Real-Time Target Tracking Through Mobile Crowdsensing

Jinyu Shi and Weijia Jia

Department of Computer Science, Shanghai Jiao Tong University, Shanghai, China
mizaoyu@sjtu.edu.cn, jia-wj@cs.sjtu.edu.cn

Abstract. In order to track a single target in real-time across a large area, we proposed a novel method which combines mobile crowdsensing and existing sparse camera networks. Tracking is proceeded by reports, which either come from cameras or smart phone users. Intra-camera tracking is performed on selected cameras to identify target, and smart phone users can report with live photo or text when seeing the target. Such schema can largely help tracking target within blind area and increase the accuracy of target identification, due to the better identification ability of human eyes. Novel validation and correction mechanisms are designed to eliminate false reports, which ensures the robustness of our method. Compared with traditional cross-camera tracking methods, our design can be performed in real-time with better performance even if the target has appearance changes during the tracking. Simulations are done using road structures of our university, which validate the accuracy and robustness of our design.

Keywords: Target tracking · Crowdsensing · Real-time

1 Introduction

Real-time target tracking is critical in lots of applications, especially those involving personal safety, such as finding the missing child/elderly, or chasing criminals/suspects, where target’s real-time movement is wanted. Putting a GPS tracking device on target is the best approach, but such preparation barely happens. Therefore, existing video surveillance systems are largely used.

The most traditional way is pure manually - by watching surveillance videos. In most cases, the screening never catches up with the movement of the target. Even worse, target’s appearance in a video record might be left out due to negligence. Hence, this method is hard to be real-time and human error might further degrade the performance. Another way is through automatic tracking across cameras. However, real-time target tracking is still hard to achieve with such approach, as the system needs to first collect video data for a period of time, and then pick the observation with the highest possibility, which might even be a wrong one. Even worse, tracking might completely fail if the target had appearance change during the tracking, such as put on/off a cloth.

© Springer International Publishing AG 2017
DOI: 10.1007/978-3-319-64786-5_1
Hence, no existing method relying on camera networks can fairly achieve real-time target tracking, and status of target in blind areas is not accessible. In this paper, we propose to combine crowdsensing and existing camera networks to solve this problem. As smart phones are all-pervading these years, there are more opportunities to leverage GPS information and wireless communication of mobile phones. Besides, the increasing ability of camera embedded in smart phones provides feasibility for the mobile crowdsensing through videos, which is needed to extract the target from the background. Especially, from IPhone 6s, iPhones start to include the new function called “Live Photo”, which breaks the clear boundary between traditional images and videos. Compared with traditional videos, taking live photo has a more natural interface, as there is no need to switch to a video mode first and then press the shutter two times to start and stop recording. Along with its ability to record the scene before pressing the shutter, it can largely reduce phone users’ response time after seeing the target. Besides, it only takes 2 times the memory space as a regular photo, which is much less than a traditional video.

In our system, tracking is proceeded by observations from either cameras or phone users. By connecting multiple reports, the target’s movement can be recovered in real-time. Target’s appearance cues are used to calculate each observation’s possibility of containing the target from selected cameras and a camera report will be immediately fired if the result possibility is higher than certain level. Phone users can choose to fire a live photo report by taking a live photo of the target, or a text report when seeing the target. Due to the existence of visual evidences in camera and live photo reports, they can be validated. But how to validate text reports, which have no appearance information, is the main challenge of our design. To ensure the robustness of our system, we designed a mechanism to form a trace tree for unvalidated text reports, where spatio-temporal information and target’s velocity model are used. Such design can also help correct the direction information from users’ report. In the end, phone users will be rewarded depending on their reports’ truthfulness, where a live photo report should be rewarded more than a text report as it provides more information and costs users more resources. Therefore, users will be encouraged to fire a live photo report. Utilizing crowdsensing to track target can help capture more patterns of the target which cameras may miss or unable to obtain, such as different angles or appearance changes. As human eyes have stronger ability in identifying people, phone users can recognize the target even if the target changed clothes. The taken live photo can capture the target’s new appearance features for more accurate tracking with our approach to be discussed in this paper.

