C_{as1} > C_{as2}$.

The above photo coverage is defined for one PoI. For a PoI list $X = \{x_1, x_2, \cdots\}$ and a photo collection F, photo coverage C_{ph} is defined as

$$C_{ph}(X, F) = \left(\sum_{x_i \in X} C_{pt}(x_i, F), \sum_{x_i \in X} C_{as}(x_i, F) \right).$$

In the rest of this chapter, photo coverage is always calculated for the same PoI list, so we omit the notation X and use $C_{ph}(F)$ to denote the photo coverage of F.

5.3 Photo Selection Algorithm

In this section, we first describe the optimization problem considered in this chapter. We identify and address two important design challenges, and then present our photo selection algorithm.

5.3.1 Problem Description

At the beginning of a crowdsourcing event, the command center issues a PoI list describing its interests and a deadline indicating how long the PoI list will be valid.

Before the deadline, photos are selected and delivered to the command center through DTN and other communication links. Specifically, when two nodes (participants) move within the wireless communication distance (Bluetooth or WiFi), they are in contact and photos can be transmitted between them. The command center can be seen as a special node, denoted by n_0, who can receive photos from other nodes when cellular or satellite connections are available.

Before the deadline, the command center has received a collection of photos F_0. Our objective is to maximize its photo coverage, i.e., max $C_{ph}(F_0)$, under the following constraints: 1) limited contact opportunities and limited bandwidth in DTN, 2) limited availability of cellular and satellite connections, and 3) limited storage space of mobile devices.

A centralized solution to the above problem will not work because the contacts among DTN nodes are not known a priori. Therefore, to maximize $C_{ph}(F_0)$ in a distributed way, nodes should optimize their local photo collections by exchanging photos with each other whenever a contact happens. This local optimization is the core of our photo selection algorithm, which will be presented in Section 5.3.4. However, such optimization requires accurate calculation of photo coverage, which poses two new challenges. First, due to coverage overlap, the coverage of a photo depends not only on itself, but also on the presence of other photos. We address this challenge by proposing a metadata management scheme in Section 5.3.2. Second, when a photo is in DTN, its *potential value to the command center* depends not only on its photo coverage, but also on how likely it can be delivered to the command center. We address this challenge by introducing *expected coverage* in Section 5.3.3.

5.3.2 Metadata Management

Due to coverage overlap, the coverage of a photo depends on the presence of other photos. For example, a photo's coverage will decrease if another photo already covers the same PoI. To calculate photo coverage accurately, it is essential to take the presence of other existing photos into account. To this end, nodes should share the metadata of their photos with each other whenever a contact happens. Thus, every node maintains its knowledge about the metadata of every other node. Such metadata is cached and used later for photo coverage calculation. Note that when

the metadata of the command center is shared, it works as an acknowledgment, telling nodes which photo has already been delivered to the command center.

Caching metadata costs very little storage space, but cache validation is a big challenge. After the node whose metadata has been cached leaves after a contact, its photos may change when it meets other nodes. However, these changes may not be known to those who have cached the metadata. Such inconsistency cannot be solved by traditional cache validation schemes due to the low connectivity of DTN. To address this issue, we propose a metadata management scheme as follows.

The inter-contact time T_{ab} between two nodes n_a and n_b has been shown to have exponential decay for many mobility models (e.g., random waypoint and Brownian motion) and real traces [71, 72, 73]. Hence, to capture the contact patterns of nodes (e.g., rescuers in the same team contact more often), we assume T_{ab} to be exponentially distributed with parameter λ_{ab}. Then, the inter-contact time between node n_a and any other node is a random variable $T_a = \min_{n_b \neq n_a} T_{ab}$, so it follows exponential distribution with parameter $\lambda_a = \sum_{n_b \neq n_a} \lambda_{ab}$.

In a contact between n_a and n_b, n_a sends n_b its photo metadata and parameter λ_a learned from historical contacts. Later, when n_b wishes to use n_a's metadata to calculate the coverage of its photos, it computes the time elapsed since their last contact, denoted as t. Then it calculates the probability that n_a has met another node within time t as

$$P\{T_a < t\} = 1 - e^{-\lambda_a t}. \tag{5.1}$$

The outcome $P\{T_a < t\}$ is compared with a predefined threshold P_{thld}. If it exceeds the threshold, the metadata is considered invalid and removed from the cache, because there is a high probability that node n_a has already met another node and therefore has updated its photos. Otherwise, the metadata of n_a is valid and can be used to calculate photo coverage. The value of P_{thld} is currently determined by simulations. It is difficult, if not impossible, to theoretically evaluate its effect, because it is hard to estimate how many photos have been updated during a contact.

5.3.3 Expected Coverage

By our model, the value of a photo can be estimated by its photo coverage. However, for photos in DTN, it is not clear if the photo coverage can be achieved since these photos may never reach the command center. Thus, the potential value of a photo depends not only on its photo coverage, but also on how likely it can be delivered to the command center.

We address this issue by jointly considering the photo coverage and the probability of data delivery from a node to the command center. To this end, we compute *delivery probability* as defined in the PROPHET routing protocol [41]. The delivery probability from node n_a to node n_b is a probabilistic metric of how likely n_a can deliver a packet to n_b. It is computed based on three heuristics: 1) the delivery probability should increase if two nodes encounter each other; 2) the delivery probability should decrease if two nodes do not encounter for a while; 3) if the delivery probability from n_a to n_b is high and that from n_b to n_c is high, then the delivery probability from n_a to n_c is also high (transitive property). The exact calculation of delivery probability can be found in Section III-A of [41]. Here, we use the delivery probability from n_i to the command center n_0, denoted as p_i, as an indicator of how likely n_i's photos can be delivered.

Intuitively, if we multiply a photo's coverage by the probability that it is delivered, the result will be the expectation of the achieved coverage, referred to as the *expected coverage*. Compared to the original photo coverage, the expected coverage is a better estimate of a photo's value, because it takes into account whether or not the photo can actually be delivered. However, the above definition of expected coverage only works for one single photo. It does not consider the possible coverage overlap between this photo and other existing photos. To consider the coverage overlap between photos and thus assess the value of photos more accurately, we extend the above definition of expected coverage to a set of photos, as detailed below.

In a contact between n_a and n_b, the two nodes can assess the value of their photos by calculating expected coverage. To consider coverage overlap between photos, expected coverage is calculated for a node set M that contains all nodes of which n_a and n_b have valid metadata. First, M includes n_a and n_b. Second, M includes the nodes whose metadata is valid according to equation (5.1). Third,

M also includes n_0 because it is important to consider what has been received by the command center. We assume the command center does not drop photos, and thus the metadata of n_0 is always valid. The expected coverage of node set M is defined as follows.

Definition 10 (Expected Coverage)**.** *Given a node set $M = \{n_0, n_1, \cdots, n_{m-1}\}$, let F_i and p_i be the photo collection and the delivery probability of node n_i ($i = 0, 1, \cdots, m-1$), respectively. Also let $b_i \in \{0, 1\}$ denote whether node n_i can deliver its photos to the command center. Then a binary string $B = b_0 b_1 \cdots b_{m-1}$ denotes one possible outcome with regard to whether each node can deliver its photos. The probability of this outcome is*

$$P_B = \prod_{i=0}^{m-1} p_i^{b_i} (1 - p_i)^{(1-b_i)}.$$

In this outcome, the photo coverage obtained by the command center is

$$C_B = C_{ph}\left(\bigcup_{b_i=1} F_i\right).$$

Hence, the expected coverage of M is defined as $C_{ex}(M) = \sum_{B \in \{0,1\}^m} P_B \cdot C_B$.

As an example, with $m = 3$, node set $M = \{n_0, n_a, n_b\}$. Since the command center always reaches itself ($b_0 = 1$), we have four cases, $B = 100, 101, 110, 111$. Therefore,

$$\begin{aligned} C_{ex}(M) \;=\; & C_{ph}(F_0) \cdot (1 - p_a)(1 - p_b) \\ & + C_{ph}(F_0 \cup F_b) \cdot p_b(1 - p_a) \\ & + C_{ph}(F_0 \cup F_a) \cdot p_a(1 - p_b) \\ & + C_{ph}(F_0 \cup F_a \cup F_b) \cdot p_a p_b. \end{aligned} \tag{5.2}$$

From formula (5.2) we can see two advantages of using expected coverage. First, it always considers the photos already received by the command center (F_0). Second, it considers all possible cases about whether the nodes can reach the command center. In this example, there are two nodes (n_a and n_b), and each node may or may not reach the command center. Each of the four possible cases is

considered with a weight indicating its probability of happening.

Via Definition 10 we calculate the expected coverage of the nodes in M. Since our initial purpose is to assess the value of photos in n_a and n_b. we also denote $C_{ex}(M)$ as $C_{ex}(F_a, F_b)$ to emphasize that purpose.

5.3.4 Photo Selection Algorithm

Recall that our objective is to maximize the photo coverage obtained by the command center. To achieve this goal in a distributed way, nodes should maximize the potential value (i.e., expected coverage) of their local photo collections by exchanging photos with each other. Specifically, when two nodes n_a and n_b are in contact, they reallocate their photos to maximize the expected coverage $C_{ex}(F_a, F_b)$. The reallocation is based on a selection pool that contains all the candidate photos, $F_a \cup F_b = \{f_1, f_2, \cdots, f_k\}$. Let s_j be the size of photo f_j. We have the following photo reallocation problem.

$$\begin{aligned} \max \quad & C_{ex}(F_a, F_b), \\ \text{subject to} \quad & \sum_{j=1}^{k} y_j s_j \leq S_a, \sum_{j=1}^{k} z_j s_j \leq S_b, \end{aligned}$$

where $y_j, z_j \in \{0, 1\}$ means whether photo f_j is selected into node n_a or n_b. S_a and S_b is the storage size of n_a and n_b.

The above problem is NP-hard since the standard 0-1 knapsack problem can reduce to it. Also, its objective function $C_{ex}(\cdot)$ is non-convex due to coverage overlap: the coverage of a photo depends not only on itself, but also on other existing photos. This makes the photo reallocation problem even more challenging. Here, we propose a greedy algorithm to solve it.

Without loss of generality we assume $p_a > p_b$, i.e., node n_a has more chances than n_b to deliver photos to the command center. Thus, n_a should have higher priority to select photos. We first fill up the storage of n_a by solving the following problem.

$$\max \quad C_{ex}(F_a, \emptyset),$$

$$\text{subject to} \quad \sum_{j=1}^{k} y_j s_j \leq S_a. \tag{5.3}$$

It is a non-convex optimization problem, so the solution is based on greedy heuristics. In each step of selection, one photo from the selection pool is selected and stored in n_a, such that the current expected coverage $C_{ex}(F_a, \emptyset)$ is greedily maximized. The selection proceeds for the remaining part of the selection pool until n_a's storage is full or no more benefit can be achieved.

Afterward, node n_b selects photos to its storage by solving a similar problem to (5.3). n_b considers what n_a has selected by maximizing $C_{ex}(F_a, F_b)$. Thus, n_b may not choose photos that are similar to (or the same as) those stored in n_a. This happens when n_a can reach the command center with a high probability, i.e., p_a is large. However, n_b uses the same selection pool as n_a, that is, the original $F_a \cup F_b$. It is possible that n_b selects a photo f_j that is already stored in n_a ($y_j = z_j = 1$). This happens when f_j is very useful but n_a cannot deliver it with a high probability.

The above algorithm assumes that the contact duration is long enough to complete the required photo transmission. In some cases, the assumption may not be valid; i.e., the contact duration is too short to transmit all required photos. With such network constraints, the algorithm should be adjusted as follows.

First, the two contacting nodes run the above algorithm to obtain a solution that maximizes the expected coverage. The solution can be different from their current photo collections. Then they transmit photos between each other so that their photo collections gradually become the same as the solution. To do this, the photos in the solution are considered one by one, in the order that they are selected. Specifically, we start by considering the first photo selected to n_a in the solution (given $p_a > p_b$). If the photo is already in n_a's photo collection, no transmission is needed. Otherwise, the photo is currently in n_b and is transmitted to n_a. Then we consider the second photo selected to n_a and check whether a transmission is needed. Similarly, all photos selected to n_a are considered and n_a's photo collection finally becomes the same as the solution. After that, the same operation proceeds for n_b. The contact may end at any time during this process, and any unfinished transmission will be discarded.

