Efficient Viewpoint Assignment for Urban Texture Documentation *

Houtan Shirani-Mehr
University of Southern California
Los Angeles, CA 90089, USA
hshirani@usc.edu

Farnoush Banaei-Kashani
University of Southern California
Los Angeles, CA 90089, USA
banaeika@usc.edu

Cyrus Shahabi
University of Southern California
Los Angeles, CA 90089, USA
shahabi@usc.edu

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 resolution urban texture documentation. We have proposed to implement PTD in two steps [10]. At the first step, termed *viewpoint selection*, a minimum number of points in the urban environment are selected from which the texture of the entire urban environment (the part visible to cameras) 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 focus on the viewpoint assignment problem. We first prove that this problem is an NP-hard problem, and therefore, the optimal solution for viewpoint assignment fails to scale as the extent of the urban environment and the number of participating users grow. Subsequently, we propose a family of heuristics for efficient viewpoint assignment to reduce the assignment running time while ensuring an almost complete texture collection. We study, profile and verify our proposed solutions comparatively by both rigorous analysis and extensive experiments.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms, Performance

*This research has been funded in part by NSF grants IIS-0238560 (PECASE), IIS-0534761, and CNS-0831505 (CyberTrust), the NSF Center for Embedded Networked Sensing (CCR-0120778) and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. 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.

1. INTRODUCTION

1.1 Motivation

The advent of Earth visualization platforms (e.g., Google Earth™, Microsoft Virtual Earth™) has inspired and enabled numerous applications. Some of these platforms 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 resolution (i.e., texture cannot be collected frequently and widely enough), particulary where the texture is dynamically changing such as environmental disaster.

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 resolution. 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]. By enabling low-cost, scalable, accurate, and real-time texture documentation, PTD empowers various time-critical applications such as eyewitness news broadcast (for instance, for coverage of live events where the environment texture is dynamically changing), urban behavior analysis, real-estate monitoring, emergency-response, and disaster management (e.g., for damage assessment in case of earthquake, hurricane, and wildfire).

1.2 Contributions

PTD is implemented as a two-step process. At the first step, called *viewpoint selection*, a set of points in the urban environment is selected from which the texture information of the entire environment can be collected. We call such points as *viewpoints*. Due to the participatory nature of PTD, available resources (e.g., users' available times) are usually limited and therefore it is critical to minimize the number of selected viewpoints (we addressed this problem in [10]). 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

[1]A prototype PTD system is developed in Information Laboratory at the University of Southern California (see http://infolab.usc.edu/projects/GeoSIM).

campaign (i.e., the specific time interval allocated for texture documentation) is maximized while all users' constraints are satisfied.

In this paper, we focus on the viewpoint assignment problem. We first prove that viewpoint assignment is an NP-hard problem by reduction from the problem of *Team Orienteering* [3]. 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). Hence, we propose various heuristics for efficient viewpoint assignment with short execution times. We categorize our proposed heuristics into two families of approaches: *individual-based* and *group-based* approaches. With individual-based approaches, we assign viewpoints to each user independent of the other users, whereas with group-based approaches we perform collective assignment for all the users as a group. Because of the exclusive nature of the individual-based approaches, they may result in non-optimal viewpoint assignment. On the other hand, group-based approaches can achieve optimal solutions as they consider the location of the users and their proximity to viewpoints and therefore assign the viewpoints to the users based on the available resources more intelligently. Based on our experiments, individual-based approaches have higher running time (on average 40% more than our proposed group-based approaches) but can collect more texture (on average 18% more texture). The choice of viewpoint assignment algorithm depends on the requirements of PTD and the amount of time users can wait to receive their assigned viewpoints.

The rest of this paper is organized as follows. In Section 2, we formally define the viewpoint assignment problem. We study the computational complexity of the problem in Section 3. We present the individual-based approaches in Section 4 and subsequently the group-based approaches in Section 5. Section 6 presents the results of the empirical analysis of our proposed heuristics. Finally, we discuss the related work in Section 7, and conclude in Section 8.

2. PROBLEM DEFINITION

In this section, we first formally describe the process of urban texture documentation. Thereafter, we introduce participatory texture documentation (PTD) as a scalable approach for texture documentation. Finally, we formalize the problem of viewpoint assignment as the second step of PTD.

2.1 Texture Documentation

Consider an urban environment which consists of various 3D elements such as buildings, trees and terrain (Figure 1(a)). Suppose the environment is modeled at the object level (i.e., a 3D model exists in which the entire environment is represented by a set of objects). Here, without loss of generality, we assume the environment is modeled by TIN (Triangulated Irregular Network) [9] (Figure 1(b)). The *texture* of the environment is the set of images mapped on the triangles of the TIN model. Correspondingly, an urban texture documentation campaign is defined as the process of collecting and mapping the texture onto the TIN model of the urban environment 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.

2.2 Participatory Texture Documentation (PTD)

With PTD, urban texture documentation is implemented by leveraging users' participation. We assume each user announces her constraints denoted by $c = (s, d, A)$ as she joins the texture documentation campaign. With c, user identifies her starting point s in the environment (i.e., her location when she joins the campaign), her desired destination d (where she intends to leave the campaign), and a maximum available time A.

