VisualVital: An Observation Model for Multiple Sections of Scenes

Jun Duan, Kang Zhang, Kevin W. Hamlen
Department of Computer Science, The Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Email: {jun.duan, kzhang, hamlen}@utdallas.edu

Abstract—**A computational methodology for reorienting, repositioning, and merging camera positions within a region under surveillance is proposed, so as to optimally cover all features of interest without overburdening human or machine analysts with an overabundance of video information. This streamlines many video monitoring applications, such as vehicular traffic and security camera monitoring, which are often hampered by the difficulty of manually identifying the few specific locations (for tracks) or frames (for videos) relevant to a particular inquiry from a vast sea of hundreds or thousands of hours of video. VisualVital ameliorates this problem by considering geographic information to select ideal locations and orientations of camera positions to fully cover the span without losing key visual details. Given a target quantity of cameras, it merges relatively unimportant camera positions to reduce the quantity of video information that must be collected, maintained, and presented. Experiments apply the technique to paths chosen from maps of different cities around the world with various target camera quantities. The approach finds detail-optimizing positions with a time complexity of $O(n \log n)$.**

I. Introduction

"Dallas, Nov. 22—President John Fitzgerald Kennedy was shot and killed by an Assassin today. ... The killer fired the rifle from a building just off the motorcade route." [1]

The tragic events of November 22, 1963 are a grim reminder of the importance of concise yet comprehensive visual surveillance of security-critical events and venues. Despite the presence of many well-trained human eye-witnesses (e.g., police officers and Secret Service members), as well as television coverage, analysts continue to debate to this day exactly who fired the fatal shot and from what position. Due to humans' limited range of attention and visual scope, it is imperative to grasp parts and details that humans easily neglect.

This is especially important when graphic data processing is involved. Humans more efficiently process visual than textual information in many contexts [2]–[4], motivating the use of visualization for data processing [5]. With advancement of sensory technology enhancement (e.g., consumer VR headsets), there is an elevated need to efficiently extract key points and timings for delivery to human users, since our brains have limited memory volume for long steaming visions or videos. By selecting these crucial scenes and positions [6], [7] and aggregating the total values of the importance of those pieces, people can control or understand the whole contents with less effort.

To this end, our research considers the specific problem of how to position and orient a limited number of video monitoring devices so as to maximize the information gain on the status of a route in a map. We anticipate that both cameras and objects may be dynamic (e.g., possibly moving observers of moving objects, such as traffic, pedestrians, or weather). Relevant applications include auto-generation of adaptable scenes for Oculus® VR with Google Maps®, which entails finding key positions and making a smooth line between them to shorten browsing time and maximize visual information along the path, or performing high-quality traffic monitoring with fewer cameras without coverage loss. In addition, this problem can be simplified into a 1-dimensional version, which is to search those key-frames inside a streaming file and find its related information instantly, such as for music and video fingerprinting [8], [9], copyright protecting [10], [11], and specific scene detection [12], [13].

Unlike prior work on related problems (see Section II), our goal is to retain and maximize overall information related to world status, not to specialize for a particular, known detail or event. For instance, prior work on traffic monitoring has demonstrated the effectiveness of probe vehicles and smartphones for helping users avoid specific, known problem states (e.g., traffic congestion) [14], [15]; and Virtual trip lines [16] can help travelers choose effective routes. But such approaches are not as applicable to identifying and analyzing arbitrary event details in which the user's interest cannot be predicted in advance (e.g., details of a terrorist attack whose relevance only become evident afterward). Our problem also generalizes beyond traffic scenarios.

Specifically, we consider the problem of observing a route (consisting of many segments, each with differing relevant details labeled) with a limited number of camera positions, so as to maximize the overall details observed by selecting optimal positions for the cameras. Each camera must fit in the model from the real world—the detail observed by camera is inversely related to its geometric distance from the object. To maximize the overall information, we invent a method to first maximize local details with sufficient positions, and then gradually merge the selected positions to decease the camera count until it meets a desired target threshold.

The contribution of the work can be summarized as follows:

- By applying human visual characteristics, we create an observing model to simulate the sight that a camera has.

- We implement a virtual camera's functionality in Google Maps®. By tagging each object with a weight (or detail), we can calculate the observed weight under different angles and distance.
- For a limited object in 2D, we find the maximal observed weight exists, and an observation position can be calculated from its minimal circumcircle.
- Our method finds optimal camera positions in $O(n \log n)$ time, and keeps the overall observed details above the baseline.

The remainder of this paper is arranged as follows. Section II summarizes related work. Section III describes the three camera observing models and formulates the problem. Section IV shows the details of our methods and mechanism of the two-phase algorithm. Experiments are implemented in Section V to show the correctness of our proposed solutions. Finally, Section VI concludes.

II. Related Work

Many research works focusing on target tracking and observation have emerged in the past few years. These applications and the popularization of Oculus® Technologies motivate our invention of approaches to observe paths and transportation within the Google Maps®. Recently, a method of tracking motion objects more effectively has been devised [17]. It creatively interlaces objects' global and local features to observe the object more accurately than former approaches.

The increasingly rapid evolution and improving affordability of VR equipment over the past decade has made consumer VR systems extremely popular in the current market. Early work on this subject innovated Head mounted Displays to bring people into virtual worlds [18]. Other work uses surround-screens to present an immersive virtual world by projection [19]. Over time, these implementations have been gradually miniaturized. Wearable VR goggles with projection are now being used in a variety of contexts. For instance, they are being used in job training related to operating precision instruments [20]. In the medical field, they are employed to create imaginal exposure or surgical proxies to cure patients psychologically and physically [21], [22].

