sensors

Article

The Device–Object Pairing Problem: Matching IoT Devices with Video Objects in a Multi-Camera Environment

Kit-Lun Tong [1], Kun-Ru Wu [2,*] and Yu-Chee Tseng [3,4,5]

[1] School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, UK; k.tong@uea.ac.uk
[2] Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan
[3] College of AI, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan; yctseng@cs.nctu.edu.tw
[4] Academia Sinica, Taipei 11529, Taiwan
[5] College of Health Sciences, Kaohsiung Medical University, Kaohsiung 80708, Taiwan
* Correspondence: wufish@nycu.edu.tw

Abstract: IoT technologies enable millions of devices to transmit their sensor data to the external world. The *device–object pairing problem* arises when a group of Internet of Things is concurrently tracked by cameras and sensors. While cameras view these things as visual "objects", these things which are equipped with "sensing devices" also continuously report their status. The challenge is that when visualizing these things on videos, their status needs to be placed properly on the screen. This requires correctly pairing visual objects with their sensing devices. There are many real-life examples. Recognizing a vehicle in videos does not imply that we can read its pedometer and fuel meter inside. Recognizing a pet on screen does not mean that we can correctly read its necklace data. In more critical ICU environments, visualizing all patients and showing their physiological signals on screen would greatly relieve nurses' burdens. The barrier behind this is that the camera may see an object but not be able to see its carried device, not to mention its sensor readings. This paper addresses the device–object pairing problem and presents a multi-camera, multi-IoT device system that enables visualizing a group of people together with their wearable devices' data and demonstrating the ability to recover the missing bounding box.

Keywords: device–object pairing; IoT; computer vision; data fusion; surveillance

Citation: Tong, K.-L.; Wu, K.-R.; Tseng, Y.-C. The Device–Object Pairing Problem: Matching IoT Devices with Video Objects in a Multi-Camera Environment. *Sensors* **2021**, *21*, 5518. https://doi.org/10.3390/s21165518

Academic Editors: Kun-chan Lan, Teen-Hang Meen, Chi-Yuan Chen and Shih-Lin Wu

Received: 19 July 2021
Accepted: 13 August 2021
Published: 17 August 2021

1. Introduction

The advance of IoT technologies enables millions of devices to transmit their sensor data to the external world. On the other hand, empowered by deep learning, today's computer vision has significantly improved its object recognition capability. When an IoT device is placed on an object, we intend to not only recognize the object in videos but also recognize the IoT device bundled with the object.

Surveillance systems are widely used in homes, buildings, and factories. However, when abnormal events occur, it usually takes a lot of human effort to check the surveillance videos. With the advance of AI technologies, automatically analyzing video content becomes feasible. RetinaNet [1] and You Only Look Once (YOLO) [2] can identify a variety of objects with high accuracy and efficiency. OpenPose [3] and Regional Multi-person Pose Estimation (RMPE) [4] can perform human pose recognition without using depth cameras or ToF (time-of-flight) sensors.

In most surveillance and security applications, the central issue is to capture abnormal people, objects, and events in the environment. This work studies the *device–object pairing problem* in surveillance videos. Previous and common practices include, but are not limited to, barcode, Radio Frequency ID (RFID), and biometric sensing (e.g., fingerprint and iris recognition). However, these methods require keeping at a very short distance to devices. On the other hand, facial recognition relies on obtaining a face database and does not work under larger shooting angles and occlusions. Furthermore, there are privacy concerns in public domains. Another way to identify persons is to exploit IoT devices, such as smartphones and smartwatches, which have become virtually the users' IDs. Personal devices can also store owners' profiles and be used in sensitive domains such as factories, hospitals, and restricted areas [5,6]. Further, with the global deployment of 5G, IoT devices nowadays can communicate almost anywhere. When tracking a group of people by camera, it would be nice to also know their purchase histories and preferable social network tools on their smartphone. This motivates us to study combining computer vision and IoT devices under multi-camera environments.

The device–object pairing system considers an environment consisting of multiple cameras, with users wearing their IoT devices walking around. Figure 1 shows our system scenario. There is an IoT network for collecting data from cameras and wearable IoT devices in the environment. We use a YOLO module to obtain and track the human objects appearing in the cameras. The homography matrices of these cameras are estimated to transform each camera view to common surface space. Because there are multiple cameras that concurrently capture the same set of people, the possibilities of occlusions and tracking failures are reduced. Our system then tries to extract objects' motion features from both visual and wearable sensors and generates (devices, object) pairs that can be displayed on video objects. Therefore, with IoT devices serving as the personal identification of users, our system can visualize user identities and sensor data for smart surveillance.