During T_C each user participates in texture documentation by taking a *participation plan* assigned by the PTD system. A user participation plan includes a list of assigned viewpoints from which the user collects texture, as well as instructions on how to move through the urban environment to traverse the assigned viewpoints while going from s to d. For a user u, we denote her participation plan as $P_u = (s, p_{s,v_1}, v_1, p_{v_1,v_2}, v_2 \ldots, v_n, p_{v_n,d}, d)$ where each v_i is a viewpoint and $p_{x,y}$ is an instructed path showing how to go from x to y. User takes P_u by going from s to the first viewpoint v_1 through p_{s,v_1}, from each viewpoint v_i to the next one v_{i+1} through $p_{v_i,v_{i+1}}$, and finally from v_n to d by taking $p_{v_n,d}$. At each viewpoint v_i she takes a panoramic image to collect the maximum texture one can gather at v_i. We assume $p_{x,y}$ is the shortest path from x to y depending on how the user is constrained to move. For example, if the user is constrained to move on the urban road network (including streets and sidewalks) $p_{x,y}$ will be the shortest road network path from x to y. For ease of notation we denote P_u as $P_u = (v_1, v_2 \ldots, v_n)$ in the rest of the paper.

For a participation plan P_u to satisfy users' constraints, the total time a user spends in PTD must be less than A. In other words, traversing P_u and taking images at each $v_i \in P_u$ should be completed in a time interval A. Specifically, if the total time required to traverse P_u is t_p, and the time required for taking images at each $v_i \in P_u$ is constant and equal to t_v, the following should hold:

$$t_p + n \times t_v \leq A$$

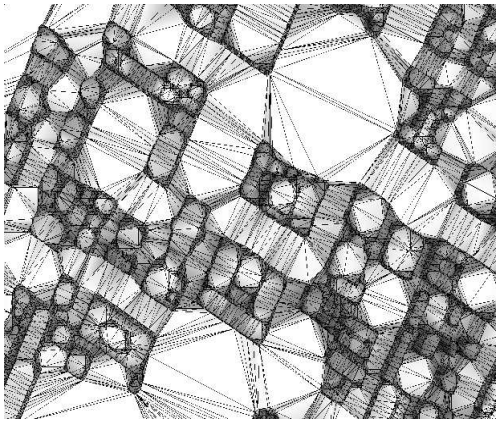
where n is the number of viewpoints in P_u. There might be no participation plan which satisfies a user constraints. In such a case the user will not participate in PTD.

2.3 Viewpoint Assignment Problem

Before formally defining viewpoint assignment, let us first review the viewpoint selection problem. Consider an urban environment represented by a TIN model, with R representing the triangles without texture. Imagine $R' \subseteq R$ as the set of triangles which can be texture mapped while respecting users movement restrictions. With viewpoint selection, a minimum set of viewpoints V is selected from which all triangles in R' can be texture mapped. The solution for viewpoint selection problem can be found in [10] and from now on we assume the viewpoints are given. The input to the problem of viewpoint assignment is the set of viewpoints V and the triangles in R'. Viewpoint assignment is the problem of finding the set of user participation plans which maximizes the number of triangles in R' that can be texture mapped during T_C, while respecting the constraints of the users.

We quantify the amount of texture that is collected while traversing a participation plan by defining a measure termed *texture score (TS)* [10]. Texture score of a set of viewpoints V measures the amount of texture that can be collected from all viewpoints in V collectively. The texture score for a viewpoint v, $TS(v)$, is defined as the total number of triangles without texture which are vis-

(a)

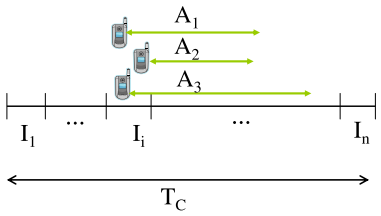


(b)

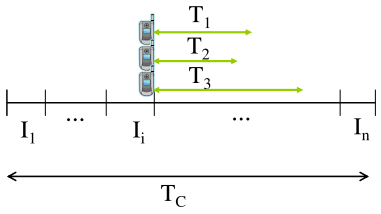
Figure 1: Representing an urban environment for texture documentation: (a) The environment and (b) its representation in TIN model.

ible to v. Similarly, the texture score for a set of viewpoints V, $TS(V)$, is defined as the number of triangles without texture visible to *any* viewpoint in V. A participation plan includes a set of viewpoints and therefore the texture score of a participation plan P_u, i.e., $TS(P_u)$, equals $TS(V_{P_u})$ where V_{P_u} is the set of viewpoints in P_u. Accordingly, the texture score of a set of user participation plans P=$\{P_{u_1}, P_{u_2}, \ldots, P_{u_n}\}$ (where P_{u_i} is the participation plan generated for user u_i) is equal to $TS(V_P)$ where V_P is the union set of viewpoints in all the participation plans in P. Assume $U = \{u_1, u_2, \ldots, u_n\}$ is the set of users who have joined during T_C. Viewpoint assignment finds a participation plan P_{u_i} for each user $u_i \in U$ joining during T_C such that $TS(P_U)$ is maximized, where P_U=$\{P_{u_1}, P_{u_2}, \ldots, P_{u_n}\}$.