The rest of the paper is organized as follows. Section 2 reviews the related work. Detailed system model is presented in Sect. 3 and simulation results are shown in Sect. 4. The conclusion is given in Sect. 5.
2 Related Work

Several methods have been developed to achieve automatic tracking in sparse camera network. The process can be divided into two steps [3]. The first step is to track target within the view of a single camera, where lots of works have been proposed with good performance [1, 2], and some in real-time [4, 5]. As blind areas exist between cameras, target’s tracks between view fields of two spatially adjacent cameras need to be corresponded, which is the second step. The basic idea of most proposed approaches is to describe each object detected in cameras (observation) as visual appearance cues and then compare it to the target’s, which was observed previously. But it is difficult to determine which observation is the target when several observations have similar visual appearance cues.

Hence, several works started to consider the spatio-temporal cues, such as the topology of cameras and transition times, to help estimate the possibilities of target’s transiting from one camera to another. One main challenge is to learn the camera topology. Javed [8] used Parzen windows to estimate the camera topology from correspondence observation data. However, they assumed that the correspondence are known. But such correspondence is not easy to obtain and is exactly the target correspondence problem we are trying to solve. Makris [3], instead, presented a method which can automatically learn the inter-camera correspondence from a large set of observations based on the assumption that, two cameras are more likely to be linked if the correlation between the leaving and entering at a certain time interval is much more likely than a random chance.

In field of crowdsensing, there exists some works related to object tracking. Frey and Antone [9] proposed an end-to-end method of cross-camera tracking using crowd-sensed mobile video data. But their method is offline. Besides, they assume much overlap in different video’s views and the video data used are opportunistic, such as videos on YouTube, which user filmed and uploaded out of their own interest. [10] proposed a system to localize remote targets based on smartphone gathered sound and image inputs. In its acoustics-based localization method, several smartphones are required to gather the sound from the target. While, such requirement can hardly be met, as it means several smartphone users need to be close to the target and the target will produce sound.

3 System Model

Our approach combines crowdsensing and existing camera networks to achieve real-time target tracking within large areas. Assume that our crowdsensing system maintains a user pool $U$ including lots of smartphone users with different ages and genders, who are willing to help track targets and share us their GPS information. They will be rewarded based on their reports’ validation result using incentive mechanism we designed in [11] after the tracking is over.

Target’s information will be sent to phone users in certain area, who can choose to report in three different ways when they observe the target. By connecting reports, the target’s movement can be recovered. But each report has to
be validated (whether it was indeed the target) for the robustness of the system. Both appearance and spatio-temporal information are used in our design, but our main focus is on the latter one.

3.1 Preparation

Considering spatio-temporal information in a crowdsensing case, everytime a phone user reports an observation, a new node is added into the network, which can be at any random place. Therefore, we need to acquire the whole map so that each new added node can find its relationship with any other node, which include fixed cameras and previous report nodes. The road topology can be obtained from an existing digital map, but more information is needed to assist the validation of reports. Hence, the first step of building the system is to train such a map, which we propose to use crowdsourcing.

To get location information automatically from a camera report, location of each camera’s entry/exit zone on the map needs to be known. Here, entry/exit zone (EEZ) is the area in a camera’s view where object can enter into or exit from. We assume that at least a text description of each camera’s installation location exists. In better cases, the longitude and latitude are also given. Although the set of EEZs in each individual camera can be automatically learned [12], the correspondence of them on specific roads are unknown. That is where crowdsourcing plays in. We send the text description of camera’s location to phone users (or show it on the map if its longitude and latitude are given), along with a snapshot of that camera’s view. Then users can point out where each EEZ is located on the map. Such task will be done for each zone by multiple users to guarantee the accuracy, and the location pointed out by most users is taken.

Meanwhile, velocity models need to be trained, which will be used to estimate the transiting possibility of target from one observation to another. When phone users freely move in the area, the system correspond their traces on the digital map. The map can be abstracted as a graph of nodes and edges, where nodes contain intersections of roads and EEZs, and edges are the roads connecting them. Each edge has its length and several velocity probability density functions (pdfs). As it might take longer time to cross a crossroad or pass a street full of attracting stores, we need to build velocity pdfs for each edge. According to [13], the average walking speed of male and female, the young and the old, are all different. Therefore, we separate the phone users into different groups according to their gender and age, and a velocity pdf is built for each classification of people on each edge.

