Geoinformatica
DOI 10.1007/s10707-012-0166-7

Users plan optimization for participatory urban texture documentation

Houtan Shirani-Mehr · Farnoush Banaei-Kashani · Cyrus Shahabi

Received: 4 May 2011 / Revised: 15 June 2012 /
Accepted: 17 July 2012

Abstract We envision participatory texture documentation (PTD) as a process in which a group of users (dedicated individuals and/or general public) with camera-equipped mobile phones participate in collaborative collection of urban texture information. PTD enables inexpensive, scalable and high quality urban texture documentation. We propose to implement PTD in two steps. At the first step, termed viewpoint selection, a minimum number of viewpoints in the urban environment are selected from which the texture of the entire urban environment (the part visible to cameras) with a desirable quality can be collected/captured. At the second step, called viewpoint assignment, the selected viewpoints are assigned to the participating users such that given a limited number of users with various constraints (e.g., restricted available time) users can collectively capture the maximum amount of texture information within a limited time interval. In this paper, we define each of these steps and prove that both are NP-hard problems. Accordingly, we propose efficient algorithms to implement the viewpoint selection and assignment problems.

This research has been funded in part by NSF grant CNS- 0831505 (CyberTrust), the NSF Integrated Media Systems Center (IMSC), the NSF Center for Embedded Networked Sensing (CENS), and in part by unrestricted cash and equipment gift from Google and Microsoft. 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.

H. Shirani-Mehr (✉) · F. Banaei-Kashani · C. Shahabi
Department of Computer Science, University of Southern California,
Los Angeles, CA 90089, USA
e-mail: hshirani@usc.edu

F. Banaei-Kashani
e-mail: banaeika@usc.edu

C. Shahabi
e-mail: shahabi@usc.edu

Published online: 11 August 2012

We study, profile and verify our proposed solutions comparatively by both rigorous analysis and extensive experiments.

Keywords Participatory data collection · Texture documentation · Sensor placement · Location-based services · Participation plan optimization · Optimization

1 Introduction

1.1 Motivation

The advent of earth visualization tools (e.g., Google Earth™, Microsoft Virtual Earth™) has inspired and enabled numerous applications. Some of these tools already include *texture* in their representation of the urban environment. The urban texture consists of the set of images/photos collected from the real environment, to be mapped on the façade of the 3D model of the environment (e.g., building and vegetation models) for photo-realistic 3D representation. Currently, urban texture is collected via aerial and/or ground photography (e.g., Google Street View). As a result, texture collection/documentation is (1) expensive, (2) unscalable (in terms of the required resources), and (3) with low temporal and/or spatial quality (i.e., texture cannot be collected frequently and widely enough).

These limitations can be addressed by leveraging the popularity of camera-equipped mobile devices (such as cell phones and PDAs) for inexpensive and scalable urban texture documentation with high spatiotemporal quality. With *participatory texture documentation*, termed PTD hereafter, a group of participants (dedicated individuals and/or general public) with camera-equipped mobile phones participate in collaborative/social collection of the urban texture information [1, 17]. By enabling low-cost, scalable, accurate, and real-time texture documentation, PTD empowers various applications such as eyewitness news broadcast, urban behavior analysis, real-estate monitoring, emergency-response, disaster management (e.g., for damage assessment in the case of earthquake, hurricane, and wildfire) and interactive visualization [12, 31]. Note that recently there have been various studies (e.g., [27]) on how to incentivize individuals to perform participatory tasks which can be leveraged to motivate individuals to participate in PTD.

PTD is implemented as a two-step process. At the first step, called *viewpoint selection*, a minimum size set of viewpoints in the urban environment is selected from which the texture information of the entire environment can be collected with a desirable quality. A viewpoint is an environment point from which one can collect texture information by imaging the surrounding area. Due to the participatory nature of PTD, available resources (e.g., users' participation time) are usually limited and, therefore it is critical to minimize the number of selected viewpoints. At the second step termed *viewpoint assignment*, the selected viewpoints are assigned to the users for texture collection. The viewpoints must be assigned such that the texture collected during the documentation campaign (i.e., the specific time interval allocated for texture documentation) is maximized while all users' constraints are satisfied.

1.2 Contributions

In this paper for the first time we introduce the problem of participatory texture documentation and model it as a two stage process: viewpoint selection and assignment. We empirically show that the current approaches to solve PTD are impractical because of high running time and therefore propose efficient heuristics for each of the steps.

We first define the viewpoint selection step and prove that it is an NP-hard problem by reduction from the minimum set cover problem [4]. Therefore, optimal solutions for the problem are rendered unscalable as the extent of the urban environment grows large; hence, we propose an efficient heuristic, termed *GVS*, with approximation guarantee to select the viewpoints. GVS solves a given instance of viewpoint selection problem by reduction to an instance of the minimum set cover problem. Based on our experiments, as compared to the naïve approach which selects the environment points by imposing a grid with the cell size of $c \times c$, GVS reduces the number of selected viewpoints by 43 % on average over different values of c.

Next, we formalize the viewpoint assignment problem and prove it is an NP-hard problem by reduction to the problem of *Team Orienteering* (different from orienteering) [5]. However, the running time of the heuristics proposed in the literature to solve the team orienteering problem is intolerably high, making them impractical solutions for the viewpoint assignment problem which requires on-the-fly assignment (considering the participatory nature of PTD). Second, we propose various heuristics for efficient viewpoint assignment with short running times. We categorized our proposed heuristics into two families of solutions: *individual-based* approaches and *group-based* approaches. With individual-based heuristics, we generate each user participation plan exclusively, independent of those of other users. Toward that end, we reduce the viewpoint assignment problem for a single user to the classical problem of orienteering [8], and accordingly adopt and extend the existing efficient heuristic solutions for the orienteering problem [6] to implement viewpoint assignment. Due to exclusive nature of individual-based, the participation plans generated by these heuristics may significantly deviate from optimal plans.

Alternatively, with our group-based heuristics we consider all users as a united group of participants. This allows for optimizing the assignment of the viewpoints among all users as a group; consequently, group-based heuristics can potentially generate near-optimal plans while maintaining high efficiency. In particular, group-based heuristics implement viewpoint assignment as a two-stage process. The main idea is to break the viewpoint assignment problem into multiple disjoint and smaller subproblems (at the first stage), where each subproblem takes a limited number of viewpoints as input and, therefore, can be solved efficiently (at the second stage). Accordingly, at the first stage group-based heuristics use various measures (e.g., proximity of the users to the viewpoints) to partition the set of viewpoints into a number of subsets, one subset per each user. At the second stage, similar to the individual-based heuristics, an orienteering heuristic is adopted to assign a subset of the viewpoints in each partition to the corresponding user of the partition. Based on our experiments, on average our individual-based approach takes 45 to 65 % more time than our proposed group-based approach for various urban environments. At

the same time, with our group-based approach we collect on average 8 to 10 % more texture in the same environment.

The rest of this paper is organized as follows. We give preliminaries in Section 3. In Section 4, we formally define the viewpoint selection and assignment problems. We describe our approaches to solve viewpoint selection and assignment problems in Sections 5 and 5.2, respectively. Section 6 presents the results of our empirical analysis of the proposed solutions. Finally, we discuss the related work in Section 2, and conclude in Section 7.

2 Related work

Texture mapping based on the images acquired by cameras is extensively studied in the literature [20, 38]. For example, in [38] a system is developed to generate texture for building exteriors from mosaics of close-range photographs acquired with commodity digital cameras. Although with our work we also use the images taken by users to generate texture information, our focus is on maximizing the amount of collected texture during the texture campaign time. Below we discuss the related work under two categories: viewpoint selection and viewpoint assignment.

2.1 Viewpoint selection

The closest body of relevant work to viewpoint selection is the literature on sensor deployment and sensing coverage with sensor networks. In [23, 28], the coverage problem is formulated as a decision problem to determine whether every point in the service area of a sensor network is covered by at least k sensors. With sensor deployment the goal is to maximize the coverage by proper sensor placement, similar to our viewpoint selection problem. However, most of the proposed approaches for sensor deployment assume simple sensing models with circular (omnidirectional or unidirectional) coverage for sensors [11, 19, 41]. Recently, Andreas et al. [24, 25] formulated the problem of sensor placement as the placement of k sensors in the environment to minimize the uncertainty about the sensing phenomena at unsensed locations. To this end, each environment point is associated with a random variable and consequently a probabilistic model such as Gaussian is used to model a joint probability distribution over a subset of environment locations A to qualify amount of information which can be collected from A. While these models properly approximate the coverage of the sensors with typical sensing modalities (e.g., sound and temperature sensors), visibility coverage is more complex; hence, rendering these approaches inapplicable for visual sensor deployment.

The most relevant work to our work is on visual sensor deployment [22, 26, 29]. The art gallery problem is a classic work in this category [26]. The most important difference with this category is that we also modeled the resolution of the texture and considered it as a constraint in texture documentation process. Furthermore, we consider spatial restrictions in viewpoint selection (namely, with our problem the selected points must be on a road network). Finally, while the previous work has focused on solving the visual sensor deployment problem in a continuous space, with

GVS we assume a discrete space for viewpoint selection to model restrictions on where users can take images.

2.2 Viewpoint assignment

We use orienteering algorithm as a building block to devise our viewpoint assignment approaches. Orienteering is a variant of traveling salesman problem with profits. The algorithms to solve orienteering fall into three categories: the exact algorithms, e.g. [14], heuristics with approximation guarantee, e.g. [7, 9], and heuristics without approximation guarantee, e.g. [39]. The first two categories are not applicable for the problems of large scale because of their large running times. Team orienteering is a variant of orienteering for multiple users [5, 39]. Similar to viewpoint assignment, orienteering is used to model the coordination of participants in data acquisition, assuming each participant has a capacity constraint. Amarjeet et al. [34, 35] studied coordination of multiple robots for environment sensing, where each robot has a resource constrain. Unfortunately, these approaches are not scalable for our texture documentation problem. For example, for less than 220 environment locations it can take more than 1,000 s to generate a robot path to collect information. In [42], the authors partition the sensing field so that the total obtained reward in each partition be the same and solved an instance of orienteering in each partition. In this paper, we proposed various heuristics to solve the viewpoint assignment problem by reduction to the orienteering problem and its variations. We also proposed a novel two-stage assignment and proved that a two-stage solution always exists. Furthermore, we experimentally verified the efficiency of our two-stage approach for large scale problems with more than 2,500 environment points, which is far larger than the problems addressed by the previous work. Based on our experiments, our two-stage assignment significantly outperforms both individual-based assignment and team orienteering assignment. Also in contrast to the previous studies, we consider users who are joining over time with different starting points, ending points and available times.

In [1], we have proposed the system design and architecture of our prototype participatory texture documentation system, GeoSIM [17]. While both this paper and [1] concentrate on the same problem, in this paper we tackle the research challenges on how to implement viewpoint selection and assignment experimentally and theoretically, while [1] presents the PTD system implementation. We have presented our preliminary solutions for viewpoint selection and viewpoint assignment in [33] and [32], respectively. This paper subsumes the previous papers by considering a new approach for environment modeling as well as considering the resolution of texture in both selecting and assigning viewpoints. We also improved the performance of our two-stage assignment by modifying our partitioning stage. In [32, 33], we proposed to model an environment with TIN model for texture documentation. Modeling an urban environment consisting of planar surfaces with TIN can become very complex. However, our proposed technique here avoids this complexity for planar surfaces. At the same time it has the benefits of TIN in modeling non-planar surfaces as for those surfaces we first model the environment with TIN. Finally, we conducted new extensive experiments to consider the effect of resolution on viewpoint selection and added a new dataset to study the behavior of our viewpoint assignment more thoroughly.