Digital mapping and Global Positioning Systems (GPSes) have opened computer vision research to applications related to transportation and localization. This is now a prolific area of study, so only some contributions related to our work are presented here.

Methods of localizing images with architectural features have been invented and implemented for Google Maps® by leveraging the GPS and Google Maps Street View® information [23]. The approach accumulates vote counts based on feature matching to find the GPS position of a query image with the highest votes. Input images containing adequate details (interest points) are crucial for successful queries. In contrast, our work focuses on navigation-related scenarios.

Decision support systems have also been leveraged to calculate vehicle routing in Google Maps® [24]. The major contribution of this work, which is unique to the research on navigation systems, is that many criteria other than distance or time are considered, and these principles influence the standards of evaluation for routing. This affects the role of weight-tagging for each path segment in our problem domain.

Cameras have been used for decades for security monitoring and other situational awareness applications. By combining the concurrent observations of many cameras, visual networks can be built to act for surveillance. Inside this network, every camera is treated as a sensor node and collaborates with the other nodes to collect instant data (in form of video and photos) synchronously [25]. In research involving camera sensor networks, most of the work entails maximizing the covered visual fields and minimizing the number or energy of camera sensors for optimization. Advances include optimized schedules and algorithms to apply camera networks to diverse scenes.

The category of monitored targets is an important aspect of this problem space that must be taken into consideration. Targets are classified into different types, such as multi-positions, regions and barriers (e.g., lines and bands). For example, the Pan and Scan Problem entails covering as many targets as possible by deploying specific numbers of cameras in a plane with vast observed targets [26]. Prior work on this problem has developed a 2-approximation algorithm that applies Voronoi tessellation to regions, with the goal of lessening the overlap of the cameras' covering fields [27].

Extensive research has examined the problem of camera coverage of region barriers. Selecting the least number of cameras to cover specific barriers is one focus of this research [28]. In most cases, the monitored objects are movable and not fixed at positions. Prior work has innovated a system of localization and tracking built atop mobile phone networks [29]. By implementing tracking mechanisms based on the virtual data derived from content inside a camera network, this work improves localization beyond what can be easily achieved with a cellular network alone. Comparing and matching objects inside the visual field facilitates the calculation of information related to targets and their locations. In another work, a system for tracking persons in a 3-dimensional area is proposed [30]. The cameras in this project detect specific human targets by marking multiple points on objects and tracing them in different cameras.

III. Notions & Models

This section presents three models that we combine to formulate the definition of the problem. The first model introduces the type of observed objects. The second one defines rules for observing and calculating *details* (or *weight*). The third one models how the position and angle of the camera affects the details, which extends the rules of the second model.

A. Observation Model

Figure 1(a) shows the basic notions used in this model, assuming a camera c is deployed at a position p, and has a fixed view scope ϕ, observing a segment ℓ. A sector is formed by the angle ϕ as c's vision range, whose depth extends from

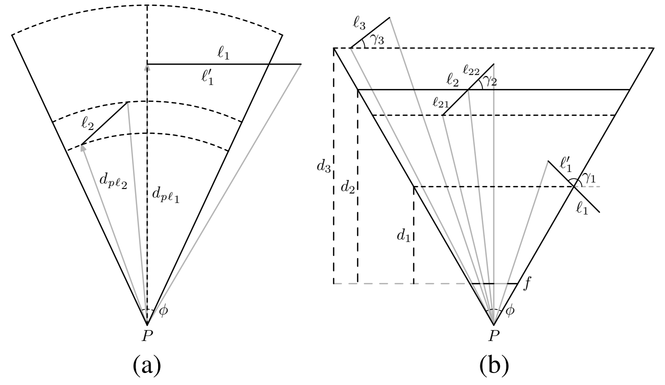

Fig. 1. Complete presentation for an observation and related notions (a) Basic observation (b) Simplified observation with projections

p to ∞. If this entire object ℓ falls in c's vision range, we say ℓ is *fully-observed*. Similarly, whenever part of ℓ stays in c's vision range, we say that ℓ is *partially-observed* by c. We denote the distance between position p and the nearest point of segment ℓ as the *viewing distance* $d_{p\ell}$ of ℓ, and the covered (or c-observed) part of ℓ as ℓ'.

In this model, the detail of segment ℓ observed by camera c is defined as ℓ's weight within a ratio of $(0,1]$, denoted as w_ℓ. The observed weight of ℓ through c is defined as w. For a single segment, w is a function of w_ℓ, $d_{p\ell}$, and ℓ'.

B. Weight Model

When the distance $d_{p\ell}$ between the camera c and the object grows, the amount of observed detail reduces; inversely, when the distance $d_{p\ell}$ is less than a specific value d_0, we cannot see more details than the original w_ℓ from a camera. So, d_0 is the *critical distance* for c's observation of objects. Thus, when ℓ is fully observed, we define the relation between the observed details w, d_0, and $d_{p\ell}$ as follows: (1) When $0 < d_{p\ell} \leq d_0$, we have $w = w_\ell$. (2) When $d_{p\ell} > d_0$, we have $w = \frac{d_0}{d_{p\ell}} w_\ell$.

The critical distance d_0 is defined before an arc $\widehat{c}$ can be drawn along all the critical positions of the camera c. Then, a sector β consists of $\widehat{c}$ and the open angle of camera c at p. Inside the area β, we assume no detail is lost in the observation. In the following, the fan-shaped lossless field is simplified into a triangle to be introduced in the next subsection.