In this paper, we use kernel density estimation (KDE) to estimate the velocity pdf on certain edge using average velocities of traces from phone users in each classification. Since pedestrian walking speed follows a normal distribution [13], a normal kernel is used here. Our design can be easily extended to support tracking containing vehicles by building velocity pdfs for different kinds of vehicles as well. To adapt to changes, such learning process should go on continuously.
3.2 Overview of Three Kinds of Reports

Camera reports (CR) are fired by cameras. Different methods of intra-camera tracking based on appearance cues can be applied to our design [6, 7]. Commonly, an appearance possibility of this observed person being the target will be calculated. If it is higher than a pre-set threshold (a loose one), a camera report will be generated and snapshot of the video record will be sent to validators. In real-time target tracking, there must exist persons who are eager and dedicated to find the target until the target is brought back. Therefore, they can perform as validators, who can help validate the authenticity of a report in real-time. In this paper, validators are tracking requesters, police officers and phone users.

Our system doesn’t require all cameras to run intra-camera tracking. Instead, only objects coming in from directly connected zones of the last observation will be analyzed. Here, directly connected zones (DCZ) means, starting from the last observation's location, there exists a path to this entry/exit zone without passing any other cameras' field of view (FOV). To find all DCZs of a location, breadth-first search (BFS) is used here, where the location is set as the tree root. When an EEZ of camera is reached, we stop exploring its neighbor nodes, and mark it as a DCZ. The searching is finished until no neighbor nodes in the area can be explored. Target's appearance features extracted from previous camera or live photo reports are only sent to cameras containing DCZs, because if target will appear at a non-DCZ, he/she must first pass another camera's entry/exit zone (a DCZ). Due to the loose possibility threshold, we assume a camera report will be fired as long as the target passes a camera's view field.

A camera report (CR) contains only one information: the video record or from target's entrance to exit. The report will pass the validation if most validators confirmed the existence of target in the video. The camera report will then produce a camera observations (CO), which contain 6 information: CO = (vr, loc, t, d, tum, tl). Here, loc and t is the location and time when target exit from the video, which can be extracted from the video. d is the heading direction when target leaves loc, which is used for further tracking analysis. In camera report, this information can be calculated automatically: the direction out from the exit zone. tum is the velocity model of the target, which indicates the pattern of target’s moving speed compared with other people. tum is used to make up the inaccuracy using velocity pdfs when target has an unusual velocity pattern. The last information tl represents the trace length that target have gone through from the initial node.

Live photo reports (LPR) require the user to provide a live photo lp containing the target, and indicate target’s heading direction d (optional) on the map. As the user might not be very close to the target, he/she can also choose to point out target’s location loc on the map instead of using the GPS information of his/her phone. Therefore, LPR = (lp, loc, d). Similar to a camera report, it will also be sent to validators, and a live photo observation (LPO) will be generated if it passes the validation. Time information t can be directly derived from the live photo file. Target’s velocity model tum and trace length tl will also be updated. To sum up, LPO = (lp, loc, t, d, tum, tl).
Text reports (TR) do not require the live photo. But the time \( t \) when user saw the target needs to be reported, as there might be already several minutes passed before the user actually report. Therefore, \( TR = (loc, t, d) \). We do not validate the text report deterministically as soon as it is fired, because there is no hard evidence to prove its truthfulness. We consider it as a truthful observation first by inserting it into a trace tree, and update its \( tum \) and \( tl \) based on its father observation. Therefore, text observation (TO) = \( (loc, t, d, tum, tl) \). Notice that, the direction information provided by phone users might not be absolutely right. Therefore, our system provides self-correcting mechanism to avoid such mistakes. To help understand, Fig. 1 is given to show how these three kinds of reports help tracking the target together.

![Diagram showing the relationship between camera, phone users, and reports]

**Fig. 1.** Overview of reports

### 3.3 Start Tracking
To initialize a tracking process, a tracking request is needed, which can either include an initial camera observation of the target or a text description/photo of the target given by the requesters (who asked to find the target). In former case, the target’s appearance features can be extracted for further tracking and a camera observation will be generated at the target’s exit zone node. In the latter case, approximate location and time when target got missing are also required. Since no appearance features can be generated for cameras to run intra-camera tracking, the tracking task will be all on phone users until one live photo observation on the target has been found. In the following subsections, we will discuss the handling process of each kind of report.

### 3.4 Text Report
1. **Trace Tree:** To validate a text report (TR), a truthful possibility of it needs to be obtained. Since there is no visual evidence, we can only use spatio-temporal cues. In case of multiple consecutive TRs, we cannot simply
calculate the transition possibility from the previous TR to this one, as the previous one might be a false report. To avoid that, we develop a trace tree which may have multiple branches. Each branch represents a possible trace of target’s movement, and only one branch will pass the validation after a camera or live photo observation (CO/LPO) is generated. The tree root should be a CO/LPO, which has already been validated.