To decide when to generate a user participation plan, we propose an iterative assignment schema. We partition T_C into small intervals of equal duration I_i (see Figure 2(a)). At the end of each subinterval $I_i \in T_C$, participation plans are generated for all users who have joined during I_i. Notice that if $|I_i| \approx 0$, each user participation plan will be generated immediately. We experimentally investigate the effect of changing I_i duration in Section 6. Figure 2 illustrates the iterative assignment schema. In Figure 2(a), users are joining PTD during I_i, each identifying an available time A_i. In Figure 2(b), we wait until the end of I_i and plan the participation of the three users who joined during I_i. Notice that in Figure 2(b), each user has less than A_i available time to complete her participation plan. The reason is that she needs to wait until the end of I_i to receive her participation plan. Throughout the paper we denote the user available time, 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. Each T_i in Figure 2(b) represents a user participation time. This iterative assignment schema can lead to optimal viewpoint assignment assuming the distribution of the user arrival time is unpredictable.



(a) Users joining during I_i



(b) Viewpoints are assigned to users at the end of I_i.

Figure 2: Iterative viewpoint assignment over time.

Algorithm 1 summarizes the viewpoint assignment in pseudocode. The algorithm takes V and R' as input. At the end of each time interval it generates participation plans for all users who joined during I_i. We denote the users who join during I_i by U_i. The method $Joined(I_i)$ returns U_i after waiting for $|I_i|$ time units (Line 3). Thereafter, $BatchAssignment(U_i, V, R')$ method, which uses either individual-based assignment or group-based assignment, finds participation plans for all users in U_i, denoted by P_{U_i}. The method takes as input the users U_i (including their constraints), the unassigned viewpoints V, and R' which represents the triangles that are not texture mapped. After calculating P_{U_i} and based on the viewpoints in P_{U_i}, V and R' are updated (Line 6). The set of participation plans generated over all the intervals I_i comprise the participation plans for all the users who join during T_C.

One can reduce the viewpoint assignment problem over T_C to viewpoint assignment in each I_i. To avoid double counting the collected texture and/or sending multiple users to collect texture from the same viewpoint, we need to take into account the participation plans generated previously (i.e., participation plans in $P_{U_{1..i-1}} = P_{U_1} \cup P_{U_2} \cup \ldots P_{U_{i-1}}$) when generating participation plans in P_{U_i}. This can be done by assuming that the triangles visible to any viewpoint in $P_{U_{1..i-1}}$ are texture mapped (assuming that users

Algorithm 1 ViewpointAssignment(V, R')

```
1: P=∅          {Initializing the result set}
2: for i=1 to n do
3:    U_i=Joined(|I_i|);
4:    P_{U_i}=BatchAssignment(U_i, V, R');
5:    P=P ∪ P_{U_i}
6:    Update(V, R', P_{U_i})
7: end for
8: return P;
```

follow their participation plans) when calculating texture scores to generate P_{U_i}. Consequently, we formalize the viewpoint assignment problem as follows:

DEFINITION 1. *Viewpoint Assignment Problem*
For each $I_i \in T_C$, viewpoint assignment finds the set of participation plans P_{U_i} which maximizes $TS(P_{U_i})$.

3. COMPLEXITY ANALYSIS

In this section, we prove that viewpoint assignment is an NP-hard problem by reduction from *Orienteering* [6] problem (which is an NP-hard problem). Orienteering is a variant of the traveling salesman problem and can be formalized as follows:

DEFINITION 2. Orienteering problem [6]
The input to the orienteering problem consists of a weighted directed 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 in G 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 the vertices visited by traversing the path P. The reward function f:$2^V \rightarrow \mathbb{Z}^+$ assigns a non-negative integer reward to any subset of the graph vertices.

Assuming that each vertex of the weighted graph G in an orienteering instance is associated with a non-negative reward, we define f as a function which returns the total reward of distinct vertices in V. For example, f returns the value 5 for a subset of graph vertices V=$\{v_1, v_2\}$ where v_1 and v_2 are associated with the rewards of 2 and 3, respectively. Team orienteering [3] is an extension of the orienteering problem which finds M ($M \geq 1$) paths (each going from v_s to v_d with a length of at most B) such that the total collected reward is maximized.

The orienteering algorithms in the literature fall into three categories: exact algorithms (such as [8]), heuristics with approximation guarantees (such as [5, 7]), and algorithms without approximation guarantee (such as [13, 4]). The algorithms in the first two categories have much longer running times as compared to those in the third category. For example, a recursive greedy algorithm with approximation guarantee is proposed in [6] which has the running time of $O((|V|B)^{O(log|V|))}$. The last category is focused more on reducing the running time without guaranteeing any bound on the collected reward.

To prove that viewpoint assignment is NP-hard, we first prove that the *SVA* problem is NP-hard; we define SVA as a special instance of viewpoint assignment in which only one user joins PTD. Thereafter, we readily conclude that the viewpoint assignment problem is NP-hard. The following lemma proves that SVA is NP-hard.

LEMMA 1. *SVA is NP-hard.*

PROOF. We prove the lemma by providing a polynomial time reduction from the orienteering problem[2]. Towards that end, we prove that given an instance of orienteering, denoted by O, there exists an instance of SVA, denoted by S, such that the solution to S can be converted to the solution of O in polynomial time. Assume a given instance O with the weighted graph G, starting vertex v_s, ending vertex v_d, and available budget B. To solve O, we find a path P_O with the length less than B which maximizes $f(V(P_O))$, where $V(P_O)$ is the set of vertices on P_O. Correspondingly, to solve S, we look for a participation plan P_S which can be finished by user participation time T and maximizes $TS(P_S)$. We propose the following mapping from O to S to reduce O to S.

