Photo Crowdsourcing for Area Coverage in Resource Constrained Environments

Yibo Wu, Yi Wang, and Guohong Cao

Department of Computer Science and Engineering
The Pennsylvania State University, University Park
Email: {yxw185, yuw124, gcao}@cse.psu.edu

***Abstract*—Photos crowdsourced from mobile devices can be used in many applications such as disaster recovery to obtain information about a target area. However, such applications often have resource constraints in terms of bandwidth, storage, and processing capability, which limit the number of photos that can be crowdsourced. Thus, it is a challenge to use the limited resources to crowdsource photos that best cover the target area. In this paper, we leverage various geographical and geometrical information about photos, called *metadata*, to address this challenge. Metadata includes the location, orientation, field of view, and range of a camera. Based on metadata, we define photo utility to measure how well a target area is covered by a set of photos. We propose various techniques to analyze such coverage and calculate photo utility accurately and efficiently. We also study the problem of selecting photos with the largest utility under a resource budget, and propose an efficient algorithm that achieves constant approximation ratio. With our design, the crowdsourcing server can select photos based on metadata instead of real images, and thus use the limited resources to crowdsource the most useful photos. Both simulation and experimental results demonstrate the effectiveness of our design.**

I. Introduction

The ever growing trend of taking and sharing photos using smart devices (smartphones, tablets, Google Glass, etc.) has enabled a variety of photo crowdsourcing applications. 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 photos taken by people using smart devices become extremely useful. There may be lots of photos taken by mobile users, and the communication infrastructure may be severely damaged. Then, the transmission of photos from mobile users to the central service or central authority will be limited by the bandwidth constraints. Thus, it is a challenge to efficiently utilize the limited bandwidth to find and collect the most useful photos that best cover the whole area.

Street view service is another application that can benefit from photo crowdsourcing. Areas like university campuses, theme parks, and outlets usually do not have street view because they are inaccessible to street view cars [3]. Since these areas have a large number of visitors, street view service can be very useful. 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. Unfortunately, the sheer amount of photos poses big challenges for image processing and storage at the server. Fully understanding the semantic of each photo by resource-intensive image processing techniques [13], [19] would be a luxury if not impossible. Thus, it is a challenge to analyze photos quickly and select those that provide the best coverage.

The challenges in the above applications are at least two-fold. First, any point inside the target area can be a point of interest. Hence the collected photos should cover as much area as possible and for each point covered, the photos should be taken from multiple angles. This requires an in-depth analysis of the coverage of photos with regard to the target area. Second, the resource is limited and not all candidate photos can be uploaded or analyzed by image processing techniques. Hence the resource must be used more efficiently to find and collect the most useful photos.

Some existing techniques may be applied to address these challenges. For example, some content-based image retrieval techniques [19], [24] 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 computation 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., computation capability and energy) to execute those programs. As another solution, description-based techniques can be applied to annotate photos by tags [16] or GPS locations [6]. However, tags may be inaccurate or insufficient for applications to determine the coverage/content of photos. Some tags may involve 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 may have different views.

In this paper, we use various geographical and geometrical information, called *metadata*, to characterize photos. Metadata includes the location, orientation, field of view, and range of a camera, which are easily accessible from the APIs and built-in sensors of most mobile devices. Whenever a photo is taken, its metadata is automatically generated and recorded. From metadata, we can infer where and how the photo is taken, and determine which area is covered by the photo.

This work was supported in part by the National Science Foundation (NSF) under grant CNS-1526425 and CNS-1421578.

Thus, based on metadata, we can quantify how well a set of photos can cover a target area, which is referred to as photo *utility*. We propose various techniques to analyze such coverage and calculate photo utility accurately and efficiently. Note that although the coverage problem has been studied in wireless sensor networks, our application and model are different. The 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 paper.