3 Preliminaries

Here, we first describe the texture documentation process. Next, we define our assumed user participation model for PTD.

3.1 Texture documentation

Consider an urban environment which consists of various 3D elements such as buildings, trees and the terrain. There are various work in literature on building 3D models of an environment, e.g., . We represent each surface of a 3D element as a set of *cells*. Without loss of generality, we assume all the cells have the same surface area a_c. We select a_c sufficiently small such that each cell can be considered as a planar surface. The *texture* of an environment is defined as the set of images mapped on the surface cells of the 3D elements of the environment, where the quality of the texture depends on the resolution of the mapped images. Next, we first describe how to construct cells for various types of surfaces. Second, we describe how to quantify the resolution of an image mapped onto a cell. Finally, we formally define the texture documentation process.

3.1.1 Surface model

The surfaces of a 3D element are either planar (such as planar building walls) or non-planar (such as the terrain). With planar surfaces, we impose a grid with the cell area of a_c and consider each cell of the grid as a surface cell (Fig. 1). On the other hand, for a non-planar surface we first convert the surface into a set of planar subsurfaces. To this end, we model a non-planar surface by TIN [16] (Triangulated Irregular Network) (Fig. 2). The TIN model represents a surface as a set of contiguous, non-overlapping triangles. Thereafter, similar to planar surfaces, we impose a grid with the cell area of a_c on each triangle. The collection of grid cells over all the triangles constitute the non-planar surface cells. Given a triangle, some of the cells closer to the TIN triangle sides may not be completely contained within the triangle. For

Fig. 1 A building with the cells depicted on one of its walls pointed by an *arrow*

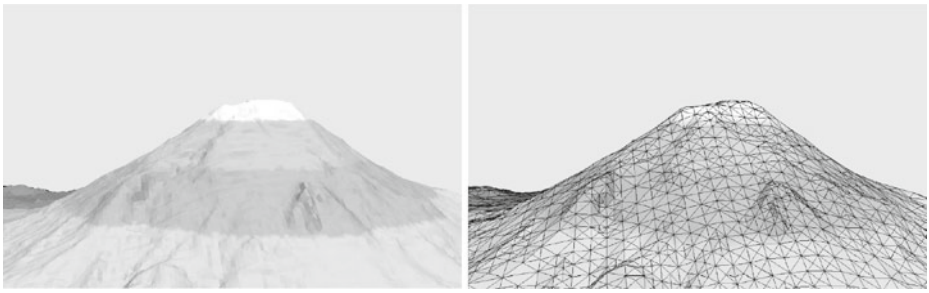


Fig. 2 **a** The environment. **b** The corresponding 3D TIN model [13]

those cells, the texture is only generated for the part of the cell overlapping with the triangle.

3.1.2 Texture resolution

The quality of the environment texture depends on the resolution of the images mapped onto the environment cells. In particular, the quality of texture for a cell is proportional to the area of the cell represented by each pixel of the image mapped on the cell, and the higher the resolution of the image, the smaller the area covered by each pixel [18]. To compute the cell area represented by each pixel of the image, we define our assumed model for the imaging process as follows, with which we consider parameters such as distance of the camera and its view angle to the cell. Imagine a small light-proof box with a small hole in one side. When the pinhole faces a light source such as a candle, one can see the inverted image of the candle on the opposite side of the box. The image is formed by light rays disseminated from the scene facing the box (Fig. 3). This imaging model is called the pinhole model [15] and despite its simplicity often represents the imaging process fairly accurately. The image of each cell c consists of a set of pixels P_c (Fig. 4). With a pinhole model, the region on c which is represented by each pixel $p \in P_c$, is in the form of a four sided polygon. The area of this polygon depends on the parameters such as the distance of the pixel to the cell, the distance of the camera and the view angel to the cell. We denote this set of parameters by S_c. Given an image of c, we define the texture

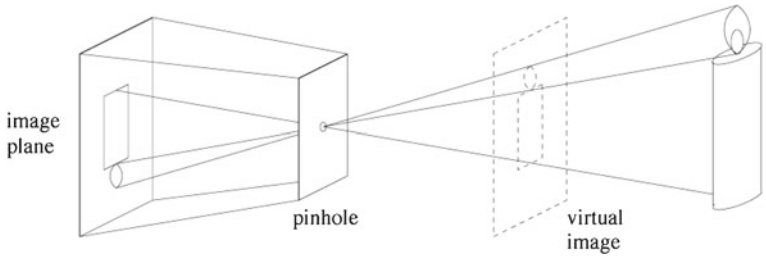


Fig. 3 The pinhole imaging model

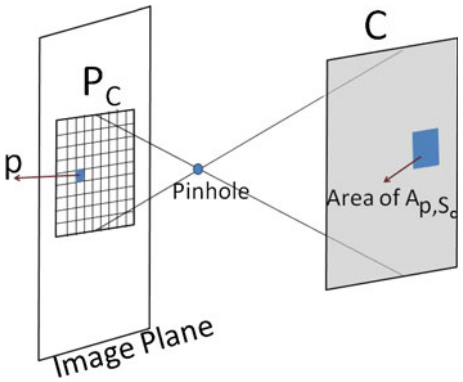


Fig. 4 A cell c and a set of pixels P_c which represents c. A pixel $p \in P_c$ represents an area of A_{p,S_c} on c

quality of c, denoted by Q_{S_c}, as the inverse of the largest area represented by the pixels in P_c. More formally, suppose the area of c represented by a pixel $p \in P_c$ is denoted by A_{p,S_c} (Fig. 4). We show in Appendix how to calculate A_{p,S_c}. We define $\max_{A_{S_c}} = \{A_{p',S_c} | p' \in P_c, \forall p \in P_c, A_{p,S_c} \leq A_{p',S_c}\}$. Consequently, we have

$$Q_{S_c} = \frac{1}{\max_{A_{S_c}}}. \tag{1}$$

3.1.3 Texture documentation process

An urban texture documentation campaign is defined as the process of collecting and mapping the texture onto the environment cells during a predefined time interval T_C (e.g., 10:00am to 2:00pm on a particular day). We assume the urban texture remains unchanged during T_C. The texture collection must be done such that the texture quality for each cell is higher than a desired minimum quality of Q_T. Suppose P is the set of points in the environment. From each point $v \in P$ one can collect texture information by imaging the surrounding area. We call each such point v a viewpoint. Accordingly, we define the texture score $TS(v)$ of a viewpoint v as the total number of environment cells visible from v with the texture quality greater than Q_T. Similarly, the texture score $TS(V)$ of a set of viewpoints $V \subseteq P$ is defined as the total number of cells visible from any viewpoint in V with a texture quality higher than Q_T.

3.2 Participatory texture documentation

With participatory texture documentation, the texture collection process is implemented by a set of users U. Users are presumed to carry similar camera-equipped cell phones. We assume each user $u \in U$ has a set of participation constraints denoted by $c = (s,d,A_T)$, where s is the user's starting point in the environment, d is the user's desired ending point (where the user intends to leave the documentation campaign), and A_T is the user's maximum available time for participation. Accordingly, the participation plan (or participation path) for a user u is define as a path $P_u = (s, v_1, v_2, \ldots, v_n, d)$ that starts from the starting point s and ends at the ending point d, while traversing a number of viewpoints v_1 to v_n, at which the user is expected to make stops for texture collection. We assume a user takes a panoramic image at a

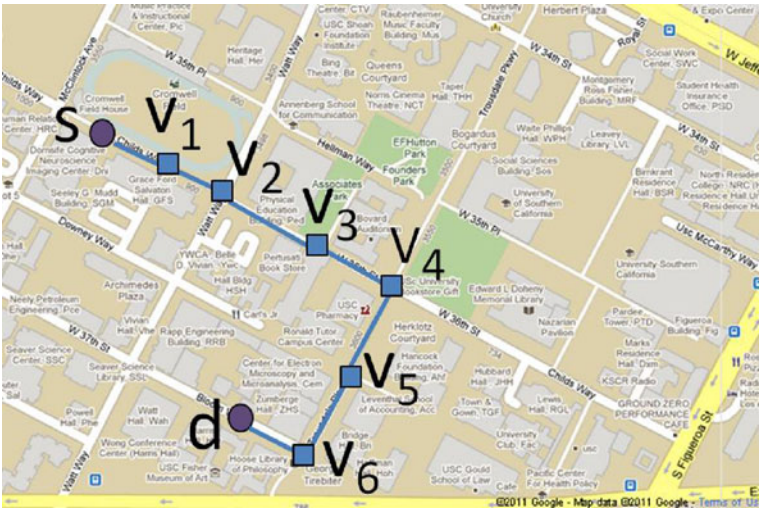


Fig. 5 An example participation path where squares are representing viewpoints and the circles are starting and ending points

viewpoint v. This ensures that the user can collect the texture for a cell c from the best view angle which maximizes the texture quality of c. In other words, among all possible view angles at v, S_c consists of a view angle which maximizes Q_{S_c} in Eq. 1. We denote this texture quality by $Q(c, v)$. Figure 5 shows a sample participation path $P_u = (s, v_1, v_2, v_3, v_4, v_5, v_6, d)$ for a user u (not shown in the figure). In this case, the sample participation path traverses the viewpoints v_1, v_2, v_3, v_4, v_5 and v_6. Furthermore, the participation path P_u for user u is said to satisfy the user constrains c if and only if the total time to traverse the participation path (i.e., the actual user participation time) is less than the user available time A_T:

$$t_p + n \times t_{v_i} \leq A_T$$

where t_p is the total time required to traverse the subpaths between successive viewpoints (assuming shortest path), and t_{v_i} is the time it takes to collect images at each viewpoint v_i along the path P_u. Finally, the texture score $TS(P_u)$ of the path P_u is defined to be equal to the texture score $TS(V_{P_u})$, where V_{P_u} is the set of viewpoints v_1 to v_n covered by P_u. Similarly, the texture score $TS(P_U)$ of a set of paths P_U is defined to be equal to the texture score $TS(V_{P_U})$, where V_{P_U} is the set of viewpoints that are covered by at least one path in P_U.

4 Problem definition

In this section, we define the viewpoint selection and assignment problems.

4.1 Viewpoint selection problem

Suppose C is the set of cells that comprise the $3D$ model of the target environment. Consider $C' \subseteq C$ as the subset of cells that are visible from at least one viewpoint in

P (note that given a finite set of viewpoints P , there might be a non-empty set of cells $C \backslash C'$ that cannot be texture mapped, regardless). Accordingly, we call a set of viewpoints $S \subseteq P$ a texture covering set, if every cell in C' is visible from at least one viewpoint in S with the quality of at least Q_T. The viewpoint selection problem is defined as the process of finding a *texture covering* set V with minimum size among all texture covering subsets of P.