Each vertex v_i of G is mapped to a viewpoint v_{p_i} in the environment. We assume the shortest distance between two viewpoints v_{p_i} and v_{p_j} is the same as the length of the shortest path between v_i and v_j. The texture score of each viewpoint v_{p_i} is the same as the reward of v_i and, correspondingly, the texture score of a set of viewpoints V_p is assumed to be the summation of texture scores of the viewpoints in V_p, i.e., $TS(V_p) = \sum_{v_p \in V_p} TS(v_p)$. We assume the time required to take an image at a viewpoint is negligible. Suppose the viewpoints correspond to v_s and v_d as user starting and ending points. The budget B in O is mapped to the distance B in S. Assume the user in S has a constant movement speed v. We can always find a participation time T for this user such that she can traverse the distance of at most B in T while moving with the constant speed v.

Given this mapping, it is easy to observe that if the answer to S is the participation plan $P_S = (\alpha_1, \alpha_2, \ldots, \alpha_k)$, the answer to O will be the path P_O, where:

$$P_O = \{(\beta_1, \beta_2, \ldots, \beta_n) \mid \alpha_k = v_i \text{ if and only if } \beta_k = v_{p_i}\}.$$

□

The following theorem follows from Lemma 1:

THEOREM 2. *Viewpoint assignment is an NP-hard problem.*

PROOF. SVA is a special instance of viewpoint assignment and is NP-hard based on Lemma 1. Therefore, viewpoint assignment is also NP-hard. □

4. INDIVIDUAL-BASED ASSIGNMENT

We categorized viewpoint assignment approaches into two categories: individual-based assignment and group-based assignment. In this section we study the individual-based approach. We first solve the SVA problem by reduction to the orienteering problem. Thereafter, we propose a solution for individual-based assignment based on SVA, 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, the individual-based approach can also guarantee an approximation error on the collected texture.

[2]A similar reduction can be provided to reduce the viewpoint assignment problem from the team orienteering problem.

4.1 SVA Heuristic

Here we explain how to solve an instance of SVA by reduction to (not from) orienteering problem. 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 is equal to the shortest path from vp_i to vp_j. If such a path does not exist then $w_{ij} = \infty$. During construction of the orienteering graph, we assume a user starting and ending points 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, respectively. A virtual viewpoint v is a viewpoint which is not obtained by viewpoint selection, and therefore, no texture is collected from v in PTD. We can set the budget based on the user participation time and her movement speed (similar to the reduction in Section 3). 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:

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

Although this reward function (texture score) is more complex than the one we used in Section 3 (i.e., summation), but it satisfies the following two conditions:

1. f is *monotone*, i.e., $f(A) \leq f(B)$ for all $A \subseteq B$.
2. 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 cannot decrease the collected 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, the orienteering algorithms with approximation guarantee can be applied to solve the constructed orienteering instance [6]. The solution to the constructed orienteering instance can be easily converted to the solution of SVA alike the reduction proposed in Section 3.

4.2 Algorithm

Here, we propose the detailed implementation of *BatchAssignment* method in Algorithm 1 based on the individual-based approach which is presented by Algorithm 2. The inputs to this method are the sets U_i (the set of users joining during I_i), V (the set of unassigned viewpoints) and R' (the set of TIN triangles without texture). Assume U_i is an ordered list, i.e., U_i=(u_1,u_2,. . . , u_m); we will discuss how to decide on the users order at the end of this section. 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 subset of triangles in R' that are visible to any viewpoint in P_{u_j} is R'_j. To prevent the double counting of the collected texture, we assume all triangles in R'_j are already texture mapped when generating the participation plans for subsequent users, i.e., for a user u_k where $k > j$. The reason is that the triangles in R'_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.

Consider an orienteering algorithm O with the approximation ratio of η, i.e., η is the ratio of the total rewards 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 the viewpoint assignment over T_C.

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

```
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, R', P_{u_j})
5:    P_{U_i}=P_{U_i} ∪ P_{u_j};
6: end for
7: return P_{U_i};
```

THEOREM 3. *The approximation ratio of the individual-based approach for users joining during T_C is*

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

where P is the set of user participation plans obtained by applying Algorithm 1 for all the users joining during T_C, and P^* is the set of optimal participation plans. η_{max} is the maximum approximation ratio of O for all the users joining during T_C. The proof of the theorem is an extension of the proof of the multi-path orienteering approximation ratio proposed in [1] and can be found in the full version of the paper.

Before running the 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 higher texture score than those of the other orderings. The intuition is that if the participation plan of a user with a long participation time is generated first, the user can collect texture from many viewpoints. Among these viewpoints, there may be numerous viewpoints which are the only possible viewpoints for the users with short participation times. In Section 6, we verify this intuition experimentally.