With metadata and utility, photo crowdsourcing becomes much more resource-friendly. When a crowdsourcing event happens, the server first asks users to upload photo metadata, and selects photos with the largest utility under a predefined resource budget. Then users upload the selected photos and get paid [15] (how users are paid is out of the scope of this paper). During this process, the unselected photos (except their metadata) will not be transmitted to or stored in the server, thus saving lots of bandwidth and storage space. All the photos are analyzed based on their metadata rather than the real images, which can significantly reduce the computational cost and delay. 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. It is provably near optimal, achieving constant approximation ratio compared to the optimal solution. Both simulations and real-world experiments are used to evaluate the performance of our design and demonstrate its effectiveness.

Our contributions are summarized as follows. First, we develop a photo crowdsourcing framework to collect photos that cover a target area. Because the framework analyzes photos by metadata rather than real images, it significantly reduces the bandwidth, storage, and computational cost of photo crowdsourcing. Second, we propose a utility metric to quantify how well a target area is covered by a set of photos, and develop various techniques to analyze such coverage and calculate utility accurately and efficiently. Third, we study the problem of selecting photos with the largest utility subject to a resource budget, and propose an efficient algorithm with constant approximation ratio. Finally, the effectiveness of the proposed techniques is evaluated through both simulations and real-world experiments based on Android smartphones.

The rest of the paper is organized as follows. Section II introduces metadata and photo utility. Section III studies how to analyze photo coverage and calculate photo utility. The budgeted max-utility problem is studied in Section IV. Section V evaluates the performance of the proposed techniques. Section VI discusses some practical issues, Section VII reviews related work, and Section VIII concludes the paper.

II. Metadata and Photo Utility

A. 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. These four parameters can be obtained from the APIs and built-in sensors of most mobile devices.

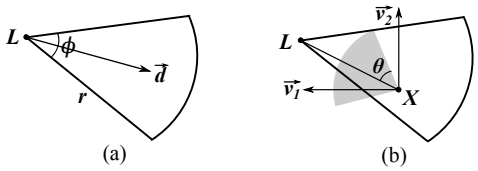


Fig. 1. (a) Coverage area of a photo. (b) Illustration of aspect coverage. Aspect $\overrightarrow{v_1}$ is covered since $\angle(\overrightarrow{v_1}, \overrightarrow{XL}) \leq \theta$; aspect $\overrightarrow{v_2}$ is not covered since $\angle(\overrightarrow{v_2}, \overrightarrow{XL}) > \theta$. In fact, all the 2θ aspects in the gray area are covered.

Fig. 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 [20], [21], [25].

B. Photo Utility

Based on the metadata, photo utility 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. 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. 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. 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 1 (Utility)**.** *Let P be a set of photos, and $1_P(\vec{v})$ be an indicator function that equals 1 if aspect $\vec{v}$ is covered by at least one photo in P and 0 otherwise. The utility of P on a given point X is $U_P(X) = \int_0^{2\pi} 1_P(\vec{v})\ d(arg(\vec{v}))$, where $\vec{v}$ is an aspect of point X and $arg(\vec{v})$ is the variable of integration changing from 0 to 2π. The utility of P on a target area A is $U_P(A) = \int_A U_P(X)\ dX$.*

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 $\vec{v}$, it is counted only once towards the total utility since $1_P(\vec{v}) = 1$. In this case, these photos contain redundant information and their total utility is less than the sum of their individual utility.

III. 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 1. 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.

A. Two-Photo Case

Consider two photos $P = \{P_1, P_2\}$ in Fig. 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 1, 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 2 (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. 2(a), there exist two critical arcs that are symmetric with respect to line $\overline{P_1P_2}$*[1].

[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.

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. 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, 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. 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 III-C. Once it is calculated, the total utility can be given as $U_P(A) = U_P(R_1) + U_P(R_2) + U_P(R_3) + U_P(R_4) = 2\theta(a_1 + a_2 + 2a_3 + a_4) + \int_{R_4} \angle P_1ZP_2\ dZ$.