4.2 Viewpoint assignment problem

Once the texture covering set V is identified, the viewpoints $v \in V$ must be assigned to the participating users $u \in U$ (by including viewpoints in their participation plan) such that the total number of the cells $c \in C'$ covered by users within the campaign time T_C is maximized. Formally, the problem of viewpoint assignment is defined as an optimization problem to find the set of participation plans $P_U = \{P_{u_1}, P_{u_2}, \ldots, P_{u_m}\}$ corresponding to the users $u_1, u_2, \ldots, u_m$ in U such that $TS(P_U)$ is maximized while each P_{u_i} satisfies the corresponding user constraints c_i, i.e., $\text{argmax}_{P_U} TS(P_U)$. With PTD, we assume users can join the texture documentation campaign progressively (not necessarily simultaneously), with a poisson arrival distribution. Accordingly, we generalize the definition of the viewpoint assignment problem by considering an iterative viewpoint assignment scheme. With this scheme, the campaign time T_C is divided into equi-length epochs, I_1 to I_l, and viewpoint assignment is repeated at each epoch to assign the remaining uncovered viewpoints (those viewpoints that are not covered at previous epochs) to the users who arrive within the current epoch I_i. With iterative viewpoint assignment, the optimization problem defined above is generalized and modified to $\text{argmax}_{P_{U_i}} TS(P_{U_i})$, where $U_i \subseteq U$ is the subset of users arriving during the ith epoch I_i. Throughout the paper we call the available time of the user, excluding the duration of time she needs to wait until the end of I_i, by user *participation time*. Consequently, we represent the constraints of a user by $c = (s, d, T)$ where T is her participation time. Note that T does not only depend on user availability but also on system properties, i.e., $|I_i|$. However, for ease of notation we assume T represents the user constraint on time.

5 Approaches

Both viewpoint selection and assignment are NP-hard problems which makes the optimal algorithms impractical. The proof sketch of the complexity results can be found in [32] and [33]. Next, we explain our approaches to solve viewpoint selection and assignment.

5.1 Viewpoint selection

Our proposed heuristic algorithm for viewpoint selection, termed GVS, is based on reduction to the *minimum set cover* problem. We first state the minimum set cover problem.

Definition 1 Let $S = \{s_1, s_2, \ldots, s_m\}$ be a collection of finite sets, s_i's , whose elements are drawn from a universal set U. Let $F = \bigcup_{i=1}^{m} s_i$ where $F \subseteq U$. Minimum set cover finds a set O with minimum cardinality where $O \subseteq S$ and $\bigcup_{s \in O} s = F$.

For example, assume $U = \{1, 2, 3, 4, 5, 6\}$ and $S = \{s_1, s_2, s_3\}$, where $s_1 = \{1, 2\}$, $s_2 = \{2\}$ and $s_3 = \{1, 3\}$. Thus, $F = \{1, 2, 3\}$ and the minimum set cover is $O = \{s_1, s_3\}$.

Such a reduction enables us to adapt the existing algorithms for minimum set cover problem to solve the viewpoint selection problem. Here, we first explain our proposed reduction, and thereafter describe our *GVS* algorithm.

5.1.1 Reduction

For each point $p \in P$, we define the *visibility* set $vs(p)$ as the set of cells in C' that are visible from p and one can collect texture for them with the resolution of at least Q_T when standing at p; i.e., $vs(p) = \{c \in C' | V(p, c) = 1 \ , \ Q(c, p) \geq Q_T\}$, where $V(p, c) = 1$ if c is visible from p. For example with $vs(p) = \{c_1, c_2, c_3\}$ the cells c_1, c_2 and c_3 are visible from p. Therefore, a user standing at p can collect texture for these cells with the quality of at least Q_T. We construct the corresponding instance of the minimum set cover problem as follows. We define $F = \bigcup_{p \in P} vs(p)$ and $S = \{vs(p) | p \in P\}$. The universal set U can be any set such that $F \subseteq U$. For example, for an instance of viewpoint selection problem in which $P = \{p_1, p_2\}$, $vs(p_1) = \{c_1, c_2\}$, and $vs(p_2) = \{c_3\}$, the corresponding set cover instance has $F = \{c_1, c_2, c_3\}$, $S = \{vs(p_1), vs(p_2)\}$ and $U = F$. If the answer to the constructed minimum set cover instance is O, then the answer to the original viewpoint selection can be derived as V, where $p_i \in V$ if and only if $vs(p_i) \in O$.

Consequently, we can use any of the heuristics for minimum set cover to solve viewpoint selection problem. To develop our viewpoint selection algorithm (see Section 5.1.2), we are inspired by the greedy minimum set cover algorithm from [10], which has a linear running time of $O\left(\sum_{i=1}^{m} s_i\right)$ where $s_i \in S$. The greedy minimum set cover algorithm works by iteratively selecting the set $s_i \in S$ that covers the greatest number of remaining uncovered elements of F. This algorithm is guaranteed to result in a sub-optimal answer with an approximation guarantee of $\ln(n) + 1$, where n is the cardinality of the set $s_j \in S$ with the largest number of elements.

5.1.2 Greedy Viewpoint Selection (GVS)

Based on the reduction in Section 5.1.1, the initial step with the viewpoint selection algorithm is to find the visibility set for each point $p \in P$. This requires performing expensive line-of-sight queries originating from p towards every cell $c \in C'$. We can avoid performing some of these queries based on the fact that for some cells it is not possible to collect texture with the minimum texture quality of Q_T if one stands at p because of the distance from p (refer to Section 4). We denote these cells by $V''(p)$. It is not possible to collect texture with the minimum quality of Q_T for any cell $c \in V''(p)$ when standing at p. Therefore, we only need to perform line-of-sight queries originating from p towards the cells in $C' \backslash V''(p)$. We represent the such cells by $V'(p)$, i.e., $V'(p) = C' \backslash V''(p)$. To find the cells in $V'(p)$, we first find a distance of $d_{\max}$ such that regardless of view angle, any cell further than $d_{\max}$ from p cannot be texture mapped with the quality of at least Q_T. In other words,

$$d_{\max} = \underset{d_{(c,p)}}{\operatorname{argmax}} \{Q(c, p) \geq Q_T\}, \tag{2}$$

where $d_{(c,p)}$ is the distance between cell c and point p. It is easy to derive $d_{\max}$ based on Eq. 6 (refer to Appendix). We can perform a range query with the radius of $d_{\max}$ centered at p to find a set $V'(p)$.

The pseudocode of our viewpoint selection algorithm, termed Greedy Viewpoint Selection (GVS), is presented in Algorithm 1. The algorithm takes as input the cells in $C' \subseteq C$ and the set of the environment points P. We explain the logic of the algorithm as follows. First, GVS computes the visibility set of each point $p_i \in P$ (lines 4–10) after finding the cells in $V'(p_i)$ for each $p_i \in P$ (line 3). Thereafter, with a greedy approach the viewpoint p with maximum visibility set size is iteratively selected among all points in P, removed from P, and added to the result set V (lines 11–16). Note that once p is added to V the corresponding cells visible to p are excluded from C' (because they are covered); consequently, the visibility set of V (and P) is updated at each iteration. The iteration terminates when V becomes a texture covering set.

To prove the correctness of GVS, note that all the cells in C' can be texture mapped by at least one point in P. During each iteration, a point is selected and all the cells visible to it are excluded. Therefore, the number of remaining cells which cannot be texture mapped from the already selected viewpoints decreases, and correspondingly $TS(V)$ increases until the iteration terminates. The returned set V is the approximate answer to the viewpoint selection problem as it may have larger cardinality than the optimal viewpoint selection answer. Assuming that the optimal answer is V_{opt}, the size of the answer returned by GVS satisfies the following inequality:

$$\frac{|V|}{|V_{\text{opt}}|} \leq \ln(VS_{\max}) + 1$$

where $VS_{\max}$ is the size of the visibility set of the point with the largest visibility set among all points in P. The above inequality guarantees that the size of V is at most $\ln(VS_{\max}) + 1$ times larger than the size of the optimal answer. This bound follows from the approximation guarantee of the greedy minimum set cover algorithm [4].

Algorithm 1 $GVS(C', P)$

1: $V = \emptyset$;
2: $vs = \emptyset$;
3: Create $V'(p)$ for every $p \in P$
4: **for** each point $p_i \in P$ **do**
5: **for** each cell $c_j \in V'(p_i)$ **do**
6: **if** ($V(p_i, c_j)$=1 and $Q(c_j, p_i) \geq Q_T$) **then**
7: $vs(p_i) = vs(p_i) \cup c_j$; {Visibility set of each point is set}
8: **end if**
9: **end for**
10: **end for**
11: CollectableTexture $= |C'|$;
12: **while** $TS(V) <$ CollectableTexture **do**
13: $p = \{p \in P | \forall p' \in P, |VS(p)| \geq |VS(p')|\}$;
14: $V = V \cup p$; {p is added to the answer set}
15: $C' = C' - vs(p)$; {C' is updated}
16: **end while**
17: **return** V;

5.2 Viewpoint assignment

The viewpoint assignment problem is NP-hard and therefore it is not practical to execute optimal algorithms to assign the viewpoints. We categorize viewpoint assignment heuristics into two categories: individual-based assignment and group-based assignment and describe each category in this section.

5.2.1 Individual-based approach

In this section we study individual-based approach. We first solve the SVA (Single user Viewpoint Assignment) problem which is a special instance of viewpoint assignment in which only one user joins PTD. We solve SVA by reduction to the orienteering problem. Thereafter, we propose the complete individual-based approach which solves an instance of SVA per user. We prove that if SVA is solved based on an orienteering algorithm with approximation guarantee on the collected reward, individual-based approach can also guarantee an approximation error on the collected texture.

SVA heuristic Here we explain how to solve an instance of SVA by reduction to (not from) orienteering [8] problem which is an NP-hard problem. Orienteering is a variant of traveling salesman problem and can be formalized as follows:

Definition 2 (Orienteering [8]) The input to an orienteering problem is given by a directed weighted graph $G = (V, E, W)$, two vertices $v_s, v_d \in V$ not necessarily distinct, and a budget $B > 0$. The goal is to find a path P from v_s to v_d of length at most B such that the collected reward is maximized. The reward is determined by applying a reward function on the set $V(P)$ of vertices visited by the path P. The reward function $f{:}2^V \rightarrow \mathbb{Z}^+$ assigns a non-negative integer reward to any subset of graph vertices.

Given an instance of SVA we describe how to construct the corresponding orienteering instance. To construct the orienteering graph G (see Definition 2), for each viewpoint vp_i we introduce a graph vertex $v_i \in G$. We introduce an edge from $v_i \in G$ to $v_j \in G$ with the weight w_{ij} which equals to the shortest urban environment path from vp_i to vp_j. If such a path does not exist then $w_{ij} = \infty$. During the orienteering graph construction, we assume user starting and ending are two *virtual* viewpoints (if they are not among the viewpoints) which comprise the starting and ending vertices of G, i.e., v_s and v_d accordingly. A virtual viewpoint v is a viewpoint which is not obtained from viewpoint selection step and therefore no texture is collected from v in PTD. We can set the budget based on user participation time and her movement speed. Finally, we need to specify the reward function for the orienteering instance. Assume the reward function is denoted by $f(V)$ where V is the set of vertices. We define the reward function as follows:

$$f(V) = TS(VP), VP = \{vp_i | v_i \in G\}.$$

This reward function satisfies the following two conditions:

- f is monotone, i.e., $f(A) \leq f(B)$ for all $A \subseteq B$.
- f is sub-modular, i.e., for all $A, B \subset V$ and $u \in V$, $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$ whenever $A \subseteq B$.

The reward function is monotone since taking images from more viewpoints will not result in collection of less texture. It is also sub-modular because by taking images from more viewpoints the texture which can be collected from a new viewpoint will not increase. As the reward function satisfies these two properties, orienteering algorithms with approximation guarantee can be applied to solve the constructed orienteering instance [8]. The solution to the constructed orienteering instance can be easily converted to the solution of SVA.