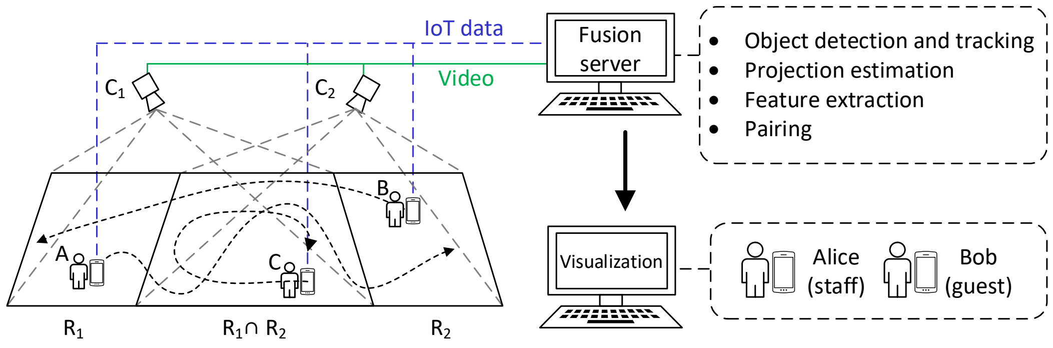


Figure 1. The device–object pairing system architecture.

We have built a prototype system that runs at 12 Frames Per Second (FPS). It contains a server and two cameras and wearable devices connected by Wi-Fi networks. We designed three mobility patterns (namely random, following, and mimicking) to evaluate its performance.

This paper is organized as follows. Related works are reviewed in Section 2. The proposed system architecture is presented in Section 3. Prototyping and evaluation results are in Section 4. Conclusions are drawn in Section 5.

2. Related Work

Surveillance has been a critical issue in smart city for decades. The study in [7] is an early work that shows how a camera with an embedded system and network connection works for smart analysis. Reference [8] proposes sensor-based surveillance with interactive multiple models to track targets. To solve the person re-identification problem, Reference [9] designs a temporal residual learning module to learn the feature of pedestrian sequences. To monitor illegal or suspicious activities, extracting skeletons by deep learning from video frames is studied in [10]. Our research work is more extensive because we consider both video and sensor data.

Human detection technologies have been rapidly developed recently. Reference [11] proposes detecting objects by a Haar Feature-based Cascade Classifier. With the advance of artificial intelligence, References [1,2] achieve real-time detection by deep learning. Skeletons can be extracted from a mono camera frame in [3,4]. For Multiple Object Tracking (MOT), Reference [12] proposes a *k*-shortest paths algorithm, Reference [13] adopts a

Kalman filter, and [14] uses a deep learning approach. Multi-camera, multi-person tracking by homography matrices is studied in [15]. In the city-scale environment, reference [16] addresses vehicles re-identification and tracking under a multi-camera scenario. Another multi-camera pedestrian tracking is proposed in [17], in which the IoT devices are edge nodes to analyze video by a deep learning module. A sensor-based IoT architecture for smart surveillance is proposed in [18]. A wireless video surveillance system is presented in [19]. On the contrary, our work explores both wearable devices (with sensors) and cameras.

Sensor fusion-based tracking has been studied in a simpler environment. Reference [20] uses a depth camera to extract skeletons and pair them with the Inertial Measurement Unit (IMU) devices carried by users. Fusion-based human and object tracking is shown in [21]. In [22], they use a camera to capture human motion by OpenPose [3] and match the motion with an IMU device to address ID association. These works all exploit user movements to identify persons. Nevertheless, their detection ranges are limited since skeletons can only be detected within a short distance. Reference [23] addresses the fusion of hand gestures and wearable sensors to identify people in crowded mingle scenarios, but it still requires a short distance to track the movements. Reference [24] is an example using a Received Signal Strength Indicator (RSSI) of Wi-Fi as signal trajectories to pair a person on camera. However, the system is unstable since the RSSI signal may suffer more interference and the variance is large. Reference [25] demonstrates how to tag personal ID on people from a drone camera through trajectory pairing. However, it is limited to one drone. Fusing computer vision and the 5G network for driving safety is proposed in [26]. A reconfigurable platform for data fusion is presented in [27].