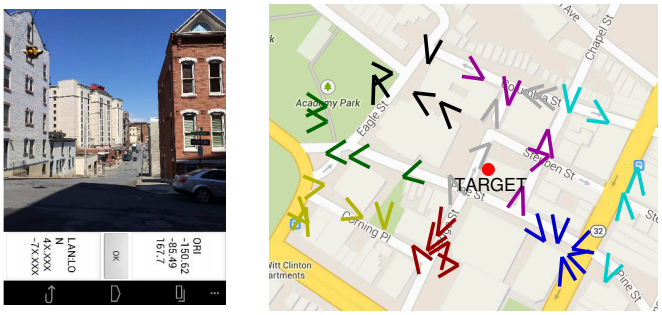


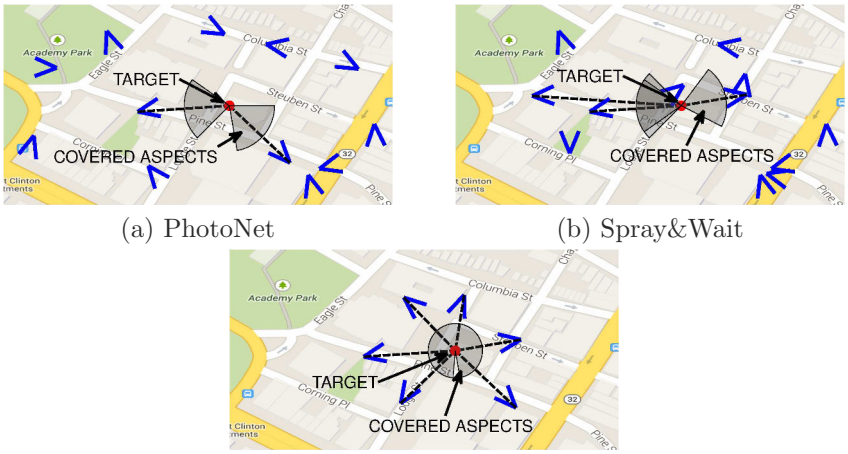
Figure 5.2. (a) Screenshot of the programmed phone. (b) We take 40 photos and assign them to 8 nodes in the trace. The locations and orientations of the photos are shown in V shapes, where the photos assigned to different nodes have different colors.

5.4 Prototype Implementation

We have implemented a proof-of-concept prototype to test whether our photo selection algorithm works for real photos. It is based on a Google Nexus 4 running Android 4.2, and it enables automatic metadata acquisition when a photo is taken (see Section 3.6.1 and our paper [74] for details). We apply the proposed photo selection algorithm to real photos to see the selection results. To run the selection algorithm, we extract the contact information among 9 nodes from an existing DTN trace, *MIT Reality*. The MIT trace records the contacts among users carrying handheld Bluetooth devices. The devices periodically detect their peers nearby, and a contact is recorded when two devices move into the transmission range of each other. Among the 9 nodes, 8 of them represent crowdsourcing participants, and the other one represents the command center. It can be considered as a rescuer carrying a satellite radio or a data mule that periodically moves back and forth between the area and the command center. We use the last 48 contacts among the 9 nodes to run the algorithm and collect photos, and use all previous contacts to learn the delivery probability of nodes.

Furthermore, we take 40 photos using the programmed phone and assign five of them to each crowdsourcing participant in the trace, as if the node took these photos for the crowdsourcing task. The locations and orientations of the photos are shown in Figure 5.2(b) in V shapes, where the photos assigned to different nodes are in different colors. The target (PoI) is a historic church near the center of the area.

Based on the photo metadata and the contact trace, we simulate the photo



(a) PhotoNet (b) Spray&Wait

(c) Our scheme

Figure 5.3. Photos delivered and aspects covered.

crowdsourcing process using three different algorithms: our scheme, PhotoNet, Spray&Wait. PhotoNet [47] is a picture delivery service that prioritizes the transmission of photos by considering location, time stamp, and color difference, with the goal of maximizing the "diversity" of the photos. Spray&Wait [40] is a classic DTN routing protocol that balances resource utilization and delivery ratio. It is used to represent a set of general-purpose DTN routing protocols that do not differentiate packets by their content.

In real crowdsourcing applications, a participant may possess hundreds of photos while the peer-to-peer communication bandwidth is at 1MB/s level and the device storage is at 1GB level. In our experiments a participant has much fewer (five) photos, so the bandwidth and storage constraints must be scaled down accordingly. We limit the number of photos that can be transferred in a contact to be three and the number of photos that can be stored in a device to be five.

Figure 5.3 shows the photos delivered to the command center at the end of the trace. For photos covering the target, dashed lines show their viewing directions and gray areas show the covered aspects (effective angle $\theta = 40°$). Both PhotoNet and Spray&Wait deliver 12 photos because there are four contacts between the participants and the command center, and three photos are transferred in each contact. PhotoNet delivers photos that are diverse in terms of location, time and color histogram, and thus the delivered photos spread across the area. However, it does not have much information about the target, with only two photos covering

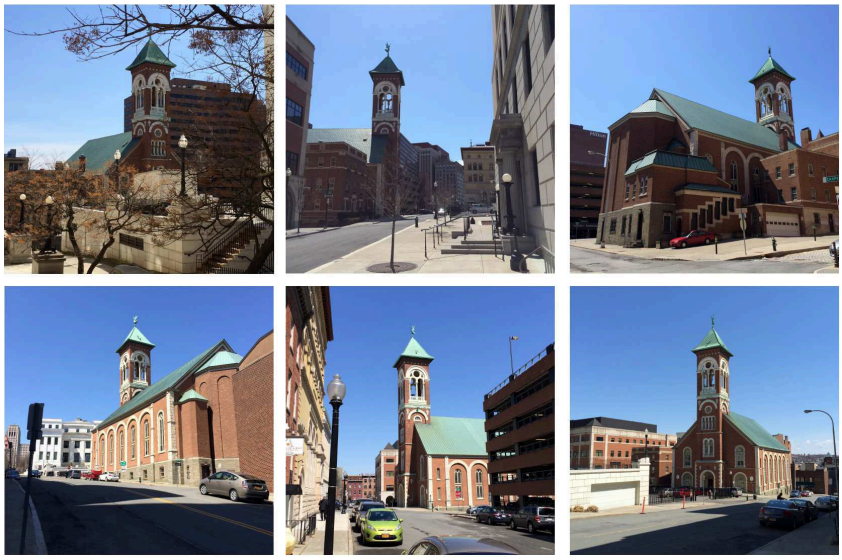


Figure 5.4. Images delivered by our scheme.

160°. Spray&Wait does not consider photos by their content, and the results are not good either. Three of the 12 photos cover 171° of the target. Among the three schemes, ours has the best performance. Although 12 photos can be delivered, six of them are not helpful for increasing the photo coverage. Thus, our scheme only delivers the six most useful photos, which cover 346° of the target.

Figure 5.4 shows the real images delivered by our scheme. As can be seen, the photos cover the church from different angles, showing a quite complete view of the church. Having complete information about the target is important in a disaster recovery scenario, where the command center needs to identify any damage related to the building and determine whether it is safe for survivors. The images delivered by the other two algorithms are not shown due to space limitation. From the photo metadata and the aspect coverage shown in Figure 5.3, it is clear that they are not as good as those delivered by our scheme.

5.5 Trace-Driven Simulations

Real photo crowdsourcing applications have a much larger scale than our demonstration. In this section, we use trace-driven simulations to further evaluate the performance of our photo selection algorithm.

Table 5.1. Simulation Settings

Parameter	Notation	Value*
photo size		4MB
effective angle	θ	30°
orientation	$\overrightarrow{d}$	$[0^\circ, 360^\circ)$
field-of-view	ϕ	$[30^\circ, 60^\circ]$
coverage range	r	$[50, 100]\cot(\phi/2)$m
valid threshold	P_{thld}	0.8
PROPHET	P_{init}, β, γ	0.75,0.25,0.98
# of nodes		97/54
simulation time		300/200 hr

*A range means that the value is randomly generated in that range; two numbers with "/" means the value in the MIT/Cambridge06 trace.

5.5.1 Simulation Setup

We assume participants move in a 6300m×6300m square region, like a town or the central area of a city. The command center issues a list of 250 PoIs randomly located in the region.

The contact information between participants is retrieved from two traces: *MIT Reality* and *Cambridge06* [75]. Similar to the MIT trace, Cambridge06 records the contacts among users carrying handheld Bluetooth devices. The devices periodically scan their peers nearby and record a contact if a peer is discovered. The scan interval for the MIT trace is five minutes, and the scan interval for the Cambridge06 trace is two minutes.

Participants may be able to communicate with the command center through various ways such as DTN links, satellite radios, etc. To simulate such links, we randomly pick about 2% of the total participants and assume they can communicate with the command center. They can be rescuers in a disaster area or commanders in a battlefield who have satellite radios, or they can be data mules that move back and forth between the interested area and the command center.

Photos are randomly generated by the participants. Photo metadata and other simulation settings are shown in Table 5.1.

5.5.2 Comparing With Other Schemes

In this subsection, we evaluate the performance of our scheme by comparing it with other solutions. The evaluation is based on the MIT trace, and the performance is measured in terms of the point coverage and aspect coverage obtained by the command center (normalized by the number of PoIs). Each data point is the average of 50 simulation runs.

The proposed photo selection algorithm (called our scheme) is compared with the following schemes.

- **BestPossible:** There is no storage or bandwidth constraint for this scheme. The only constraint is contact opportunity. Nodes will try to replicate every useful photo to everyone, and achieve the best possible coverage.
- **NoMetadata:** Disable the metadata caching and metadata management component from our scheme.
- **Spray&Wait:** The binary spray and wait protocol with four allowed copies, as described in [40].
- **ModifiedSpray:** It is based on Spray&Wait. When a node transmits photos to another node, it transmits the photo with the most photo coverage first. When a node receives a photo and its storage is full, it first removes the photo with the least photo coverage.

Fig. 5.5 shows the results, where the storage size is 0.6GB and the number of created photos is 250 per hour. As time goes, more photos are delivered and thus the coverage increases. Our scheme covers 70% of PoIs after 150 hours of crowdsourcing. This time would be significantly shorter in real world applications because there could be much more participants than those in the DTN traces.

Comparing the five schemes, BestPossible provides the performance upper bound, and our scheme performs close to it with a maximum of 10% less point coverage and 17% less aspect coverage. NoMetadata performs worse than our scheme, which shows that metadata caching and metadata management can improve performance even though the inter-contact time used in metadata management does not strictly follow exponential distribution in real traces. Spray&Wait has the

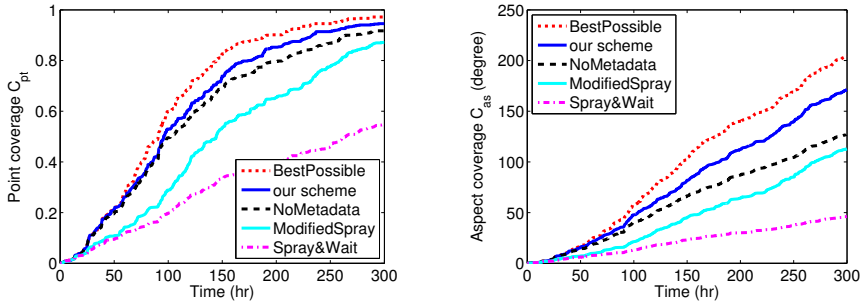


Figure 5.5. Comparing our scheme with other four schemes.

worst performance because it is designed for routing general data and does not consider which photo is more useful. At 150 hours, it has 49% less point coverage and 69% less aspect coverage than our scheme. ModifiedSpray outperforms Spray&Wait because it considers the coverage of photos. However, like previous utility-based routing algorithms, ModifiedSpray prioritizes photos by their individual coverage without considering the overlap (redundancy) between photos. This is why it still underperforms our scheme. At 150 hours, ModifiedSpray has 26% less point coverage and 38% less aspect coverage than our scheme.

5.5.3 The Effects of Short Contact Duration

We usually assume the contact duration is long enough to do the required photo transmission. In this subsection, we evaluate how our scheme performs under short contact duration. The transmission bandwidth is set as 2MB/s, which can be achieved by Bluetooth 3.0+hs, WiFi ad hoc mode, and WiFi Direct. Thus, a 10 minute contact results in 1.2GB transmission capacity, enough for the transmission between two nodes with 0.6GB storage.

The results are shown in Fig. 5.6. Since 10 minutes means no limit on the contact duration, its performance is the same as before. When the contact duration is reduced to 2 minutes, the performance only decreases by about 1%. This is because our scheme prioritizes the transmission of important photos, and thus those photos are still delivered to the command center even though the contact duration reduces by 80%. Nonetheless, the performance drops faster when the contact duration is further reduced and becomes insufficient for transmitting important photos. In the 30 seconds case, only 5% of the photos can be transmitted,

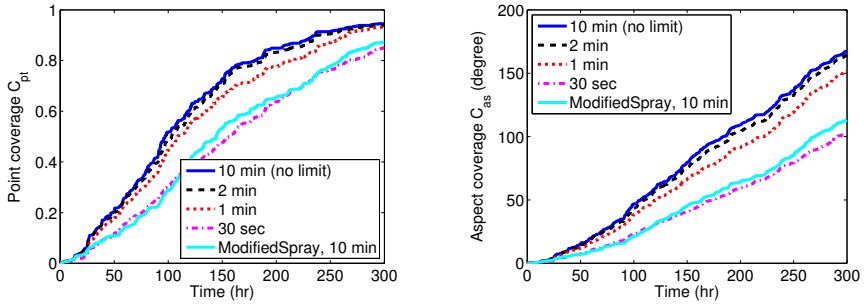


Figure 5.6. The effects of short contact duration.