Algorithm The individual-based approach for viewpoint assignment is presented in Algorithm 2. The input to the algorithm are the sets U_i (the users joined during I_i), V (unassigned viewpoints) and C' (cells not already texture mapped). Assume U_i is an ordered list, i.e., $U_i = (u_1, u_2, \ldots, u_m)$ (we later discuss how to decide the order of these users). A user participation plan P_{u_j} is generated by solving an instance of SVA for each user u_j starting from u_1. Assume the set of cells in C' and in the visibility set of any viewpoint in P_{u_j} is C'_j. To prevent double count of the collected texture we assume all the cells in C'_j are already texture mapped when generating the participation plans for next users, i.e., for a user u_k where $k > j$. The reason is that the cells in C'_j will be texture mapped by u_j when she takes P_{u_j}. This update is done at line 4 and by the *Update* method which removes the viewpoints in P_{u_j} from V as well.

Algorithm 2 Individual-based(U_i, V, C')

```
1: P_{U_i}=∅                          {Initializing the output.}
2: for j=1 to m do
3:     P_{u_j}=SVA(u_j, V);           {Solving SVA for each u_j ∈ U_i.}
4:     Update(V, C', P_{u_j})
5:     P_{U_i}=P_{U_i} ∪ P_{u_j};
6: end for
7: return P_{U_i};
```

Consider an orienteering algorithm O with the approximation ratio of η, i.e., η is the ratio of the total reward collected by the optimal orienteering solution to the one collected by O. With O the value of η depends on the budget B. Assume SVA in Algorithm 2 is solved by reduction to O. The following theorem proves the approximation ratio of viewpoint assignment over T_C.

Theorem 1 *The approximation ratio of individual-based approach for users joined during T_C is*

$$\frac{TS(P^*)}{TS(P)} = 1 + \eta_{\max}, \tag{3}$$

where P is the set of users participation plans obtained by applying individual-based assignment for all the user joined during T_C and P^* is the set of optimal participation plans. $\eta_{\max}$ is the maximum approximation ratio of O for all the users joined during T_C.

The proof of the theorem is an extension of the multi-path orienteering approximation ratio proof proposed in [2].

Before running Individual-based algorithm, we need to order the users in U_i. Intuitively, sorting the users in ascending order of their participation times results in the collection of more texture score than the other orderings. The reason is if the participation plan of a user with a large participation time being generated first, the user collects texture from lots of viewpoints. Among these viewpoints, there may be a huge number of viewpoints which are the only choices for the users with small participation times to gather texture from. In Section 6, we experimentally verify this intuition.

5.2.2 Group-based assignment

In this section we propose a two-stage approach to implement group-based assignment. In the first stage, partitioning stage, we create a sub-problem for each user which includes a subset of unassigned viewpoints. We call the viewpoints within a sub-problem a *partition*. The idea of this stage is to break the viewpoint assignment problem into multiple disjoint sub-problems quickly where each subproblem has a few number of viewpoints and therefore is easy to solve. In the second stage and to create participation plans for users, we solve an instance of SVA for each of the partitions generated in the first stage. In this section, we first formally define each of these two stages and prove the existence of two-stage assignment. Thereafter, we propose efficient heuristics for the partitioning stage. The second stage solves an instance of SVA for each partition and the heuristics to solve SVA are already discussed in Section 5.2.1. Finally, we propose the complete algorithm for two-stage assignment.

To form a partition corresponding to user u, denoted by $part_u$, we first categorize the viewpoints based on the constraints of u. For each user we can identify three different categories of viewpoints. The first category includes the ones from which u cannot collect texture from. These viewpoints are infeasible for u to collect texture from and we represent them as $I(u)$ and call them *infeasible* viewpoints for u. If the time needed to reach to viewpoint v from s, collecting texture from v and going to d be more than T, then u cannot collect texture from v. The other category of viewpoints includes those which only u can gather texture from. We denote this set of viewpoints as $D(u)$ and term them *dedicated* viewpoints to u. The last category of viewpoints are those which in addition to u other users can collect texture from as well. We denote this set of viewpoints as $S(u)$ and term them *shared* viewpoints between users. For example, consider two users u_1 and u_2 having the constraints of $c_1 = (s_1, d_1, T_1)$ and $c_2 = (s_2, d_2, T_2)$, accordingly. We denote the time it takes to go from x to y as $t(x, y)$ and assume that the time required to gather texture from a viewpoint v is negligible. If $T_1 = 25$ and $t(s_1, v) = 30$ then $v \in I(u_1)$. Presume $T_1 = 30$, $T_2 = 30$, $t(s_1, v) = 10$, $t(v, d_1) = 10$, $t(s_2, v) = 10$ and $t(v, d_2) = 10$. In this case, $v \in S(u_1)$ and $v \in S(u_2)$ as both u_1 and u_2 can collect texture from v. In the same setting if we change $t(v, d_2) = 30$ then $v \in D(u_1)$ as only u_1 can gather texture from v now.

The method for categorizing the viewpoints is presented in Algorithm 3. It takes as input the set of unassigned viewpoints V and users U_i (including their constraints). For each viewpoint v it iterates over the users and checks whether a user u_j can collect texture from v at line 3. We represent the constraints of u_j as $c_j = (s_j, d_j, T_j)$

and the time required to collect texture from v as t_v (t_v is the same for all the viewpoints). If u_j can collect texture from v, u_j is added to a set called $CG(v)$ which includes the users who can collect texture from v. Thereafter, the algorithm iterates over the viewpoints and for each viewpoint v checks the cardinality of $CG(v)$ and based on that it updates dedicated and shared viewpoints.

Algorithm 3 Categorization(V, U_i)

```
for each v ∈ V do
    for each u_j ∈ U_i do
        if t(s_j, v) + t_v + t(v, d_j) > T_j then
            I(u_j) ← I(u_j) ⋃ v;
        else
            CG(v) ← CG(v) ⋃ u_j;
        end if
    end for
end for
for each v ∈ V do
    if |CG(v)| == 1 then
        D(u_j) ← D(u_j) ⋃ v (where u_j ∈ CG(v))
    else
        S(u_j) ← S(u_j) ⋃ v (where u_j ∈ CG(v))
    end if
end for
```

Clearly, all the viewpoints in $D(u)$ are in $part_u$. Any viewpoint in $I(u)$ is not in $part_u$ and a subset of viewpoints in $S(u)$ are in $part_u$. As the subproblems and hence the partitions should be disjoint, we need to assign a shared viewpoint to only one partition. We now formally define a user partition. The partition for a user $u \in U_i$ is defined as follows:

$$part_u = S_1 \cup S_2 \text{ where } S_1 = D(u) \text{ and } S_2 \subseteq S(u).$$

Two partitions should be disjoint as well:

$$\forall i \neq j,\ part_{u_i} \cap part_{u_j} = \emptyset. \tag{4}$$

In the second stage of two-stage assignment, we solve an instance of SVA for each sub-problem, i.e., for each $part_u$. The set of SVA solutions for all the partitions forms the solution of viewpoint assignment problem. Assume the set of participation plans for users obtained by solving viewpoint assignment optimally to be $P^* = \{p^*_{u_1}, p^*_{u_2}, \ldots, p^*_{u_n}\}$, where $p^*_{u_j}$ is the optimal participation plan generated for user $u_j \in U_i$. The optimal two stage approach is the one which gives a set of participation plans $P = \{p_{u_1}, p_{u_2}, \ldots, p_{u_n}\}$, p_{u_j} is the generated participation plan for $u_j \in U_i$, where $TS(P) = TS(P^*)$. In [32] we proved that two-stage assignment exists. In the reminder of this section, we propose our partitioning approaches and finally we present our complete two-stage viewpoint assignment algorithm.

At the beginning of the partitioning stage all the viewpoints dedicated to particular individuals are assigned to the corresponding partitions. We also add the starting and ending viewpoints (which can be virtual viewpoints as discussed in Section 5.2.1) to the corresponding partitions. This may lead to overlapping between partitions which we ignore as the number of overlapping starting and ending viewpoints is very small as compared to the total number of viewpoints. Notice that such an overlap does not result in double count of collected texture as in the second stage and after

solving an instance of SVA per user, each viewpoint is assigned to only one user participation plan. The main challenge in the partitioning stage is to assign the shared viewpoints. This problem can be viewed as a classification problem. Classification is the task of assigning objects to one of several predefined categories [37]. In our problem, a predefined category is a user partition $part_{u_i}$ including the starting and ending viewpoints for u_i and all the viewpoints in $D(u_i)$. Notice that during assigning shared viewpoints, we assume user starting and ending viewpoints are also dedicated viewpoints.

We propose here two heuristic approaches to assign the shared viewpoints. With the first approach, termed the distance-based approach, we define a distance measure between a viewpoint and a partition and assign a viewpoint to its nearest partition. With the second approach, termed the nearest-neighbor based approach, we decide on the partition for a viewpoint v based on the partition of the nearest-neighbor of v among the already assigned viewpoints. In [32] we considered all the viewpoints in a partition in order to assign a shared viewpoint. This approach may lead to inefficient assignment as it is sensitive to the order of assigning the shared viewpoints. For example, assume all the shared viewpoints except a viewpoint v have distance of at most d to all users but v has the distance of $d' \gg d$ to users. If we first assign v to a user u, it can prevent the other viewpoints to be assigned to u. To prevent such scenarios we only consider dedicated viewpoints to assign a shared viewpoint. Next, we explain each partitioning approach in detail.

Distance-based partitioning A shared viewpoint in the partitioning stage is assigned to its nearest partition in the distance-based partitioning. We define two alternative distance metrics to measure the distance between a viewpoint v and a partition $part_{u_i}$. Intuitively, v is close to $part_{u_i}$ if it is close to the dedicated viewpoints within $part_{u_i}$. Therefore, we define the first distance metric as the summation of the distances between v and each dedicated viewpoint within $part_{u_i}$.

$$dist(v, part_{u_i}) = \sum_{j=1}^{|D(u_i)|} dist_v(v, v_j),\ v_j \in D(u_i)$$

where $dist(v, part_{u_i})$ is the distance between v and $part_{u_i}$ and $dist_v(x, y)$ is the distance between viewpoints x and y.

A user participation plan includes a path which traverses some viewpoints within the user partition. Motivated by this, for the next distance metric, we define $dist(v, part_{u_i})$ as the distance between v and the minimum length path in $part_{u_i}$ which includes all the existing members of $part_{u_i}$.

$$dist(v, part_{u_i}) = dist_{\text{path}}(v, path_{u_i}),$$

where $path_{u_i}$ is the path with the shortest length covering all the dedicated viewpoints within $part_{u_i}$. As finding $path(part_{u_i})$ is an NP-Hard problem (by reduction to Traveling Salesman Problem) we approximate $path_{u_i}$ by constructing minimum spanning tree within $part_{u_i}$ and converting it to approximate traveling salesman problem tour. The distance between v and $path_{u_i}$ ($dist_{\text{path}}(v, path_{u_i})$) is the minimum length required to expand $path_{u_i}$ to include v. After defining the distance metrics, we assign a shared viewpoint to its nearest partition. In case of tie, we select

one of the nearest partitions randomly. We call the partitioning based on the first distance metric DPOINTS and the partitioning based on the second distance metric DPATH.