In this work, we develop machine learning-based feature extraction and a more efficient fusion model with time synchronization under multi-camera environments.

3. Fusion-Based Device–Object Pairing

Figure 2 shows the proposed data fusion process. The hardware components include multiple surveillance cameras, some smart wearable devices with built-in IMU (accelerometer and magnetometer), a fusion server, and an IoT network. To pair devices with video objects, the data collected from cameras and wearable devices will be fused based on features extracted. The main software components include: (A) IoT network, (B) projection estimation, (C) local object detection and tracking, (D) global object tracking, (E) feature extraction, and (F) device and global object pairing. Section 3.1 introduces our IoT network. How to map camera views to a common ground space is addressed in Section 3.2. The object tracking task for each camera is discussed in Section 3.3. How to merge all cameras' views to a global view is addressed in Section 3.4. Then, feature extraction and device–object pairing are covered in Sections 3.5 and 3.6, respectively.

3.1. IoT Network

In our system, multiple cameras are deployed to fully cover the surveillance site. Users appearing in front of cameras are expected to put on their wearable devices (otherwise, such users will be marked as "unknown" by the pairing module). An IoT network is designed to manage these devices. Data are exchanged by MQTT (Message Queuing Telemetry Transport [28]), which is a lightweight, publish–subscribe protocol allowing message transportation among massive devices without obvious latency. Furthermore, NTP (Network Time Protocol) is adopted to synchronize time among all components.

For convenience, we use smartphones to simulate wearable devices. Each smartphone has IMUs, including a three-axis accelerometer and magnetometer. These sensors periodically report their data to the fusion server at a rate of 50 Hz in JSON format [29]. Readings are marked by timestamps and pre-processed by a low pass filter. Furthermore, each device is uniquely distinguishable by its ID.