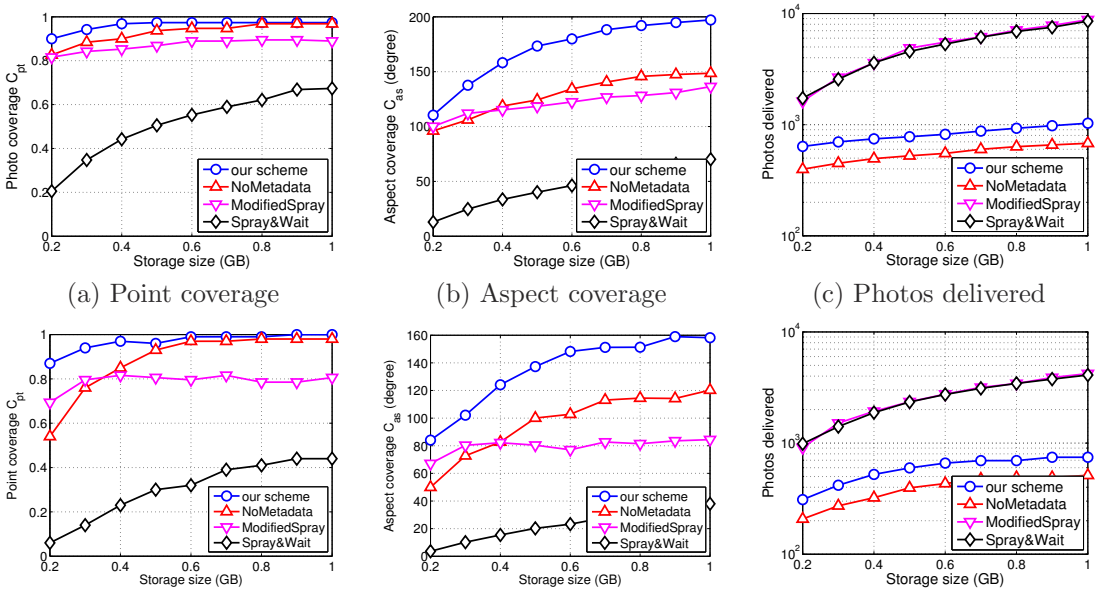
and the performance degrades to a similar level of ModifiedSpray with 10 minutes duration. In sum, our scheme maintains good performance unless the contact duration is drastically reduced (e.g., by 95%), in which case the performance is still comparable to ModifiedSpray with 10 minutes duration.

5.5.4 The Effects of Storage Capacity

In this subsection, we evaluate the effects of storage capacity. At the end of a simulation run, the point coverage and aspect coverage obtained by the command center are recorded (normalized by the number of PoIs). We also record the number of photos delivered to the command center.

Fig. 5.7 summarizes the results, where the number of generated photos is 250 per hour. Fig. 5.7(a)(b)(c) is based on the MIT trace and Fig. 5.7(d)(e)(f) is based on the Cambridge06 trace. From Fig. 5.7(a)(b)(d)(e), we can see that increasing storage capacity generally improves photo coverage since more photos can be delivered to the command center. For our scheme and NoMetadata, the performance is improved mainly because with larger storage space, a useful photo has more copies stored in different nodes, and thus it is more likely to be delivered. However, ModifiedSpray is not affected too much by storage space since the number of copies for a photo is limited to four.

Fig. 5.7(c)(f) shows the number of delivered photos in logarithmic scale. In our scheme and NoMetadata, photos are transferred to the command center only if they can contribute to the photo coverage. Thus, the number of delivered photos in our scheme and NoMetadata is dramatically less than that in ModifiedSpray and Spray&Wait.



(a) Point coverage (b) Aspect coverage (c) Photos delivered

(d) Point coverage (e) Aspect coverage (f) Photos delivered

Figure 5.7. The effects of storage capacity. (a)(b)(c) are based on the MIT trace and (d)(e)(f) are based on the Cambridge06 trace.

5.5.5 The Effects of the Number of Generated Photos

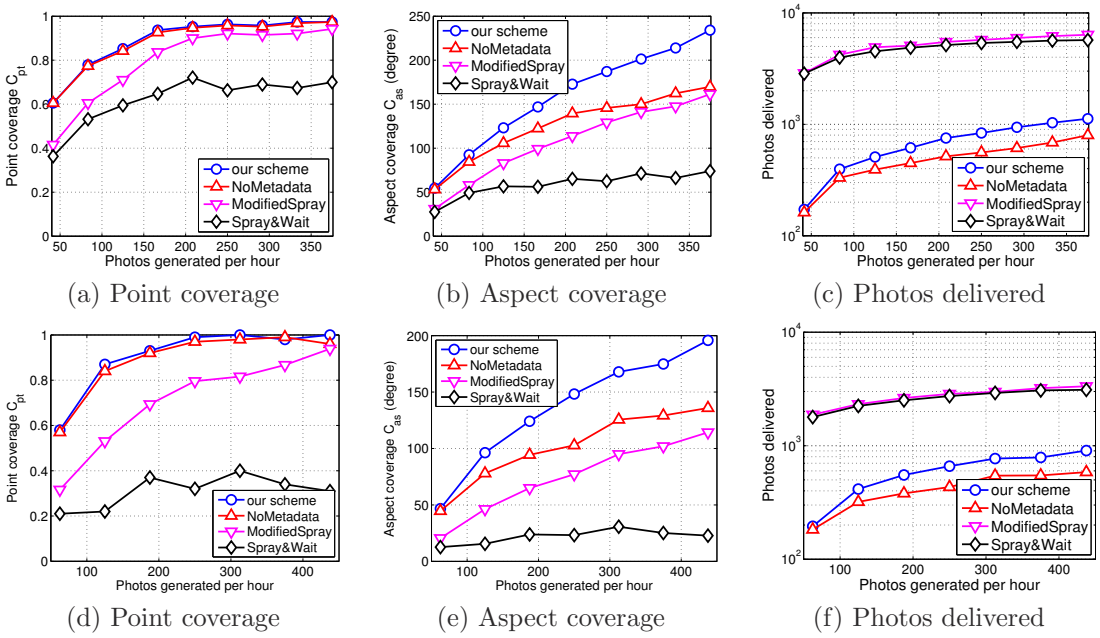
The number of generated photos has two opposite effects: 1) network and storage contention is exacerbated with more generated photos; and 2) photo coverage can be increased if more useful photos are generated and received by the command center.

Fig. 5.8 summarizes the results, where the storage size is fixed at 0.6GB. As shown in Fig. 5.8(a)(b)(d)(e), our scheme can significantly improve the point coverage and aspect coverage when more photos are generated. This is because our scheme is able to select an increasing amount of useful photos from an increasing amount of candidate photos, and thus the good effect (more candidate photos) overcomes the bad effect (more contention). Similarly, NoMetadata and ModifiedSpray follow the same trend because they consider photo coverage and thus can select useful photos from candidate photos. Spray&Wait fluctuates because it cannot select useful photos from candidate photos, hence suffering from the exacerbated resource contention.

Fig. 5.8(c)(f) shows the number of delivered photos in logarithmic scale. Again, our scheme and NoMetadata deliver much fewer photos than the other two schemes.

However, these photos achieve good coverage and contain little redundancy. Consider the case where 250 photos are generated per hour in Fig. 5.8(c). Our scheme delivers around 800 photos to the command center. Since there are 250 PoIs in total, on average each PoI is covered by 3.2 photos. With 30° effective angle (Table 5.1), the 3.2 photos would cover $3.2 \times 2 \times 30° = 192°$ if they do not overlap with each other at all. In fact, from Fig. 5.8(b), we can see that the actual aspect coverage on each PoI is about 180°, which shows that the redundancy between the 3.2 photos is only 12°.

To summarize the simulation results, our scheme performs close to the best possible results where no bandwidth or storage constraint is enforced. Our scheme significantly outperforms Spray&Wait, which represents a set of general-purpose routing protocols that do not consider packet utility in routing. Our scheme also outperforms ModifiedSpray, which is similar to previous utility-based approaches in that they prioritize packets by their individual utility, without considering the overlap (redundancy) between photos.



(a) Point coverage (b) Aspect coverage (c) Photos delivered

(d) Point coverage (e) Aspect coverage (f) Photos delivered

Figure 5.8. The effects of the number of generated photos. (a)(b)(c) are based on the MIT trace and (d)(e)(f) are based on the Cambridge06 trace.

Chapter 6

A Metadata-Enhanced Crowdsourcing System for Mobile Videos

6.1 Background

Video technology has improved dramatically in the past decade. Large, heavy digital cameras and camcorders have been replaced with smaller, lighter smartphones capable of shooting Ultra High Definition (UHD) videos. In the near future, a variety of wearable devices such as smartglasses, smartwatches, and smart jewelry will make video recording more user-friendly. Taking a video may only require users to press a button, rather than taking their smartphones out of pockets and performing multiple touch screen actions. Through smartphones or wearable devices, the convenience of video recording is expected to significantly increase the number of mobile videos recorded and shared over the Internet.

The videos, if managed properly, could enable many useful and critical applications. For example, terrorist attacks and gun violence have been increasingly common around the world. In such incidents, authorities often identify the suspects from surveillance videos. If all the mobile videos taken around the time and location of the incidents were collected, they could have served as additional resources for identifying and locating the suspects, especially in places not covered

by security cameras. In fact, officers investigating 2016 Brussels bombings were unable to trace a suspect along his escape route due to the absence of security cameras, and they had to ask people who inadvertently filmed or photographed the suspect to contact the police [76]. Besides crime investigation, mobile videos can be used in many other applications, such as locating a missing child or investigating traffic accidents.

To enable the above applications, one possible approach is to ask mobile users to upload all videos to the cloud or cloudlets, and then use content-based techniques to enable user search [50]. This approach, however, is extremely resource hungry. Uploading all videos consumes lots of bandwidth and energy on mobile devices, while storing and processing videos consumes a large amount of disk space and processing power on the server side. To make things worse, only a small part of the uploaded videos may be useful to the applications. Thus, the resources spent on uploading, storing, and processing all other videos are wasted.

Due to the aforementioned issues, a better approach is to let mobile devices upload high level video descriptions initially, and only ask for actual video files in response to user queries. However, existing geo-tagging features on mobile devices only tag videos with coarse location information, which is not sufficient to reveal the view of the camera. Even at the same location, cameras facing different directions will have different views. Besides location, existing systems (e.g., YouTube) rely heavily on user-generated annotations. Tagging each video manually is not convenient and discourages public participation. More importantly, these annotations may be inaccurate or miss important information. For example, when a tourist inadvertently films a suspect, the video may be tagged as travel related, nothing to do with the suspect. Thus, it is difficult for applications to find the videos they need simply based on annotations.

To address these problems, we design and implement *VideoMec*, a system that organizes and indexes all videos taken by mobile devices based on the idea of metadata, which supports efficient queries for applications to find and fetch the requested videos. VideoMec consists of two components, a VideoMec server and a VideoMec app (see Fig. 6.1). When a video is recorded using the VideoMec app on a mobile device, its metadata is automatically generated and uploaded to the VideoMec server. Since metadata is very small, the uploading does not consume

much bandwidth and energy. In the VideoMec server, the uploaded metadata is organized and indexed in a database. Then, the application can query the database to find the desired videos, by providing information such as when and where the event happened, and from which angle or distance the event should be captured. Thus, when a crime happens at certain time and location, every video taken around that time and location can be found and used to search for the suspect, regardless of whether mobile users are aware of the suspect or not.

If the video description also includes user-generated annotations, the application can provide extra keywords in the query, and the returned videos should either match the keywords or satisfy the metadata requirements. After processing the query, the VideoMec server returns the matching videos if they have been uploaded; otherwise, the server notifies the corresponding mobile devices, which automatically upload the videos and get paid.

When uploading videos, it is possible that mobile devices cannot upload all requested videos in time due to limited wireless bandwidth and large video files. For example, the latest smartphones can shoot UHD videos at about 40 Mbps bitrate, and then uploading one-minute UHD video would take 40 minutes for a 1 Mbps cellular connection. Since some applications are time-sensitive, requiring videos to be available within a short time, it is important to decide which videos are more valuable to the application based on the metadata, and then upload them first. To achieve this goal, we design an *upload decision engine* that decides which videos (possibly a short clip in a long video) should be uploaded given bandwidth and time constraints. The decision engine considers two typical application scenarios, one prioritizing videos that provide good time coverage, and the other prioritizing videos at the right locations.

In this chapter, we present VideoMec, a metadata-enhanced crowdsourcing system for mobile videos. By using the uploaded metadata to serve queries and only uploading queried video files, the proposed crowdsourcing system is capable of handling a very large number of mobile videos while reducing the bandwidth and energy consumption of mobile devices. VideoMec supports comprehensive queries including time, location, angle, and distance information, for applications to find desired videos quickly and accurately from the distributed video repository. Moreover, VideoMec's upload decision engine selects the most valuable videos to

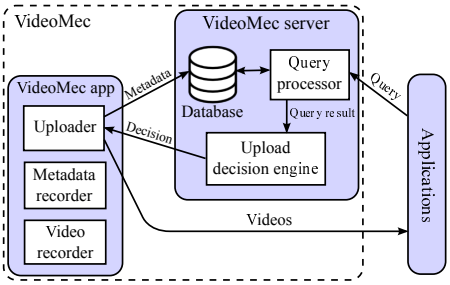


Figure 6.1. The VideoMec system.