Therefore, combining with the observation model, the observed weight (or details) w can be defined as follows: Since d_0 is a parameter of the camera, we can introduce a constant ratio α to formulate a relation $d_0 = \alpha w_\ell$ to simplify the formula

$$w = \begin{cases} \dfrac{\ell'}{\ell} w_\ell & \text{if } 0 < d_{p\ell} \leq d_0 \\ \dfrac{\ell'}{\ell} \cdot \dfrac{d_0}{d_{p\ell}} w_\ell = \dfrac{\ell' \alpha}{\ell d_{p\ell}} w_\ell^2 & \text{if } d_{p\ell} > d_0 \end{cases}$$

C. Camera Projection Model

We see different lengths of a line when viewing a segment from its side and from one of its ends. Obviously, if a line object could project its entire side on the camera screen (without losing any details), this object should be completely inside the open fan-shaped vision sector β and perpendicular to the viewing direction. Here, the angel between the line object and the screen is denoted by γ. When γ grows to its maximum $\frac{\pi}{2}$, the line object is vertical to the camera screen and projects a dot on it, in which case we consider no detail can be observed. Based on the two critical conditions, our angle-continuous projection is modeled as follows.

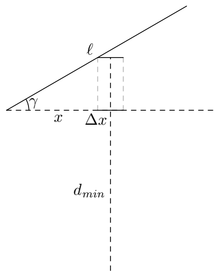

Fig. 2. Accumulation for the calculus step

Given a camera c with view scope ϕ at p, a line object ℓ with weight w_ℓ, and viewing distance $d_{p\ell}$ of ℓ, the weight w of ℓ's image can be approximated by $w = w_\ell |\cos\gamma|$. This is due to two considerations: (1) Computation can be drastically reduced from calculating values on an arc to calculating its corresponding line segment. (2) The difference between the length of an arc and its correspondent straight line in the observation becomes insignificant when the viewing distance is large. Also, $\Delta\ell$ is an increment when $d_{p\ell}$ increases from d_{min}, shown in Figure 2. So a piecewise function with integral is formulated. Taking the weight model into consideration, the length on the camera is

$$w = \begin{cases} 0 & \text{if } \gamma = \frac{1}{2}\pi \\ \dfrac{\ell'}{\ell} w_\ell |\cos\gamma| & \text{if } d_{p\ell} \in (0, d_0] \\ & \text{and } \gamma \in [0, \frac{1}{2}\pi) \cup (\frac{1}{2}\pi, \pi] \\ \displaystyle\int_0^{\ell'|\cos\gamma|} \frac{w_\ell d_0}{\ell(d_{p\ell} + x|\tan\gamma|)} \, \mathrm{d}x & \text{if } d_{p\ell} > d_0 \text{ and } \gamma \neq \frac{1}{2}\pi \end{cases}$$

Substituting $d_0 = \alpha w_\ell$ and evaluating the integral, we obtain

$$w = \begin{cases} 0 & \text{if } \gamma = \frac{1}{2}\pi \\ \dfrac{\ell'}{\ell} w_\ell |\cos\gamma| & \text{if } d_{p\ell} \in (0, d_0] \text{ and} \\ & \quad \gamma \in [0, \frac{1}{2}\pi) \cup (\frac{1}{2}\pi, \pi] \\ \dfrac{\alpha w_\ell^2}{\ell |\tan\gamma|} \ln\left(1 + \dfrac{\ell' \sin\gamma}{d_{p\ell}}\right) & \text{if } d_{p\ell} > d_0 \text{ and} \\ & \quad \gamma \in (0, \frac{1}{2}\pi) \cup (\frac{1}{2}\pi, \pi) \\ \dfrac{\ell' \alpha}{\ell d_{p\ell}} w_\ell^2 & \text{if } d_{p\ell} > d_0 \text{ and } \gamma \in \{0, \pi\} \end{cases}$$

In typical application scenarios, the camera usually observes segments from a long distance. Thus, the meaning of $d_{p\ell}$ is also changed, which indicates the distance between the nearest point in object ℓ' and position p. This significantly

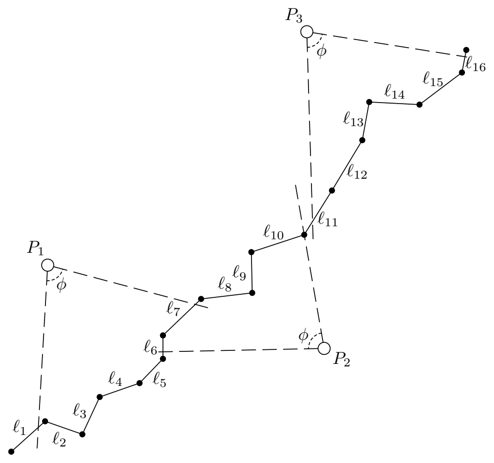

Fig. 3. Camera c observes poly-line L at different positions P_1, P_2, P_3

simplifies the computation for the weight-lossless area. The similar relation $d_0 = \alpha w_\ell$ can still be easily proved and holds after the transformation.

D. Problem Formulation

Given a series of continuous segments, a poly-line $L = \{\ell_0, \ell_1, \ell_2, \dots, \ell_n\}$, with each segment ℓ_i assigned a weight w_{ℓ_i} (to denote the quantity of details of this segment), and a camera c whose viewing angle is ϕ, the total amount of observed detail is $w = \sum_{i=0}^{n} w_i$.

Having built the model, we formulate the problem as finding a set of positions that satisfies the following three requirements:

- fully covering each segment of a poly-line object,
- having a bounded number of spots (positions),
- maximizing weight under the spot limits.