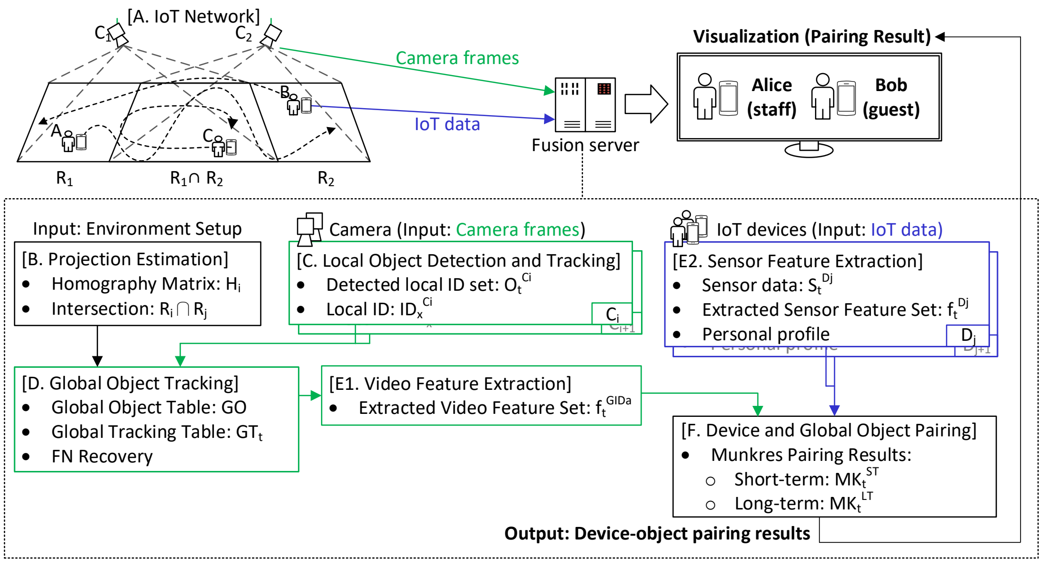

[A. IoT Network]
C1
C2
A
B
C
R1
R1∩ R2
R2
Camera frames
IoT data
Fusion server
Visualization (Pairing Result)
Alice (staff)
Bob (guest)
Input: Environment Setup
[B. Projection Estimation]
Homography Matrix: Hi
Intersection: Ri∩Rj
Camera (Input: Camera frames)
[C. Local Object Detection and Tracking]
Detected local ID set: Ot^Ci
Local ID: IDx^Ci
Ci
Ci+1
IoT devices (Input: IoT data)
[E2. Sensor Feature Extraction]
Sensor data: St^Dj
Extracted Sensor Feature Set: ft^Dj
Personal profile
Dj
Dj+1
[D. Global Object Tracking]
Global Object Table: GO
Global Tracking Table: GTt
FN Recovery
[E1. Video Feature Extraction]
Extracted Video Feature Set: ft^GIDa
[F. Device and Global Object Pairing]
Munkres Pairing Results:
Short-term: MKt^ST
Long-term: MKt^LT
Output: Device-object pairing results

Figure 2. Data fusion procedure.

There are multiple cameras. For each camera, an M-JPEG (Motion Joint Photographic Experts Group) server is set up for streaming frames by HTTP (Hypertext Transfer Protocol) in the JPEG format. To validate our framework, we do not use the keyframe method [30], where complete frames are interleaved by compressed frames, causing broken textures when a frame is lost. Therefore, all frames are complete frames, and no retransmission is performed. Every frame from camera C_i is assigned a timestamp t and is denoted as $F_t^{C_i}$. Frames are remapped to solve the radial distortion problem caused by lens, which is achieved by using the chessboard photos test [31] to calculate the internal and external parameters and the lens distortion coefficients.

3.2. Projection Estimation

To relate the views among multiple cameras, we shall map the pixel space of each camera to a common ground space. Our approach is to estimate a homography matrix H_i for each camera C_i that transforms each camera pixel to a ground coordination. Let the errors E_i caused by the transformation follow a normal distribution model and the coverage region of C_i be R_i. We show how to determine (i) H_i, (ii) E_i and (iii) $R_i \cap R_j$ for each pair of C_i and C_j.

To find H_i, we design a lightweight human-assisted process. We place only a few markers on the ground and send a designated person to walk in the field arbitrarily. The person passes these markers from time to time. Whenever passing a marker, they will stop for a while and record the measured value before moving forward. This stopping behavior serves as an indication to cameras that they are right on a marker. In addition, another wearable device is attached to trace them. In some sense, the cameras localize the person when they stop at a marker, and the wearable sensor tracks their trajectory between two markers while walking. By calibrating their location to a marker, the drifting problem of IMU tracing is significantly relieved. This procedure can be repeated arbitrarily for all camera coverage regions, and during the procedure, the person's ground trajectory is mapped with camera views to learn the mapping between pixels and ground trajectories. Lots of such mappings can be found only within a few minutes.

In fact, the above steps serve as a labeling process to map C_i's pixel space to the ground space. Through object detection, the center of a human object is regarded as their location in C_i's pixel space. Their trajectory in C_i is then partitioned into segments (called *v_segments*) according to those stopping points (markers). On the IMU side, their trajectory is also partitioned into segments (called *i_segments*) at places where C_i reports stopping events. An *i_segment* is modeled by the recursion: $L_k = L_{k-1} + \vec{d_t} * stride\_len$, where L_0 is the location of the starting marker, L_k is their location at the k-th stride, $\vec{d_k}$ is the normalized vector of the k-th stride (obtained from magnetometer), and $stride\_len$ is a predefined value. The recursion stops when encountering the next marker. We then match the endpoints of all $i_segment$s to ground markers by the maximal likelihood of inter-marker distances and absolute $i\_segment$ distances. This derives the ground markers of all $i_segment$s' endpoints. With known endpoints of each $i\_segment$, we calibrate it by rescaling. Therefore, we obtain many $(v\_segment, i\_segment)$ pairs for each camera C_i with their pixel locations labeled by the $i\_segment$.

Given a large number of $(v\_segment, i\_segment)$ pairs for C_i, we can derive its homography matrix H_i for the projection. By [32], the objective is to meet

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & h_2 \\ h_3 & h_4 & h_5 \\ h_6 & h_7 & h_8 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H_i \begin{bmatrix} u \\ v \\ 1 \end{bmatrix},$$

where (u, v) is a pixel and (u', v') a ground point. It requires at least four pairs to solve H_i. As an $i\_segment$ is more likely to contain errors due to sensor data drifting, we suggest retrieving more knowns to minimize the drifting problem. We then apply the least-square method to find H_i.