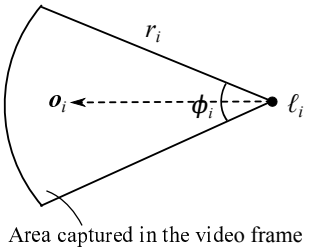


Figure 6.2. Metadata at timestamp t_i.

upload when bandwidth and time resources are limited. Finally, we have implemented a prototype of VideoMec, evaluated its performance and demonstrated its effectiveness based on real experiments.

6.2 Metadata and Its Storage

6.2.1 Metadata

A mobile video can be characterized by its metadata; i.e., when, where and how the video is taken. The metadata includes the start and end time of a video, denoted as t_s and t_e, respectively. Between t_s and t_e, we define a series of timestamps $t_0, t_1, t_2, \ldots$ such that $t_0 = t_s$ and $t_{i+1} - t_i = 1$ second for all $i = 0, 1, 2, \cdots$. Then each timestamp t_i is associated with four parameters $(\ell_i, \boldsymbol{o}_i, \phi_i, r_i)$ which jointly determine an area on the Earth's surface that is viewable in the current video frame.

As shown in Fig. 6.2, location ℓ_i (also referred to as *video location*) is the geographic coordinates of the camera, including longitude x_i and latitude y_i. Orientation $\boldsymbol{o}_i$ is the direction to which the camera is facing. It is a vector coming from the camera aperture and perpendicular to the image plane. Field of view ϕ_i specifies how wide the camera can see. Objects outside the field of view will not appear in the video. Coverage range r_i specifies how far the camera can see. It is the distance beyond which objects are no longer clearly recognizable in the video.

Besides the above metadata, VideoMec uses frame rate and resolution available in the video header as video quality metrics. It also uses file size as a cost metric, since a larger video requires more network resources to upload.

6.2.2 Metadata Storage

When a mobile video is recorded, its metadata is automatically generated and uploaded to the VideoMec server. The uploaded metadata can be stored in a relational database for future queries. However, as mobile videos continue to grow exponentially, there should be efficient ways to index and query metadata. We propose to use *R*-tree* [77] to index the metadata. R*-tree is a tree-based data structure designed for efficient indexing of spatial information such as geographical coordinates. It has been commonly used in geographical information systems (GIS) to store locations of objects such as gas stations or shapes of objects such as roads, and then find answers quickly to queries such as "find the nearest gas station" or "find all road segments within 5 miles of my location". Although R*-tree has been used in GIS, it has never been applied to video query and selection, and video metadata cannot be directly stored in R*-tree.

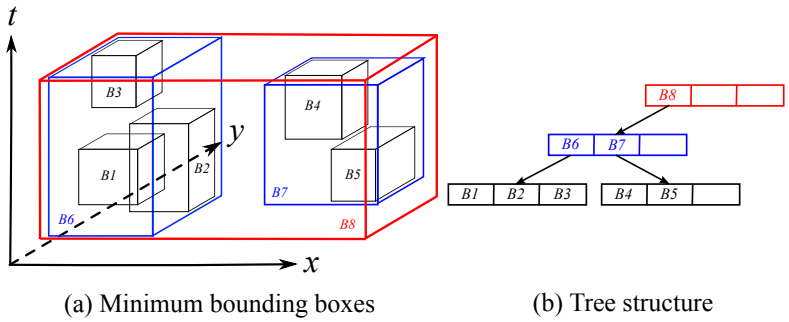
To use R*-tree, we convert the raw metadata of a video to the *minimum bounding box* of the video in a 3D space, which can then be stored in R*-tree for efficient query and selection. More specifically, consider the 3D space formed by the two dimensional plane of the Earth's surface[1] (called x-y plane) and the third dimension of time (called t-axis). Each timestamp t_i and its corresponding location $\ell_i = (x_i, y_i)$ maps to a point (t_i, x_i, y_i) in the t-x-y space. As time goes from t_s to t_e, point (t_i, x_i, y_i) moves in the t-x-y space and forms a trace. The minimum bounding box of the trace is defined as $(t_{min}, t_{max}, x_{min}, x_{max}, y_{min}, y_{max})$, where

$$
\begin{aligned}
t_{min} &= t_s, & t_{max} &= t_e,\\
x_{min} &= \min_i x_i, & x_{max} &= \max_i x_i,\\
y_{min} &= \min_i y_i, & y_{max} &= \max_i y_i.
\end{aligned}
$$

This minimum bounding box represents the spatiotemporal boundary of the video, and it is stored in R*-tree.

Fig. 6.3 shows an example of R*-tree containing five videos. The left figure depicts the minimum bounding box of each video as B_1 through B_5. The key idea of R*-tree is to group nearby objects and represent them with their minimum

[1]The Earth's surface is not flat. Here we use a plane instead of a sphere for the convenience of illustration.



(a) Minimum bounding boxes (b) Tree structure

Figure 6.3. An example of a R*-tree containing five videos.

bounding box in the next higher level of the tree. In this example, the left three boxes, B_1, B_2, B_3, are close to each other. Hence, they are grouped together and their minimum bounding box, B_6, is generated. Similarly, the right two boxes B_4 and B_5 are grouped and their minimum bounding box B_7 is generated. At a higher level, B_6 and B_7 are further grouped and their minimum bounding box, B_8, is generated.

The right figure shows the structure of the corresponding R*-tree. The leaf nodes store the minimum bounding box of each video, and the nodes at higher levels store larger bounding boxes which aggregate multiple videos. Based on this tree structure, consider a query to "find all videos whose minimum bounding boxes intersect with a given box", where two boxes intersect with each other if they contain at least one common point. If the given box does not intersect with a larger bounding box, e.g., B_7, it also cannot intersect with any of the contained boxes, i.e., B_4 or B_5. Therefore, the entire subtree rooted at B_7 can be skipped in the search. As R*-tree has mechanisms to maintain its balance after insertion or deletion, it guarantees a search time of $O(\log n)$ for the above query, where n is the total number of videos in the R*-tree.

6.3 Query Design and Processing

In VideoMec, an application can query the server to retrieve videos of interest. In this section, we present the supported queries and describe how those queries are processed.

6.3.1 Query Design

An application typically searches for videos related to a particular event. For example, a police officer may search for videos related to a traffic accident and seek evidence to determine the responsible party. Therefore, the approximate time duration and location of the queried event should be included in the query. The location of the queried event is denoted as ℓ_e. The query may also contain angle, distance, or quality requirements as follows. The query may include one or more angles $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots$ from which the event should be captured, in order to avoid obstructions (e.g., buildings and trees) blocking the line of sight. The query can also specify the distance d between video location and event location, based on how large the interested objects would appear in the video. Since the required angle or distance may not be met completely, a deviation value a_{dev} or d_{dev} is given to indicate how much deviation is allowed from the required value. Finally, a minimum acceptable frame rate or resolution can be provided to filter out videos of low quality. Fig. 6.4(a) shows an example of query with all the requirements specified.

6.3.2 Query Processing

Queries are processed using the *filter-refine paradigm*. The filter step leverages R*-tree to quickly filter out most videos that are irrelevant to the query. Then the refinement step checks the metadata of each remaining video to determine whether it matches the query or not. Details of the two steps are elaborated below.

6.3.2.1 Filter Step

We derive the necessary conditions that a video must satisfy in order to match the query. The conditions can be easily checked using R*-tree, and thus videos not satisfying these conditions can be quickly found and filtered out.

Consider the query shown in Fig. 6.4(a). We redraw the query in Fig. 6.4(b) and consider a video frame located at ℓ_i. For the video frame to capture the event, the distance between the video location ℓ_i and the event location ℓ_e must be within the coverage range of the camera, i.e., $\left\|\overrightarrow{\ell_i \ell_e}\right\| \leq r_i$. However, this condition cannot be efficiently checked using R*-tree because R*-tree only contains information about

video locations while this condition also involves the coverage range. Thus, we replace r_i with a predefined constant R representing the upper bound of all r_i and obtain a necessary condition that can be efficiently checked using R*-tree:

$$\left\|\overrightarrow{\ell_i\ell_e}\right\| \leq R. \tag{6.1}$$

When a required distance d and an allowed deviation d_{dev} are given, the distance between the video location and the event location should also satisfy

$$\left|\left\|\overrightarrow{\ell_i\ell_e}\right\| - d\right| \leq d_{dev}. \tag{6.2}$$

If there are required angles $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots$ and an allowed deviation a_{dev}, the video frame should capture the event from an angle close to one of the required angles:

$$\min_j \angle\left(\overrightarrow{\ell_e\ell_i}, \boldsymbol{a}_j\right) \leq a_{dev}. \tag{6.3}$$

For the video frame to match the query, its location ℓ_i must satisfy inequalities (6.1), (6.2), and (6.3) simultaneously[2]. On the x-y plane, the set of all video locations that satisfy the three inequalities form a closed area, as the colored annular sector shown in Fig. 6.4(b). To simplify the shape of that area, we find its minimum bounding rectangle shown as the dashed rectangle in Fig. 6.4(b). The minimum bounding rectangle and the time duration of the event together define a minimum bounding box in the t-x-y space, referred to as *query box*. For a given query, VideoMec derives its query box and uses R*-tree to find all videos whose minimum bounding boxes intersect with the query box. Then only the returned videos may match the query, and they will be further evaluated in the refinement step.

6.3.2.2 Refinement Step

In this step, the metadata of the videos returned from the filter step is further checked to generate the final query result.

First, if there are any video quality requirements in the query such as the

[2]Depending on if there is any specified distance or angle, inequality (6.2) or (6.3) may or may not exist.

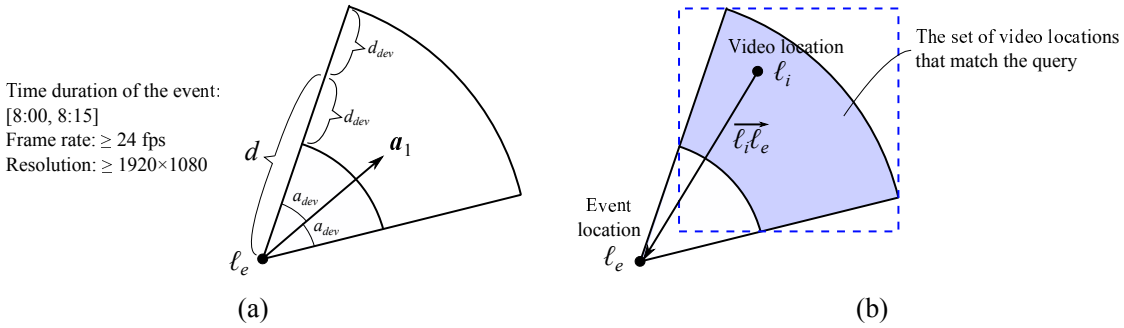


Figure 6.4. An example query. (a) All the requirements specified in the query. Note that there is only one required angle. (b) The set of video locations matching the query. For simplicity, assume that $d + d_{dev} \leq R$.

minimum frame rate or the minimum resolution, videos not satisfying the quality requirements will be discarded. Then for each remaining video, since the metadata can be different for different timestamps, it is necessary to check the metadata at each timestamp to see which part of the video matches the query and which part does not. Specifically, we check whether the following four conditions hold.

1. Timestamp t_i is within the time duration of the queried event.
2. The distance between the video location and the event location is within the coverage range of the camera, i.e., $\left\|\overrightarrow{\ell_i\ell_e}\right\| \leq r_i$.
3. Orientation $\boldsymbol{o}_i$ is appropriate so that the event location is within the field of view of the camera, i.e., $\angle\left(\overrightarrow{\ell_i\ell_e}, \boldsymbol{o}_i\right) \leq \phi_i/2$.
4. If distance or angle is specified in the query, inequalities (6.2) or (6.3) should be satisfied.

A video frame matches the query if all four conditions hold, which means that the queried event is captured in the video frame (2nd and 3rd conditions) at the required time (1st condition) and from the required angle and distance (4th condition). When multiple video frames from the same video with consecutive timestamps all match the query, the video clip containing these frames will be returned as the output. For example, if the frames at timestamps t_2, t_3, t_4 match the query but those at t_0, t_1 and $t_5, t_6, \ldots$ do not, the video clip $[t_2, t_4]$ will be returned as an output. After all videos obtained from the filter step are checked, a set of video clips is returned as the final query result.

6.4 Upload Decision Engine

Videos returned from query processing should be uploaded by mobile devices. When uploading videos, it is possible that mobile devices cannot upload all requested videos in time due to limited wireless bandwidth and large video files. The upload decision engine decides which videos (possibly a short clip in a long video) should be uploaded given bandwidth and time constraints. Based on two typical application scenarios, we propose two different strategies.

6.4.1 Time-Based Strategy

The Time-Based Strategy (TBS) is motivated by applications which require uploaded videos to cover a certain period of time. Since multiple devices can take videos at the same time, uploading a video whose time interval is already covered by other videos does not increase the total time coverage. Thus, it is a challenge to select videos with the most time coverage under bandwidth and time constraints.

6.4.1.1 TBS Problem

Given a set of videos, we consider the problem of selecting a subset of them such that every mobile device can upload its selected videos within a time limit and the total time coverage of the selected videos is maximized. When selecting videos, it is sometimes necessary to cut a long video into smaller pieces and only upload part of the video due to time constraints. Hence, we divide a video into *segments* of fixed length L, and decide whether each segment should be selected. Having smaller L increases the flexibility of selecting "good" segments from different videos and different devices, thus improving the overall time coverage. However, if L is too small (e.g., one second), the quality of experience suffers since the selected videos contain many small segments and may switch scenes too frequently. The choice of L will be evaluated in Section 6.5. Now, we formally define the problem of selecting video segments as follows.

