2017 IEEE 37th International Conference on Distributed Computing Systems

EV-Matching: Bridging Large Visual Data and Electronic Data for Efficient Surveillance

Gang Li*†, Fan Yang*†, Guoxing Chen*†, Qiang Zhai*†, Xinfeng Li†, Jin Teng†,
Junda Zhu‡, Dong Xuan†, Biao Chen‡, and Wei Zhao‡
† Dept. of Computer Science and Engineering, The Ohio State Univ.
{lgang, yanfan, chenguo, zhaiq, lixinf, tengj, xuan}@cse.ohio-state.edu
‡ Univ. of Macau {jdzhu, bchen, weizhao}@umac.mo

***Abstract*—Visual (*V*) surveillance systems are extensively deployed and becoming the largest source of big data. On the other hand, electronic (*E*) data also plays an important role in surveillance and its amount increases explosively with the ubiquity of mobile devices. One of the major problems in surveillance is to determine human objects' identities among different surveillance scenes. Traditional way of processing big *V* and *E* datasets separately does not serve the purpose well because *V* data and *E* data are imperfect alone for information gathering and retrieval. Matching human objects in the two datasets can merge the good of the two for efficient large-scale surveillance. Yet such matching across two heterogeneous big datasets is challenging. In this paper, we propose an efficient set of parallel algorithms, called *EV-Matching*, to bridge big *E* and *V* data. We match *E* and *V* data based on their spatiotemporal correlation. The *EV-Matching* algorithms are implemented on Apache Spark to further accelerate the whole procedure. We conduct extensive experiments on a large synthetic dataset under different settings. Results demonstrate the feasibility and efficiency of our proposed algorithms.**

I. Introduction

Visual surveillance is a major way to monitor people's locations, behaviors and activities. Surveillance cameras are widely deployed in public places, especially in safety-sensitive areas, and the amount of cameras is huge. According to reports, there is one surveillance camera for every 14 people in Britain, and at least $24,000$ cameras have been installed in Chicago [11]. Surveillance videos have become the biggest big data we need to deal with.

Electronic data also plays an important role in surveillance. Currently, there are roughly 6 billion active cell phones in the world [20]. Almost every person has one or multiple electronic devices, including mobile phones, tablets and laptops. These devices constantly emit electronic signals to communicate with network infrastructure. These signals carry the holder's electronic identity (*EID*), such as IMSI for GSM, UMTS and LTE, WiFi MAC and Bluetooth ID which can be utilized to locate and track the device holder in surveillance systems.

*The first four authors are co-primary authors. This work is partially supported by Natural Science Foundation of China (NSFC) under grants: 61373115, 61402356 and 61572398, the Macau FDCT project 009/2010/A1, 061/2011/A3 and University of Macau project MYRG112, the Fundamental Research Funds for the Project Funded by China Post doctoral Science Foundation (2015M572565) and the Fundamental Research Funds for the Central Universities (xkjc2015010). Any opinions, findings, conclusions, and recommendations in these publications are those of the authors and do not necessarily reflect the views of the funding agencies.

For surveillance applications, one of the major problems is how to precisely determine human objects' identities among different surveillance scenes. Traditionally, visual (*V*) data and electronic (*E*) data are processed separately for such purpose. However, *V* data and *E* data are imperfect alone for information gathering and retrieval. Yet they are highly complementary to each other for surveillance purposes. *V* data is intuitive and accurate but searching through the massive videos, either by human operators or through computer vision techniques, is inefficient. Compared with *V* data, *E* data is relatively light-weight, and electronic surveillance based on localization and tracking is readier to be carried out. But the propagation of *E* signal is highly dependent on a good transmission environment. The range error of *E* localization is relatively large. According to the characteristics of *E* and *V* data, we can merge the good of both worlds for efficient large-scale surveillance.

In this paper, we study the combination of *E* data and *V* data for better surveillance. Informally put, we want to find the visual images (visual identity or *VID*) of a person carrying a specific mobile device in massive surveillance videos. For example, a crime happened and the police have the *EID*s appearing around the crime scene when it occurred. They want to figure out the activities of these *EID*s' holders in surveillance videos over previous months in order to find the suspects. Technically we intend to match the *EID*s in electronic location logs with *VID*s in surveillance videos. With this matching, we are further able to fuse these two big and heterogeneous datasets, and retrieve the E and V information for a person at the same time with one single query. We call this *EV-Matching*. Spatiotemporal information is what these two datasets share in common and what we utilize to perform the fusion. To deal with these two *EV* datasets which are hugely diversified in quality and processing, a two-stage *E*-filtering and *V*-identification strategy was proposed to reduce the visual processing burden and get accurate matching results in [24]. However, this method only works on single *EID-VID* matching and it does not work in fusing big *EV* datasets. Naive parallelization of this algorithm to make multiple machines work on different *EID-VID* pair brings no further benefit because of duplicated computation during the matching.

We design distributed algorithms to streamline efficient and accurate *EV-Matching*. We develop the *EID* set splitting

1063-6927/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDCS.2017.89

algorithm to reduce the processing burden of *V* data. The *E* location dataset is firstly processed and a fraction of video frames where each *EID* may appear are selected. Then the common *VID*s appearing in the video frames filtered in the previous stage will be figured out as the matched *VID*s of the corresponding *EID*s. We call this stage *VID* filtering. With *EV-Matching*, multiple *EID-VID* pairs can be matched at the same time. In fact, a large portion of the video frames selected with the *EID* set splitting algorithm can be reused for other *EID*s. This will further reduce the processing burden of *V* data. We extend the whole procedure to a distributed version and implement it with MapReduce.

Moreover, our *EV-Matching* algorithm also supports elastic matching sizes. This is very important for surveillance because the *EID-VID* range of interest may be different, from a single person to the entirety of people in a city. With this algorithm, we can find the *VID* corresponding to one specific *EID* without matching other *EID*s and *VID*s. We can also choose to match multiple *EID* and *VID* pairs simultaneously, or even achieve universal dataset matching. Universal matching is the extreme case, which actually gets each *VID* in the whole videos labeled with its corresponding *EID*. After universal labeling, it will be more efficient to do future queries because all the *EV* raw data has been processed and indexed. Note that the larger the matching size is, the less time it costs per *EID-VID* pair.