Everytime a TR is fired, if the previous observation is a CO/LPO, the system appends this TR to it as a child node. By setting one observation as the child of another, we assume target moved from father node directly to the child node. Any other reports between them along the timeline are considered as false reports in this branch, which must have joined other branches already. If the previous report is also a TR, a trace tree must have been built. Each time, the system appends current TR to the deepest branch as leaf node and then calculate the transition possibility based on that, because a deeper branch means more reports have been generated on this possible trace, which is more likely to be true. If the result possibility value is lower than a possibility threshold PT, it indicates a low transition possibility from the father node to this one for the target. Therefore, we remove the leaf node of the longest branch before appending current report, and repeat until the PT requirement is met. Detailed process to form the trace tree is illustrated in Algorithm 1.

Algorithm 1. Append TRn to Trace Tree
1: \( TT \leftarrow \) Text report trace tree
2: repeat
3: Append \( TR_n \) to the longest branch in \( TT \)
4: \( SP_{nd} \leftarrow \) all short paths without considering \( TR_n.d \)
5: for Each path \( P \) in \( SP_{nd} \) do
6: Calculate \( p_{v}^{d}(P) \) by solving problem 2
7: Adjust \( p_{v}^{d}(P) \) to \( p_{v}^{nd}(P) \) using Eq. 6
8: \( P_{nd}^{*} \leftarrow \arg \max_{P} \{ p_{v}^{nd}(P) \} \)
9: \( p_{v}^{nd} \leftarrow \max \{ p_{v}^{nd}(P) \} \)
10: Calculate \( p_{d}^{nd} \) using Eq. 6
11: Calculate \( p_{v}^{nd} = p_{v}^{nd} * p_{d}^{1-w} \)
12: if \( P_{nd}^{*} \) meets with \( TR_n.d \) then
13: \( p_{n} \leftarrow p_{v}^{nd} \)
14: else
15: \( SP_{d} \leftarrow \) all short paths considering \( TR_n.d \)
16: Calculate \( p_{n}^{d} \) by repeating line 5 to 11
17: \( p_{n} \leftarrow \max \{ b * p_{v}^{nd}, p_{v}^{d} \} \)
18: Correct \( TR_n.d \) according to selected path
19: if \( p_{n} < PT \) then
20: Remove \( TR_n \)'s father node from \( TT \)
21: \( p_{n} \geq PT \)

But if the initial report is a TR, no CO/LPO can be the tree root. In such case, it will be appended to a virtual tree root. If the second report is still a
TR, we first try append it to the first report. If the result possibility value is lower than the threshold, we append it directly to the virtual root. For later consecutive text reports, similar operations will be performed.

2. **Truthful possibility**: Suppose $O_n$ is the current text observation report. Its truthful possibility is defined as follows.

$$p_n = p_v(O_f, O_n)^w * p_d(O_{sf}, O_f, O_n)^{1-w}$$  \hspace{1cm} (1)

The possibility is composed of two parts. The first part $p_v$ indicates how likely the target will move from the father observation ($O_f$) in this branch to the current node within the transiting time and with the heading directions. While the second part $p_d$ shows the possibility of the trace that $O_{sf}$ (grandfather observation), $O_f$ and $O_n$ form. Therefore, $p_v$ represents the transiting possibility which relies on target's passing time, but $p_d$ is only related to the trace itself. $w$ is the weight that balances these two parts' influence on the final possibility $p_n$, and can be chosen based on experience.

3. **Transiting possibility**: To calculate $p_v$, a path that target might take from father node to current node needs to be found first. To do so, we need to consider the velocity pdf of each edge along the path in target's classification and the transiting time $tt = O_{n.t} - O_{f.t}$. Besides, as it is less likely for a person to take a large detour to the destination, we only consider all short paths (SPs) whose total length is no larger than $a \times l_{max}$, where $l_{max}$ is the length of the shortest path and $a$ is the coefficient of relaxation, which is larger than 1. Among those short paths, we need to pick out the most likely one. Besides, the heading direction of the father observation $O_f$, $d$ and the current one $O_n$, $d$ should be taken into considerations. If $O_f$ is a CO, the exit direction is 100% correct, as even if the target turned back, he would be captured by the same camera again. However, if $O_f$ is a LPO/TO, the target might return to the father node without being reported again. Unlike cameras, it is impossible for phone users to stay at the same place forever. But as $O_f$, $d$ has already been corrected (introduce later), we consider it as valid. However, we should have some doubts on $O_n$, $d$'s validity.