Definition 11 (TBS Problem)**.** *Consider a set of devices* $D_1, D_2, \ldots, D_d$*, where each device* D_i *has an average bandwidth* B_i *and a set of video segments* $V_{i,1}, V_{i,2}, \ldots, V_{i,n_i}$*. Each segment* $V_{i,j}$ *has a start time* $t_s(V_{i,j})$*, end time* $t_e(V_{i,j})$*, and file size*

$s(V_{i,j})$. Given an upload time limit T, the TBS problem is to select a subset of segments $\mathcal{V} \subseteq \{V_{i,j} \mid (i = 1, \ldots, d) \wedge (j = 1, \ldots, n_i)\}$ such that every device can upload its segments in $\mathcal{V}$ within the time limit, i.e., $\sum_{V_{i,j} \in \mathcal{V}} s(V_{i,j}) \leq B_i T$ for every $i = 1, \ldots, d$, and the total length of $\bigcup_{V_{i,j} \in \mathcal{V}} [t_s(V_{i,j}), t_e(V_{i,j})]$ is maximized.

Theorem 6. *The TBS problem is NP-hard.*

Proof. We reduce a known NP-hard problem, the 0-1 knapsack problem, to the TBS problem. In the 0-1 knapsack problem, there are a set of items, each with a value and weight. Given a knapsack with a weight limit, the problem is to find a subset of items such that their total weight is no more than the weight limit and their total value is maximized.

For any instance of the 0-1 knapsack problem, we can construct an instance of the TBS problem as follows. We construct a mobile device D_1, where the network bandwidth limit $B_1 T$ is set to the weight limit of the knapsack, and the number of video segments n_1 is set to the number of items. For the j-th segment $V_{1,j}$, its length $t_e(V_{1,j}) - t_s(V_{1,j})$ is set to the value of the j-th item, and its file size $s(V_{1,j})$ is set to the weight of the j-th item. Also, we ensure that the segments do not overlap in time because a device cannot record two videos simultaneously.

A solution $\mathcal{V}$ to this instance of the TBS problem maximizes the total length of $\bigcup_{V_{1,j} \in \mathcal{V}} [t_s(V_{1,j}), t_e(V_{1,j})]$. Since the segments do not overlap, it actually maximizes $\sum_{V_{1,j} \in \mathcal{V}} [t_e(V_{1,j}) - t_s(V_{1,j})]$. When the segments are seen as items, $\mathcal{V}$ is a subset of items which satisfies the weight constraint and maximizes the total value of items. Thus, $\mathcal{V}$ is also a solution to the 0-1 knapsack problem. This completes the reduction and hence the proof. □

The proof above shows that the TBS problem is at least as hard as the knapsack problem. Unfortunately, the TBS problem is much harder than the knapsack problem, because its objective function is *non-additive*; i.e., selecting a video segment may not increase the total time coverage. The amount of increase depends on how much the segment overlaps with others.

6.4.1.2 TBS Algorithm

Due to the hardness of the problem, we propose an algorithm based on several heuristics. First, let $\Delta D_i = \max\{0, \sum_{j=1,\ldots,n_i} s(V_{i,j}) - B_i T\}$ denote how much

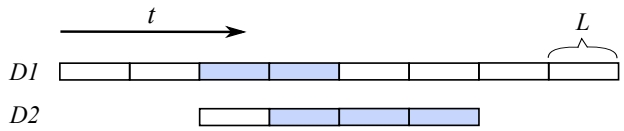


Figure 6.5. Suppose D_1 can upload 2 of its 8 segments, and D_2 can upload 3 of its 4 segments, so $\Delta D_1 > \Delta D_2$. If D_2 selects first, no matter what it selects, D_1 can avoid overlap and achieve a total time coverage of $5L$. If D_1 selects first, it may select two segments in the middle (shown in color). Then D_2 cannot avoid overlap and the total time coverage is at most $4L$.

video segments in device D_i cannot be uploaded due to bandwidth constraints. If $\Delta D_i = 0$, all D_i's segments can be uploaded. Then, segments in other devices which have time overlap with D_i's segments need to be updated. That is, if a segment partially or fully overlaps with D_i's segments, the overlapping part should be discarded since it is no longer useful. By doing so, the video segments in other devices becomes shorter, and thus for those devices, ΔD_i should be checked to see if it is now equal to zero. This process, referred to as *constraint checking procedure*, identifies devices for which no selection is needed.

After the constraint checking procedure, the remaining devices have $\Delta D_i > 0$, and we need to decide which video segments to upload for each device. We start from the device with the smallest ΔD_i. The reason is that the device with smaller ΔD_i throws away fewer segments during selection and thus is less likely to make mistakes (i.e., select segments that overlap with others). This heuristic is illustrated in Fig. 6.5.

For the device with the smallest ΔD_i, we select a subset of its video segments for uploading. Because segments in one device do not have time overlap, the total time coverage of those segments is equal to the sum of the length of each segment. Hence, the selection becomes a 0-1 knapsack problem where the items are video segments, the weight of a segment is its file size, and the value of a segment is its length. The knapsack problem can be solved using a polynomial time approximation scheme via dynamic programming [78]. It's time complexity is $O(n_i^3 \epsilon^{-1})$, and it has an approximation ratio of $(1 - \epsilon)$.

Simply applying the knapsack problem may not be the best solution. The knapsack problem values each segment by its length, but a segment overlapping with other segments in other devices is often less valuable than a "unique" segment

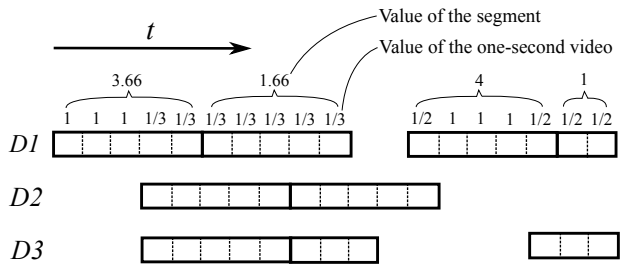


Figure 6.6. Value adjustments for segments in D_1.

which does not overlap with any other segments, even though the two segments have the same length. Thus, *the value of segments should be adjusted based on their time overlap.*

As an example, consider the first segment in D_1 in Fig. 6.6, where $L = 5$ seconds. For each second of the video, let d' denote the number of devices that have video segments in that second. Then, $d' = 0$ for the first three seconds of the video, and $d' = 2$ for the last two seconds[3]. The value of each one-second video is then decreased from 1 to $\frac{1}{1+d'}$. For example, for D_1, its value is one in the first three seconds, and its value is $\frac{1}{3}$ in the fourth and the fifth second, and then its value for the first segment is 3.66.

After value adjustment, the video segments are selected by solving the knapsack problem. Then, segments in other devices overlapping with the selected segments should be updated to remove the overlapping parts. The update may cause some devices to have $\Delta D_i = 0$, so a constraint checking procedure is needed and we are back to the first step. The entire process repeats until the selection is done for all devices. Then, the VideoMec server groups the selected segments that are continuous in time and in the same video, and notifies the devices which part of their videos should be uploaded. Then, the notified devices will upload the corresponding segments. Algorithm 1 shows the formal description of the TBS algorithm.

[3]For simplicity, each timestamp is rounded to its closest whole second, so that two one-second videos either completely overlap, or do not overlap at all.

Algorithm 1 TBS algorithm

```
1: while selection is not done for some devices do
2:     while any device has ΔD_i = 0 do
3:         selection is done for that device;
4:         update segments in other devices to remove the
5:         overlapping parts;
6:     end while
7:     pick device D_i where i = arg min_i ΔD_i;
8:     for all segments V_{i,j} in D_i do
9:         adjust their values based on the time overlap;
10:    end for
11:    solve the 0-1 knapsack problem for D_i;
12:    selection is done for D_i;
13:    update segments in other devices to remove overlap;
14: end while
```

6.4.2 Location-Based Strategy

The Location-Based Strategy (LBS) is motivated by applications that have strict requirements on the locations of the uploaded videos. For example, a video capturing a suspect's face or a vehicle's license plate should have an appropriate angle and a close distance, so that the face or the plate number can be clearly recognized. Such information can be specified as angle and distance requirements in the query, and videos that are close to the required angle and distance should be uploaded first. Hence, we have the following optimization problem.

6.4.2.1 LBS Problem

Similar to the TBS problem, we consider how to select a subset of video segments such that the selected segments can be uploaded in time and their total value is maximized. Here the value of a segment is determined based on the video location, as described below.

Recall that at a given timestamp t_i, $\overrightarrow{\ell_i\ell_e}$ denotes the vector from the video location to the queried event location. Let $a_{diff} = \min_j \angle(\overrightarrow{\ell_e\ell_i}, \boldsymbol{a}_j)$ be the smallest difference between the actual viewing angle and any required angles. Then according to Section 6.3, only video frames satisfying $a_{diff} \leq a_{dev}$ may be included in the query result. Thus, we have $a_{diff} \in [0, a_{dev}]$ for all video frames considered in

the LBS problem. Similarly, let $d_{diff} = \left| \left\| \overrightarrow{\ell_i \ell_e} \right\| - d \right|$ be the difference between the actual viewing distance and the required distance. We have $d_{diff} \in [0, d_{dev}]$. Then, the value of the video frame at timestamp t_i is defined as

$$\mu \left(1 - \frac{a_{diff}}{a_{dev}}\right) + (1 - \mu) \left(1 - \frac{d_{diff}}{d_{dev}}\right),$$

where $\mu \in [0, 1]$ is a predefined constant representing the relative importance of angle against distance ($\mu = 0.5$ in our experiments). For each one-second video $[t_i, t_{i+1}]$, its value is the average value of the frame at t_i and the frame at t_{i+1}. Then, the value of a segment, denoted as $v(V_{i,j})$, is the total value of all the one-second videos. The LBS problem is formally defined as follows.

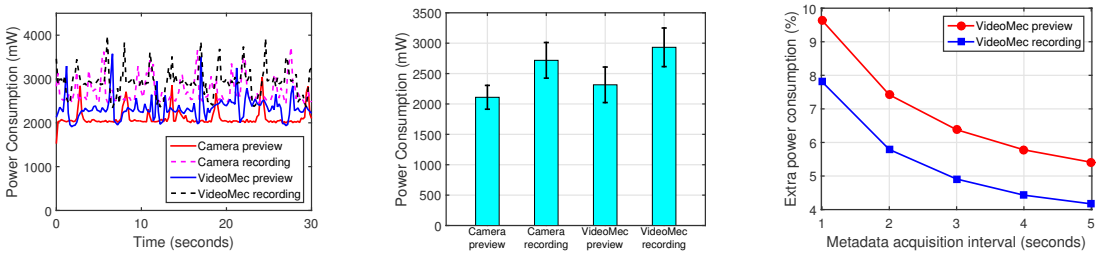
Definition 12 (LBS Problem)**.** *Consider a set of devices $D_1, D_2, \ldots, D_d$, where each device D_i has an average bandwidth B_i and a set of video segments $V_{i,1}, V_{i,2}, \ldots, V_{i,n_i}$. Each segment $V_{i,j}$ has a value $v(V_{i,j})$ and file size $s(V_{i,j})$. Given an upload time limit T, the LBS problem is to select a subset of segments $\mathcal{V} \subseteq \{V_{i,j} \mid (i = 1, \ldots, d) \wedge (j = 1, \ldots, n_i)\}$ such that every device can upload its segments in $\mathcal{V}$ within the time limit, i.e., $\sum_{V_{i,j} \in \mathcal{V}} s(V_{i,j}) \leq B_i T$ for every $i = 1, \ldots, d$, and the total value $\sum_{V_{i,j} \in \mathcal{V}} v(V_{i,j})$ is maximized.*

6.4.2.2 LBS Algorithm

Different from the TBS problem, the LBS problem is easier because its objective function is *additive*; i.e., the total value of segments equals to the sum of the value of each segment. Thus, selecting the best segments for all devices is equivalent to selecting the best segments for each device independently. Hence, the LBS algorithm solves the following problem for each device D_i.

$$\max \sum_{j \in \mathcal{J}} v(V_{i,j}),$$
$$\text{s.t.} \sum_{j \in \mathcal{J}} s(V_{i,j}) \leq B_i T,\ \mathcal{J} \subseteq \{1, \ldots, n_i\}$$

This is a 0-1 knapsack problem where the items are video segments. As mentioned before, it can be solved by a polynomial time approximation scheme which



(a) Raw measurement data

(b) Average and standard deviation

(c) Extra power vs. metadata acquisition interval

Figure 6.7. Power consumption of the VideoMec app vs. the Android Camera app.

gives a $(1-\epsilon)$ approximation. Because the value achieved for each device is at least $(1-\epsilon)$ times the optimal value for that device, the total value achieved by the LBS algorithm is also at least $(1-\epsilon)$ times the optimal value of the LBS problem. Therefore, the LBS algorithm has an approximation ratio of $(1-\epsilon)$.