Though our goal is to explore the problem of fusing large surveillance visual data and electronic data, we start with a simplified and clean model. Our *EV-Matching* algorithm assumes the *EID* and *VID* are both readily available for the person of interest. However, such assumption is not always true in real-world environment. For example, some people do not carry any electronic device so they have no *EID*s. Also, due to occlusion or imperfect vision algorithms, we may fail to extract the *VID*s for some people. In such situations, we adapt our algorithm and try to get the best matching results. Although human intervention may be involved, our *EV-Matching* can still shoulder human's workloads.

In short, we have the following key contributions:

– We design the *EV-Matching* algorithm to fuse big *EV* datasets for efficient large-scale surveillance. *EID* set splitting is developed to reduce the visual processing burden. Practical settings are also considered to accommodate our algorithms to real-world situations.

– The whole matching procedure is extended to the MapReduce framework. We utilize the mechanism of $(key, value)$ shuffle in MapReduce to implement the key operation in our algorithms. This design not only improves the time-efficiency in a parallel way, but also reduces the visual processing burden because of the overlap in filtered video frames.

– Our algorithm supports single, multiple and universal *EID-VID* matching. Multiple *EID*s can be matched to their corresponding *VID*s simultaneously. Even universal labeling can be performed in an efficient way.

– We implement our algorithm on Apache Spark [2]. Large-scale evaluations are conducted on a 14-node cluster and the results demonstrate the matching accuracy and efficiency of our algorithm.

The rest of this paper is organized as follows. Section II reviews related work. Section III introduces the problems we are going to address. Section IV details our *EV-Matching* algorithm design and Section V presents algorithm parallelization with MapReduce. Section VI reports our experiment results. Finally, Section VII concludes this paper.

II. Related Work

Efficient large-scale surveillance systems have been a hot research area in recent years. There are three categories of related work.

A branch of video surveillance research particularly of interest is human re-identification. Re-identifying the person from different camera views is the basis for consistent labeling across multiple cameras [3]–[6]. Ding et. al. [8] propose M-Clip for retrieving human objects among large surveillance videos. Doretto et. al. [9] discussed the human re-identification problem in camera networks and used appearance-based algorithms to address visual surveillance needs. Zhao et. al. [27] used mid-level filters learned from patch clusters to identify visual patterns and distinguish persons. Yang et. al. [26] fused the color distributions over different color names to generate features for human re-idenfication.

More similar to our work are approaches that fuse sensing data from multiple sensors for identifying or tracking people. Fan et. al. [15] proposed a particle-filtering based motion sensor fusion approach for self-tracking. Roetenberg et. al. [21] proposed a system that consists of magnetic sensors and inertial sensors for motion tracking. Teixeira et. al. [22], [23] leveraged the existing camera infrastructure with initial sensors to identify and localize people. There are also some literatures focusing on electronic and visual sensing data. Teng et. al. [24] proposed an electronic and visual sensor fusion based system to identify an object's appearance model from visual data. Papaioannou et. al. [19] proposed an indoor positioning system which fuses visual detections captured by camera infrastructure with WiFi radio data. EV-Linker [10] and IdentityLink [18] both tried to link the device with the user by leveraging eavesdropped wireless data and visual signals. Li et. al. [14] implemented ForeSight system which can dynamically integrate the information observed in the visual domain and the electronic domain to match the vehicles observed with high accuracy.

Big spatial data fusion on moving objects aims to efficiently process a database on moving objects and their locations. The key problems include: location model (imprecision of localization), indexing (R-tree, Quadtree), spatial and temporal range query and object dynamics (information update). There are works on clustering of moving trajectories for traffic shedding [13] and extensive queries [25]. Recent works [1] consider Hadoop-GIS on MapReduce. The main challenges are skew of spatial data (load imbalance) and objects in the boundaries (inaccurate results). Online profiling [16] can be used to improve load balance in big data processing platforms.

III. Problem Definition

Electronic data and visual data are pervasive and both of them can be leveraged to capture people's positions in public places. For example, base stations can sense and estimate a mobile phone user's location, which can be called *E-Location*. Meanwhile, public cameras can capture human figures and estimate their real-world locations, which can be called *V-Location*. Within a period of time, such as one day, one *EID*'s *E-Location*s accumulate and an entire *E-Trajectory* is generated. *V-Trajectory* is a linkage of the *V-Location*s of a single person with human re-identification or visual tracking methods. Then one person has one *E-Trajectory*, if the person uses only one phone in this period of time, and multiple *V-Trajectory* segments, because of occlusions and appearance variations. If we want to match some *EID*s with their corresponding *VID*s, we need to associate *E-Trajectories* and *V-Trajectories* of a group of people in large temporal and spatial scale.

Next, we describe the data we consider in this paper, and the problem we are trying to deal with.

A. Data description

– Raw *E-Data*: It contains *EID*s (e.g., IMSI, WiFi MAC) and timestamps when these *EID*s were captured. *E-Location*s can be estimated using the position of the devices or base stations that capture these *EID*s, or using other localization methods if more information is available, such as electronic signal strength.

– Raw *V-Data*: Basically, it contains timed video data. *VID* and their locations can be extracted from videos. Such extraction can be very time intensive.

B. Problems

We have a large amount of *E-Data* and *V-Data* generated by the same group of people across different time periods within a large area. We are trying to deal with the following problems:

– *Matching:* How to match the same person's *EID* and *VID*. To begin with a clean matching model, we have the following assumptions:

1) *VID consistency:* In a period of time, we can successfully extract the same *VID* with some methods (such as appearance similarity) with a high probability.
2) *VID completeness:* When the *EID* of a person is recorded, we assume the *VID* of the same person is also in *V* dataset. (We will weaken this assumption in practical settings in Section IV.)

From the first assumption, we can connect the *V-Location*s of the same person to generate the *V-Trajectory*. Meanwhile, we have each *EID*'s location records at different timestamps. This kind of spatiotemporal information is able to distinguish one *EID*'s large-scale trajectory from the others' (two people are rarely at the same position all the time). Based on this and the second assumption, we can find the *VID* which is most likely to have the same trajectory with this *EID*, and matching is accomplished.

– *Parallel Processing:* How to parallelize the matching procedure to match multiple people at one time on a large-scale *EV* dataset. This is also a big data problem. We aim to implement our algorithm in MapReduce framework which is one of the most popular frameworks for big data processing. When using MapReduce, we need design proper *map* and *reduce* functions to fit our matching algorithms into the distributed framework.

The problems of matching and parallel processing will be addressed in Section IV and V, respectively.

IV. EV-Matching Algorithm

In this section, we first explain how we try to tackle this *Matching* problem. Next, we propose algorithms for both ideal and practical settings, followed by necessary analysis.