Nearest-neighbor based method Intuitively when a user collects texture from a viewpoint, it is efficient for him to next collect texture from the nearby viewpoints to save the resources. Based on this intuition, we propose a nearest-neighbor based approach to decide the partition of a shared viewpoint. Consider a shared viewpoint such as a. To decide a partition for a, we look at the nearest-neighbor of a within the dedicated viewpoints in the partitions. More specifically, consider the set of dedicated viewpoints within partitions to be $N = \{v | v \in D(u_j), u_j \in U_i\}$. Presume that the nearest-neighbor of a in N to be $NN(a) = b$, where $b \in part_{u_k}$. We assign a to $part_{u_k}$ and u_k is the candidate user to collect texture from a. We call this partitioning approach NN for short.

Algorithm The group-based assignment based on two-stage assignment is presented in Algorithm 4. $TwoStage$ method takes as input users U_i, unassigned viewpoints V and the cells not texture mapped C'. The partitioning stage is executed in lines 1–15 while the second stage is performed in lines 16–20. At the partitioning stage, the partitions are first initialized by adding starting and ending viewpoints, i.e., v_{s_i} and v_{d_i} for user u_i correspondingly, to the partitions (lines 1–3). As discussed in Section 5.2.1 these viewpoints can be virtual viewpoints. Thereafter, the viewpoints are categorized and viewpoints which are infeasible for all the users are pruned. Subsequently, a viewpoint $v \in D(u_j)$ is assigned to $part_{u_j}$. For a shared viewpoint v the users who can collect texture from v are first found (line 12) and then v is assigned to one partition based on the methods discussed in Section 5.2.2. Finally an instance of SVA is executed for each partition to generate user participation plans.

Algorithm 4 TwoStage(U_i, V, C')

```
1: for each u_j in |U_i| do
2:     part_{u_j}=part_{u_j} ∪ v_{s_j} ∪ v_{d_j};   {Initializing the partitions}
3: end for
4: Categorization(V, U_i);
5: Prune(V);   {A viewpoint which is infeasible for all users is removed}
6: for each v in V do
7:     if v in D(u_j) then
8:         part_{u_j}=part_{u_j} ∪ v;
9:         V=V-v;   {Excluding the assigned viewpoints from V}
10:    end if
11: end for
12: for each v in V do
13:    u_v = {u ∈ u_i | v ∈ S(u)}   {Finding all the users who can collect texture from v}
14:    Assign(u_v,v)   {Assign(u_v) assigns v to one of the partitions}
15: end for
16: for each u_j in U_i do
17:    p_{u_j}=SVA(u_j, V);
18:    Update(V, C', P_{u_j})
19:    P_{U_i}=P_{U_i} ∪ P_{u_j};
20: end for
```

6 Experiments

In this section, we experimentally evaluate our proposed solutions. We first describe our experimental setup, and next we present our experimental results.

6.1 Experimental methodology

Our experimental system is implemented in Java, and runs on a typical Intel 2.66GHz PC with 3.25GB RAM. The operating system is Windows XP SP2. For each setting, we tested the algorithm by running it 10 times to compute the average values. Next, we present our experimental results.

In this section, we first describe how to model the environment, and thereafter we explain the cell phone camera settings used in our experiments.

6.1.1 Environment modeling

We conducted our experiments using two real-world datasets. The first dataset, LA dataset, is the elevation data of Los Angeles area, from USGS (http://data.geocomm.com), covering a 10 km $\times$ 10 km area. The second dataset, USC dataset, is the elevation data of University of Southern California campus covering a 1.5 km $\times$ 1.5 km area. The road network data for LA dataset is acquired from NAVTEQ (http://www.navteq.com). For USC dataset, we assume all points with elevation zero comprise the road network pathways; hence, the road network includes both roads and sidewalks. Users are assumed to only move on these road networks.

The USC dataset includes the USC building coordinates and consists of planar surfaces. Therefore, we created the cells for this dataset by imposing the grids with the resolution of 2 m $\times$ 2 m on the building walls. In this way, more than 100,000 cells are generated. The LA dataset covers the terrain and therefore is non-planar. To find the cells for this dataset we first modeled the dataset in TIN model which resulted in 8,000 triangles. Subsequently, we imposed the grid with the cell size of 20 m $\times$ 20 m on the triangle faces to create the cells. The number of generated cells for this dataset is more than 182,000. We used larger cell size for this dataset because the LA digital elevation model has lower resolution than USC dataset.

Moreover, to quantize the road network space, we impose a grid on the road network and use the intersection of the road network segments and grid cells as the collection of viewpoints. This approach enables us to emulate various viewpoint selection restrictions by changing the granularity of the imposed grid. With our experiments, we imposed grids with different granularity. The number of resulting viewpoints P by imposing different grids is shown in Fig. 6 for each dataset.

6.1.2 Cell phone camera settings

We assume texture documentation is performed using the Nokia N95 cell phone camera[1] in our experiments. The focal length[2] of this camera is 5.6 mm and the size

[1] http://www.forum.nokia.com/devices/N95/

[2] The camera focal point is the point where all parallel rays will be focused to. The distance between camera lens and focal points is focal length [21].

Grid Cell Size	$20m \times 20m$	$40m \times 40m$	$60m \times 60m$	$80m \times 80m$
|P|	54207	13659	5977	3393

(a) LA

Grid Cell Side	$2.5m \times 2.5m$	$5m \times 5m$	$7.5m \times 7.5m$	$10m \times 10m$
|P|	57056	14317	6362	3637

(b) USC

Fig. 6 Sizes of the viewpoint sets with each dataset

of image plane is 4.8 mm $\times$ 3.6 mm. The camera is 5.0 MP and has the resolution of 2582 $\times$ 1944 pixels.

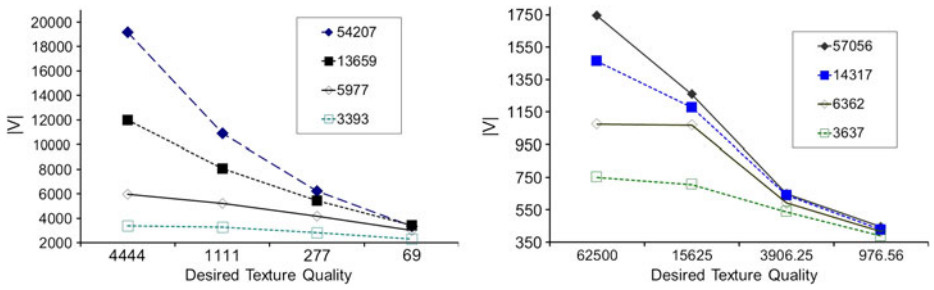
6.2 Experimental results

Below, we explain the results of our proposed algorithms for the viewpoint selection and viewpoint assignment problems respectively.

6.2.1 Viewpoint selection

Based on our experiments, the optimal algorithm for viewpoint selection problem takes more than a day for $|P| = 25$ (i.e., when P includes 25 randomly picked points from the road network imposed on LA dataset) and $Q_T = 2,500$ (throughout the rest of the paper we assume that the unit for resolution is m^{-2}). Also, its running time increases exponentially which makes it impractical. We observed that for $|P| < 25$, $\frac{|V|}{|V_{\text{opt}}|} \geq 95$ %, where $|V|$ and $|V_{\text{opt}}|$ are the number of viewpoints calculated by GVS and optimal algorithms, respectively. Therefore, GVS clearly outperforms the optimal algorithm in efficiency and scalability, while providing almost optimal result. In this section, we study the effect of different parameters on the performance of the GVS algorithm.

Size of the selected set of viewpoints With this experiment, we evaluate the effect of the desired texture resolution Q_T, on the number of selected viewpoints $|V|$. The result is shown in Fig. 7 where each curve is generated for a specific $|P|$. Using GVS,



(a) LA (b) USC

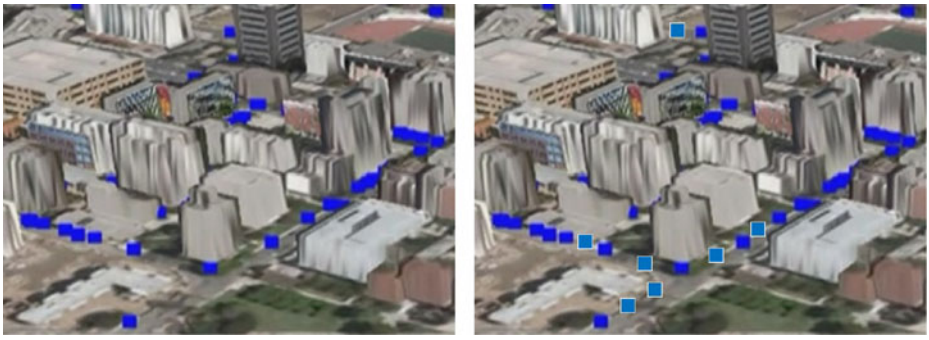
Fig. 7 $|V|$ vs. desired texture resolution (Q_T)

on average $|V|$ is 43 % (89 %) less than $|P|$ for LA (USC) dataset over all the cases. There are two reasons that explain the GVS performance difference with the two datasets: (1) the variation between elevations is much higher in the LA dataset as compared to the USC dataset, and as a result the probability that a cell is visible to two different viewpoints with the resolution of at least Q_T decreases and (2) with the LA dataset users can only collect texture from streets whereas with the USC dataset users can collect texture from side walks as well. This additional constraint also reduces the size of the intersection between visibility sets and consequently increases the number of viewpoints. Regardless of this difference, the reduction in the number of viewpoints significantly improves the scalability of any viewpoint assignment algorithm.

As expected, $|V|$ increases as the desired texture resolution grows, because from a same viewpoint we can capture texture with desired resolution for fewer cells when Q_T increases. Hence, GVS selects more viewpoints. Similarly, increasing the number of road network points results in larger $|V|$, because more cells become visible to the road network viewpoints. The set of selected viewpoints for part of USC area is shown in Fig. 8a for $Q_T = 0$ and Fig. 8b for $Q_T = 2{,}500$.

Collected texture Viewing the texture for all cells in C (the collection of all environment cells) might not be possible from the points in P (see Section 4). Here, we measure the percentage of texture which can be collected from a set of selected viewpoints. This percentage is denoted by $\rho = \frac{TS(V)}{|C|}$. For example, $\rho = 50$ % states that only 50 % of the environment cells can be texture mapped by taking images at the viewpoints. Figure 9 illustrates how ρ varies for different desired resolutions and road network points. For both datasets, decreasing the Q_T increases the value of ρ since texture can be collected for more cells.

For the same Q_T, increasing the number of road network points raises the value of ρ as more texture can be collected from more road network points. Suppose ρ_1 and ρ_2 represent ρ for the LA datasets that have the two largest number of road network points (i.e., ρ_1 and ρ_2 correspond to 54207 and 13659 road network points, respectively). On average, the difference between ρ_1 and ρ_2 is less than 9 %. The same argument holds for the USC dataset as well. The reason is that in the case of



(a) Q_T =0 (b) Q_T =2500

Fig. 8 Selected viewpoint (blue rectangles) for different values of (Q_T)

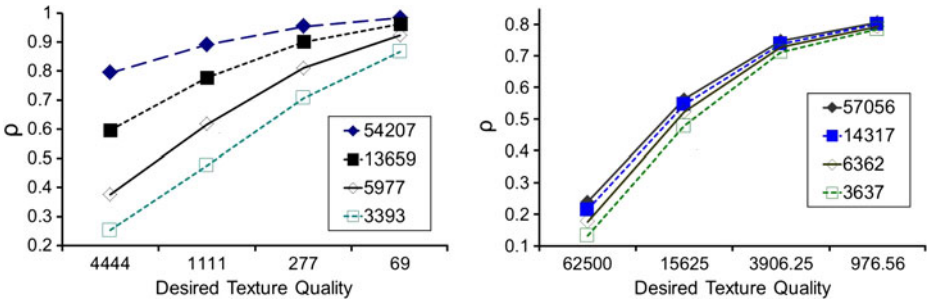


Fig. 9 ρ vs. desired texture resolution (Q_T)

having the largest number of road network points the distance between a point and its neighbor is very small and hence their visibility sets are almost the same. Therefore, the amount of collectable texture is not significantly larger than the case in which we have the next largest number of road network points.