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 [14]. By tuning the number of line segments that replace an arc, we can control the tradeoff between accuracy and computational complexity.

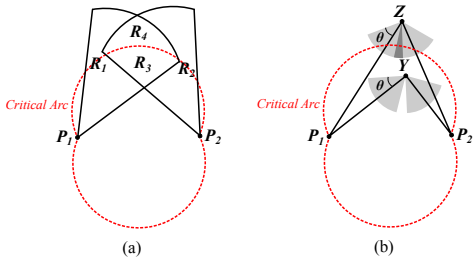
B. Multi-Photo Case

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

1) Coverage Area Partition: The coverage area of photos naturally partitions the target area into small, non-overlapping regions, as shown in Fig. 3(a). We have the following property after the partition.

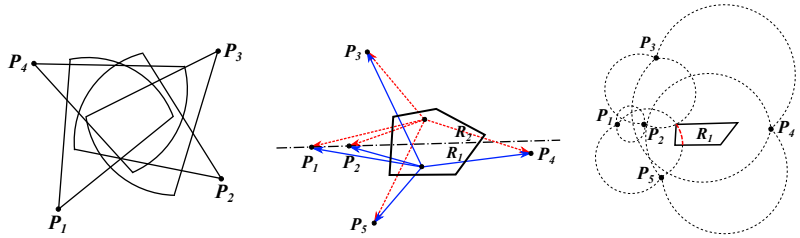
Property 1. *Given a region obtained from coverage area partition, all points in it are covered by the same set of photos.*

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



(a) (b)

Fig. 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.



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

Fig. 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.

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. 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 $\overline{P_2P_4}$.

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

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

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 $\overset{\frown}{P_1P_2}, \overset{\frown}{P_2P_3}, \ldots, \overset{\frown}{P_nP_1}$ 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. 3(b). We perform critical arc partition on R_1 in Fig. 3(c). R_1 is completely outside the critical arcs $\overset{\frown}{P_1P_2}$ and $\overset{\frown}{P_2P_3}$, 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 $\overset{\frown}{P_3P_4}$ and $\overset{\frown}{P_4P_5}$, which means for $\forall X \in R_1$, $\angle P_3XP_4 \geq 2\theta$ and $\angle P_4XP_5 \geq 2\theta$. $\overset{\frown}{P_5P_1}$ 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) 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 (1) 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 $U_P(R'') = \int_{R''} U_P(X)\ dX = 2\theta a|I_1| + \sum_{i\in I_2} \int_{R''} \angle P_iXP_{i+1}\ dX$, where a is the area of R''. We will discuss how to calculate integral $\int_{R''} \angle P_iXP_{i+1}\, dX$ in Section III-C. 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.

C. 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 [18]. 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.

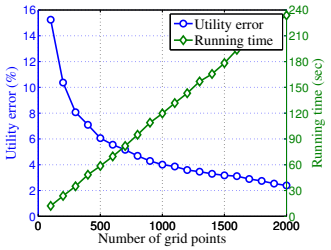


Fig. 4. Utility error and running time of the grid point method.

D. 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 1. 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. 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.

IV. 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 paper, we study one fundamental problem called the budgeted max-utility problem.

A. 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 3 (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.*

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 [8].

B. 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)

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

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 1. *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.* [11], 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 $\vec{v}$, we have $1_P(\vec{v}) \leq 1_Q(\vec{v})$ 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

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

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

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

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

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

(a) (b) (c)

Fig. 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.

V. 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)?

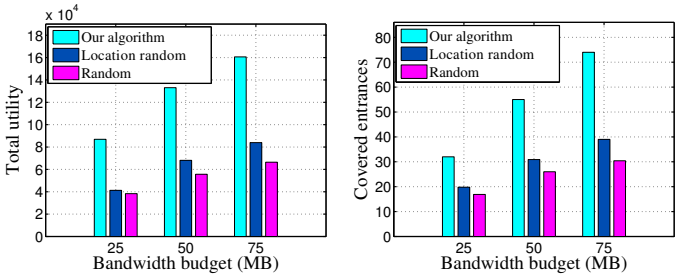
We answer questions 1) and 2) by using real-world experiments, and answer question 3) with simulations.