A. Preliminaries

Directly extracting all information such as *E-Location*s and *V-Location*s for matching is too time consuming, which is undesirable. To mitigate this problem, we propose to use "rough" *E-Location*s and *V-Location*s to select a small portion of the whole data, and perform detailed extraction only on this portion of data. So we introduce the concept of *EV-Scenario*.

Definition 1. *EV-Scenario* is a snapshot of the *EID* and *VID* sets appearing in a specific spatial region at a single time point. It is comprised of *E-Scenario* (with *EID* set only) and *V-Scenario* (with *VID* set only).

We divide the whole spatial region into a bunch of smaller regions that we call scenarios. A scenario can be the region covered by one camera, the region of one room covered by several cameras or even a hexagonal cell if we generate the view of the whole region by combining the views of all cameras and divide it uniformly as shown in Figure 1. As described in Section III, all *E-Data* and *V-Data* are embedded into *EV-Scenarios*. At a single time point, *EID*s appearing in one scenario make up an *E-Scenario*. An *E-Scenario*'s corresponding *V-Scenario* is the set of *VID*s appearing simultaneously in the same scenario. An *E-Scenario* together with its corresponding *V-Scenario* comprise an *EV-Scenario*.

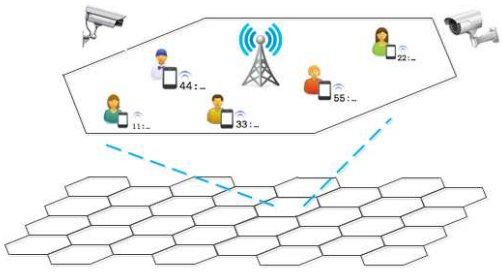


Fig. 1: EV scenario

After introducing the concept of *EV-Scenario*, matching *EID* and *VID* by comparing the whole *E-Trajectory* and *V-Trajectory* becomes matching the "sub-trajectories" of *EID* and *VID*. That is, to find a list of *EV-Scenario*s such that only

one *EID* and one *VID* appear in all these *EV-Scenario*s, and we can match them safely.

However, searching for such a list of *EV-Scenario*s for each person separately may be not efficient enough because extracting information, especially *VID*, is very time consuming. If we can reuse *EV-Scenario*s for different people, the efficiency will be improved. The basic idea is that one *EV-Scenario* can help distinguish all people inside the *EV-Scenario* from all those outside. So one *V-Scenario* has the potential to be reused by all *EID*s contained in the corresponding *E-Scenario*. And we only need to process this *V-Scenario* once.

Another idea to further reduce the number of *EV-Scenario*s we need: *VID*s that have been already matched may help distinguishing those remain unmatched. For example, consider such two *E-Scenario* : one contains EID_1 and EID_2 and the other contains only EID_1. Matching EID_1 and VID_1 is simple because the only VID_1 in the second *V-Scenario* can be safely matched to EID_1. To match EID_2, we can get its corresponding VID_2 from the first *V-Scenario* by ruling out VID_1 that appears in the second *V-Scenario*.

Ideally we assume the consistency between *EID*s and *VID*s. That is, if a person appears in one scenario, his or her *EID* and *VID* are exclusively contained in exact one same *EV-Scenario* at one time point.

We deal with this ideal case first and discuss the practical settings afterwards.

B. Algorithm for Ideal Setting

Under the ideal setting, if only one *EID* appears throughout a list of *E-Scenario*s, only one *VID* would appear throughout the list of corresponding *V-Scenario*s, and vice versa. Since dealing with *E-Data* is much faster than *V-Data*, we have the following scheme for *EV-matching*.

Our scheme has two stages: E stage (dealing with *E-Scenario*s) and V stage (dealing with *V-Scenario*s). In E stage, *EID* set splitting is performed to get a list of *E-Scenario*s for each *EID* such that each *E-Scenario* list can distinguish the corresponding *EID* from others. Then the *VID* corresponding to each *EID* can be extracted from the corresponding *V-Scenario* list by *VID* filtering in V stage.

1) EID Set Splitting: In this stage, we try to find a list of *E-Scenario*s for each *EID* to distinguish them. The basic idea is that one *E-Scenario* can distinguish the *EID*s within it from the rest. We introduce *set splitting* mechanism to achieve this goal.

Denote a group of *EID*s that are undistinguishable with each other by a set with these undistinguished *EID*s as its elements. Initially, all *EID*s are in one set because no information is available to distinguish them. One *E-Scenario* can split one undistinguishable *EID* set into to two undistinguishable *EID* sets: one containing *EID*s from the original set that appear in the *E-Scenario* and the other containing the rest *EID*s in the original set. We call this procedure *set splitting*. Such splitting operations can be performed iteratively. Notice that during the set splitting procedure, each *EID* is contained in one and only one set. We call the set that consists of all these undistinguishable *EID* sets a *partition*. At the beginning, the partition contains only one undistinguishable *EID* set which contains all *EID*s. By the end of the algorithm, hopefully, each set in the partition contains only one *EID*. And a list of *E-Scenario*s that can distinguish each *EID* can be found accordingly. Following are the notations and algorithms are presented in Algorithm 1.

– $U_{eid} = \{EID_1, EID_2, \cdots, EID_n\}$ is the universal set of *EID*s

– P_{eid} is a partition on U_{eid}, $P = \{A_1, A_2, \cdots, A_k\}$, $\bigcup_{i=1}^{k} A_i = U_{eid}$, and $\forall i \neq j$, $A_i \bigcap A_j = \emptyset$, each A_i is an undistinguishable *EID* set

– C is an *E-Scenario*. It is presented as a set of EIDs appearing in the *E-Scenario*

– $S_{sce} = \{C_1, C_2, \cdots, C_m\}$ is the universal set of *E-Scenario*s.

Algorithm 1 Algorithm for ideal setting

```
MAIN()
    ES = SetSplit({U_eid}, S_sce)
    return VFilter(ES)

SetSplit(P_eid, S_sce)
    while ||P_eid|| < n & S_sce ≠ ∅ do
      select C from S_sce
      S_sce = S_sce \ {C}
      P_eid = SplitBy(P_eid, C)
      if P_eid changes then
        Record C
      end if
    end while
    return Recorded Cs

SplitBy(P_eid, C)
    P' = ∅
    for A ∈ P_eid do
      A' = A ⋂ C
      P' = P' ⋃ {A', A \ A'}
    end for
    return P'
```

Remark. If an *EV-Scenario* can not effectively split one set, which may happen when the *EV-Scenario* contains all the *EID*s in the set or none, such *EV-Scenario* will be skipped. Only effective *EV-Scenario* will be recorded.