Running time We also measured the average running time of *GVS* algorithm. Our results are shown in Fig. 10. As expected the running time of the algorithm increases when the number of road network points increases. Interestingly, increasing the value of Q_T decreases the running time. This is due to the fact that for larger Q_T values, the size of visibility sets for environment points are smaller. Therefore, the reduced set-cover instances for large Q_T values are less complex to solve. The running time in both cases is less than 16 min.

6.2.2 Viewpoint assignment

We selected a set of viewpoints obtained from viewpoint selection with more than 2,500 for LA dataset and 1,000 for USC dataset to evaluate viewpoint assignment. We precalculated the distances between road network vertices ba able to quickly compute the distances between viewpoints. Precalculation of road network distances is extensively studied in the literature (e.g. [30]). We presume the time required to

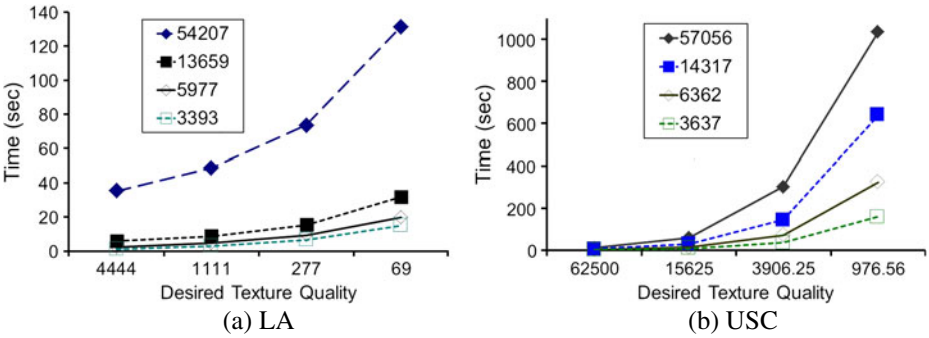


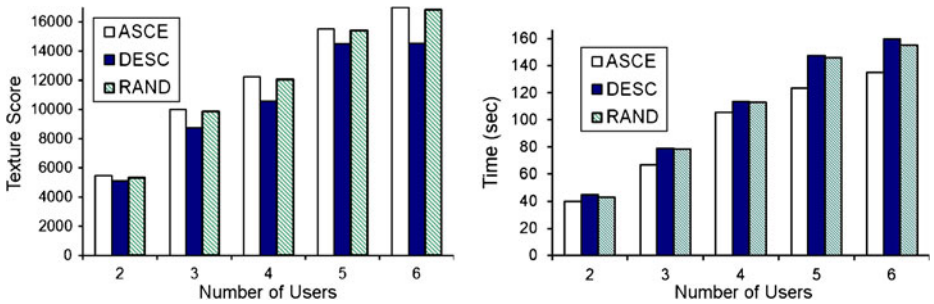
Fig. 10 Running time vs. desired texture resolution (Q_T)

gather texture from a viewpoint is 1 min and a user moves on the road network with the speed of 100 meters per minute. We assume the user participation time is a random number between 15 and 75 (25) min for LA (USC) dataset unless otherwise stated. Although we selected a smaller rang for participation time with the USC dataset, it should not imply that we expect the corresponding experimental results have the same values because: (1) the distribution of viewpoints and the extent of the areas are different, and (2) users move differently around LA and USC. More specifically, users are moving on streets in LA but on both streets and sidewalk in USC.

The viewpoint assignment algorithms proposed throughout the paper extensively uses SVA to generate users' participation plans. We used the orienteering heuristic proposed in [6] to implement SVA. This method has been empirically found to be one of the best orienteering heuristics [3]. It consists of two steps. In the first step, the initialization step, multiple paths are constructed greedily based on distance while ignoring the collected reward. In the second step, by applying operators such as insertion/deletion of vertices and moving vertices between initial paths, the collected reward of the best path is improved. The total reward of the best path can also decrease during this step to avoid local optima. More details on this method can be found in [6]. We run either individual-based or group-based over each $I_i \in T_C$ to generate participation plans for users in U_i. Here we first evaluate the efficiency of the individual and group-based assignment, and thereafter, we study the effect of $|I_i|$ on the efficiency of viewpoint assignment algorithm.

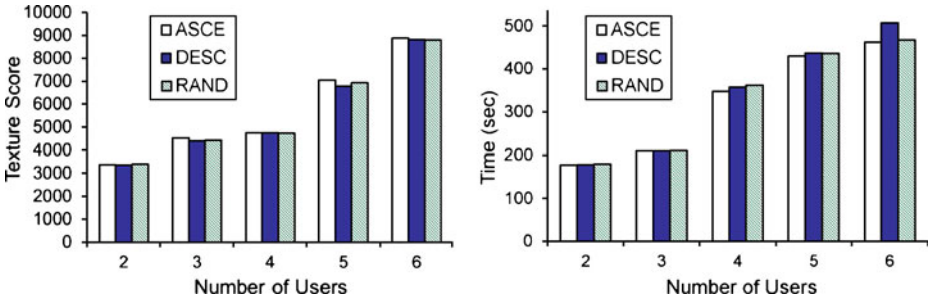
Individual-based assignment With individual-based assignment, we iterate over an ordered list of users and generate the participation plan for each user. Here, we experimentally studied the effect of ordering users based on their participation times on the efficiency of individual-based assignment. We considered three different user orderings: (1) random, in which users are randomly ordered, (2) descending, in which users are sorted in descending order of their participation times, and (3) ascending, where users are sorted in ascending order of their participation times. We denote individual-based assignment based on the first ordering as RAND, the second one as DESC, and the last one as ASCE.

Figures 11 and 12 show the average running time of generating user participation plans and collected texture for different individual-based assignment approaches and



(a) Texture score vs. number of users. (b) Running time vs. number of users.

Fig. 11 Individual-based assignment (LA dataset)



(a) Texture score vs. number of users. (b) Running time vs. number of users.

Fig. 12 Individual-based assignment (USC dataset)

different number of users for LA and USC datasets, respectively. As described in Section 5.2.1, ASCE collects more texture than the other two approaches. At the same time, it is the fastest approach. The reason is that the ASCE approach starts by solving the simpler SVA instances which are instances with fewer number of input viewpoints. This decreases the complexity of SVA instances for users with the largest participation times and hence more input viewpoints. The difference between the ordering approaches is less significant for the USC dataset as compared to the LA dataset.

Although the average participation time is smaller for the USC dataset, the number of viewpoints from which a user can collect texture (i.e., the input points to consecutive SVA instances) is significantly larger for the USC dataset (on average more than 6 times of that of the LA dataset). This is because of the smaller extent covered by the USC dataset and the fact that users can move more freely around USC. The large number of input viewpoints to SVA instances makes individual-based approach slower with the USC dataset as compared to the LA dataset.

Group-based assignment Here, we focus on evaluating the efficiency of the two-stage assignment. We first experimentally evaluate the overhead of the partitioning stage, and thereafter, evaluate the efficiency of the two-stage approach assuming different partitioning techniques proposed in Section 5.2.2.

Overhead of the partitioning stage Here, we set the user participation time for both datasets to a random value between 15 and 75 min and evaluate the running time of the partitioning stage for two-stage assignment. Our results are shown in Figs. 13 and 14 for LA and USC datasets, respectively. For the LA dataset, the running time of the NN approach and the DPOINTS approach are significantly smaller than DPATH approach (because of the simplicity of these approaches). On the other hand, for the USC dataset the running time of all the approaches are very similar. The reason is that the number of dedicated points are very small for the USC dataset because: (1) the extend of the area is smaller, and (2) users move more freely on both sidewalks and streets. We also evaluated the overall overhead of the partitioning stage and in the worst case it is less than 2 % of the total running time for the two-stage approach.

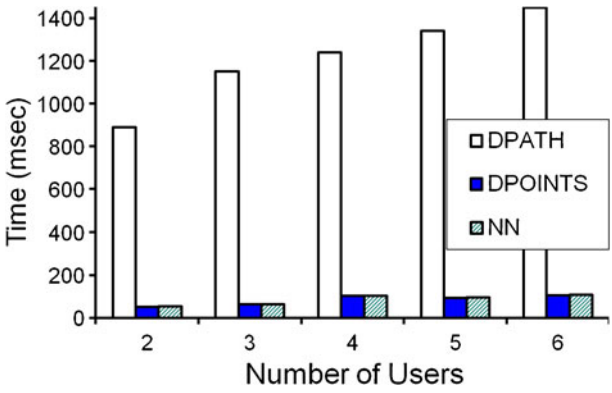


Fig. 13 Running time of the partitioning stage (LA dataset)

Comparison of different partitioning approaches Here we compare the efficiency of our proposed partitioning heuristics. The results are shown in Figs. 15 and 16. Figure 15 shows the average running time in generating a user participation plan for different partitioning approaches, while Fig. 16 shows how the collected texture changes for different partitioning approaches. In all cases, the two-stage approach with NN results in the highest overall collected texture. Considering all the cases together, the collected texture with this assignment approach is 6 % (8 %) higher than the two-stage assignment with DPOINTS and 24 % (19 %) higher than the two-stage assignment with DPATH for LA (USC) dataset. The reason is that two-stage assignment with NN considers only the nearest dedicated viewpoint for the assignment. In contrast, two-stage assignment with DPOINTS considers all the viewpoints in the partitions when assigning a shared viewpoint, where most of them may not be in the optimal solution. Also, the path we construct in a partition, i.e., $path_{u_i}$ with DPATH partitioning, can be very different from the final user participation plan. In terms of the running time, over all the cases and as expected based on the complexity of the approaches, two-stage assignment with NN and DPOINTS partitioning have lower running times than the two-stage assignment with DPATHS.

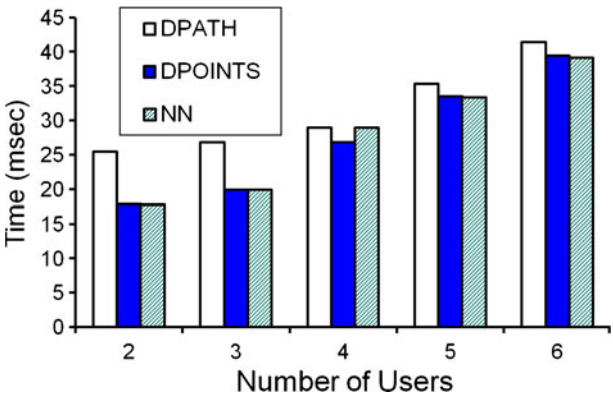


Fig. 14 Running time of the partitioning stage (USC dataset)

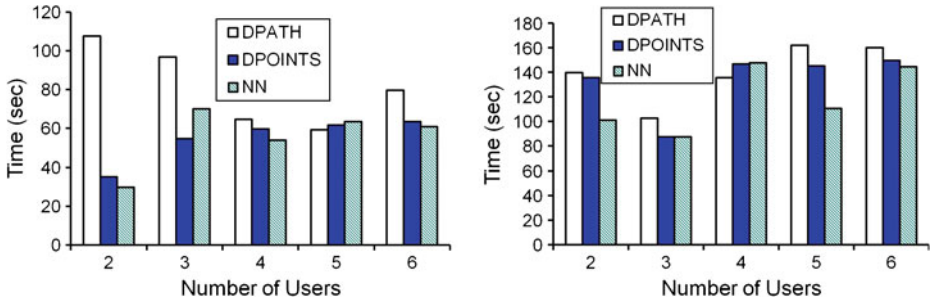


Fig. 15 Running time vs number of users for different two-stage approaches

Comparison of viewpoint assignment approaches In this section we compare individual-based and group-based assignment with each other and with the team orienteering heuristic. We used the algorithm in [39], denoted by TO hereafter, with slight modifications to implement approximate team orienteering. TO is faster and more efficient than other existing heuristics for team orienteering. For each category of viewpoint assignment heuristics we selected a representative which outperforms the others in the same category. Thus, we compared ASCE (an individual-based assignment), NN (a group-based assignment), and TO. Our results are shown in Figs. 17 and 18 for texture score and average running time in generating user participation plans for different number of users.