A. Real-World Experiments

1) Obtaining Metadata: We use a Nexus 4 phone running Android 5.1.1 to take photos, with metadata automatically obtained and recorded in the phone. Specifically, location is obtained from the GPS module, and orientation is calculated based on the information from the inertial measurement unit (IMU) [9], [10], [22]. The error of these two parameters will be discussed in Section VI. Field of view $\phi = 2\arctan(\frac{w}{2f})$, where w is the width of the image sensor and f is the focal length, both available from Android APIs [1]. In our experiment, all photos have the same field of view, $38.4°$. Finally, range is a little trickier to determine as it is affected by many practical factors such as focal length, camera quality, and the requirement of the application. Different applications may use different ranges depending on whether they need a close look at the covered objects or they are satisfied with photos taking far away from objects. In our experiment, we use 100m as a reference range.

2) Experimental Setup: As shown in Fig. 5(a), the target area (large pentagon) 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. 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.



(a) (b)

Fig. 6. Experiment results: (a) utility vs. bandwidth budget; (b) covered entrances vs. bandwidth budget.

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 total.

3) Results: Fig. 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 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. 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 III. 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.

B. Simulations

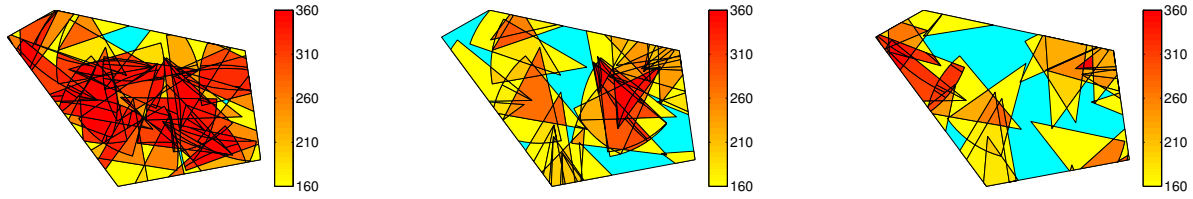
We generate photos with random locations and orientations in a 400m by 400m square, and set the 200m by 200m square in the center as the target area. The photos have 60° field of view and 100m range. The file size of each photo is generated randomly in $[2, 4]$MB. In each experiment, we select photos 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.

Fig. 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. 8(a), where $B = 100$MB and $\theta = 60^\circ$. 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.

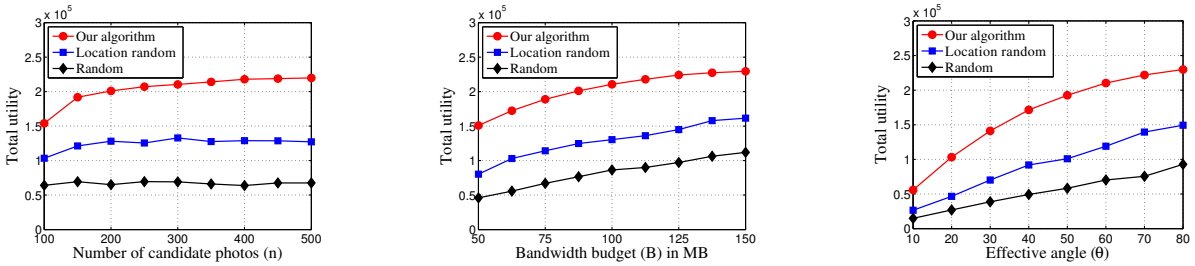
Fig. 8(b) shows the impact of the bandwidth budget (B) with $n = 300$ and $\theta = 60^\circ$. Again, diminishing returns can be observed for our algorithm, and a turning point can be found roughly at $B' \approx 125$MB. Note that in real applications, the turning point is not determined subjectively. 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

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



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

Fig. 7. Heat maps of the photos selected by the three algorithms. This figure is better viewed in color.