2) VID Filtering: After *EID* set splitting, we get a list of effective *E-Scenario*s for each *EID*. This list can be treated as a large-scale, course-grained trajectory of this *EID* and no other *EID*s share the same trajectory. Then it is possible to find the *VID* corresponding to this *EID* with person re-identification methods because this *VID* is supposed to be the only one having the same trajectory with this *EID*. In each corresponding *V-Scenario* on this trajectory, several human figures can be extracted with human detection methods. Appearance or even gait features [12] of each *VID* can be

obtained. The similarity of two *VID*s $sim(VID_1, VID_2)$ is defined in Equation 1. f_{VID} represents the feature vector of a specific *VID* and $dist(f_1, f_2)$ is the normalized vector distance between f_1 and f_2.

$$sim(VID_1, VID_2) = 1 - dist(f_{VID_1}, f_{VID_2}) \quad (1)$$

VID similarity reflects the probability that two *VID*s represent the same person. Let $P(VID_1 = VID_2)$ denote the probability that two *VID*s represent the same person. P is positive-correlated with sim, but it is difficult to define the conversion function from sim to P. Assuming there are k *VID*s in a scenario S: $VID_1, VID_2, \cdots, VID_k$, we can simplify the probability that one *VID*, say $VID^\star$, is in S to $P(VID^\star \in S) = max_{i=1,\cdots,k}\{sim(VID^\star, VID_i)\}$, and the probability that $VID^\star$ is not in S to $P(VID^\star \notin S) = 1 - max_{i=1,\cdots,k}\{sim(VID^\star, VID_i)\}$ [24].

If for an *EID**, the scenario list selected from *EID* set splitting algorithm is $\{S_1, S_2, S_3\}$ and we know that the corresponding $VID^\star$ appears in S_1 S_2 and S_3. Then for each *VID* in these scenarios, $P(VID = VID^\star) = P(VID \in S_1)P(VID \in S_2)P(VID \in S_3)$. In every scenario, we choose the *VID* with the largest probability to be $VID^\star$ as the final result.

So far, we successfully match *EID** with corresponding $VID^\star$ after processing only a few *V-Scenario*s. This procedure can be repeated for other *EID-VID* matching.

C. Algorithm for Practical Setting

1) Practical Settings: In real world situation, *E-Scenario*s and *V-Scenario*s are not necessarily consistent, i.e. the *EID* and *VID* of the same person may not be in the same *EV-Scenario*. Major practical issues are as follows:

– *Drifting EID*: Some *EID*s may appear in wrong *E-Scenario*s (neighbor cell) because of electronic noise. This is possible especially for those who are actually located near the boundary of a scenario.

– *Missing EID*: For people who do not carry any electronic device, they have no *EID*. This will result in some additional *VID*s appearing in *V-Scenario* lists when matching other *EID*s with their corresponding *VID*s.

– *Missing VID*: Due to occlusion and miss detection, we may fail to extract the *VID*s corresponding to a *EID* from some *V-Scenario*s.

In this setting, the previous algorithm can not be applied directly. Modifications are needed. We propose *vague zone* in *EID* Set Splitting to deal with *drifting EID* and *matching refining* process to handle *missing EID* and *missing VID*.

2) EID Set Splitting: Due to electronic noise, some *EID*s may be slightly out of the scenarios which they should be in. To tackle this problem, we introduce vague zones in the scenarios. As shown in Figure 2, the area of a scenario is divided into two parts: inclusive zone (the region far from the border) whose *EID*s are considered confidently included in this scenario, vague zone (the region near the border) whose *EID*s are also included and are marked as vague (which means that it is not sure if the *EID*s should appear in this scenario). The area outside the vague zone (outside of the scenario) is denoted as exclusive zone whose *EID*s are excluded from this scenario.

Accordingly, we slightly modify the definition of *EV-Scenario* by extending one single time point to a certain period of time. Specifically, we count the occurrence of *EID*s within this time period. The *EID*s which appear mostly are considered in the inclusive zone, the ones who appear adequately are considered in the vague zone, and the ones who appear occasionally are considered in the exclusive zone. Each *EID* within an *EV-Scenario* is associated with an attribute value which is either inclusive or vague indicating whether the *EID* is in the inclusive zone or vague zone.

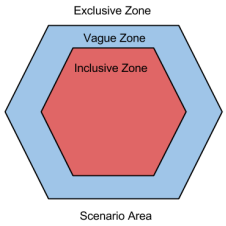


Fig. 2: Vague zone of scenario in spatial domain

Next, we describe our algorithm. Intuitively, we should try to avoid using *EV-Scenario*s with the target *EID* in the vague zone to distinguish that *EID*. So when performing set splitting, we should not simply split one set into two sets as previous mentioned. In fact, we focus on splitting the *EID*s that are inclusive both in the original set and the given *E-Scenario*. One example is shown in Figure 3.

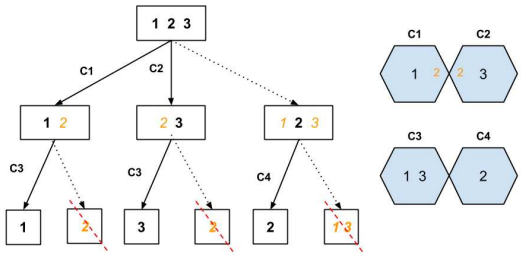


Fig. 3: An example of algorithm for practical setting

3) VID Filtering: After EID set splitting, we get a list of effective *E-Scenario*s for each *EID*. Due to introduction of vague zone, the corresponding *VID* may or may not show up in the corresponding *V-Scenario*s. Notice that for the list of effective *E-Scenario*s of a given *EID*, this *EID* only appears in the inclusive zone or the exclusive zone. We can use the same similarity measurement as in the ideal setting. However, we may not get acceptable result in one run because of the practical issues we mentioned before. We need some refining mechanism.

4) Matching Refining: *VID* missing is the the most challenging problem for *VID* filtering. The performance depends on the missing rate. If missing rate is 0, it becomes the the

ideal setting. But if the missing rate is relatively high, *VID* filtering may fail to find the right *VID*. To handle this, we can collect the corresponding *EID*s of these undistinguished *VID*s and go through the EID set splitting and VID filtering again to refine the result until it is acceptable. However, if missing rate is too high, human intervention may be required to manually find the right *VID*.