The process is as follows. We first find all short paths by not considering $O_n$, $d$, which we use $SP_{nd} = SP(O_1^f, O_n^{nd})$ to indicate in this paper. After the system select the most likely path $P_{nd}^*$ among them, if $P_{nd}^*$ meets with $O_n$, $d$, the finding process is finished. However, if $O_n$, $d$ is not met, the system find $SP_d = SP(O_f^d, O_n^{nd})$, which take $O_n$, $d$ into considerations. Similarly, a most likely path $P_{d}^*$ is found in this case. When selecting $P_{nd}^*$ and $P_{d}^*$, the first possibility $p_{nd}^*$ and $p_{d}^*$ have already be obtained. As considering the direction or not will affect the result of the second possibility $p_d^*$, the system calculate $p_{nd}^*$ and $p_{d}^*$ first under these two cases. Then, the truthful possibility $p_{nd}^*$ and $p_{d}^*$ can be obtained. However, as phone users had reported a direction after all, which should be more likely to be the correct one than any other random directions, we have to multiple $p_{nd}^*$ with a coefficient $b < 1$ to diminish it a little bit. After that, the larger one among $p_{nd}^*$ and $p_{d}^*$ will be picked, and $O_n$, $d$ will be replaced by the direction implied in the winning case. That is the self-correcting mechanism of the direction information, which is part of the process of forming a trace tree.
(a) Find all short paths
To find all short paths with length smaller than $a \cdot l_s$, the algorithm $K^*$ [14] is applied. The algorithm can find the $k$ shortest paths between a pair of vertices in a given weighted graph one by one in a non-decreasing order with respect to the total cost of the path. In our application, the cost of each edge is the length of the road. To find all short paths, we simply set a large $k$ initially, and stop the iteration when a newly found path's length breaks the restriction. The $K^*$ algorithm is developed based on the famous pathfinding algorithm $A^*$ [15], which uses heuristics to guide its search. $A^*$ has been proved to examine the fewest possible nodes than any other optimal algorithm employing the same heuristic, and $K^*$ inherited such advantage from $A^*$. Besides, $K^*$ can operate on-the-fly, which means that the full problem graph is not required to be presented in the main memory. Instead, the nodes can be generated as needed, which ensures a good performance for a large problem graph. Both the time complexity and space complexity of algorithm $K^*$ are $O(m + nlog n + k)$.

(b) Pick out the most likely path
After finding all short paths, we will solve an optimization problem on each path to calculate the possibility of target's traversing within the transiting time. The goal is to find an average velocity $v_k^*$ on each edge along the path that can result in an overall highest possibility. But since the length of each edge is different, longer edge's possibility value should have larger weight in the final result. Therefore, the optimization problem is expressed as follows.

$$
\max_{p'} p' = \frac{1}{\sum_{k=1}^{M} \text{Len}(k) p_{df}(v_k)} \sum_{k=1}^{M} \text{Len}(k) p_{df}(v_k)
$$

$$
st. \sum_{k=1}^{M} \frac{\text{Len}(k)}{v_k} = tt
$$

$$
v_k \geq \frac{\text{Len}(k)}{tt}, k = 1, \ldots, M
$$

(2)

$
\text{Len}(k)$ is the length of edge $k$, and $M$ is the total number of edges along the path. $p_{df}(v_k)$ represents the possibility that a person within target's classification will have an average velocity of $v_k$ when passing edge $k$. For each $v_k$, it must be larger than or equal to $\text{Len}(k)/tt$, because otherwise, it is impossible for the target to traverse the path within $tt$.

This nonlinear constrained optimization problem exists multiple methods to solve, such as SQP [16], Interior Point [17] and etc. An important step is to choose the initial point. In our system, we define the pattern of a person's velocity (velocity model) as a value $vm \leq 1$. For example, if $vm = 0.3$, it means this person usually walk faster than 30% of the public in his classification. For an edge $k$, the estimated velocity $E_k$ of a target with $vm$ on this road would thus satisfy $\int_{-\infty}^{E_k} p_{df}(k) = vm$. In other words, $cdf_k(E_k) = vm$, where $cdf(\cdot)$ is the cumulative distribution function. We use $cdf_k^{-1}(x)$ to indicate the estimated velocity on edge $k$ based on a velocity model value $x$. Hence, to find a
feasible initial point, we simply solve Eq. 3 to find an \( x \) that meets the transting time requirement. After \( x \) is found, \( V_0 = \{ v_{k0} = cdz_{k0}(x), k = 1, \ldots, M \} \) is the initial point value. Since a person are less likely to change his/her walking habit drastically, the initial velocities are set with the same velocity model value so that the result velocity model values on each edge is close to each other.