(a) (b) (c)

Fig. 8. Simulation results: (a) utility vs. number of candidate photos; (b) utility vs. bandwidth budget; (c) utility vs. effective angle.

the target area). Thus on average, every selected photo brings roughly the same increase in utility.

Fig. 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.

VI. Discussions

A. Location and Orientation Error

A 2011 study shows that the GPS module in smartphones has a median error of 5m-8.5m [26], which is reasonably small compared to the range of a photo or the size of a target area. As GPS technology keeps evolving and its accuracy keeps increasing [2], we expect the error to be even smaller for the newest mobile devices. Note that military GPS service can achieve much higher accuracy than civilian GPS service, so we already have the technique but not deployed due to various reasons. For orientation, an average error of 1.3°-3.4° is obtained for a method using all three IMU sensors (accelerometer, magnetometer, gyroscope) [22]. The authors also find that new devices with more advanced hardware and OS are more accurate. Hence, location and orientation error should not be an issue in our design.

B. Occlusion

Occlusion means that the view of an object is blocked by obstacles, though the object is considered to be covered based on the metadata. Since everything behind the obstacle is not visible in the photo, the photo's coverage area should be changed accordingly. Then the coverage area may no longer be a circular sector. It can be an arbitrary shape depending on the locations and shapes of obstacles.

Unfortunately, occlusion is difficult to detect unless the image itself is processed and understood, which requires a large amount of computational resources. A camera parameter, depth of field, may be used to detect occlusion if occlusion causes the object to be out of focus [22]. However, it cannot detect the case where objects and obstacles 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 notice that occlusion issue is orthogonal to the contributions of this paper. 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 is often visible from other directions.

C. 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) [12] 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.

VII. Related Work

Photo or video crowdsourcing has been increasingly popular due to its ability to provide a variety of useful applica-

tions, such as disaster recovery [27], 3D object modeling [6], floor plan reconstruction [5], [7], and many others [4], [17]. Although many studies use geographic location in their design, very few take advantage of camera orientation or other photo/video metadata to reduce the processing and analysis cost. FOCUS [9] uses orientation to cluster videos taken in a small area like a stadium. This helps organize crowdsourced videos but it is different from our problem, i.e., selecting photos to best cover a target area.

One closely related work is SmartPhoto [22], which studies photo utility and selection for target points. Compared to point coverage, area coverage is much harder to analyze because there are infinite number of points in an area, and each point can be covered by different photos from different angles. We propose a new utility metric suitable for area coverage and study how to analyze such coverage by effectively partitioning the area. Experiment results show that our method is much faster and more accurate than sampling the area with grid points. Moreover, we study the budgeted max-utility problem where each photo is associated with a resource cost, which is not considered in SmartPhoto. Also, the problem we consider is more generic, and our solution can be applied to any quantifiable resources in practice.

Another related research field is camera sensor networks. In [20] and [21], Wang *et al.* proposed the full-view coverage model which considers an object to be full-view covered if no matter which direction it faces, there is always a camera capturing its front image. 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 [23], [25] 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, our utility metric is more general because it measures the degree of coverage as a continuous function. More importantly, we conduct real-world experiments to demonstrate that our utility metric can truly reflect the coverage of real photos.

VIII. Conclusion

In this paper, we leveraged photo metadata to address the challenge of resource limitations in photo crowdsourcing applications. By collecting and analyzing metadata instead of real images, a significant amount of bandwidth, storage, and computational resources can be saved. Based on metadata, we defined photo utility to measure how well a target area is covered by a given set of photos. 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. We proposed various techniques to analyze such 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.