Algorithm 2 Algorithm for matching refining

```
MAIN()
    P_eid = {U_eid}
    while true do
      ES = SetSplit(P_eid, S_sce)
      MM = VFilter(ES)
      if MM is acceptable then
        break
      end if
      update P_eid
    end while
    return MM
```

D. Algorithm Analysis

We first analyze the correctness and complexity of the EID set splitting algorithm for ideal setting.

1) Analysis for Ideal Setting: Correctness of the algorithm is proved as follows.

Theorem 4.1. In ideal setting, with the *E-Scenario*s recorded by the end of Algorithm 1, all *EID*s can be distinguished in order if there are adequate *EV-Scenario*s generated from *EV* datasets.

We sketch the proof here. We use a binary tree to demonstrate the status of the set splitting procedure:

– Each node represents an undistinguishable *EID* set. The tree root represents the initial set which contains all *EID*s;

– One effective *E-Scenario* splits one node (set) into two nodes (sets): left child contains the *EID*s appearing in the *E-Scenario* and right child contains the rest;

– By the end of the algorithm, each leaf node contains only one *EID*.

The post-order traversal of this binary tree gives a order according which all *EID*s can be distinguished, the process is as follows:

– For the left-most child (it should be a leaf according to the splitting process), we can distinguish the *EID* by looking for the intersection of the *E-Scenario*s used to split all its ancestor nodes since the *EID* is the only one appearing in all these *E-Scenario*s.

– Exclude the distinguished *EID*s from all the nodes and recorded *E-Scenario*s. The left-most node will become an empty set, delete it. The set contained in the parent node of the deleted node will become identical with the set contained in its right child node. So merge the parent node and its right child node. After these exclusion, deletion and merge operations, the binary tree and all recorded *E-Scenario*s will become a solution to the subproblem of distinguishing all the *EID*s except for the distinguished *EID*.

– Repeat previous two steps until all *EID*s are distinguished.

Now we analyze the efficiency of the algorithm.

Theorem 4.2. In ideal setting, $\log(n) \backsim (n-1)$ effective *E-Scenario*s are adequate to distinguish n *EID*s.

We sketch the proof here.

– Lower bound: Given an *E-Scenario*, there are only two cases for each *EID*: appear or not. So $\log(n)$ *E-Scenario*s can distinguish n *EID*s at most. However, such lists of *E-Scenario*s do not always exist.

– Upper bound: Consider the number of sets in the partition. At the beginning, the partition contains one set (with all n *EID*s). Each effective scenario increase the number of sets in the partition by at least one. The maximum number of sets in the partition is n (each set contains exactly one element). So $(n-1)$ effective scenarios are sufficient to distinguish all n *EID*s.

Note that this efficiency is very important to reduce the visual processing burden.

2) Analysis for Practical Setting: Since there may be multiple vague *EID*s in multiple sets, the whole *EID* distinguishing process slows down. More *E-Scenario*s are needed to distinguish *EID*s according to the percentage of vague *EID*s.

Correctness of the algorithm is proved as follows.

Theorem 4.3. In practical setting, with the *E-Scenario*s recorded by the end of Algorithm 2, all *EID*s can be distinguished in order if there are adequate *EV-Scenario*s generated from *EV* datasets.

The proof is similar as the ideal setting. Except that when one effective *E-Scenario* splits one node: left child contains the *EID*s appearing in the *E-Scenario* with inclusive attributes (if they are inclusive in both the *E-Scenario* and the original node) or vague attribute (otherwise), while the right child contains the rest *EID*s with their original attribute and *EID*s appearing in both the *E-Scenario* and the original node with vague attribute;

Now we analyze the efficiency of the algorithm.

Theorem 4.4. In practical setting, $\log(n) \backsim n^2$ effective *E-Scenario*s are adequate to distinguish n *EID*s.

– Lower bound: Similar as the ideal setting.

– Upper bound: In the worst case, we need n effective *E-Scenario*s to distinguish each *EID*. So n^2 effective *E-Scenario*s are sufficient to distinguish all n *EID*s.

V. Parallelization with MapReduce

In the large-scale *EV* matching problem, the dataset usually has a large spatiotemporal span and its volume is big, especially for the visual data. We propose to use MapReduce, one of the most popular frameworks for big data processing, to parallelize *EV* matching algorithm. Both *EID* set splitting and *VID* filtering algorithms are implemented in MapReduce framework.

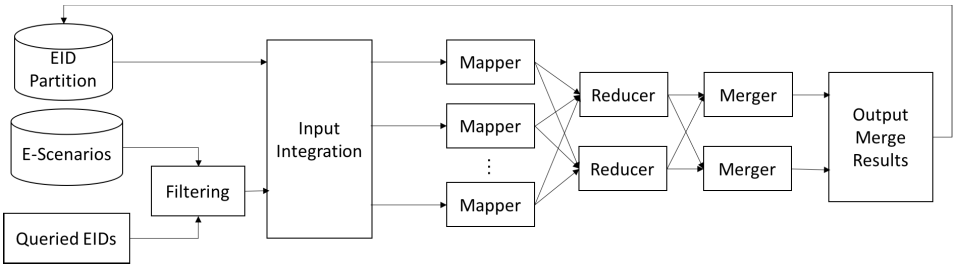


Fig. 4: EID set splitting workflow

A. MapReduce Basics

MapReduce is a programming model for processing large-scale datasets in a distributed way among multiple machines. Its execution process has four stages: split, map, shuffle and reduce. The input data is firstly split into smaller chunks and sent to different machines (mappers). During the map stage, each chunk is transformed into a $(key, value)$ pair based on the logic defined in the map function. Then all the $(key, value)$ pairs from all mappers are shuffled, sorted to put in order and grouped. In the reduce stage, a few machines (reducers) aggregate the shuffled pairs with the same key and output the final results. During the entire process, all data are stored in an underlying distributed file system. Data assignment, Map/Reduce task scheduling, and task failure recovery are managed by a master machine.

B. Parallelization of EID Set Splitting

The set splitting described in Algorithm 1 is a single-thread procedure and inefficient, because in each iteration only one scenario is selected and set splitting affects merely the *EID*s contained in that scenario. We consider to fetch a number of scenarios in each iteration and split the set more efficiently in a parallel way.

To parallelize the set splitting, we exploit the shuffle of $(key, value)$ pairs in MapReduce framework to implement set intersection operation of *EID* partitions and *E-Scenario*s. We break the procedure into four steps, preprocess, map, reduce and merge. Algorithm 3 defines the functions for the four steps of one iteration. The worklow of the whole procedure is shown in Figure 4.