As expected, the texture score of TO is slightly better than the other assignment approaches. On average NN outperforms ASCE by more than 10 % (8 %) in terms of collected texture for the USC (LA) dataset. This is because NN considers all users for generating user participation plans. In contrast, the ASCE method generates a user participation plan independent of those of other users. As for running time, NN significantly outperforms the others. Averaging over all cases, it is 65 % (45 %) faster than ASCE for the USC (LA) dataset. NN is also more than 70 % (60 %) better than TO for the USC (LA). The reason is the complexity of the operators and heuristics which are used with the TO heuristic. Also, although both NN and ASCE

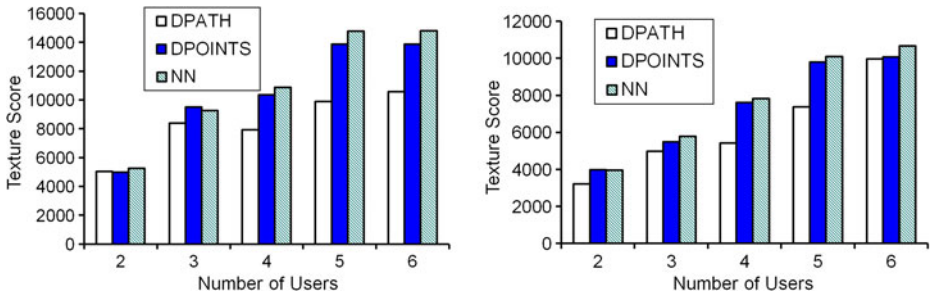
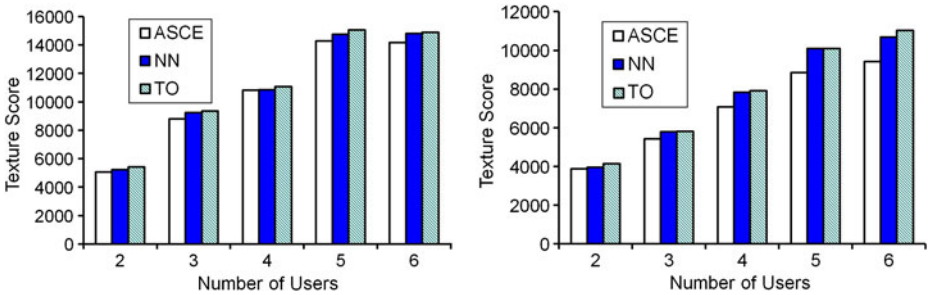


Fig. 16 Texture score vs number of users for different two-stage approaches

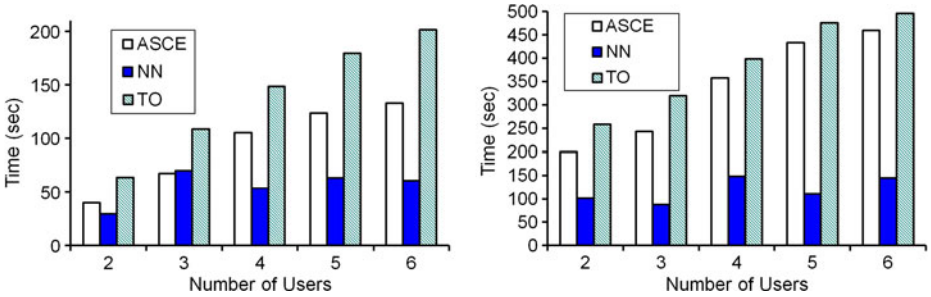


(a) LA (b) USC

Fig. 17 Texture score vs number of users for different viewpoint assignment approaches

approaches solve independent sub-problems by reduction to orienteering, the size of each sub-problem with the two-stage approach is much smaller than that with the ASCE approach. At the same time NN partitioning is fast, and therefore, the total running time of the NN approach is lower than that of the ASCE assignment. The reduction of running time for NN in comparison of ASCE is larger with the USC dataset as compared to the LA dataset. This is because of the fact that the number of viewpoints is larger in USC dataset and hence the effect of partitioning in reducing the number of inputs to SVA instances is more pronounced. Notice that the two-stage approach generates user participation plans by solving instances of SVA problems. After solving each instance, a user participation plan is generated, and hence there is no need to wait for completion of the whole algorithm to generate user participation plans. In this way, on average each user waits for less than 50 (25) s to get her participation plan with the USC (LA) dataset.

Viewpoint assignment over time Here we study the effect of $|I_i|$ on the efficiency of viewpoint assignment. We assume users arrival rates follow a Poisson distribution [40] which is a common distribution to model arrival of individuals in different processes. We assume T_C has the length of one hour and users arrival time follows a Poisson distribution with the mean equals to three arrivals per minute. Each user



(a) LA (b) USC

Fig. 18 Running time vs number of users for different viewpoint assignment approaches

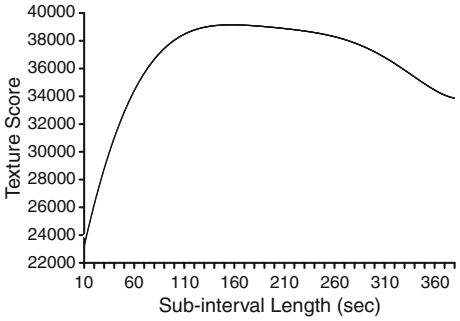


Fig. 19 Collected texture for different $|I_i|$s

available time is a random number between 5 and 20 min. Figure 19 shows the effect of $|I_i|$ on the collected texture for NN approach and for USC dataset. As similar pattern is observed for the other dataset, we only present the results for USC dataset.

As shown in Fig. 19, there is a trade-off in selecting $|I_i|$. By increasing $|I_i|$, we have the opportunity to optimize participation plans across a larger pool of users and hence, better chance to increase the amount of collected texture. On the other hand, increasing $|I_i|$ results in wasting users available time. In this experimental setting the optimal $|I_i|$ is 130 s as figure shows.

7 Conclusion and future work

In this paper, we introduced the problem of participation planning for urban texture documentation. We modeled the problem as a two-step process: viewpoint selection and viewpoint assignment. We studied the complexity of each step, and since the problems are both NP-hard, we proposed efficient heuristic solutions to solve them. In particular, we proposed a greedy algorithm with approximation guarantee, termed GVS, to solve viewpoint selection. We also proposed two categories of heuristics to assign the viewpoints which include individual-based and group-based approaches. We verified the efficiency of our proposed approaches by extensive experiments on real datasets. Based on our experiments, our proposed group-based two-stage approach outperforms the individual-based approach in terms of both running time and amount of collected texture. In particular, the two-stage approach collects on average 8 to 10 % more texture than the individual-based approach and takes 45 to 65 % less time to assign viewpoints for various urban environments. Moreover, comparison with the current approaches for team orienteering, two-stage assigns the viewpoints 60 to 70 % faster but collects at most 6 % less texture. As part of our future work, we plan to extend our proposed approaches for documenting data with other modalities, such as voice and temperature.

Appendix: Calculating A_{p,S_c}

The image of each cell c consists of a set of pixels P_c. With a pinhole model, the region represented by each pixel $p \in P_c$ is in the form of a four sided polygon which area depends on a set of parameters S_c. S_c includes camera focal length f, camera

view angle θ (the angle between camera image plane and c) and camera distance to c represented by d . Finally, we denote the area represented by $p \in P_c$ on c, according to the setting in S_c, by A_{p,S_c}. Here, we elaborate on how to calculate A_{p,S_c}.First, assume the camera image plane is parallel to c, i.e., view angle is zero and $\theta = 0$ (we later consider the general case). Consider a pixel p with the height of p_v and the width of p_h. In this case, the polygon represented by p is a rectangle with the side lengths of l_v (vertical side) and l_h (horizontal side). l_v is coplanar with c and has the length of

$$l_v = \frac{p_v \times d_c}{d_p}, \tag{5}$$

where d_c is the distance between camera lens and point o on c where o, camera lens and center of p_v are on the same line (Fig. 20). d_p is the distance between midpoint of p_v and the pinhole (both d_p and d_c can be calculated based on the parameters in S_c). Equation 5 is derived based on the similarity of triangles formed by l_v, p_v and the pinhole. In a similar way, we can find the length of l_h which is coplanar with c. Therefore, A_{p,S_c} is derived as follows:

$$A_{p,S_c} = l_h \times l_v = \frac{p_h \times p_v \times d_c^2}{d_p^2}. \tag{6}$$

A_{p,S_c} is not exactly the same for all the pixels in P_c and depends on d_c and d_p. Note that p_h and p_v are the intrinsic camera properties. Equation 6 holds for a set of cells whose images are completely captured on the camera image plane. To find this set, we calculate the *field of view* of camera [15]. The field of view of a camera is the portion of scene space that actually projects onto the camera image plane. For a camera with the image plane diameter of I_d, the field of view ϕ is $2 \arctan \frac{I_d}{2f}$ [15]. The image of a cell within the field of view of is completely captured on the camera plane and Eq. 6 holds for it.