\[
\sum_{k=1}^{M} \frac{Len(k)}{cdf_{k}^{-1}(x)} = tt
\]  

(3)

(c) Correct \( p' \)

After the optimization result \( p' \) and \( V^* = \{ v^*_k, k = 1, \ldots, M \} \) are obtained, we need to correct the \( p' \) value to a more accurate possibility \( p \).

There are two considerations when adjusting \( p' \). Firstly, a person are less likely to deviate from his walking habit a lot. Therefore, a weighted \( vm \) deviation (vmd) from \( O_f . tvm \) is calculated. Since \( V^* = \{ v^*_k, k = 1, \ldots, M \} \) is obtained, the corresponding \( vm \) value on each edge \( k \) can be calculated: \( vm_k = cdf_k(v^*_k) \). Furthermore, the deviation on a longer edge should contribute more to \( vmd \). Hence, \( vmd \) is defined as in Eq. 4. If \( vmd \) is pretty small, even if the optimization result \( p' \) is rather low, we should adjust the possibility to a higher value, because the result \( vm \) fits the target’s walking habit. In the same way, \( p' \) with a higher \( vmd \) should be diminished more. Since the optimization result is obtained based on database trained by the public, if the target has a unique walking habit, the optimization result \( p' \) wouldn’t be high. Therefore, possibility needs to be adjusted to a higher value to avoid such mistakes.

\[
vmd = \frac{1}{\sum Len(k)} \sum_{k=1}^{M} Len(k)|vm_k - tvm|
\]  

(4)

Secondly, the validity of \( O_f . tvm \) should be considered. Initially, we set \( tvm = 0.5 \) as no target’s movement information can be obtained. Therefore, the initial several updated \( tvm \) might not be accurate, which shouldn’t play much role in adjusting the possibility. Therefore, the total length of the trace that has been used to update \( tvm \) (\( tvmLen \)) is included. The larger the \( tvmLen \), more influence \( vmd \) has. To achieve the above two requirements, we adjust \( p' \) as follows.

\[
vmp = \text{base}^{-vmd}, \quad p_v = \frac{\text{tvmLen}}{\sqrt{p'^* + \frac{\text{tvmLen}}{\beta}}}
\]  

(5)

Here, \( vmp \) indicates a \( vm \) possibility of the corresponding \( vm \) results along the path. The constant \( \beta \) is a number larger than 1, which helps transfer \( vmd \) into a possibility less than or equal to 1. \( \beta \) is another constant to fit the scale. With such definition, a larger \( tvmLen \) will make \( vmp \) more influential in the correction. The system calculates this \( p_v \) possibility for all short paths and pick one with the highest value, and then calculate its \( p_d \) possibility.
4. Trace possibility: Possibility \( p_d(O_{sf}, O_f, O_n) \) implies how likely the trace from \( O_{sf} \) to current node \( O_n \) passing \( O_f \) will occur. The definition is based on the assumption that people are less likely to take a large detour path to the destination. However, there is a time gap threshold \( T \), as the assumption about detour is only reasonable when consecutive reports don’t have a large time gap between each other. Therefore, if the transiting time \( t_t \) is larger than time threshold \( T \), we restart the tracking process and treat \( O_n \) as the first observation. If current report has no grandfather node, we simply set \( p_d \) as 1. In normal case, \( p_d \) is defined as follows.

\[
p_d = \frac{SPL(O_{sf}, O_n)}{SPL(O_{sf}, O_f) + SPL(O_f, O_n)}
\]  

(6)

Here, \( SPL(a, b) \) returns the length of the shortest path from \( a \) to \( b \). There might be 2 cases when finding short paths: consider \( O_n, d \) or not. When considering the direction, \( SPL \) finds a shortest path with \( O_n, d \) satisfied. Combining the two possibility \( p_v \) and \( p_d \) together, we can obtain the truthful possibility \( p_n \).