Given a set of positions, we further investigate whether there exists a shortest path for the camera to pass all the positions.

IV. Algorithm Design

For the two problems above, we need to build the position set first and then the orbits can be searched in the set. To find the set, we perform two major steps. First, we compute a *lossless weight set* (LWS) that deploys the fewest number of points to cover the target line without losing any details. That is, the weight to be observed at these positions should be equal to the original weight of the object. Second, we compute a *points constricted set* (PCS) by revising and deleting positions to retain the maximized total observed weight and reduce the number to meet the boundary on the number of positions. The weight obtained from the positions in PCS is apparently smaller than or equal to that from those in LWS, and reaches its maximum under the number limit (threshold).

A. Lossless Weight Set

To find the LWS for a poly-line, an efficient method is used to sequentially record positions. The way of collecting the positions for LWS starts along a poly-line that is parallel to the object. Each time, we choose a position where the vision lossless range of the camera can cover the most weight on the line ℓ_i with the best viewing angle ($\gamma = 0$). This procedure of selecting positions continues until the length of ℓ_i is equal to 0. Algorithm 1 sketches the procedure for computing the LWS for a single side of the poly-line.

Algorithm 1: Algorithm for building LWS (ABLWS) on one side[1]

```
Input  : $L = \{\ell_1, \ell_2, \dots, \ell_n\}$; $\{w_{\ell_1}, w_{\ell_2}, \dots, w_{\ell_n}\}$; $d_0$; $\phi$
Output : $LWS$
1  $LWS = \emptyset$;
2  Choose an endpoint $e$ (either from $\ell_1$ or $\ell_n$) along $L$;
3  for $i = 1$ to $n$ do
4      Construct a poly-line $\ell_{ip}$ perpendicular to $\ell_i$ at $e$;
5      Place $c$ at a position $p$ on $\ell_{ip}$ at distance $d_0$ from $e$
         (where $d_0$ is $c$'s critical distance). Position $p$ could
         be at either side of $\ell_i$. Without loss of generality, we
         assume it is on one specific side for all;
6      Adjust the bisector $b$ of $c$'s view scope so $b$ is per-
         pendicular to poly-line $\ell_i$ and view scope open to $\ell_i$;
7      while $len(\ell_i) > 0$ do
8          Begin with one end of $\ell_{ip}$ and move $c$ along the
             poly-line $\ell_{1p}$. Record the position $p$, direction of
             camera's view scope $b$, and observed weight $w$
             when the total weight inside the view scope
             reaches its maximum;
9          Delete the part of $\ell_1$ whose weight has been
             recorded in the last step;
10         LWS $\leftarrow (p, w)$;
11     end
12 end
```

Algorithm 1 states that the camera at each position p in LWS always observes the maximal weight when a line object is larger than the range c with viewing angle ϕ. Thus, all the positions are consecutively selected into LWS along the moving trace of the camera. If a line or the remaining part of a line is smaller than the range c, its position is revised and combined to decrease the number of positions in the following algorithms.

If it still does not meet the limit, some positions are combined and adjusted to cover the part covered by the camera at more than one previous position. Since the LWS elements are chosen sequentially along poly-line $L_p = \ell_{1p}, \ell_{2p}, \dots, \ell_{np}$, the better options are always to replace the two nearby elements in the order of LWS by one best candidate to reduce the count of the set. In particular, two non-consecutive positions cannot be selected as a replaced pair because the new cover range overlaps the part between positions.

Consequently, this motivates a procedure to find which pair of consecutive positions should be replaced by a new position

[1]Positions can also be chosen from both sides, but doing so requires the camera to cross the object under observation, which is undesirable and should be avoided. If all the selected positions are kept on the same side, the orbit never intersects the polyline.

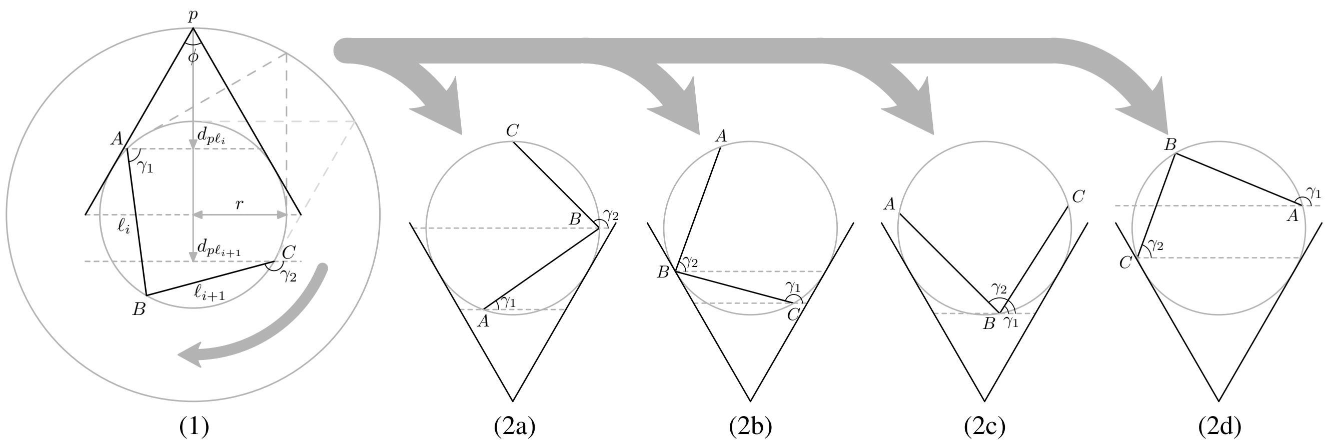

Fig. 4. (1) All possible situations of camera view (2) 4 different angle relations between γ_1 and γ_2

with the sum coverage of both former positions and the least loss of observed weight in the current LWS. Our approach attempts all pairs of neighboring positions in LWS and identifies candidate positions.