Assuming the projection error E_i of H_i following a normal distribution, $E_i \backsim N(\mu_i, \sigma_i^2)$, we collect all (u', v') calculated by H_i and its corresponding label (u'', v'') recorded by IMU. The distances between all (u', v') and (u'', v'') are mapped to a normal distribution to find μ_i and σ_i of E_i.

By mapping all pixels of C_i to the ground, we can obtain the coverage R_i of C_i. The next task is to find the intersection $R_i \cap R_j$ for each pair of C_i and C_j. The shape of R_i is close to a polygon. Therefore, the overlapping area $R_i \cap R_j$ is also close to a polygon. The related backgrounds can be found in [33,34].

Figure 3 shows an example. Figure 3a,b are views taken by C_1 and C_2 at the same time. There are six tiny markers on the ground (we do not require cameras to recognize these markers). Through human detection, trajectories of people are obtained. Meanwhile, wearable sensors also derive their trajectories. Figure 3c shows the rescaled IMU trajectories. Finally, by H_1 and H_2, the overlapping region $R_1 \cap R_2$ is obtained in Figure 3d.

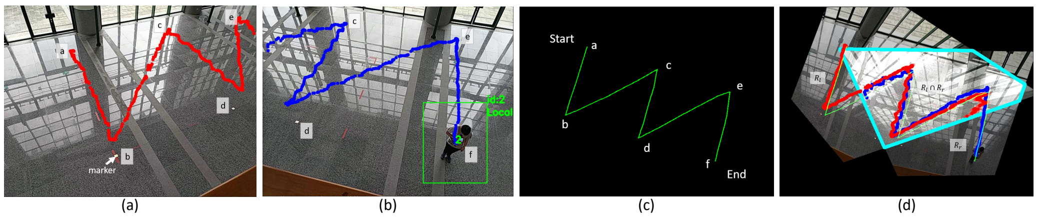


Figure 3. (**a**) Trace found by C_1. (**b**) Trace found by C_2. (**c**) IMU trajectory partitioned by markers. (**d**) Overlapping region $R_1 \cap R_2$.

3.3. Local Object Detection and Tracking

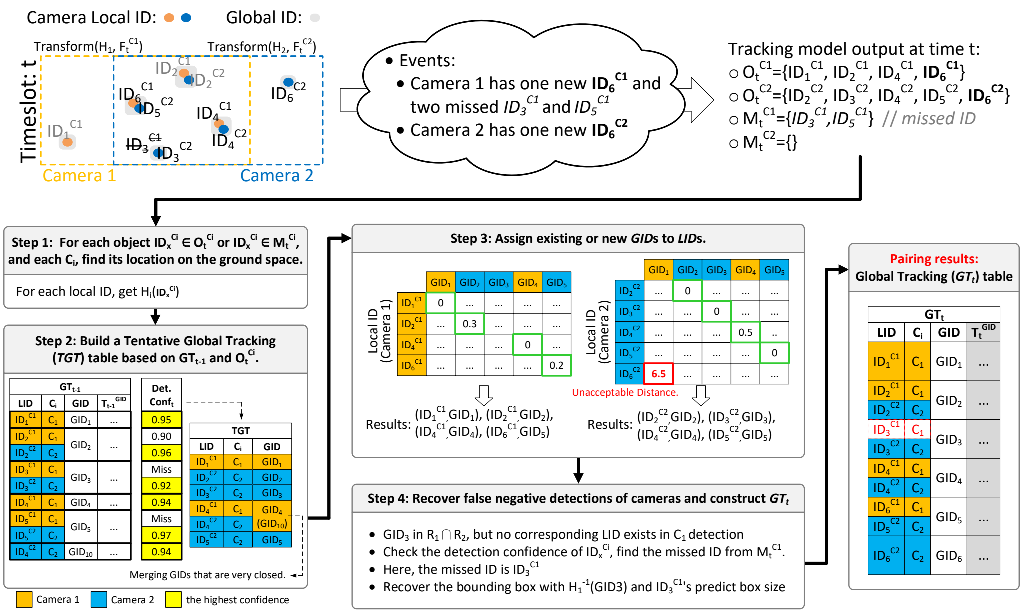
Each camera C_i needs to detect and track human objects locally. Since this task has been extensively researched, we will only discuss how we use existing tools to solve this problem. First, for each $F_t^{C_i}$, it is sent to YOLOv3 [2], a real-time deep learning model for object detection, to retrieve a set of bounding boxes representing detected human objects. We have also tried skeleton models by OpenPose [3], but since the detection time is longer, we adopt YOLO in the rest of the discussion.