Update Target Velocity Model. Once \( p_n \) was obtained, a single path \( P^* \) from \( O_f \) to \( O_n \), and the average velocity on each edge \( e^* \) along the path have all been determined. It is time to update \( O_n, tvm \). In a text report trace tree, each node maintains a local \( tvm \), and each \( tvm \) is updated based on its father node's \( tvm \) through exponential weighted moving average (EWMA). We use EWMA as it can update the velocity model smoothly and reflect the recent change of target's walking speed. Besides, longer the edge, more influence the velocity model derived from this edge should have when updating \( O_n, tvm \). Therefore, we round the length of each edge \( k \) along the path to an integer \( t_k \), and then perform the following assignment for edge \( k \) for \( t_k \) times. Therefore, the update assignment will run \( \sum t_k \) times in total. Here, \( \alpha \) should be small, as multiple updates will be performed.

\[
tvm = (1 - \alpha) tvm + \alpha v m_k
\]  

(7)

Although TR will not be validated immediately, request still needs to be sent to users in new estimated area (within radius \( R \) from the reported location). As less users and cameras mean longer report interval time, \( R \) should be inversely proportional to camera and user density in the searching area.

3.5 Live Photo Report

Now we introduce the handling process of a live photo report (LPR). The first step is validation, because nothing will proceed if it fails. Except from confirming whether target exits in the live photo, validators also need to point out the target from several moving objects detected in the live photo. Although automatic target identification can be run on live photo, it might fail when there are much changes in target's appearance. For a camera report, it has to run automatically,
because it is impractical for validators to validate all detected objects in all those real-time video records. Besides, the reported direction can also be validated if the validator can deduct that from the live photo. If the report passes the validation, the system would extract target’s appearance feature, which will be used for further analysis in camera and sent to phone users.

If the previous report is a text report (TR), the TR trace tree needs to be validated, and such validation result of TRs can be used for incentive mechanism. The procedure is the same as forming the trace tree for TRs. The system considers the LPR as a TR in such case and find the right branch to append it to. Only ancestors of this LPR in that branch are kept and validated. All other nodes fail the validation. If the reported direction has already been validated, then we consider the direction when finding short paths. If it is not validated by validators, such appending process is still required in order to correct the direction information even if the previous report is not a text report.

Algorithm 2. Report Handling

1: if Camera Report (validated) then
2: Extract appearance features, send to phone users
3: Find DCZs and run intra-camera tracking
4: if Previous report is a text report then
5: Validate text report trace tree
6: else if Live Photo Report (validated) then
7: Extract appearance features, send to phone users
8: if Previous report is a text report then
9: Validate text report trace tree
10: else if Direction not validated by validators then
11: Correct direction
12: else if Text Report then
13: Send info to phone users in estimated area
14: if Previous report is not a text report then
15: Append to previous report (form trace tree)
16: else
17: Append to appropriate branch in trace tree
18: Update $t_{\text{tm}}$ and $t_{\text{f}}$ based on its father/previous report

3.6 Camera Report

Camera report also requires validation first. But validators only need to confirm target’s existence in the video. For CR, new DCZs should be found again from this observation, which intra-camera tracking will run on. Same as LPR, if the previous one is a text report, validation of the TR trace tree is required with the same procedure. But direction must be considered in this case, as direction information of a camera report is 100% correct. An overview of the handling process of these 3 kinds of reports are shown in Algorithm 2.
4 Simulation

To evaluate the performance of our schema in tracking the target, we did a simulation using the road structure of Shanghai Jiao Tong University campus (Fig. 2). Our simulations mainly focused on testing the performance in case of text reports, therefore no camera and appearance cues detail will be included. We asked 20 volunteers with smartphones to freely move within the area and share us their GPS information for one week. Their movement records were separated into multiple traces by a stopping interval time longer than 20 min. In total, 207 traces had been collected. For simulation, we used one trace everytime as the testing trace which simulated the actual target’s movement and all the other traces to train the velocity model.

For each test, we simulated the reports during the tracking period and applied our schema on them. From the start of the testing trace to its end, the tracking time was equally divided into x periods. Within each period, a random moment was selected as the time when one report is fired. Therefore, we simulated x reports for each testing trace. The report type was randomly picked from 4 categories with possibility shown in Table 2, where $\gamma_1 + \gamma_2 + \gamma_3 = 1$. Since live photo and camera reports are validated by the crowds, and those which fail the validation will not have any influence to the tracking result, we only considered validated LPR and CR in this simulation. For false text report, the reported location was selected randomly in the area. All other reports just used the actual location on the testing trace at the selected moment.