B. Points Constricted Set

The first step is to find positions and directions of the cameras to cover the range at 2 consecutive positions. Our model shows that the total observed weight is related to the angle formed by the two neighboring line segments. We begin with the easiest case—the three endpoints are collinear. The camera must move away in the same direction until covering exactly the 2 segments. If the three endpoints are not collinear, however, a series of positions can be found based on the circumcircle of the three endpoints. Assume that $\ell_i = AB$ and $\ell_{i+1} = BC$ are covered by a camera c at positions p' and p'' of LWS. If the area of the circumcircle of A, B, and C can be covered by c with open angle ϕ at a single position p_i, then ℓ_i and ℓ_{i+1} are included, too. From the formula of our camera projection model, it is known that w is a monotonic decreasing function of $d_{p\ell}$. So the nearest position to observe this circumcircle with any directions of c is the farthest vertex of the isosceles triangle formed by angle ϕ and the line through radius r, in which a half part of the circumcircle is inscribed. The candidate positions also compose a circle $\bigcirc p$ and are shown in Figure 4. Observe that points inside $\bigcirc p$ may not fully cover ℓ_i and ℓ_{i+1}, and points outside $\bigcirc p$ must get smaller weight than points on $\bigcirc p$ due to the inverse relation between w and $d_{p\ell}$.

The second step is to select the best position on circle $\bigcirc p$ from candidate positions. This situation can occur when one object is blocked by another at some positions on the circle. Therefore, these positions are excluded from comparison. In Figure 4, conditions (2a) and (2b) are the excluded ones. For the left two conditions, we choose the best position based on the formula from our model

$$w = \frac{\alpha w_\ell^2}{\ell\,|\tan\gamma|}\ln\!\left(1+\frac{\ell'\sin\gamma}{d_{p\ell}}\right) \quad \text{when } d_{p\ell} > d_0 \text{ and } \gamma \in (0, \tfrac{1}{2}\pi) \cup (\tfrac{1}{2}\pi, \pi)$$

In this function, w is related to $d_{p\ell}$ and closely related to γ. This is because the two left situations, (2c) and (2d), restrict $d_{p\ell}$ to a range, and can only be changed less than the radius of the circumcircle. Consequently, finding the right γ is the key point in this search. Furthermore, we know the value of the angle formed by the two line objects, and can easily derive the relation between γ_1 and γ_2, which are the angles formed by the objects and the screen of the camera, respectively. In addition, the total weight is the sum of these two observed weights. As a result, the following steps of calculation find the critical point of γ_1 (or γ_2) which maximizes w within its interval.

After computing the best location on the circle, the final task is to adjust the distance between the camera and the two line objects. Usually the camera remains stationary or is relocated a bit towards the objects. When entering the circle, the camera at some spots can get more weight without losing any coverage for the objects; on other spots, it cannot keep the objects fully covered if it moves inside.

We design the following two algorithms based on the analysis above. Algorithm 2 calculates the best position between any two consecutive positions of LWS, and Algorithm 3 deletes and replaces the positions for the camera to get the total maximal weight subject to the number restriction. In the algorithms, p_{max} denotes the position where c can fully cover ℓ_i, ℓ_{i+1} and maximize the observed weight; w_{max} denotes the observed maximal weight.

Assume that we select all the positions on one side of poly-line L; the side should be designated in advance. Also, the camera is always positioned over the critical distance from the poly-line, ensuring a near-maximal total weight observed without losing details in each calculation.

In the last step, p_{max} is inclined to be chosen in the later side of the record. That is because if the candidate positions are continuous, the poly-line is more likly to be divided into small segments at the beginning. Since the positions are chosen greedily, the poly-line can be broken into two discontinuous parts. Thus, we have another algorithm based on algorithm 2

Algorithm 2: Algorithm for Finding the Candidate Position (AFCP)

```
Input  : $\ell_i = AB, \ell_{i+1} = BC$; $p_j, p_{j+1} \in LWS$; $d_0$; $\phi$
Output : $p_{max}$, $w_{max}$
1  Calculate the angle $\alpha = \measuredangle ABC$ ;
2  if $\alpha = 0$ or $\alpha = \pi$ then
3      Locate the position $P_{max}$ to exactly cover $\ell_i$ and $\ell_{i+1}$ and use the same view direction as $c$ did at $p_j$ (or $p_{j+1}$);
4      Calculate $w_{max}$ with $c$ at $p_{max}$;
5  else
6      Draw the circumcircle $\bigcirc P$ of $A$, $B$, and $C$ to get its center $P$ and radius $r_P = PA$ [31];
7      Draw a circle $\bigcirc P'$ with $P$ as its center and $r = r_P / \sin\frac{\phi}{2}$ as its radius;
8      Get the equation of $w_t = w_i + w_{i+1}$ for $\ell_i$ and $\ell_{i+1}$ based on the formula of $w$ and $\gamma$, by using $\gamma_{i+1} = f(\gamma_i, \alpha)$ to replace $\gamma_i$;
9      Insert the (2c) and (2d) conditions in Figure 4 to calculate maximal values, denoted by $w_{t1}$ and $w_{t2}$;
10     Compare the two values and choose the position $p'$ of the larger $w_t$;
11     Relocate towards $P$ to exactly cover $\ell_i$ and $\ell_{i+1}$ and use the same view direction as $c$ did at $p'$;
12     Calculate $w_{max}$ and $p_{max}$ with the current $d_{p\ell}$;
13 end
14 Return $w_{max}$ and $p_{max}$ ;
```