Second, we need to determine if a human object detected in $F_t^{C_i}$ has also appeared in $F_{t-1}^{C_i}$. This is achieved by Simple Online and Realtime Tracking (SORT) [13], a tracking algorithm based on the Kalman filter. The outputs are a detected ID set $O_t^{C_i} = \{ID_x^{C_i}, ID_y^{C_i}, \ldots\}$ corresponding to all bounding boxes and a miss ID set $M_t^{C_i} = \{ID_a^{C_i}, ID_b^{C_i}, \ldots\}$. We call

$ID_x^{C_i} \in O_t^{C_i}$ the *local* ID assigned to object x at time t by camera C_i. If $ID_x^{C_i}$ appears in both $O_t^{C_i}$ and $O_{t-1}^{C_i}$, object x is regarded as the same person. On the other hand, false negative ($x \in O_{t-i}^{C_i}$, but x is not detected as an object at t) and ID switching (x receives different IDs at $t-1$ and t) may happen. If an object in $O_{t-i}^{C_i}$ disappears in $O_t^{C_i}$, the corresponding predicted bounding boxes from Kalman filter are included in $M_t^{C_i}$. In fact, ID switching is not uncommon when YOLO continuously fails to detect an object or network packets are lost continuously, making SORT regard it as a new person after it reappears. It also happens when a detected bounding box drifts far away from its previous location in a new frame. We will discuss how to reduce such confusion later.

3.4. Global Object Tracking

The next objective is to merge all $O_t^{C_i}$s detected by all cameras under a global domain. Assume that at time $t-1$ a Global Tracking table GT_{t-1} as shown in Figure 4 is obtained. The contents of GT_{t-1} include: (i) the local ID (LID) assigned to object x, (ii) the source camera C_i that captures x, (iii) the global ID (GID) assigned to x, and (iv) the trajectory T_{t-1}^{GID} of x. Note that a GID may be associated to multiple LIDs if the object is captured by more than one camera.



Camera Local ID: ● ●
Global ID: ●
Transform(H₁, F_t^C1)
Transform(H₂, F_t^C2)
Timeslot: t
ID₁^C1
ID₂^C1
ID₂^C2
ID₆^C1
ID₅^C2
ID₄^C1
ID₄^C2
ID₃^C1
ID₃^C2
ID₆^C2
Camera 1
Camera 2
• Events:
• Camera 1 has one new ID₆^C1 and two missed ID₃^C1 and ID₅^C1
• Camera 2 has one new ID₆^C2
Tracking model output at time t:
○ O_t^C1={ID₁^C1, ID₂^C1, ID₄^C1, ID₆^C1}
○ O_t^C2={ID₂^C2, ID₃^C2, ID₄^C2, ID₅^C2, ID₆^C2}
○ M_t^C1={ID₃^C1,ID₅^C1} // missed ID
○ M_t^C2={}
Step 1: For each object ID_x^Ci ∈ O_t^Ci or ID_x^Ci ∈ M_t^Ci, and each C_i, find its location on the ground space.
For each local ID, get H_i(ID_x^Ci)
Step 2: Build a Tentative Global Tracking (TGT) table based on GT_t-1 and O_t^Ci.
GT_t-1
LID
C_i
GID
T_t-1^GID
Det. Conf_t
0.95
0.90
0.96
Miss
0.92
0.94
Miss
0.97
0.94
TGT
GID₁₀
(GID₁₀)
Merging GIDs that are very closed.
Camera 1
Camera 2
the highest confidence
Step 3: Assign existing or new GIDs to LIDs.
Local ID (Camera 1)
Local ID (Camera 2)
0.3
0.2
0.5
6.5
Unacceptable Distance.
Results: (ID₁^C1,GID₁), (ID₂^C1,GID₂), (ID₄^C1,GID₄), (ID₆^C1,GID₅)
Results: (ID₂^C2,GID₂), (ID₃^C2,GID₃), (ID₄^C2,GID₄), (ID₅^C2,GID₅)
Step 4: Recover false negative detections of cameras and construct GT_t
• GID₃ in R₁∩R₂, but no corresponding LID exists in C₁ detection
• Check the detection confidence of ID_x^Ci, find the missed ID from M_t^C1.
• Here, the missed ID is ID₃^C1
• Recover the bounding box with H₁⁻¹(GID3) and ID₃^C1's predict box size
Pairing results:
Global Tracking (GT_t) table
GT_t
GID₁
GID₂
GID₃
GID₄
GID₅
GID₆

Figure 4. An example of creating GT_t from GT_{t-1}.