– *Preprocess:* The input for preprocess are a list of *EID*s which need to be matched with their *VID*s, an *EID* partition which could be the set U_{eid} or the result from previous iteration and a list of *E-Scenario*s which contains untouched *E-Scenario*s in the database. Note that each *E-Scenario* is actually an *EID* set and it has a unique set *ID* and a timestamp. We randomly choose a timestamp and select all the *E-Scenario*s with this timestamp. Then we further filter out the *E-Scenario*s which do not contain any of the *EID*s which need to be matched. The remained *E-Scenario*s are integrated with the latest *EID* partition into a complete input for next step.

– *Map:* The input of the map function is an *EID* set which is the element of the output from preprocess. We use the element of the *EID* set as the key and the set *ID* as the value. Such $(key, value)$ pairs are emitted as the output of mappers. After this stage, all the $(key, value)$ pairs with the same *EID* as the key will be shuffled to the same reducer.

– *Reduce:* The reducer receives multiple *EID* set *ID*s and a single *EID* which appears in the intersection of these sets. It directly emits a $(key, value)$ pair with the *EID* set *ID*s as the key and the received *EID* as the value. All these pairs from many reducers will be aggregated by the key and sent to merge step.

– *Merge:* The merge function is similar to the reduce function. It receives a list of *EID* set *ID*s and a list of *EID*s which is the intersection of those sets. In fact, this intersection is just one element of the new partition. The merger emits the set *ID*s and their intersection as the output. Then all the intersections from all mergers are collected to form a new *EID* partition. And set *ID*s of the scenarios which contribute to the new partition will be recorded.

So far, one iteration completes and a finer partition of *EID* universal set can be fed into the next iteration. After all

Algorithm 3 Algorithm for one iteration

Preprocess
  Input: $List < EID > eidlist$, $EIDPartition\ eidpart$, $List < E-Scenario > escelist$
  $eidsetlist1 \leftarrow$ Filter $escelist$ by a random time stamp
  $eidsetlist2 \leftarrow$ Filter $eidsetlist1$ by $eidlist$
  $eidsetlist3 \leftarrow$ Integrate $eidsetlist2$ with $eidpart$
  return $eidsetlist3$

Map
  Input: $EIDSet\ eidset$
  for each eid in $eidset$ **do**
    emit(eid, $EIDSetID$)
  end for

Reduce
  Input: $EID\ eid$, $List < EIDSetID > eidsetidlist$
  emit($eidsetidlist$, eid)

Merge
  Input: $List < EIDSetID > eidsetislist$, $List < EID > eidlist$
  emit($eidsetidlist$, $eidlist$)

the *EID*s are distinguished, which means each of these *EID*s makes up one set by itself in the partition, the *EID* splitting stage completes and *VID* filtering is triggered.

C. Parallelization of VID Filtering

After *EID* set splitting, a list of *E-Scenario*s is selected for each *EID*. The corresponding *V-Scenario*s are then selected for each *EID*. For *VID* filtering, we need detect human objects in the visual data, extract object features and compare the features to figure out the same *VID* appearing in the *V-Scenario*s. Such processing can be time consuming. To speedup, we use MapReduce to parallelize human detection and feature extraction by processing different *V-Scenario*s on different mappers. Because these visual operations require no data dependancy. *VID* features are computed and stored in distributed storage system. Then we reorganize the processed *V-Scenario*s as the input for another MapReduce procedure. The *V-Scenario*s in the selected list of one *EID* will be conveyed to the same mapper to do feature comparison. In this way, *VID* feature comparisons for multiple *EID*s are performed in parallel.

VI. Evaluation

In this section, we evaluate our proposed algorithm on a synthetic dataset in real-world clusters. We will present the results below.

A. Experiment Setup

We set up a cluster with 14 machines. Each machine has a four-core (2.4 GHz) processor, 8 GB RAM and 2 TB storage. All machines are connected via gigabit Ethernet over a local switch. Apache Spark 1.3.1 and Hadoop 2.7.0 are deployed on each machine.

In our experiments, we have a database with 1000 human objects each associated with an *EID* and a *VID*. All the *VID*s are images of humans from the CUHK02 [17] dataset. The images are extracted snapshots of humans captured by cameras from different views. We assign WiFi MAC addresses to the human objects as their captured *EID*s. For *EV-Scenario* generation, we randomly choose a number of human objects from the database and distribute them across a 1000 m $\times$ 1000 m spatial region which consists of several cells. For mobility, we employ the random waypoint model [7] to control each human object's movement in terms of location, velocity and acceleration change.

B. Evaluation Results

We compare our algorithm with *EDP*, a baseline method proposed in [24]. However, *EDP* can only handle one *EID* at one time. For fair comparison with our parallelized method, we adapt *EDP* to MapReduce framework by assigning each mapper one *EID* matching task. We use two metrics to evaluate the performance of our algorithm, time efficiency and accuracy, both of which are key concerns when dealing with large-scale surveillance data. We have two experiment settings. In one setting, we change the number of *EID*s which need to be matched. In the other one, we vary the *EID* density in the region, i.e., the average number of human objects in each cell. In the evaluation results, we use *SS* to denote our algorithm since it is based on set splitting.

Both of our algorithm and *EDP* have two stages, *E* stage and *V* stage. *E* stage selects a bunch of scenarios for each *EID* and these scenarios are the input for the *V* stage. Thus the number of selected scenarios directly affects the computation time of the *V* stage. More scenarios mean more computation. Figure 5 and Figure 6 show the number of selected scenarios under different number of matched *EID*s and different densities. Note that reused scenario is only counted once. Generally, our algorithm selects much fewer scenarios than *EDP*. This is because our algorithm aims to reuse each selected scenarios for multiple *EID*s while it is highly random for a scenario selected for one *EID* to be reused for other *EID*s in *EDP*. In Figure 6, when the density increases, the number of selected scenarios by our algorithm decreases and converges around 40, which is the opposite to *EDP*. This is because each selected scenario is more likely to be reused when each cell has more *EID*s. Figure 7 shows the average number of scenarios needed for each matched *EID*. We can see that our algorithm needs about one more scenario for each *EID* than *EDP*. This results in more comparisons of *VID* features in the *V* stage of our algorithm.

Figure 8 and Figure 9 show the results on processing times including both the processing time for *E* stage and *V* stage. We can see that *E* stage costs negligible time while the time spent in *V* stage dominates the total time because feature extraction and comparison is more computation intensive. Although more feature comparisons are needed for each *EID*, our algorithm is still faster than *EDP* because *EDP* needs to process much more scenarios in its *V* stage.