Algorithm 3: Algorithm for Creating the PCS (ACPCS)

```
Input  : $L$; $d_0$; $\phi$; $LWS = \{p_1, p_2, \ldots, p_{|LWS|}, \}$; $limit$
Output : $PCS$
1  $PCS = LWS$;
2  Initialize an ordered set $TMP = \{t_1, t_2, \ldots, t_{|LWS|-1}\}$ and and $t_i = \{p, w\}$;
3  if $|LWS| \leq limit$ then
4      Return $PCS$;
5  else
6      for $i = 1$ to $|LWS| - 1$ do
7          $t_i = AFCP(p_i, p_{i+1})$;
8      end
9      while $|LWS| > limit$ do
10         Find $t_i$ satisfying $p_i.w + next(p_i).w - t_i.w = min\{p_1.w + next(p_1).w - t_1.w,\ p_2.w + next(p_2).w - t_2.w,\ \ldots\}$;
11         $PCS = PCS - next(p_i)$ ;
12         $p_i = t_i$;
13         if $prev(t_i)$ exists then
14             $prev(t_i) = AFCP(prev(p_i), p_i)$;
15         end
16         if $next(t_i)$ exists then
17             $next(t_i) = AFCP(p_i, next(p_i))$;
18         end
19         $TMP = TMP - t_i$;
20     end
21     Return $PCS$;
22 end
```

to justify the best positions in those sub-polylines. Again, we always stay on one side.

V. Simulation

To verify the correctness and evaluate the performance of our algorithms, we conducted experiments on geographic data obtained from Google Maps®. Streets of 4 different cities around world are chosen for the simulation. Some cities and districts have neat planning such that all their streets follow a specific pattern. For example, Ginza in Tokyo follows a rectangular pattern. In other cities, such as the districts in Cairo, Egypt, most of the streets evolved historically without any regular pattern. Our simulation results show that all the calculated camera spots can cover all the streets when the number of spots is decreasing. Also, it retains over 90% of observed details when removing 20% camera spots.

A. Experiment Setup

The experiment setting receives two inputs: a camera with fixed parameters and selected routes (i.e. streets in the cities). It outputs the observed details with the boundaries of all camera spots. A real-world 24mm wide-angle camera is simulated, with view scope $\phi = 84°$, viewing distance $dis \in (0, \infty)$, and maximal lossless distance $d_0 = \sqrt{2}$. All the routes are listed in Table I, which are approximated as poly-lines, and each line segment is set with weight (or detail) according to its traffic volume (or importance). For example, a path from the Great Pyramid at Giza to the Egyptian Museum is weighted and transformed to a poly-line based on its relative coordinates.

TABLE I
Routes Information

No.	City	Segments	Baseline Spots	Total Weight
1	Dallas	19	155	148
2	Paris	18	216	146
3	Tokyo	25	153	362
4	Cairo	42	226	426

For each poly-line, a set of baseline spots are first computed by finding the minimum sum of camera spots to fully cover the line without losing any detail. We then collect the total weights with different boundaries of different spots. The performance of the algorithm is evaluated by examining how slowly (or quickly) the observed details are lost when reducing the number of spots.

B. Performance

The performance of our experiment can be evaluated by comparing the baseline and results from our algorithm, as shown in Figure 5. The first step is to draw a line with the maximal weights without covering the entire poly-line within