5. GROUP-BASED ASSIGNMENT

In this section we propose a two-stage approach to implement group-based assignment. At the first stage, termed *partitioning stage*, we create a subproblem for each user which includes a subset of the unassigned viewpoints. We call the viewpoints within a subproblem a *partition*. The idea of this stage is to break the viewpoint assignment problem into multiple disjoint subproblems, where each subproblem has a limited number of viewpoints and therefore can be efficiently solved. At the second stage, to create participation plans for users we solve an instance of SVA for each of the partitions generated at the first stage. In this section, we first formally define each of these two stages and prove the existence of a two-stage assignment solution. 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 4. Finally, we discuss the complete algorithm for the two-stage assignment.

To form a partition corresponding to a user u, denoted by $part_u$, we first categorize the viewpoints based on the constraints of u. For each user u, we can identify three different categories of viewpoints. The first category includes the viewpoints from which u cannot collect texture. Collecting texture from these viewpoints is infeasible for u and we denote them by $I(u)$. If the total time

needed to go from s to a viewpoint v, collecting texture from v, and going to d is more than T, then u cannot collect texture from v. The other category of viewpoints includes those from which only u can collect texture. We denote this set of viewpoints as ***D(u)*** and term them the viewpoints *dedicated* to u. The last category of viewpoints are those from which u among other users can collect texture. We denote this set of viewpoints as $S(u)$ and term them the *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 $c_2 = (s_2, d_2, T_2)$, respectively. We denote the time it takes to go from x to y as $t(x, y)$, and assume that the time required to collect 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, $t(v, d_2)$=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. Assuming the same setting, if we have $t(v, d_2)$=30, then $v \in D(u_1)$, because only u_1 can collect 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 the set of 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 (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 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 the dedicated and shared viewpoint sets.