6.5 Performance Evaluations

In this section, we present evaluation results of VideoMec. The VideoMec app was developed on Android smartphones, with the functionality of video recording, metadata recording, and video and metadata uploading. The app has been installed on four smartphones, two Samsung Galaxy S5 running Android 4.4.2 and two Motorola Nexus 6 running Android 5.1.1. The VideoMec server runs on a Dell OptiPlex 9020 desktop with Intel Core i7-4770 3.4GHz CPU.

6.5.1 Metadata Acquisition

When a mobile user starts recording a video, a timer is started and it triggers the metadata acquisition process every second to record the current timestamp, location, orientation, field of view, and coverage range (see Section 3.6.1 and our paper [79] for details). When the video recording is finished, the timer is terminated and additional information such as frame rate, resolution, and file size is recorded.

We use the Monsoon power monitor to measure the power consumption of metadata acquisition. More specifically, we measure and compare the power consumption of the VideoMec app and the standard Android Camera app on a Galaxy

S5 phone [4]. For each app, we measure the power consumption of the *preview* state where camera preview is shown on the screen but no video is being recorded, and the *recording* state where a video is being recorded. During the experiments, the camera records the same scene, so that the scene displayed on the screen has the same brightness. The results are shown in Fig. 6.7, where Fig. 6.7(a) is the raw measurement data and Fig. 6.7(b) is the average power consumption and the standard deviation. Compared to the Android Camera app, our app consumes 9.6% more power in the preview state, and 7.8% more power in the recording state. Based on the measured power consumption, a fully charged Galaxy S5 phone (2800mAh, 3.7V) can record videos for 3.53 hours using our app, which is only 17 minutes less than using the standard Android Camera app.

The power consumption of our app can be further reduced by increasing the metadata acquisition interval. For example, location usually does not change rapidly and it can be updated every five or ten seconds. The locations between two updates can be inferred by interpolation. Fig. 6.7(c) shows that increasing the metadata acquisition interval to five seconds reduces the extra power consumption to 4-5%.

6.5.2 Effectiveness of Query Processing

In this part, we evaluate the effectiveness of query processing by comparing it with a content-based approach, which requires all video files to be uploaded and uses computer vision algorithms to find desired videos.

6.5.2.1 Experimental Setup

To compare VideoMec with a content-based approach, we have implemented a query processor based on optical character recognition (OCR). This query processor accepts a word as a query, and outputs a set of video frames containing the word. Specifically, for each video, it extracts one frame per second, and for each extracted frame, it runs a text localization algorithm [80] to determine which part of the frame contains text. Then the localized text is recognized by Tesseract

[4]The geo-tagging feature of the standard Android Camera app is disabled by turning GPS and network off.

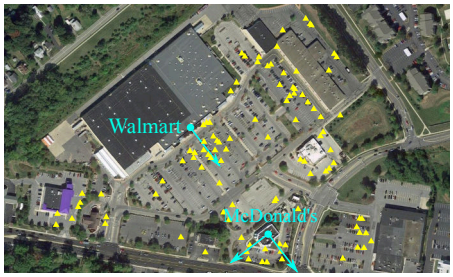


Figure 6.8. Satellite image of the shopping plaza. Yellow triangles are the initial locations of the 100 videos.

Figure 6.9. An example of correctly recognized logos.

OCR [81]. Since the recognition is subject to errors, edit distance [82] is used to determine whether the recognized text is *approximately* the same as the given word. A video frame matches the query if the minimum edit distance between the recognized text and the given word is less than a threshold.

Using VideoMec and the OCR-based query processor, we conduct the following experiments. We take 100 videos using four smartphones around a shopping plaza. The metadata (150 KB) is automatically generated and uploaded to the VideoMec server. The video files (10.7 GB) are also uploaded for the OCR-based query processor to use. Then we consider two queries: "find all videos containing the Walmart logo" and "find all videos containing the McDonald's logo". Here we consider queries on store logos because they are large, clear, and can be easily recognized by OCR.

The two queries are processed by both VideoMec's query processor and the OCR-based query processor. In addition, the ground truth is manually generated.
VideoMec. For VideoMec's query processor, we set up queries by specifying the locations, angles and allowed deviation from the specified angles. For Walmart, the event location is the position of the store logo on the building, and the specified angle is the angle from which we get the front view of the logo. For McDonald's, since it has two store logos on two sides of the building, the query location is set to be the center of the building, and there are two specified angles, one for each logo. These setups are shown as blue circles and arrows in Fig. 6.8. The allowed deviation from the specified angles is 45°, and other requirements such as time and distance are left unspecified.

		VideoMec	
	Logo?	Yes	No
Truth	Yes	492	26
	No	5	2530

		OCR	
	Logo?	Yes	No
Truth	Yes	295	223
	No	16	2519

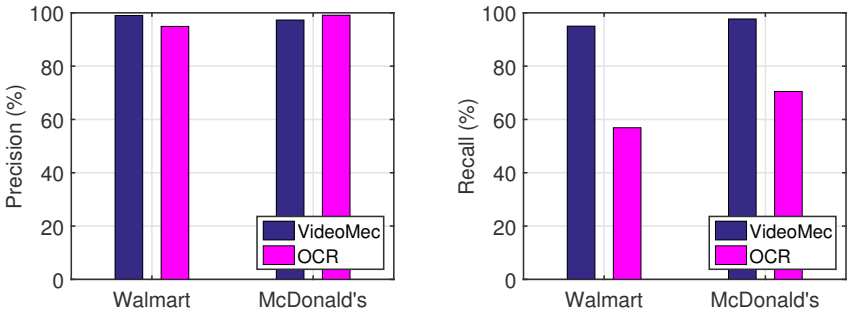
(a) Walmart

		VideoMec	
	Logo?	Yes	No
Truth	Yes	295	7
	No	8	2743

		OCR	
	Logo?	Yes	No
Truth	Yes	213	89
	No	2	2749

(b) McDonald's

Table 6.1. Confusion matrices of the query results. Each number represents the number of frames in the corresponding category.



(a) Precision (b) Recall

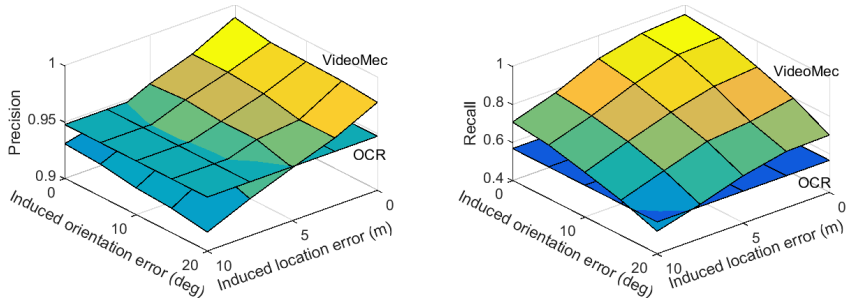
Figure 6.10. Precision and recall of the query results. Precision=true positive/(true positive+false positive). Recall=true positive/(true positive+false negative).

OCR. For the OCR-based query processor, the queries are simply two words, "Walmart" and "McDonalds". An example of correctly recognized logos are shown in Fig. 6.9.

Ground truth. Ground truth is obtained by visually checking each extracted frame to see whether the logo appears clearly and completely.

6.5.2.2 Results

By comparing the query results of VideoMec and OCR with the ground truth, we obtain several confusion matrices as shown in Table 6.1. Each number in the table



(a) Precision (b) Recall

Figure 6.11. Precision and recall of the Walmart query vs. induced location and orientation error.

	VideoMec		OCR	
Preparation	Upload metadata Build database	0.24s 0.022s	Upload videos	730s
Processing	Processing Upload videos	0.004s 139s	Processing	3466s

Table 6.2. Query processing time of the Walmart query. Each upload time is the longest time it took for any of the four smartphones to upload the corresponding data through WiFi.

represents the number of frames in the corresponding category. False judgments (i.e., false positives and false negatives) of VideoMec are mainly caused by large location and orientation error, which only happens occasionally. OCR, on the other hand, has more false judgments due to the imperfect text localization and recognition. Based on the confusion matrices, we calculate the precision and recall of queries for VideoMec and OCR, as shown in Fig. 6.10. Both VideoMec and OCR have very high precision ($\geq 95\%$), which means that among the frames they report as containing the logos, most indeed contain the logos. Although the recall of VideoMec is still high ($\geq 95\%$), the recall of OCR drops significantly, which means that OCR often fails to report frames that actually contain the logos. This is because OCR has difficulty of locating and recognizing text viewed from the side; i.e., when there is a large difference between the actual viewing angle and the specified angle.

We further evaluate the impact of location and orientation error on VideoMec's query processing performance. The obtained metadata already contains some er-

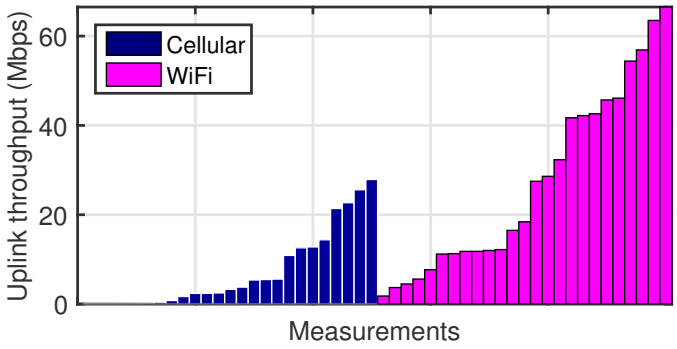


Figure 6.12. Uplink throughput measurements.

rors, but these errors cannot be controlled. Hence, we induce additional errors into the metadata and evaluate the sensitivity of precision and recall to errors in Fig. 6.11. For reference, the precision and recall of OCR are also drawn in the figure. They are not affected by metadata errors and thus appear as flat surfaces. As can be seen, the precision is not affected much by the induced errors. Even with 10 m location error and 20° orientation error, the precision is still around 92%. On the other hand, the recall decreases more when the induced errors become large. This is because when the metadata errors become larger, the number of false negatives increases much faster than the number of false positives. Nevertheless, the recall remains above 80% as long as the induced location error is less than 5 m and the induced orientation error is less than 10°. Compared to the recall of OCR (57%), the recall of VideoMec is almost always higher.

In addition to precision and recall, the query processing time is also compared in Table 6.2. For VideoMec, the metadata needs to be uploaded and stored into the database beforehand. Then the query can be processed quickly and videos matching the query are uploaded. If a time limit is given, VideoMec can run the TBS algorithm or the LBS algorithm to further bound the upload time (at the cost of uploading fewer, but more important videos). On the other hand, OCR takes much more time to prepare since it requires all videos to be uploaded, although many of them are not useful. It also takes more time to process the query because like most computer vision algorithms, OCR is computationally expensive.

6.5.3 Effectiveness of TBS and LBS algorithms

In this subsection, we evaluate the performance of the algorithms used in VideoMec's upload decision engine. To obtain the uploading bandwidth used in the algorithms, we collect 50 uplink throughput measurements using our smartphones at various locations. In each measurement, we upload a 1 MB video to the VideoMec server and calculate the throughput. Half of the uploading used cellular networks and half used WiFi. The median throughput measured for cellular networks is 3.3 Mbps, and the median for WiFi is 18.4 Mbps. The results are shown in Fig. 6.12. Those measurements are used as traces to compare our algorithms with other algorithms under the same bandwidth setting.

Five different queries are executed based on 100 collected videos. For each query, the selected videos should be uploaded to the VideoMec server. Based on the measured bandwidth, each phone will randomly pick a measurement value to be its available wireless bandwidth. Since the wireless bandwidth is limited, TBS or LBS is used to determine which part of the videos should be uploaded. For TBS, the performance is evaluated by the time coverage of the uploaded videos. For LBS, the performance is evaluated by the total value of the uploaded videos. Since there are five queries, and 10 bandwidth measurements are used for each query, we have 50 experiment runs, and each data point is averaged over those 50 runs.

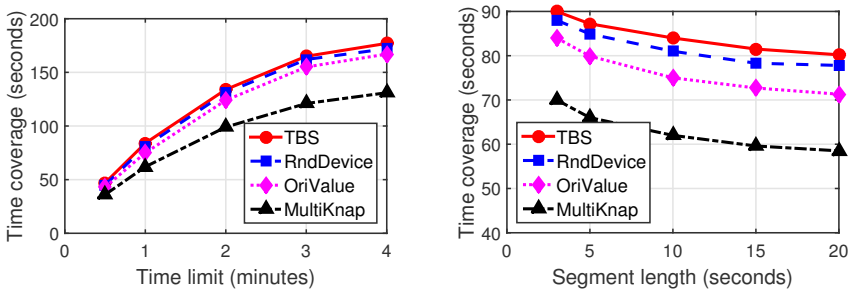
6.5.3.1 TBS Algorithm

The TBS algorithm is compared with the following three algorithms:

- **RndDevice:** It is based on the TBS algorithm. When deciding which device to select next, it randomly picks a device instead of choosing the one with the smallest ΔD_i.

- **OriValue:** It is based on the TBS algorithm. Instead of value adjustment before solving the knapsack problem, it uses the original value (i.e., the length) of the video segments.