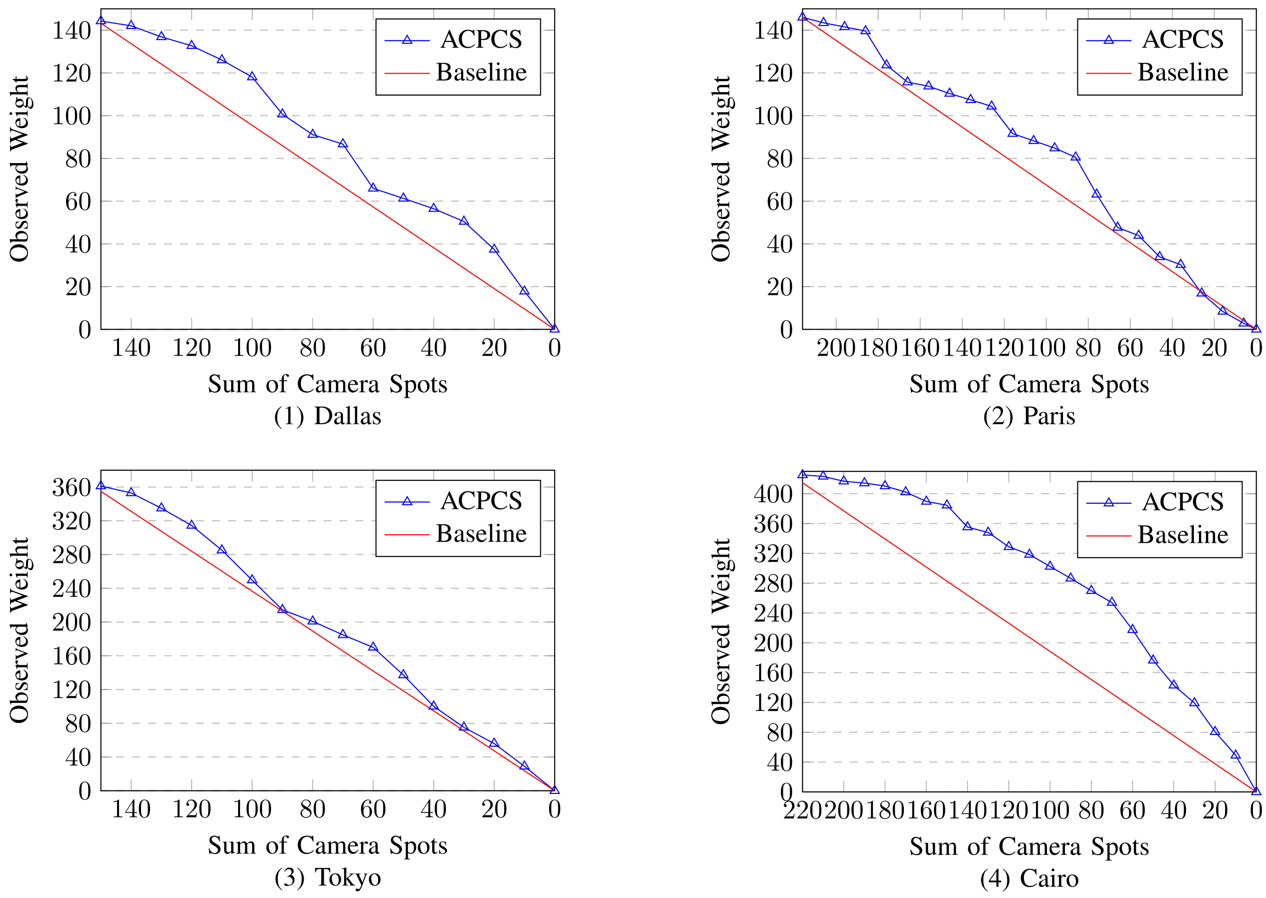

Fig. 5. Detail-preservation when decreasing the number of spots on sampled path data from four different cities (Dallas, Paris, Tokyo, Cairo)

the limits of camera spots. In the second step, we compute the sum of the weights when decreasing the number of spots.

Figure 5 shows that at the beginning of decreasing spots (among the first 20% of span), our method can keep more than 90% of the original weight. For all the four cities, the results show that our algorithms maintain more weights than the baseline set 98% of the time in the collected samples. The only scenario in which the total weight is lower than the baseline weight happens when the limit of spots is under 8%. When the limit is below 8%, one must set the camera spots far away from the observed lines in order to cover the full range. Moreover, our experiment shows that with an increasing number of segments, our method generates better results, as compared for (2) and (4) in Figure 5.

Our experiment reveals that with a series of weighted objects, the segments of the highest weights are selected first. The system always attempts to merge the objects of lower weights, and is inclined to cover those of higher weights with better angles when weights must be reduced. These features are exactly our objectives when designing the algorithms.

C. Time Complexity

By analyzing the pseudo-codes, the average run time of our algorithms is $O(n \log n)$. The first phase takes $O(n)$ to segment a poly-line. In the second phase, a shadow list recording the difference of elements is generated and sorted. Then, a while-loop inserts elements into the list and decreases the count toward the limit, obtained from input as parameter a. So Phase 2 is $O(n) + O(n \log n) + O(an + c) = O(n \log n)$. Since a is an input constant less than n, the overall time complexity of our algorithm is $O(n \log n)$. The efficiency has also been demonstrated by running the experiment.

VI. Conclusions

This paper has explored the problem of how to determine a minimal number of spots for a camera to cover an entire range of scenes in a geographical region with minimal loss of details. We have proposed a visual model, which consists of projection, measurement of weights, and observation range, given a set of geometrical and geographical constraints.

Having defined the problem, we propose a 2-phase algorithm to gradually remove and/or merge camera positions and eventually meet the requirements. Most importantly, the algorithm finds the maximal observed weight for a single spot. For situations of multiple spots, the solution is NP-hard; thus, we propose a heuristic yet efficient approach and an algorithm with time complexity of $O(n \log n)$.

We have also experimentally verified our method with several realistic geographical scenarios. Our results greatly outperform the baseline where the weights are evenly distributed.

As the future work, more experiments will be conducted for the scenes with single dimensions or objects with multiples features. We will apply our approach to geographical visualization systems and evaluate its effectiveness in such applications.

VII. Acknowledgments

This research was supported in part by NSF award #1054629 and AFOSR award FA9550-14-1-0173.

References

[1] T. Wicker, “Kennedy is killed by sniper as he rides in car in Dallas; Johnson sworn in on plane,” *The New York Times*, p. 1, Nov. 22, 1963.

[2] A. V. Pandey, A. Manivannan, O. Nov, M. Satterthwaite, and E. Bertini, “The persuasive power of data visualization,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2211–2220, 2014.

[3] J. L. Cybulski, S. Keller, and D. Saundage, “Metaphors in interactive visual analytics,” in *Proceedings of the 7th International Symposium on Visual Information Communication and Interaction (VINCI)*, 2014, pp. 212–215.

[4] D. Heukelman and S. E. Obono, “Exploring the African village metaphor for computer user interface icons,” in *Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT)*, 2009, pp. 132–140.

[5] J. Heer and B. Shneiderman, “Interactive dynamics for visual analysis,” *Communications of the ACM (CACM)*, vol. 55, no. 4, pp. 45–54, 2012.