Matching accuracy is defined as the percentage of the correctly matched *EID*s. An *EID* is correctly matched only when the majority of the *VID*s chosen from the scenarios for this *EID* is the right *VID*. We compare the accuracy of our algorithm with *EDP* under different number of matched *EID*s and different densities. The results are given in Table I and Table II. Over multiple runs for each parameter setting, the average accuracy rates of our algorithm are over 85% and comparable with those of *EDP*.

Matched *EID*s	200	400	600	800
SS	92.42%	90.60%	91.50%	89.12%
EDP	93%	92%	88.21%	87.70%

TABLE I: Accuracy with respect to the number of matched *EID*s

Density	30	60	100	160
SS	92.04%	90.22%	88%	87.13%
EDP	91%	87%	89%	88.20%

TABLE II: Accuracy with respect to the density

We also evaluate our matching algorithm and compare it with *EDP* under practical settings. We consider two practical settings, *EID* missing for example some people do not carry

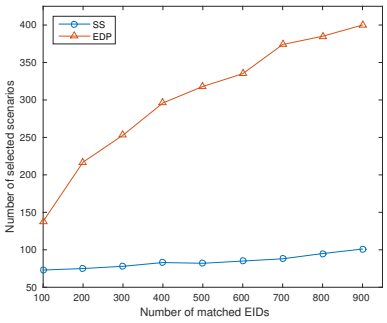


Fig. 5: Number of selected scenarios vs Number of matched *EID*s

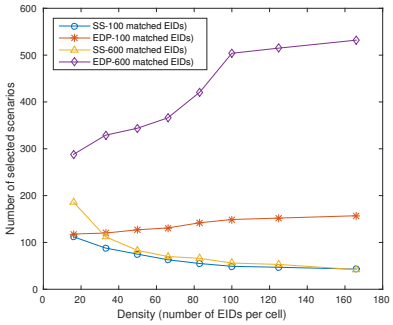


Fig. 6: Number of selected scenarios vs Density

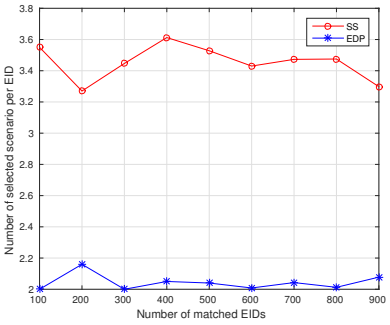


Fig. 7: Average number of selected scenarios per matched *EID*

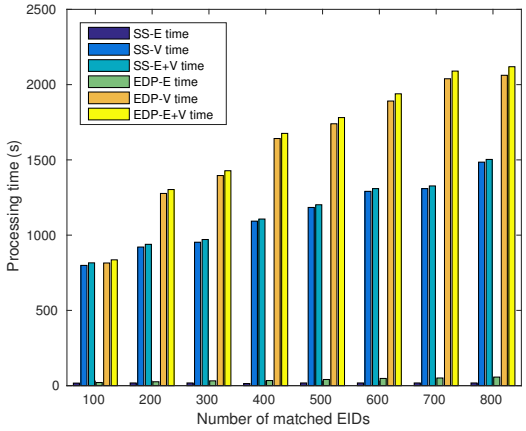


Fig. 8: Processing time vs Number of matched *EID*s

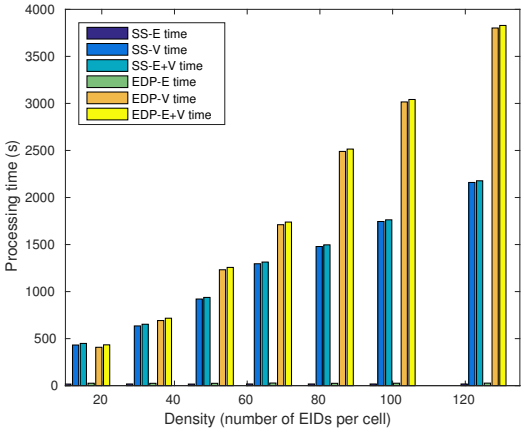


Fig. 9: Processing time vs Density

their phones, and *VID* missing for example human objects are miss detected. In Figure 10, we measure the matching accuracy under different *EID* missing rates. Generally the accuracy drops when the *EID* missing rate rises. However, even when the missing rate goes up to 50%, the matching accuracy is still good at around 85%. Figure 11 shows the matching accuracy under different *VID* missing rates. We can see that *VID* missing has more negative impacts on the matching accuracy. However, by matching refining, our accuracy is still above 80% even when the missing rate is as high as 10% which is far below the state-of-art human detection performance. Also our algorithm yields better accuracy than *EDP*.

VII. Conclusions

In this paper, we propose to match heterogeneous *EV* data according to spatiotemporal information to achieve efficient large-scale surveillance. In the matching procedure, *EID* set splitting algorithm is designed to reduce the amount of visual data to be processed. To cope with the big data problem, MapReduce is utilized to fuse *EV* data in parallel. Elastic matching size is supported by this algorithm and multiple *EID-VID* matching can be performed simultaneously. We implement the distributed algorithms on Spark and test it on a large synthetic *EV* dataset. The evaluation results demonstrate the feasibility and efficiency of our algorithms.

References

[1] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang, and J. Saltz. Hadoop gis: a high performance spatial data warehousing system over mapreduce. *Proceedings of the VLDB Endowment*, 6(11):1009–1020, 2013.

[2] Apache Spark. http://spark.apache.org.

[3] A. Bedagkar-Gala and S. K. Shah. A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270–286, 2014.

[4] W. Cai, C. Li, and S. Luan. Soi rf switch for wireless sensor network. *arXiv preprint arXiv:1701.01763*, 2017.

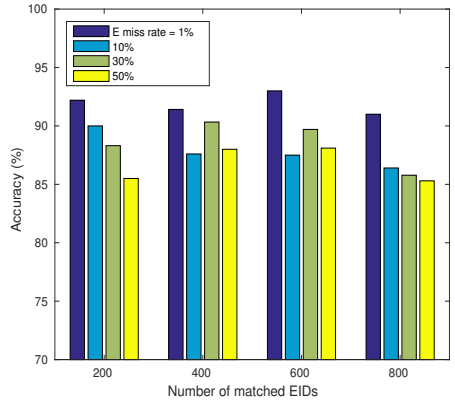
[5] W. Cai, J. Xu, and S. Wang. Low power si class e power amplifier for healthcare application. *International Journal of Electronics Communication and Computer Engineering*, 7(6):290, 2016.