- **MultiKnap:** It considers the TBS problem as the multiple independent knapsack problem. Each knapsack problem maximizes the time coverage for



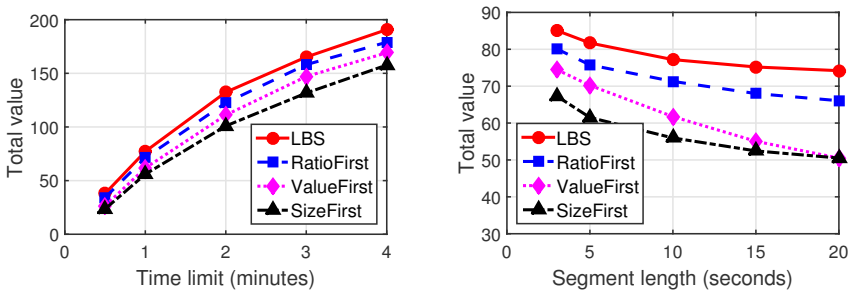
(a) (b)

Figure 6.13. Comparisons between different TBS algorithms.

one device, while the time overlap among devices is ignored. It uses the polynomial time approximation scheme to solve each knapsack problem.

Fig. 6.13 (a) shows the time coverage as a function of the time limit, where the segment length L is fixed at 10 seconds. When using the polynomial time approximation scheme to solve the knapsack problem, parameter ϵ is set to 0.01. As shown in the figure, the two heuristics used in the TBS algorithm have some benefits. Choosing the device with the smallest ΔD_i instead of choosing randomly improves the time coverage by 2-4%, and using the adjusted value rather than the original value improves the coverage by 6-12%. Compared to MultiKnap, which maximizes the time coverage for each device but ignores the time overlap among devices, the TBS algorithm achieves 30-36% more coverage. For all four algorithms, the time coverage increases as the time limit increases. The increase, however, slows down gradually because with more videos uploaded, the remaining videos will be more likely to overlap with the uploaded videos, and thus contribute less coverage when being uploaded.

Fig. 6.13 (a) shows the time coverage as a function of the segment length, where the upload time limit T is fixed at 1 minute. As can be seen, smaller segment length increases the flexibility of selecting different parts of the video, and thus improves performance. Since switching to a different segment too frequently hurts user experience, a moderate segment length such as 10 seconds is used to balance the tradeoff between time coverage and user experience.



(a) (b)

Figure 6.14. Comparisons between different LBS algorithms.

6.5.3.2 LBS Algorithm

The LBS algorithm has $(1-\epsilon)$ approximation ratio where $\epsilon > 0$ can be arbitrarily small. To evaluate its performance, we compare it with the following heuristic based algorithms:

- **RatioFirst:** It solves the knapsack problem for each device by selecting segments from the largest value-to-size ratio to the smallest, until the upload time limit is reached.
- **ValueFirst:** It is similar to RatioFirst, but selects segments from the largest value to the smallest.
- **SizeFirst:** Similar to RatioFirst, but selects segments from the smallest size to the largest.

Fig. 6.14(a) shows the total value as a function of the time limit, where the segment length L is fixed at 10 seconds. As can be seen, the LBS algorithm performs 4-11% better than RatioFirst, 12-43% better than ValueFirst, and 21-61% better than SizeFirst. We also find that the performance difference between the LBS algorithm and the heuristic based algorithms is larger when the time limit is shorter, because with a longer upload time, good segments will always be selected even though the heuristic based algorithms do not have the optimal selection. For all four algorithms, the total value increases as the time limit increases, though the increase does not slow down as much as that in Fig. 6.13(a). Here the slow down occurs because some devices have selected all their segments and thus additional upload time does not provide any benefits.

Fig. 6.14 (b) shows the total value as a function of the segment length, where the upload time limit T is fixed at 1 minute. Similar to Fig. 6.13(b), the total value decreases as the segment length increases. Thus, in the LBS problem, it is also important to choose the appropriate segment length to achieve a better tradeoff between total value and user experience.

Chapter 7

Conclusions and Future Work

7.1 Summary

In this dissertation, we designed and evaluated various metadata-based techniques to support resource-aware crowdsourcing in wireless networks. Metadata includes camera location, orientation, field of view, coverage range, and other parameters such as focal length, which are easily accessible from the embedded sensors of most off-the-shelf mobile devices. Metadata describes where and how a photo/video is taken, and can be used to measure the value of crowdsourced photos/videos. More importantly, metadata is very small and consumes very little bandwidth, storage, and computing resources, which addresses the challenges of resource constraints in crowdsourcing applications. Based on metadata, we designed various solutions to support resource-aware crowdsourcing, which are summarized as follows.

In Chapter 3, we designed SmartPhoto, a resource-aware framework which can select the most useful photos to cover some points of interest. A metric called photo utility was proposed to evaluate the value of photos based on how many angles of the interested points are covered, which can be calculated from metadata. Three optimization problems regarding the tradeoffs between photo utility and resource usage, namely max-utility, min-selection and min-selection with k-coverage, have been studied. The performance bounds of the proposed algorithms were theoretically proved. We have implemented SmartPhoto in a testbed using Android smartphones, and proposed techniques to improve the accuracy of the collected metadata and mitigate the occlusion and out-of-focus issues. Results based on

real implementations and extensive simulations validated the effectiveness of the proposed algorithms.

In Chapter 4, we extended the proposed framework so that it can also select photos to cover an area of interest. Since such coverage is affected by the angle or viewpoint from which the photo is taken, it is very different from previous coverage problems studied in wireless sensor networks. To address the challenge, we extended the definition of photo utility to measure how well the entire target area is covered. Various techniques have been proposed to analyze area coverage and calculate photo utility accurately and efficiently. We further studied the budgeted max-utility problem, where the server selects photos with the largest utility under a resource budget. As the problem was proved NP-hard, we proposed an efficient approximation algorithm that achieves constant approximation ratio. Results of simulations and real-world experiments demonstrated the effectiveness of the proposed techniques.

In Chapter 5, we studied the photo crowdsourcing problem in scenarios like disaster recovery and battlefield, where the cellular network can be partly damaged or overloaded with extensive requests. In these scenarios, photos related to points of interest have to be transmitted to the command center through DTN. Due to bandwidth and storage limitations in DTN, nodes need to select and transmit the most valuable photos. We measured the value of photos based on their metadata, and designed a distributed photo selection algorithm to maximize the total value of photos delivered to the command center considering bandwidth and storage constraints. Results based on a proof-of-concept prototype and extensive simulations demonstrated the effectiveness of our design.

In Chapter 6, we designed VideoMec, a crowdsourcing system that organizes and indexes the metadata of all videos taken by mobile devices. By using the uploaded metadata to serve queries and then only uploading the queried video files, VideoMec has much better scalability while reducing the bandwidth and energy consumption of the mobile devices. VideoMec supports comprehensive queries for applications to find videos quickly and accurately from the distributed video repository. For time-sensitive applications where not all identified videos can be uploaded in time, VideoMec selects the most important videos or video segments to upload given bandwidth and time constraints. A prototype of VideoMec has

been implemented, and various experiments have been conducted to demonstrate its effectiveness.

7.2 Future Directions

Our work to date has provided a series of techniques for resource-aware crowdsourcing in wireless networks. There are other research directions worth further studies. Next we outline several directions for future work that one could pursue.

- **Extension to three-dimensional metadata and coverage model:** So far the metadata and the coverage model considered in our studies are two-dimensional. The metadata, including location, orientation, etc., is represented as 2D information on the map. The coverage is only calculated along the longitude and latitude of the earths surface, without considering altitude. In practice, users can take photos/videos towards the top of a high skyscraper or towards its ground entrance, resulting in totally different scenes. To handle these cases, it is necessary to extend the metadata and coverage model to the third dimension in space.

  Specifically, for location, the altitude of the camera needs to be recorded together with the longitude and latitude. Orientation, represented by one angle in 2D, will be represented by three angles (azimuth, pitch, and roll) in 3D. Field of view, also represented by one angle in 2D, will be represented by two angles (horizontal field of view and vertical field of view) in 3D. The resulting 3D metadata will define a pyramid-shaped 3D space covered by the camera. With this 3D coverage model, we can accurately compute whether a 3D point can be captured in a 2D image, and if so, where it will appear in the image. Furthermore, we can map any 3D scene to a 2D image as long as we know the 3D metadata of the photo/video and the geometric structure of the 3D scene.

  The 3D coverage model can help us detect occlusion much more accurately. For example, we can check the altitude of the camera, the interested targets, and the objects in between. Occlusion occurs when the objects in between is high enough to block the line-of-sight between the camera and the targets.

Based on this idea, effective occlusion detection techniques may be developed. Moreover, the 3D coverage model can be used to develop much finer optimization models for photo/video selection, though 3D coverage is more complex than 2D coverage and extensive studies are needed to address the challenge.

- **Crowdsourcing for target tracking:** In our studies, the targets to be covered are static objects or events with known locations. In some applications such as law enforcement, the target can be a person or a vehicle on the move. The proposed techniques may be adjusted or new techniques may be developed to efficiently track moving targets using crowdsourced photos/videos. The key challenge here is that when tracking a moving target, its current location is unknown to the server. The server may know the previous locations of the target and may use domain knowledge or movement prediction algorithms [83, 84] to predict the current location of the target. The prediction will indicate the area where the target may appear as well as the probability that the target appear at each point in the area. Thus, photos/videos should be crowdsourced based on the prediction to maximize the probability of finding the target. Such optimization will be challenged by the probability-based coverage models, the mobility of mobile users, and the resource constraints at both the server and the user side.

- **Integrating with content-based approaches:** Our metadata-based solutions and content-based approaches can be complementary to each other. On one hand, metadata-based solutions support new applications which cannot be supported by content-based approaches. For example, when a tourist inadvertently films a suspect in a video, content-based techniques may label the video as travel related, nothing to do with the suspect. Then the video will never be considered useful for identifying the suspect unless its metadata is considered. On the other hand, content-based approaches can address some issues such as occlusion in metadata-based approaches. Metadata-based approaches can be used as a filter step to quickly eliminate photos/videos that do not match the required spatiotemporal information (e.g., location, angle, etc.). Then, content-based approaches can be applied to refine the search

so that photos/videos in which the target is occluded are not selected or uploaded. Consequently, developing a system where metadata-based and content-based approaches work seamlessly together will be an important next step, though the high bandwidth and computational cost of content-based approaches must be carefully considered in resource constrained environments.

Bibliography

[1] Alon, N. and J. H. Spencer (2006) *We the Media: Grassroots Journalism by the People, for the People*, O'Reilly Media, Sebastopol, California.

[2] Huang, Y., Y. Wang, and C. White (2014) "Designing a Mobile System for Public Safety Using Open Crime Data and Crowdsourcing," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*.

[3] Liu, S. B., L. Palen, J. Sutton, A. L. Hughes, and S. Vieweg (2008) "In Search of the Bigger Picture: The Emergent Role of On-Line Photo Sharing in Times of Disaster," in *ISCRAM*.

[4] Zhai, Z., T. Kijewski-Correa, D. Hachen, and G. Madey (2012) "Haiti Earthquake Photo Tagging: Lessons on Crowdsourcing In-Depth Image Classifications," in *ICDIM*.

[5] Bohez, S., J. Mostaert, T. Verbelen, P. Simoens, and B. Dhoedt (2014) "Management of Crowdsourced First-person Video: Street View Live," in *Proc. of ACM International Conference on Mobile and Ubiquitous Multimedia*.

[6] Snavely, N., S. M. Seitz, and R. Szeliski (2006) "Photo tourism: Exploring photo collections in 3D," in *Proc. ACM International Conference and Exhibition on Computer Graphics and Interactive Techniques (SIGGRAPH)*.

[7] "Street View," `http://www.google.com/maps/views/streetview/`.

[8] LiKamWa, R. and L. Zhong (2015) "Starfish: Efficient Concurrency Support for Computer Vision Applications," in *ACM MobiSys*.

[9] Wan, J., D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li (2014) "Deep Learning for Content-Based Image Retrieval: A Comprehensive Study," in *ACM International Conference on Multimedia*.

[10] Yan, T., V. Kumar, and D. Ganesan (2010) "CrowdSearch: Exploiting Crowds for Accurate Real-time Image Search on Mobile Phones," in *ACM MobiSys*.

[11] Qin, C., X. Bao, R. Roy Choudhury, and S. Nelakuditi (2011) "TagSense: A Smartphone-based Approach to Automatic Image Tagging," in *ACM MobiSys*.

[12] Crandall, D. and N. Snavely (2012) "Modeling people and places with internet photo collections," *Commun. ACM*, **55**(6).

[13] Jin, H., L. Su, H. Xiao, and K. Nahrstedt (2016) "INCEPTION: Incentivizing Privacy-Preserving Data Aggregation for Mobile Crowd Sensing Systems," in *ACM MobiHoc*.

[14] Chen, Y., B. Li, and Q. Zhang (2016) "Incentivizing Crowdsourcing Systems with Network Effects," in *IEEE INFOCOM*.

[15] Peng, D., F. Wu, and G. Chen (2015) "Pay As How Well You Do: A Quality Based Incentive Mechanism for Crowdsensing," in *ACM MobiHoc*.

[16] Manweiler, J., P. Jain, and R. R. Choudhury (2012) "Satellites in Our Pockets: An Object Positioning System using Smartphones," in *ACM Mobisys*.