[6] V. Lam, S. Phan, D.-D. Le, D. A. Duong, and S. Satoh, “Evaluation of multiple features for violent scenes detection,” *Multimedia Tools and Applications*, vol. 76, no. 5, pp. 7041–7065, 2017.

[7] E. B. Nievas, O. D. Suarez, G. B. García, and R. Sukthankar, “Violence detection in video using computer vision techniques,” in *Proceedings of the 14th International Conference on Computer Analysis of Images and Patterns (CAIP)*, 2011, pp. 332–339.

[8] D. Milano, “Content control: Digital watermarking and fingerprinting,” White Paper, Rhozet, Harmonic Inc., 2012.

[9] S. Lee and C. D. Yoo, “Robust video fingerprinting for content-based video identification,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 7, pp. 983–988, 2008.

[10] A. Hampapur, K.-H. Hyun, and R. M. Bolle, “Comparison of sequence matching techniques for video copy detection,” in *Storage and Retrieval for Media Databases, SPIE Proceedings 4676*, 2002, pp. 194–201.

[11] J. Yuan, L.-Y. Duan, Q. Tian, S. Ranganath, and C. Xu, “Fast and robust short video clip search for copy detection,” in *Advances in Multimedia Information Processing, 5th Pacific Rim Conference on Multimedia (PCM)*, 2004, pp. 479–488.

[12] A. Datta, M. Shah, and N. D. V. Lobo, “Person-on-person violence detection in video data,” in *Proceedings of the 16th International Conference on Pattern Recognition (ICPR)*, 2002.

[13] C. Jansohn, A. Ulges, and T. M. Breuel, “Detecting pornographic video content by combining image features with motion information,” in *Proceedings of the 17th ACM International Conference on Multimedia (MM)*, 2009, pp. 601–604.

[14] B. Hoh, T. Iwuchukwu, Q. Jacobson, D. Work, A. M. Bayen, R. Herring, J.-C. Herrera, M. Gruteser, M. Annavaram, and J. Ban, “Enhancing privacy and accuracy in probe vehicle-based traffic monitoring via virtual trip lines,” *IEEE Transactions on Mobile Computing*, vol. 11, no. 5, pp. 849–864, 2012.

[15] V. Astarita, G. Guido, and V. P. Giofré, “Co-operative ITS: Smartphone based measurement systems for road safety assessment,” *Procedia Computer Science*, vol. 37, pp. 404–409, 2014.

[16] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson, “Virtual trip lines for distributed privacy-preserving traffic monitoring,” in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2008, pp. 15–28.

[17] L. Cehovin, M. Kristan, and A. Leonardis, “Robust visual tracking using an adaptive coupled-layer visual model,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 941–953, 2013.

[18] J. C. Chung, M. R. Harris, F. P. Brooks, H. Fuchs, M. T. Kelley, J. Hughes, M. Ouh-Young, C. Cheung, R. L. Holloway, and M. Pique, “Exploring virtual worlds with head-mounted displays,” in *Three-Dimensional Visualization and Display Technologies, SPIE Proceedings 1083*, 1989, pp. 42–52.

[19] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, “Surround-screen projection-based virtual reality: The design and implementation of the CAVE,” in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1993, pp. 135–142.

[20] T. P. Grantcharov, V. B. Kristiansen, J. Bendix, L. Bardram, J. Rosenberg, and P. Funch-Jensen, “Randomized clinical trial of virtual reality simulation for laparoscopic skills training,” *British Journal of Surgery*, vol. 91, no. 2, pp. 146–150, 2004.

[21] B. O. Rothbaum, L. F. Hodges, D. Ready, K. Graap, and R. D. Alarcon, “Virtual reality exposure therapy for Vietnam veterans with posttraumatic stress disorder,” *Journal of Clinical Psychiatry*, vol. 62, no. 8, pp. 617–622, 2001.

[22] R. M. Satava, “Virtual reality, telesurgery, and the new world order of medicine,” *Journal of Image Guided Surgery*, vol. 1, no. 1, pp. 12–16, 1995.

[23] A. R. Zamir and M. Shah, “Accurate image localization based on Google Maps street view,” in *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, 2010, pp. 255–268.

[24] L. Santos, J. Coutinho-Rodrigues, and C. H. Antunes, “A web spatial decision support system for vehicle routing using Google Maps,” *Decision Support Systems*, vol. 51, no. 1, pp. 1–9, 2011.

[25] S. Soro and W. Heinzelman, “A survey of visual sensor networks,” *Advances in Multimedia*, vol. 2009, 2009, doi:10.1155/2009/640386.

[26] M. P. Johnson and A. Bar-Noy, “Pan and scan: Configuring cameras for coverage,” in *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM)*, 2011, pp. 1071–1079.

[27] P. Kulkarni, P. Shenoy, and D. Ganesan, “Approximate initialization of camera sensor networks,” in *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN)*, 2007, pp. 67–82.

[28] H. Ma, M. Yang, D. Li, Y. Hong, and W. Chen, “Minimum camera barrier coverage in wireless camera sensor networks,” in *Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM)*, 2012, pp. 217–225.

[29] A. Kansal and F. Zhao, “Location and mobility in a sensor network of mobile phones,” in *Proceedings of the 17th ACM SIGMM International Workshop on Network and Operating Systems Support for Digital Audio & Video (NOSSDAV)*, 2007.

[30] K. Heath and L. Guibas, “Multi-person tracking from sparse 3D trajectories in a camera sensor network,” in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, 2008, pp. 1–9.

[31] D. Pedoe, *Circles: A Mathematical View*, ser. International Series of Monographs on Pure and Applied Mathematics. Dover Publications, 1957.