Algorithm 3 Categorization(V, U_i)

```
1: for each v ∈ V do
2:    for each u_j ∈ U_i do
3:       if t(s_j, v) + t_v + t(v, d_j) > T_j then
4:          I(u_j) ← I(u_j) ⋃ v;
5:       else
6:          CG(v) ← CG(v) ⋃ u_j;
7:       end if
8:    end for
9: end for
10: for each v ∈ V do
11:    if |CG(v)| == 1 then
12:       D(u_j) ← D(u_j) ⋃ v (where u_j ∈ CG(v))
13:    else
14:       S(u_j) ← S(u_j) ⋃ v (where u_j ∈ CG(v))
15:    end if
16: 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:

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

where all partitions are disjoint, i.e., $\forall i \neq j, part_{u_i} \cap part_{u_j} = \emptyset$.

At the second stage, we solve an instance of SVA for each subproblem, i.e., for each $part_u$. The set of SVA solutions for all the partitions comprises the solution for the viewpoint assignment problem.

Assume the optimal set of participation plans is $P^* = \{p^*_{u_1}, p^*_{u_2}, \ldots, p^*_{u_n}\}$, where $p^*_{u_j}$ is the optimal participation plan generated for user $u_j \in U_i$. The optimal two-stage assignment is the one which gives a set of participation plans $P = \{p_{u_1}, p_{u_2}, \ldots, p_{u_n}\}$, (where p_{u_j} is the participation plan generated for $u_j \in U_i$) such that $TS(P) = TS(P^*)$. Next, we prove that two-stage assignment exists.

5.1 Existence

The following lemma proves that two-stage assignment can lead to optimal viewpoint assignment.

LEMMA 4. *Two-stage assignment can generate the optimal viewpoint assignment solution.*

PROOF. To prove the lemma, we show that given a set of participation plans generated by optimal viewpoint assignment, we can construct an instance of the two-stage assignment and obtain a set of participation plans with the same texture score. Assume the optimal solution of viewpoint assignment is $P^* = \{p^*_{u_1}, p^*_{u_2}, \ldots, p^*_{u_n}\}$, where each $p^*_{u_j}$ is the optimal participation plan for user $u_j \in U_i$. For each user $u_j \in U_i$, we construct a partition $part_{u_j}$=$S_1 \cup S_2$ where: $S_1 = D(u_j)$ and $S_2 = \{v_i | v_i \in S(u_j), v_i \in p^*_{u_j}\}$.

Running optimal SVA (SVA which is reduced to optimal orienteering) on $part_{u_j}$ results in a participation plan for u_j with a texture score at least equals to the texture score of $p^*_{u_j}$, because the optimal SVA collects the maximum possible texture score for each partition. The partitions are also disjoint. Therefore, the overall collected texture score, $TS(P)$, is at least $TS(P^*)$. □

5.2 Partitioning Approaches

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 4) to the corresponding partitions. This may result in overlapping between partitions. However, we can safely ignore the overlaps because the number of the overlapping starting and ending viewpoints is very small in comparison to the total number of viewpoints. Notice that such overlaps will not result in the double counting of the collected texture because at the second stage (after solving an instance of SVA per user) each viewpoint is assigned to only one user participation plan. The main challenge at 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 [12]. With our problem, a predefined category is a user partition $part_{u_i}$, including starting and ending viewpoints for u_i and all the viewpoints in $D(u_i)$.

We propose two categories of heuristics to assign the shared viewpoints. In the first category, termed *distance-based* approach, we define a distance measure between a viewpoint and a partition, and use it to assign a viewpoint to its nearest partition. In the second category, termed *nearest-neighbor based* approach, we decide the partition of a viewpoint v based on the partition of the nearest-neighbor of v among the viewpoints which are already assigned. We exploit these two categories of approaches to make the partitioning stage fast and at the same time effective. We next explain each category in detail.

5.2.1 Distance-Based Partitioning

With distance-based partitioning, a shared viewpoint is assigned to its nearest partition in the partitioning stage. We define two 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 all the viewpoints within $part_{u_i}$. Therefore, we define the first distance metric as the summation of the distances between v and each viewpoint within $part_{u_i}$.

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

where $dist(v, part_{u_i})$ is the distance between v and $part_{u_i}$, and $dist_v(x, y)$ is the environment 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 traverses all the existing members of $part_{u_i}$:

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

where $path_{u_i}$ is the path with the shortest length covering all the viewpoints within $part_{u_i}$. As finding $path_{u_i}$ 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 tour. The distance between v and $path_{u_i}$ ($dist_{path}(v, path_{u_i})$) 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. We call the partitioning based on the first distance metric DPOINTS, and the partitioning based on the second distance metric DPATH.

5.2.2 Nearest-Neighbor Based Method

Intuitively, when a user collects texture from a viewpoint, she should next collect texture from the nearby viewpoints to save the resources. Based on this intuition, we propose a nearest-neighbor based approach to decide to which partition a shared viewpoint should be assigned. Consider a shared viewpoint such as a. To decide on the partition for a, we look at the nearest-neighbor of a within the partitions. More specifically, consider the set of viewpoints within partitions, i.e., $N = \{v | v \in part_{u_j}, u_j \in U_i\}$. Suppose the nearest-neighbor of a in N is $NN(a) = b$, 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 partitioning, for short.

5.3 Algorithm

The group-based assignment is summarized by Algorithm 4. This algorithm is an implementation of $BatchAssignment$ in Algorithm 1, based on the two-stage assignment. As input, the $TwoStage$ method takes users U_i, unassigned viewpoints V, and the triangles without texture R'. The partitioning stage is executed in Lines 1 to 14 while the second stage is performed in Lines 15 to 19. 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) to the partitions (Lines 1-3). As discussed in Section 4, these viewpoints can be virtual viewpoints. Thereafter, the viewpoints are categorized and the viewpoints which are infeasible for all the users are pruned. Subsequently, each 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 identified (Line 12) and then v is assigned to one of the existing partitions based on the methods discussed in Section 5.2. Finally an instance of SVA is executed for each partition to generate the user participation plans.