References

[1] Android developers. http://developer.android.com/index.html.
[2] Gps accuracy. http://www.gps.gov/systems/gps/performance/accuracy/.
[3] Street view. http://www.google.com/maps/views/streetview/.
[4] F. Chen, C. Zhang, F. Wang, and J. Liu. Crowdsourced live streaming over cloud. In *IEEE INFOCOM*, 2015.
[5] S. Chen, M. Li, K. Ren, and C. Qiao. Crowdmap: Accurate reconstruction of indoor floor plans from crowdsourced sensor-rich videos. In *IEEE ICDCS*, 2015.
[6] D. Crandall and N. Snavely. Modeling people and places with internet photo collections. *Commun. ACM*, 55(6), 2012.
[7] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li. Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In *ACM MobiCom*, 2014.
[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
[9] P. Jain, J. Manweiler, A. Acharya, and K. Beaty. Focus: Clustering crowdsourced videos by line-of-sight. In *ACM Sensys*, 2013.
[10] E. Koukoumidis, L.-S. Peh, and M. R. Martonosi. Signalguru: Leveraging mobile phones for collaborative traffic signal schedule advisory. In *ACM MobiSys*, 2011.
[11] A. Krause and C. Guestrin. A note on the budgeted maximization of submodular functions. Technical report, CMU-CALD-05-103, 2005.
[12] D. Lee and K. Plataniotis. Toward a no-reference image quality assessment using statistics of perceptual color descriptors. *IEEE Transactions on Image Processing*, 25(8), 2016.
[13] R. LiKamWa and L. Zhong. Starfish: Efficient concurrency support for computer vision applications. In *ACM MobiSys*, 2015.
[14] R. Nurnberg. Calculating the area and centroid of a polygon in 2d. www.ma.ic.ac.uk/~rn/centroid.pdf.
[15] D. Peng, F. Wu, and G. Chen. Pay as how well you do: A quality based incentive mechanism for crowdsensing. In *ACM MobiHoc*, 2015.
[16] C. Qin, X. Bao, R. Roy Choudhury, and S. Nelakuditi. Tagsense: A smartphone-based approach to automatic image tagging. In *ACM MobiSys*, 2011.
[17] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan. Scalable crowd-sourcing of video from mobile devices. In *ACM Mobisys*, 2013.
[18] A. Sommariva and M. Vianello. Product gauss cubature over polygons based on green's integration formula. *BIT Numerical Mathematics*, 47, 2007.
[19] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li. Deep learning for content-based image retrieval: A comprehensive study. In *ACM International Conference on Multimedia*, 2014.
[20] Y. Wang and G. Cao. Barrier coverage in camera sensor networks. In *ACM MobiHoc*, 2011.
[21] Y. Wang and G. Cao. On full-view coverage in camera sensor networks. In *IEEE INFOCOM*, 2011.
[22] Y. Wang, W. Hu, Y. Wu, and G. Cao. Smartphoto: A resource-aware crowdsourcing approach for image sensing with smartphones. In *ACM MobiHoc*, 2014.
[23] Y. Wu, Y. Wang, W. Hu, X. Zhang, and G. Cao. Resource-aware photo crowdsourcing through disruption tolerant networks. In *IEEE ICDCS*, 2016.
[24] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: Exploiting crowds for accurate real-time image search on mobile phones. In *ACM MobiSys*, 2010.
[25] Z. Yu, F. Yang, J. Teng, A. C. Champion, and D. Xuan. Local face-view barrier coverage in camera sensor networks. In *IEEE INFOCOM*, 2015.
[26] P. A. Zandbergen and S. J. Barbeau. Positional accuracy of assisted gps data from high-sensitivity gps-enabled mobile phones. *Journal of Navigation*, 64(3), 2011.
[27] Z. Zhai, T. Kijewski-Correa, D. Hachen, and G. Madey. Haiti earthquake photo tagging: Lessons on crowdsourcing in-depth image classifications. In *ICDIM*, 2012.