[17] Gao, R., M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li (2014) "Jigsaw: Indoor Floor Plan Reconstruction via Mobile Crowdsensing," in *ACM MobiCom*.

[18] Chen, S., M. Li, K. Ren, X. Fu, and C. Qiao (2015) "Rise of the Indoor Crowd: Reconstruction of Building Interior View via Mobile Crowdsourcing," in *ACM SenSys*.

[19] Chen, S., M. Li, K. Ren, and C. Qiao (2015) "CrowdMap: Accurate Reconstruction of Indoor Floor Plans from Crowdsourced Sensor-Rich Videos," in *IEEE ICDCS*.

[20] Chen, F., C. Zhang, F. Wang, and J. Liu (2015) "Crowdsourced Live Streaming over Cloud," in *IEEE INFOCOM*.

[21] Dong, J., Y. Xiao, M. Noreikis, Z. Ou, and A. Yla-Jaaski (2015) "iMoon: Using Smartphones for Image-based Indoor Navigation," in *ACM SenSys*.

[22] Dong, J., Y. Xiao, Z. Ou, Y. Cui, and A. Yla-Jaaski (2016) "Indoor Tracking Using Crowdsourced Maps," in *ACM/IEEE IPSN*.

[23] Jiang, Y., X. Xu, P. Terlecky, T. Abdelzaher, A. Bar-Noy, and R. Govindan (2013) "MediaScope: Selective on-demand media retrieval from mobile devices," in *ACM/IEEE IPSN*.

[24] Akyildiz, I. F., T. Melodia, and K. R. Chowdhury (2007) "A survey on wireless multimedia sensor networks," *Comput. Netw.*, **51**(4), pp. 921–960.

[25] Rinner, B. and W. Wolf (2008) "A Bright Future for Distributed Smart Cameras," *Proceedings of the IEEE*, **96**(10), pp. 1562 – 1564.

[26] Soro, S. and W. Heinzelman (2009) "A Survey of Visual Sensor Networks," *Advances in Multimedia*, **2009**, pp. 1–22.

[27] Li, F., J. Barabas, and A. L. Santos (2010) "Information processing for live photo mosaic with a group of wireless image sensors," in *Proc. ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN)*.

[28] Shen, Y., W. Hu, M. Yang, J. Liu, and C. T. Chou (2012) "Efficient background subtraction for tracking in embedded camera networks," in *Proc. ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN)*.

[29] Gong, X., J. Zhang, D. Cochran, and K. Xing (2013) "Barrier coverage in bistatic radar sensor networks: cassini oval sensing and optimal placement," in *ACM MobiHoc*.

[30] Hong, J., J. Cao, Y. Zeng, S. Lu, D. Chen, and Z. Li (2009) "A location-free Prediction-based Sleep Scheduling protocol for object tracking in sensor networks," in *IEEE ICNP*.

[31] Wan, P.-J., X. Xu, and Z. Wang (2011) "Wireless coverage with disparate ranges," in *ACM MobiHoc*.

[32] Younis, O., M. Krunz, and S. Ramasubramanian (2007) "Coverage Without Location Information," in *IEEE ICNP*.

[33] Johnson, M. P. and A. Bar-Noy (2011) "Pan and Scan: COnfiguring Cameras for Coverage," in *IEEE INFOCOM*.

[34] Hörster, E. and R. Lienhart (2006) "On the optimal placement of multiple visual sensors," in *Proc. ACM International Workshop on Video Surveillance and Sensor Networks (VSSN)*.

[35] Wang, Y. and G. Cao (2011) "On Full-View Coverage in Camera Sensor Networks," in *IEEE INFOCOM*.

[36] ——— (2011) "Barrier Coverage in Camera Sensor Networks," in *ACM MobiHoc*.

[37] ——— (2013) "Achieving full-view coverage in camera sensor networks," *ACM Transactions on Sensor Networks*, **10**(1).

[38] Wu, Y. and X. Wang (2012) "Achieving Full View Coverage with Randomly-Deployed Heterogeneous Camera Sensors," in *IEEE ICDCS*.

[39] Yu, Z., F. Yang, J. Teng, A. C. Champion, and D. Xuan (2015) "Local Face-View Barrier Coverage in Camera Sensor Networks," in *IEEE INFOCOM*.

[40] Spyropoulos, T., K. Psounis, and C. S. Raghavendra (2005) "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *ACM SIGCOMM workshop on Delay-tolerant networking*.

[41] Lindgren, A., A. Doria, and O. Schelen (2003) "Probabilistic routing in intermittently connected networks," in *ACM SIGMOBILE Mobile Computing and Communications Review*.

[42] Lee, K., Y. Yi, J. Jeong, H. Won, I. Rhee, and S. Chong (2010) "Max-Contribution: On Optimal Resource Allocation in Delay Tolerant Networks," in *IEEE INFOCOM*.

[43] Gao, W. and G. Cao (2010) "On Exploiting Transient Contact Patterns for Data Forwarding in Delay Tolerant Networks," in *IEEE ICNP*.

[44] Balasubramanian, A., B. N. Levine, and A. Venkataramani (2007) "DTN routing as a resource allocation problem," in *ACM SIGCOMM*.

[45] ——— (2008) "Enhancing Interactive Web Applications in Hybrid Networks," in *ACM MobiCom*.

[46] Krifa, A., C. Barakat, and T. Spyropoulos (2008) "An Optimal Joint Scheduling and Drop Policy for Delay Tolerant Networks," in *WoWMoM Workshop on Autonomic and Opportunistic Communications*.

[47] Wang, H., M. Uddin, G. Qi, T. Huang, T. Abdelzaher, and G. Cao (2011) "PhotoNet: A similarity-aware image delivery service for situation awareness," in *ACM/IEEE IPSN*, pp. 135–136.

[48] Weinsberg, U., Q. Li, N. Taft, A. Balachandran, V. Sekar, G. Iannaccone, and S. Seshan (2012) "CARE: Content Aware Redundancy Elimination for Challenged Networks," in *ACM HotNets*.

[49] Chen, F., C. Zhang, F. Wang, and J. Liu (2015) "Crowdsourced Live Streaming over Cloud," in *IEEE INFOCOM*.

[50] Simoens, P., Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan (2013) "Scalable Crowd-Sourcing of Video from Mobile Devices," in *ACM Mobisys*.

[51] Bao, X. and R. R. Choudhury (2010) "MoVi: Mobile Phone based Video Highlights via Collaborative Sensing," in *ACM MobiSys*.

[52] Jain, P., J. Manweiler, A. Acharya, and K. Beaty (2013) "FOCUS: Clustering Crowdsourced Videos by Line-of-Sight," in *ACM Sensys*.

[53] Kim, S. H., Y. Lu, J. Shi, A. Alfarrarjeh, C. Shahabi, G. Wang, and R. Zimmermann (2015) "Key Frame Selection Algorithms for Automatic Generation of Panoramic Images from Crowdsourced Geo-tagged Videos," in *Proc. of International Symposium on Web and Wireless Geographical Information Systems*.

[54] Lee, U., E. Magistretti, M. Gerla, P. Bellavista, and A. Corradi (2009) "Dissemination and Harvesting of Urban Data Using Vehicular Sensing Platforms," *IEEE Transactions on Vehicular Technology*, **58**(2).

[55] Blanz, V., P. Grother, P. J. Phillips, and T. Vetter (2005) "Face Recognition Based on Frontal Views Generated from Non-Frontal Images," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 454–461.

[56] Phillips, P. J., W. T. Scruggs, A. J. O'Toole, P. J. Flynn, K. W. Bowyer, C. L. Schott, and M. Sharpe (2007) "FRVT 2006 and ICE 2006 Large-scale Results," *National Institute of Standards and Technology, Tech. Rep. NISTIR 7408*.

[57] Hochbaum, D. S. (ed.) (1996) *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company.

[58] Wu, Y., Y. Wang, W. Hu, and G. Cao (2016) "SmartPhoto: A Resource-Aware Crowdsourcing Approach for Image Sensing with Smartphones," *IEEE Transactions on Mobile Computing*, **15**(5).

[59] Dobson, G. (1982) "Worst-Case Analysis of Greedy Heuristics for Integer Programming with Nonnegative Data," *Mathematics of Operations Research*, **7**(4), pp. 515–531.

[60] "Android Developers," `http://developer.android.com/index.html`.

[61] Meier, R. (2010) *Professional Android 2 Application Development, 2nd Ed.*, Wiley Publishing, Inc.

[62] Williams, G. (2012) *Linear Algebra With Applications*, Jones & Bartlett Learning.

[63] Ray, S. F. (2002) *Applied Photographic Optics*, 3 ed., Focal Press, Oxford, UK.

[64] Gabarda, S. and G. Cristobal (2007) "Blind image quality assessment through anisotropy," *Journal of the Optical Society of America A*, **24**(12), pp. B42–B51.

[65] Nurnberg, R., "Calculating the area and centroid of a polygon in 2d," `www.ma.ic.ac.uk/~rn/centroid.pdf`.

[66] Sommariva, A. and M. Vianello (2007) "Product Gauss cubature over polygons based on Green's integration formula," *BIT Numerical Mathematics*, **47**.

[67] Garey, M. R. and D. S. Johnson (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman.

[68] Krause, A. and C. Guestrin (2005) *A Note on the Budgeted Maximization of Submodular Functions*, *Tech. rep.*, CMU-CALD-05-103.

[69] Wu, Y., Y. Wang, and G. Cao (2017) "Photo Crowdsourcing for Area Coverage in Resource Constrained Environments," in *IEEE INFOCOM*.

[70] Lee, D. and K. Plataniotis (2016) "Toward a No-Reference Image Quality Assessment Using Statistics of Perceptual Color Descriptors," *IEEE Transactions on Image Processing*, **25**(8).

[71] Cai, H. and D. Y. Eun (2008) "Toward Stochastic Anatomy of Inter-meeting Time Distribution under General Mobility Models," in *ACM MobiHoc*.

[72] Gao, W., Q. Li, B. Zhao, and G. Cao (2012) "Social-aware Multicast in Disruption Tolerant Networks," *IEEE/ACM Transactions on Networking*.

[73] Zhu, H., L. Fu, G. Xue, Y. Zhu, M. Li, and L. Ni (2010) "Recognizing Exponential Inter-Contact Time in VANETs," in *IEEE INFOCOM*.

[74] Wu, Y., Y. Wang, W. Hu, X. Zhang, and G. Cao (2016) "Resource-Aware Photo Crowdsourcing Through Disruption Tolerant Networks," in *IEEE ICDCS*.

[75] Leguay, J., A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft (2006) "Opportunistic Content Distribution in an Urban Setting," in *ACM SIGCOMM Workshop on Challenged Networks*.

[76] (2016), "New video shows alleged Brussels bomber's escape," The Washington Post.

[77] Beckmann, N., H.-P. Kriegel, R. Schneider, and B. Seeger (1990) "The R*-tree: an efficient and robust access method for points and rectangles," in *ACM SIGMOD*.

[78] Kellerer, H., U. Pferschy, and D. Pisinger (2004) *Knapsack Problems*, Springer Berlin Heidelberg.

[79] Wu, Y. and G. Cao (2017) "VideoMec: A Metadata-Enhanced Crowdsourcing System for Mobile Videos," in *ACM/IEEE IPSN*.

[80] Neumann, L. and J. Matas (2012) "Real-Time Scene Text Localization and Recognition," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*.

[81] Kay, A. (2007) "Tesseract: an Open-Source Optical Character Recognition Engine," *Linux Journal*.

[82] Manning, C. D., P. Raghavan, and H. Schutze (2008) *Introduction to Information Retrieval*, Cambridge University Press.

[83] Li, M., A. Ahmed, and A. J. Smola (2015) "Inferring Movement Trajectories from GPS Snippets," in *ACM WSDM*.

[84] Baratchi, M., N. Meratnia, P. J. Havinga, A. K. Skidmore, and B. A. Toxopeus (2014) "A Hierarchical Hidden Semi-Markov Model for Modeling Mobility Data," in *ACM UbiComp*.

Vita

Yibo Wu

Yibo Wu received the B.E. degree in Information Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2012. He enrolled in the Ph.D. program in Computer Science and Engineering at The Pennsylvania State University in August 2012.

Publications during the Ph.D. study:

- Yibo Wu and Guohong Cao, "VideoMec: A Metadata-Enhanced Crowdsourcing System for Mobile Videos," ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2017.
- Yibo Wu, Yi Wang, and Guohong Cao, "Photo Crowdsourcing for Area Coverage in Resource Constrained Environments," IEEE International Conference on Computer Communications (INFOCOM), 2017.
- Yibo Wu, Yi Wang, Wenjie Hu, Xiaomei Zhang, and Guohong Cao, "Resource-Aware Photo Crowdsourcing Through Disruption Tolerant Networks," IEEE International Conference on Distributed Computing Systems (ICDCS), 2016.
- Yibo Wu, Yi Wang, Wenjie Hu, and Guohong Cao, "SmartPhoto: A Resource-Aware Crowdsourcing Approach for Image Sensing with Smartphones," IEEE Transactions on Mobile Computing (TMC), vol. 15, no. 5, pp. 1249-1263, 2016.