Algorithm 4 TwoStage(U_i, V, R')

1: **for** each u_j in U_i **do**
2: $part_{u_j}$=$part_{u_j} \cup v_{s_j} \cup v_{d_j}$; {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: $part_{u_j}$=$part_{u_j} \cup v$;
9: V=V-v; {Excluding the assigned viewpoints from V}
10: **end if**
11: **end for**
12: **for** each v in V **do**
13: $u_v = \{u \in U_i | v \in S(u)\}$ {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: $Update(V, R', P_{u_j})$
19: P_{U_i}=$P_{U_i} \cup P_{u_j}$;
20: **end for**

6. EXPERIMENTS

In this section, we evaluate our proposed heuristics by empirical analysis. We first describe our experimental setup and then present our experimental results.

6.1 Methodology

We conducted our experiments using a real dataset, i.e., the elevation data of Los Angeles area (from USGS[3]) covering a $10km \times 10km$ area. We assume users are constrained to move on the road network which is acquired from NAVTEQ[4]. The TIN model for the urban environments covered by the dataset includes 8000 triangles. We used the approach presented in [10] to generate the TIN model and to select the viewpoints in the urban environment. The number of selected viewpoints is 975. We precalculate the distances between the road network vertices and use that to quickly compute the distances between the viewpoints. We assume the time required to capture images at each viewpoint is one minute, and a user moves on the road network with the speed of 100 meters per minute.

The viewpoint assignment algorithms proposed in this paper extensively uses SVA to generate user participation plans. We used the orienteering heuristic proposed in [4] to implement SVA. This method has been empirically found to be one of the best orienteering heuristics [2]. It consists of two steps, where at the first step (initialization step) multiple paths are constructed by a greedy approach based on distance, while ignoring the collected reward. At 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 [4].

[3]http://data.geocomm.com
[4]http://navteq.com

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 20 times to compute the average values. Next, we present our experimental results.

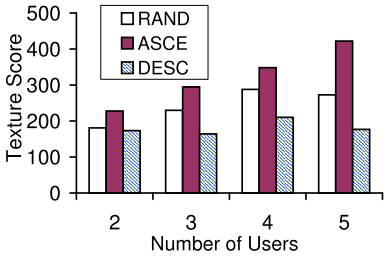
6.2 Results

We run *BachAssignment* (either individual-based or group-based) in Algorithm 1 over each $I_i \in T_C$ to generate participation plans for users in U_i. Here, we first evaluate the efficiency of individual and group-based assignment. Thereafter, we study the effect of changing $|I_i|$ on the efficiency of the viewpoint assignment algorithm (Algorithm 1).

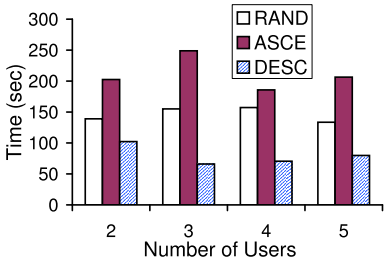
6.2.1 Individual-based Assignment

With individual-based assignment, we iterate over an ordered list of users and generate a participation plan for each user. Here, we study the effect of ordering users based on their participation times on the efficiency of the individual-based assignment Algorithm. We consider three different user orderings: (1) random, with which users are randomly ordered, (2) descending, with 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 subsequently, the last one as ASCE.

Figure 3 shows the average running time for different individual-based assignment approaches and different number of users. We assume user participation time is a random number between 30 and 60 minutes. As described in Section 4, ASCE results in collecting more texture as compared to the other two approaches. However, it has the largest running time. The reason is that the size of SVA instances for consecutive users are decreasing much slower in ASCE than the other two approaches. In other words, the number of viewpoints which are input to SVA instances are decreasing much slower in ASCE. DESC is the fastest individual-based assignment approach as it reduces the size of consecutive SVA instances faster than other approaches.



(a) Texture score vs. number of users.



(b) Running time vs. number of users.

Figure 3: Individual-based assignment

6.2.2 Group-based Assignment

Here, we focus on evaluating the efficiency of the two-stage assignment algorithms. We first experimentally evaluate the overhead of the partitioning stage, and thereafter, evaluate the efficiency of the two-stage assignment based on different partitioning techniques proposed in Section 5.

6.2.2.1 Overhead of the Partitioning Stage. Using the same settings as those of the experiments in the previous section, we evaluate the running time of the partitioning stage for two-stage assignment. The results are shown in Figure 4. As shown in the figure, the running time of the NN approach and the DPOINTS approach are significantly smaller than that of the DPATH approach (because of the simplicity of these approaches) and are very close to each other. We also evaluated the overall overhead of the partitioning stage; in the worst case, the overhead is less than 6% of the total running time for the two-stage algorithm. The is due to precalculation of the shortest paths which makes the partitioning stage fast.

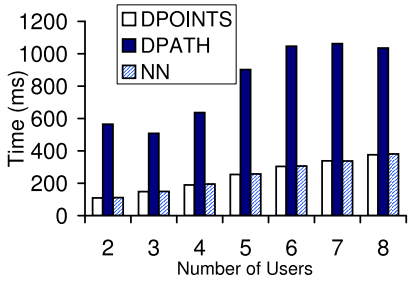


Figure 4: Running time of the partitioning stage

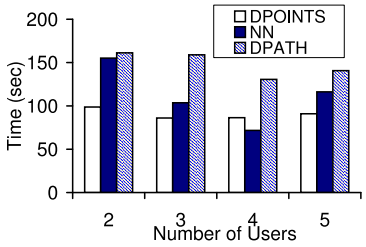


Figure 5: Average running time vs. number of users with two-stage assignment

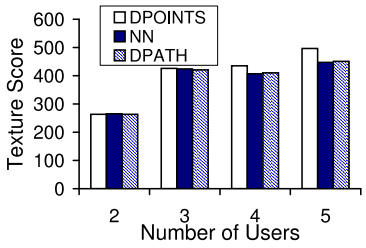


Figure 6: Collected texture vs. number of users with two-stage assignment

6.2.2.2 Comparison of Different Partitioning Approaches. Here, we compare the efficiency of our partitioning heuristics. The users participation times are selected randomly between 45 and 75 minutes. The results are shown in Figure 5 and Figure 6. Figure 5 shows the average running time to generate a user participation plan based on different partitioning approaches. Figure 6 shows how the collected texture changes when different partitioning approaches are used. In all cases, the two-stage assignment with DPOINTS for partitioning results in the highest overall collected texture. All in all, the collected texture with DPOINTS is 8% and 5% higher than those of DPATH and NN, respectively. We attribute this to the fact that NN is very biased to a single viewpoint which may not be in the suboptimal solution. In contrast, DPOINTS considers all the viewpoints in the partitions when assigning a shared viewpoint. Also, the path we construct in a partition (i.e., $path_{u_i}$) using DPATH can be very different from the final suboptimal path which the user takes in her participation plan. In terms of running time, over all the cases the two-stage algorithm with NN and DPOINTS partitionings have similar running times, lower than that of the two-stage algorithm with DPATHS partitioning.