We now consider the case in which the view angle is non-zero ($-90 \leq \theta < 0$ or $0 < \theta \leq 90$). In this case, the area represented by p is not necessarily a rectangle but still a four sided polygon where each side of the polygon is represented by a side of p. Assume the bottom side of p is denoted by p_b. We explain how to calculate l_b (the side of the polygon represented by p_b) and the approach can be extended to calculate the other sides. Consider a plane which contains l_b and p_b. We call this plane $P_\perp$. The projection of θ on $P_\perp$ is denoted by $\theta_\perp$ (Fig. 21). Based on

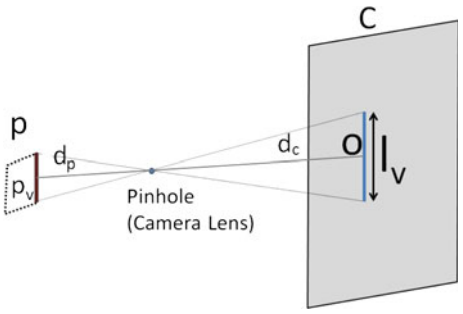


Fig. 20 A pixel p with the side length p_v where p_v represents the length of l_v on cell c

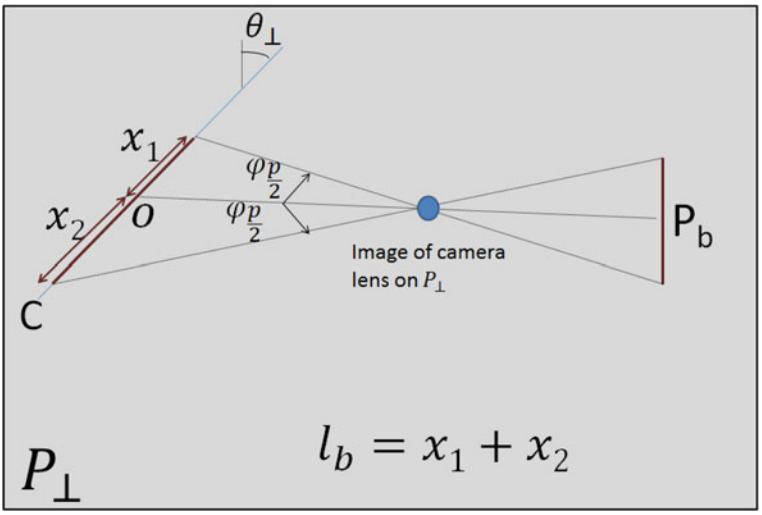


Fig. 21 The top-view of a camera pixel p and a cell c

the figure, $l_b = x_1 + x_2$ where x_1 and x_2 can be calculated according to the law of sines [36], i.e.,

$$x_1 = \frac{d_{c_\perp} \sin\left(\frac{\phi_p}{2}\right)}{\cos\left(\frac{\phi_p}{2} - \theta_\perp\right)} \quad \text{and} \quad x_2 = \frac{d_{c_\perp} \sin\left(\frac{\phi_p}{2}\right)}{\cos\left(\frac{\phi_p}{2} + \theta_\perp\right)}.$$

ϕ_p is the field of view for p (the field of view of a camera which only consists of pixel p) and can be derived similar to field of view of camera. $d_{c_\perp}$ is the distance between camera lens and point o, where o is at the intersection of l_b and the line passing through camera lens image on $P_\perp$ and the center of p_b. Similarly, we can calculate the other sides of the polygon represented by p. Additionally, we can find the angle between the sides based on the equation of planes form from the image sides and pixel sides. Consequently, we can derive A_{p,S_c}.

References

1. Banaei-Kashani F, Shirani-Mehr H, Pan B, Bopp N, Nocera L, Shahabi C (2010) Geosim: a geospatial data collection system for participatory urban texture documentation. Special Issue of IEEE Data Eng Bull 33(2):40–45
2. Blum A, Chawla S, Karger DR, Lane T, Meyerson A, Minkoff M (2003) Approximation algorithms for orienteering and discounted-reward tsp. In: FOCS, pp 46–55
3. Borgstrom PH, Singh A, Jordan BL, Sukhatme GS, Batalin MA, Kaiser WJ (2008) Energy based path planning for a novel cabled robotic system. In: IROS, pp 1745–1751
4. Chakravarty S, Shekhawat A (1992) Parallel and serial heuristics for the minimum set cover problem. J Supercomput 5(4):331–345
5. Chao I, Golden BL, Wasil EA (1996) The team orienteering problem. Eur J Oper Res 88(3): 464–474
6. Chao IM, Golden BL, Wasil EA (1996) A fast and effective heuristic for the orienteering problem. Eur J Oper Res 88(3):475–489

7. Chekuri C, Korula N, Pál M (2008) Improved algorithms for orienteering and related problems. In: SODA 2008. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp 661–670
8. Chekuri C, Pal M (2005) A recursive greedy algorithm for walks in directed graphs. In: FOCS 2005. IEEE Computer Society, Washington, DC, USA, pp 245–253
9. Chen K, Har-Peled S (2008) The euclidean orienteering problem revisited. SIAM J Comput 38(1):385–397
10. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms, 2nd edn. McGraw-Hill Science/Engineering/Math
11. Dhillon SS, Chakrabarty K (2003) Sensor placement for effective coverage and surveillance in distributed sensor networks. Wireless communications and networking, 2003. WCNC 2003, vol 3. IEEE, pp 1609–1614
12. Engel J, Pasewaldt S, Trapp M, Döllner J (2012) An immersive visualization system for virtual 3d city models. In: Proceedings of the 20th international conference on GeoInformatics. IEEE GRSS
13. Evans W, Kirkpatrick D, Townsend G (1997) Right triangular irregular networks. Tech. rep., Algorithmica, Tucson, AZ, USA
14. Fischetti M, Gonzalez JJS, Toth P (1998) Solving the orienteering problem through branch-and-cut. INFORMS J Comput 10(2):133–148
15. Forsyth DA, Ponce J (2002) Computer vision: a modern approach. Prentice Hall Professional Technical Reference
16. Fowler RJ, Little JJ (1979) Automatic extraction of irregular network digital terrain models. In: SIGGRAPH '79: proceedings of the 6th annual conference on Computer graphics and interactive techniques. ACM, New York, NY, USA, pp 199–207
17. http://infolab.usc.edu/projects/GeoSIM
18. Leachtenauer JC, Driggers RG (2001) Surveillance and reconnaissance imaging systems: modeling and performance prediction. Artech House, Boston
19. Guestrin C, Krause A, Singh AP (2005) Near-optimal sensor placements in gaussian processes. In: ICML 2005. ACM, New York, NY, USA, pp 265–272
20. Guillou E, Meneveaux D, Maisel E, Bouatouch K (2000) Using vanishing points for camera calibration and coarse 3d reconstruction from a single image. Vis Comput 16(7):396–410
21. Hecht E (2002) Optics. Addison-Wesley
22. Hörster E, Lienhart R (2006) On the optimal placement of multiple visual sensors. In: VSSN 2006. ACM, New York, NY, USA, pp 111–120
23. Huang CF, Tseng YC (2003) The coverage problem in a wireless sensor network. In: WSNA 2003. ACM, New York, NY, USA, pp 115–121
24. Krause A, Guestrin C (2009) Optimizing sensing: from water to the web. Comput 42: 38–45
25. Krause A, Rajagopal R, Gupta A, Guestrin C (2009) Simultaneous placement and scheduling of sensors. In: IPSN '09: proceedings of the 2009 international conference on information processing in sensor networks. IEEE Computer Society, Washington, DC, USA, pp 181–192
26. Lee DT, Lin AK (1986) Computational complexity of art gallery problems. IEEE Trans Inf Theory 32(2):276–282
27. Lee JS, Hoh B (2010) Dynamic pricing incentive for participatory sensing. J Perv Mob Comp 6:693–708
28. Meguerdichian S, Koushanfar F, Potkonjak M, Srivastava MB (2001) Coverage problems in wireless ad-hoc sensor networks. In: INFOCOM, pp 1380–1387
29. Murray AT, Kim K, Davis JW, Machiraju R, Parent RE (2007) Coverage optimization to support security monitoring. Comput Environ Urban Syst 31(2):133–147
30. Samet H, Sankaranarayanan J, Alborzi H (2008) Scalable network distance browsing in spatial databases. In: SIGMOD conference, pp 43–54
31. Semmo A, Trapp M, Kyprianidis JE, Döllner J (2012) Interactive visualization of generalized virtual 3d city models using level-of-abstraction transitions. ACM GIS 31(3):885–894
32. Shirani-Mehr H, Banaei-Kashani F, Shahabi C (2009) Efficient viewpoint assignment for urban texture documentation. In: GIS '09: proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, pp 62–71
33. Shirani-Mehr H, Banaei-Kashani F, Shahabi C (2009) Efficient viewpoint selection for urban texture documentation. In: Third international conference on geosensor networks
34. Singh A, Krause A, Guestrin C, Kaiser WJ (2009) Efficient informative sensing using multiple robots. J Artif Intell Res 34(1):707–755

35. Singh A, Krause A, Kaiser WJ (2009) Nonmyopic adaptive informative path planning for multiple robots. In: IJCAI'09: proceedings of the 21st international jont conference on Artifical intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 1843–1850
36. Spiegel MR (1992) Mathematical handbook of formulas and tables, 28th printing edn. McGraw Hill
37. Tan PN, Steinbach M, Kumar V (2005) Introduction to Data Mining, 1st edn. Addison Wesley
38. Tsai F, Lin HC (2007) Polygon-based texture mapping for cyber city 3d building models. Int J Geogr Inf Sci 21(9):965–981
39. Vansteenwegen P, Souffriau W, Berghe GV, Oudheusden DV (2009) A guided local search metaheuristic for the team orienteering problem. Eur J Oper Res 196(1):118–127
40. Wolff RW (1990) A note on pasta and anti-pasta for continuous-time markov chains. Oper Res 38(1):176–177
41. Wu CH, Lee KC, Chung YC (2007) A delaunay triangulation based method for wireless sensor network deployment. Comput Commun 30(14–15):2744–2752
42. Zhang B, Sukhatme GS (2008) Adaptive sampling with multiple mobile robots. In: IEEE international conference on robotics and automation

Houtan Shirani-Mehr is currently a PhD student in Computer Science at the University of Southern California, Information Laboratory. He received his B.S. degree in Computer Engineering from Sharif University of Technology in 2004 and his M.S. degree in Information and Computer Science from University of California, Irvine in 2007. His research interests include Distributed Optimization, Planning, Data Mining, Security and Privacy. He has authored nine peer-reviewed research papers along with a pending patent.

Farnoush Banaei-Kashani is a Research Associate and Associate Director of Research at NSF's Integrated Media Systems Center (IMSC) at the University of Southern California. He received his

B.S. in Computer Engineering from Sharif University of Technology in 1996, and his M.S. and PhD Degrees in Computer Networks and Computer Science from the University of Southern California in 2002 and 2006, respectively. He has authored more than thirty research articles in the areas of Sensor and Peer-to-Peer Databases, Distributed Databases, and Spatial Databases, Data Streams, Cloud Computing, and Social Network Analysis. Dr. Banaei-Kashani regularly serves on the program committee of various database conferences.

Cyrus Shahabi is a Professor and the Director of the Information Laboratory (InfoLAB) at the Computer Science Department and also the Director of the NSF's Integrated Media Systems Center (IMSC) at the University of Southern California. He is also the CTO and co-founder of a USC spin-off, Geosemble Technologies. He received his B.S. in Computer Engineering from Sharif University of Technology in 1989 and then his M.S. and PhD Degrees in Computer Science from the University of Southern California in May 1993 and August 1996, respectively. He authored two books and more than hundred-fifty research papers in the areas of databases, GIS and multimedia. Dr. Shahabi was an Associate Editor of IEEE Transactions on Parallel and Distributed Systems (TPDS) from 2004 to 2009. He is currently on the editorial board of the VLDB Journal, IEEE Transactions on Knowledge and Data Engineering (TKDE), ACM Computers in Entertainment and Journal of Spatial Information Science. He is the founding chair of IEEE NetDB workshop and also the general co-chair of ACM GIS 2007, 2008 and 2009. He chaired the nomination committee of ACM SIGSPATIAL for the 2011–2014 terms. He regularly serves on the program committee of major conferences such as VLDB, ACM SIGMOD, IEEE ICDE, ACM SIGKDD, and ACM Multimedia. Dr. Shahabi is a recipient of the ACM Distinguished Scientist award in 2009, the 2003 U.S. Presidential Early Career Awards for Scientists and Engineers (PECASE), the NSF CAREER award in 2002, and the 2001 Okawa Foundation Research Grant for Information and Telecommunications. He was an organizer of the 2011 National Academy of Engineering "Japan-America Frontiers of Engineering" program, an invited speaker in the 2010 National Research Council (of the National Academies) Committee on New Research Directions for the National Geospatial-Intelligence Agency, and a participant in the 2005 National Academy of Engineering "Frontiers of Engineering" program.