Table 1. Tracking accuracy, $P = 0.3$, $x = 5$

<table>
<thead>
<tr>
<th>$\gamma_1/\gamma_2$</th>
<th>0%</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>98.5%</td>
<td>96.6%</td>
<td>93.9%</td>
<td>90.2%</td>
<td>86.1%</td>
</tr>
<tr>
<td>10%</td>
<td>95.9%</td>
<td>93.8%</td>
<td>90.7%</td>
<td>86.7%</td>
<td>82.1%</td>
</tr>
<tr>
<td>20%</td>
<td>92.7%</td>
<td>90.4%</td>
<td>87%</td>
<td>82.5%</td>
<td>76.9%</td>
</tr>
<tr>
<td>30%</td>
<td>88.6%</td>
<td>85.8%</td>
<td>81.6%</td>
<td>76.3%</td>
<td>69.3%</td>
</tr>
<tr>
<td>40%</td>
<td>83.8%</td>
<td>80.2%</td>
<td>75%</td>
<td>68.4%</td>
<td>59.7%</td>
</tr>
</tbody>
</table>

![Simulation map](image)

Table 2. Simulated report types, possibilities and expected results

<table>
<thead>
<tr>
<th>Report type</th>
<th>Possibility</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR (true location, true direction)</td>
<td>(1-P)$\gamma_3$</td>
<td>pass validation, direction not modified</td>
</tr>
<tr>
<td>partial true (true location, wrong direction)</td>
<td>(1-P)$\gamma_2$</td>
<td>pass validation, direction corrected</td>
</tr>
<tr>
<td>false (wrong location)</td>
<td>(1-P)$\gamma_1$</td>
<td>fail validation</td>
</tr>
<tr>
<td>Live Photo/Camera Report (only validated ones)</td>
<td>P</td>
<td>/</td>
</tr>
</tbody>
</table>
Our schema was tested under different scenarios (different combinations of variable $x$, $P$, $\gamma_1$ and $\gamma_2$). For each scenario, we defined the tracking accuracy as the number of correctly handled text reports (have the expected result in Table 2 after applying our schema) divide by the total number of text reports among the simulations for total 207 test cases. Such simulation was repeated for 10 times to get an average of the tracking accuracy. In the simulations, $PT = 0.6$, $w = 0.9$, base $=100$, $\beta = 100$, $\alpha = 0.1$, $a = 1.5$, and $b = 0.9$, which are the other parameters used in our schema.

![Fig. 3. Tracking accuracy](image)

(a) $x = 5$, $\gamma_1 = 10\%$, $\gamma_2 = 10\%$

(b) $P = 0.3$, $\gamma_1 = 10\%$, $\gamma_2 = 10\%$

We first fixed $P = 0.3$, $x = 5$, and tested the tracking accuracy under different $\gamma_1$ and $\gamma_2$. When the percentage of false TRs and partial true TRs increase, the tracking accuracy gradually decreases but still maintains at an acceptable level. Detailed results are given in Table 1. Next, we fixed $x = 5$, $\gamma_1 = 10\%$, $\gamma_2 = 10\%$, and gradually decreased $P$ from 1.0 to 0. From the result shown in Fig. 3(a), we can see that even when $P = 0$, where all reports are text reports, our tracking schema is still able to handle all the reports with accuracy over 80%. To show the performance of our design under different density of reports, we change $x$ from 3 to 10, where $P = 0.3$, $\gamma_1 = 10\%$ and $\gamma_2 = 10\%$. As shown in Fig. 3(b), the denser the reports, the higher the tracking accuracy.

Among all 207 testing traces, there exist 16 traces with unusual velocities (walk faster or slower than most of the people). Our design successfully learned their velocity model gradually. Therefore, the simulation results on those special traces are almost the same as on the whole trace set.

5 Conclusion

In this paper, we proposed a new method to track single target in real-time, which combines crowdsensing and existing sparse camera network. Smart phone
users are involved to help build maps with velocity models, report observation of the target, and validate uploaded visual reports. Along with the existing camera networks, our method can help tracking the target in blind area, which pure camera tracking cannot achieve. An inventive mechanism is proposed to tackle the truthfulness of user reports, which ensures the robustness of our design. Simulation results are presented to validate our design.

Acknowledgement. This work is supported by Chinese National Research Fund (NSFC) Key Project No.61532013; National China 973 Project No.2015CB352401; Shanghai Scientific Innovation Act of STCSM No.15JC1402400 and 985 Project of Shanghai Jiao Tong University with No.WF220103001.

References