[6] W. Cai, X. Zhou, and X. Cui. Optimization of a gpu implementation of multi-dimensional rf pulse design algorithm. In *Bioinformatics and Biomedical Engineering,(iCBBE) 2011 5th International Conference on*, pages 1–4. IEEE, 2011.

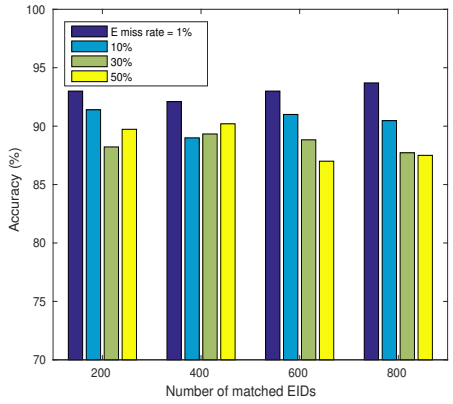
[7] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502, 2002.

[8] S. Ding, G. Li, Y. Li, X. Li, Q. Zhai, A. C. Champion, J. Zhu, D. Xuan, and Y. F. Zheng. Survsurf: human retrieval on large surveillance video data. *Multimedia Tools and Applications*, 76(5):6521–6549.

[9] G. Doretto, T. Sebastian, P. Tu, and J. Rittscher. Appearance-based person reidentification in camera networks: problem overview and current approaches. *Journal of Ambient Intelligence and Humanized Computing*, 2(2):127–151, 2011.

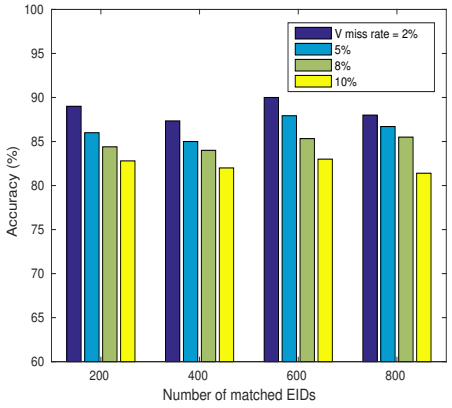


(a) *SS*

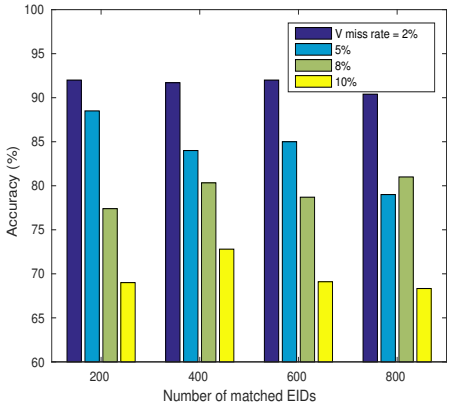


(b) *EDP*

Fig. 10: Accuracy vs *EID* missing



(a) *SS*



(b) *EDP*

Fig. 11: Accuracy vs *VID* missing

[10] S. Du, J. Hua, Y. Gao, and S. Zhong. Ev-linker: Mapping eavesdropped wi-fi packets to individuals via electronic and visual signal matching. *Journal of Computer and System Sciences*, 2015.

[11] foxnews. Security camera surge in Chicago sparks concerns of 'massive surveillance system', 5 Nov. 2014. http://www.foxnews.com/.

[12] J. Han and B. Bhanu. Statistical feature fusion for gait-based human recognition. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–842. IEEE, 2004.

[13] J. Jiang, H. Bao, E. Y. Chang, and Y. Li. Moist: a scalable and parallel moving object indexer with school tracking. *Proceedings of the VLDB Endowment*, 5(12):1838–1849, 2012.

[14] D. Li, Z. Lu, T. Bansal, E. Schilling, and P. Sinha. Foresight: Mapping vehicles in visual domain and electronic domain. pages 1995–2003, 2014.

[15] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 421–430. ACM, 2012.

[16] G. Li, X. Li, F. Yang, J. Teng, S. Ding, Y. F. Zheng, D. Xuan, B. Chen, and W. Zhao. Traffic at-a-glance: Time-bounded analytics on large visual traffic data. *Transactions on Parallel and Distributed Systems*, 2017. doi:10.1109/TPDS.2017.2684158.

[17] W. Li and X. Wang. Locally aligned feature transforms across views. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3594–3601. IEEE, 2013.

[18] L. T. Nguyen, Y. S. Kim, P. Tague, and J. Zhang. Identitylink: user-device linking through visual and rf-signal cues. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 529–539. ACM, 2014.

[19] S. Papaioannou, H. Wen, A. Markham, and N. Trigoni. Fusion of radio and camera sensor data for accurate indoor positioning. In *Mobile Ad Hoc and Sensor Systems (MASS), 2014 IEEE 11th International Conference on*, pages 109–117. IEEE, 2014.

[20] J. Pramis. Number of mobile phones to exceed world population by 2014, 2Nov. 2013. http://www.digitaltrends.com/mobile/mobile-phone-world-population-2014/.

[21] D. Roetenberg, P. J. Slycke, and P. H. Veltink. Ambulatory position and orientation tracking fusing magnetic and inertial sensing. *Biomedical Engineering, IEEE Transactions on*, 54(5):883–890, 2007.

[22] T. Teixeira, D. Jung, G. Dublon, and A. Savvides. Pem-id: Identifying people by gait-matching using cameras and wearable accelerometers. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pages 1–8. IEEE, 2009.

[23] T. Teixeira, D. Jung, and A. Savvides. Tasking networked cctv cameras and mobile phones to identify and localize multiple people. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 213–222. ACM, 2010.

[24] J. Teng, J. Zhu, B. Zhang, D. Xuan, and Y. F. Zheng. Ev: efficient visual surveillance with electronic footprints. In *INFOCOM, 2012 Proceedings IEEE*, pages 109–117. IEEE, 2012.

[25] G. Trajcevski, H. Ding, P. Scheuermann, R. Tamassia, and D. Vaccaro. Dynamics-aware similarity of moving objects trajectories. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, page 11. ACM, 2007.

[26] Y. Yang, J. Yang, J. Yan, S. Liao, D. Yi, and S. Z. Li. Salient color names for person re-identification. In *Computer Vision–ECCV 2014*, pages 536–551. Springer, 2014.

[27] R. Zhao, W. Ouyang, and X. Wang. Learning mid-level filters for person re-identification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 144–151. IEEE, 2014.