6.2.3 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 [13], denoted by TO hereafter, with slight modifications to implement the approximate team orienteering. TO is faster and more efficient as compared to other existing heuristics for the team orienteering [13] problem. For each category of viewpoint assignment heuristics, we select a representative which outperforms others in the same category. Therefore, we compare ASCE (from individual-based assignment), DPOINTS (from group-based assignment) and TO. We assume random user participation times between 30 and 60 minutes. The results are shown in Figures 7 and 8. These figures show the texture score and the average running time of the solutions versus variable number of users.

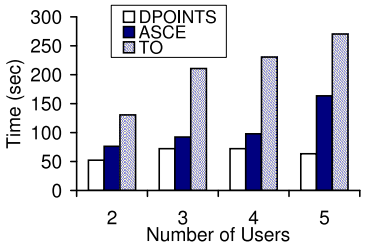


Figure 7: Running time of the viewpoint assignment approaches

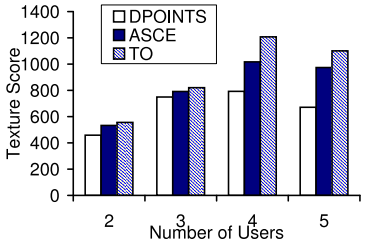


Figure 8: Collected texture with various viewpoint assignment approaches

As expected, the texture score of TO is slightly better than the other assignment approaches. On average ASCE outperforms DPOINTS by 18% in terms of the collected texture. As for running time, DPOINTS significantly outperforms others. Averaging over all cases, it is 40% faster than ASCE and 75% faster than TO. The reason is the complexity of the operators and heuristics which are used with TO. Also, although both DPOINTS and ASCE approach solve independent subproblems by reduction to orienteering, the size of each subproblem within the two-stage assignment is much smaller than that of the ASCE approach. At the same time DPOINTS partitioning is fast, and therefore, the total running time of DPOINTS is lower than ASCE.

6.2.4 Viewpoint Assignment Over Time

Here, we study the effect of changing $|I_i|$ on the efficiency of viewpoint assignment. We assume user arrival times follows a Poisson distribution with the mean of 10 (min). We assume T_C is two hours. Each user participation time is a random number between 30 and 60 minutes. We presume 10 users are joining PTD in the first 30 minutes of T_C. We set $|I_i|$ to one, two, five and 10 minutes and evaluate the amount of collected texture. As the results are similar for both individual-based and group-based assignment we only present the results with DPOINTS. Figure 9 shows the effect of changing $|I_i|$ on the collected texture.

As the figure shows, there is a trade-off in selecting $|I_i|$. By increasing $|I_i|$, we have the opportunity to optimize participation plans by including more users, and hence, increase the amount of collected texture. On the other hand, increasing $|I_i|$ results in wasting users available time. Given our experimental settings, the optimal $|I_i|$ is two minutes.

7. RELATED WORK

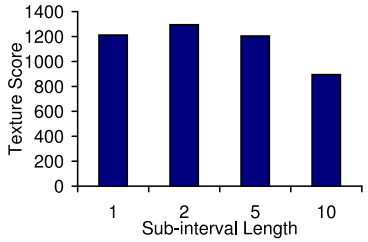


Figure 9: Effect of $|I_i|$

Orienteering is a variant of the traveling salesman problem with profits. The algorithms to solve orienteering fall into three categories: the exact algorithms (e.g., [8]), heuristics with approximation guarantee (e.g., [5, 7]), and heuristics without approximation guarantee (e.g., [13]). The first two categories are not applicable with large problem sizes because of long running times. Team orienteering as a variant of orienteering for multiple users is studied in [3] and [13]. Orienteering is used to model the coordination of participants in data acquisition, assuming each participant has a capacity constraint. Amarjeet et al. [11] studied coordination of multiple robots for environment sensing, where each robot has a resource constraint. In [14], the authors partition the sensing field so that the total obtained reward in each partition is the same and an instance of orienteering is solved in each partition. In this paper, we proposed various heuristics to solve viewpoint assignment problem by reduction to orienteering problem and its variations and also proposed a novel two-stage assignment algorithm and proved that optimal two-stage solution exists. Also, we consider users who are joining over time with various constraints (i.e., starting, ending and available time).

8. CONCLUSION AND FUTURE WORK

In this paper, we introduced the problem of viewpoint assignment for participatory texture documentation. We studied the complexity of the problem and as the problem is NP-hard, we proposed different categories of heuristic solutions for this problem. We evaluated our proposed solutions using a real dataset. We plan to study the scalability of our solutions using larger datasets in the future. Also as part of our future work, we intend to extend our proposed approaches for documenting other urban data modalities, such as sound and temperature.

9. REFERENCES

[1] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. In *FOCS*, pages 46–55, 2003.

[2] P. H. Borgstrom, A. Singh, B. L. Jordan, G. S. Sukhatme, M. A. Batalin, and W. J. Kaiser. Energy based path planning for a novel cabled robotic system. In *IROS*, pages 1745–1751, 2008.

[3] I. Chao, B. L. Golden, and E. A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474, 1996.

[4] I.-M. Chao, B. L. Golden, and E. A. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489, 1996.

[5] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. In *SODA 2008*, pages 661–670, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[6] C. Chekuri and M. Pal. A recursive greedy algorithm for walks in directed graphs. In *FOCS 2005*, pages 245–253, Washington, DC, USA, 2005. IEEE Computer Society.

[7] K. Chen and S. Har-Peled. The euclidean orienteering problem revisited. *SIAM J. Comput.*, 38(1):385–397, 2008.

[8] M. Fischetti, J. J. S. Gonzalez, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS J. on Computing*, 10(2):133–148, 1998.

[9] R. J. Fowler and J. J. Little. Automatic extraction of irregular network digital terrain models. In *SIGGRAPH '79: Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, pages 199–207, New York, NY, USA, 1979. ACM.

[10] H. Shirani-Mehr, F. Banaei-Kashani, and C. Shahabi. Efficient viewpoint selection for urban texture documentation. In *Third International Conference on Geosensor Networks*, July 2009.

[11] A. Singh, A. Krause, C. Guestrin, W. J. Kaiser, and M. A. Batalin. Efficient planning of informative paths for multiple robots. In *IJCAI*, pages 2204–2211, 2007.

[12] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, 1 edition, May 2005.

[13] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196(1):118–127, July 2009.

[14] B. Zhang and G. S. Sukhatme. Adaptive sampling with multiple mobile robots. In *IEEE International Conference on Robotics and Automation*, 2008.