We need to construct GT_t at t from GT_{t-1}. There are four tasks:

1. For each object $ID_x^{C_i} \in O_t^{C_i}$ or $ID_x^{C_i} \in M_t^{C_i}$, and each C_i, find x's location on the ground space.
2. Build a Tentative Global Tracking table (TGT).
3. Assign existing or new GIDs to LIDs.
4. Recover false negative detections of cameras and construct GT_t.

The first task can be achieved by H_i of C_i. Here, we regard the central pixel of an object's bounding box as its location. The corresponding result is denoted as $H_i(ID_x^{C_i})$.

The second task is to build TGT based on GT_{t-1} and the new tracking results $O_t^{C_i}$. TGT has the same three entries as GT_{t-1} (i.e., LID, C_i, and GID). We copy GT_{t-1} into TGT with three modifications: (i) For each LID in $M_t^{C_i}$, we exclude it in TGT since the object is not detected at t. (ii) Depending on the detection confidence at t, for each GID, only the entry with the highest confidence is kept. (iii) If two GIDs at $t-1$ become very close to each other at t, they will be merged into one GID (we keep the earlier comer here). For example, in Figure 4, $ID_3^{C_1}$ and $ID_5^{C_1}$ disappear at t, so they are removed. Furthermore, $GID_4(ID_4^{C_1})$ and $GID_{10}(ID_4^{C_2})$ become too close, the latecomer GID_{10} is merged with the earlier comer GID_4.

The third task is to assign GIDs to LIDs. First, distance matrices between all GIDs and all LIDs of all cameras are built. Given a GID_i and a local $ID_x^{C_j}$ detected by C_j, we define $d(GID_i, ID_x^{C_j})$ as the Euclidean distance between the location of GID_i and $H_j(ID_x^{C_j})$ (when GID_i and $ID_x^{C_j}$ fall in the same row in TGT, the distance is 0). Then we run the Hungarian Matching algorithm [35,36] to pair GIDs and $LIDs$ in the distance matrix. Since the matched pairs might encounter a false negative match, we execute a threshold test by checking each matched object $ID_x^{C_i}$ (with the highest detection confidence) in TGT and the corresponding paired object $ID_y^{C_j}$. Let $\tilde{d} = d(H_i(ID_x^{C_i}), H_j(ID_y^{C_j}))$. The confidence that $ID_x^{C_i}$ and $ID_y^{C_j}$ belong to the same global object is written as $E_i(\tilde{d}) \cdot E_j(\tilde{d})$. Intuitively, $E_i(resp., E_j)$ is the confidence that an object at an error distance of $\tilde{d}$ is acceptable. An example is shown in Figure 4. Note that a pair with distance 0 is always matched (so they can be optimized from the Hungarian algorithm). For a LID that is not matched with a GID in the above process, or the remainder, a new GID is assigned to it (GID_1 and $ID_6^{C_2}$ is such cases). The final results are a number of (LID, GID) pairs.

In the fourth task, we first recover false negative detections for each C_i. As we have the intersection region $R_i \cap R_j$ for each pair of C_i and C_j, if there is a discrepancy such that an $ID_x^{C_i}$ appears in $R_i \cap R_j$ but no corresponding $ID_x^{C_j}$ exists in C_j's detection, a miss detection may happen for C_j. We select the best matched missing LID in $M_t^{C_j}$, and test its predicted location from SORT against C_i's detection $H_i(ID_x^{C_i})$. If the test is passed, we can recover the missing bounding box by applying H_j^{-1} on $H_i(ID_x^{C_i})$ and the predicted box size from $M_t^{C_j}$, and add it to C_j's local detection list $O_t^{C_j}$. The new bounding box of C_j should belong to the same GID of C_i's. Finally, we can compile all (LID, GID) pairs to construct the GT_t at t.

3.5. Feature Extraction

In order to pair video objects with devices, we have to extract some common features to compare their similarity. Let $f_t^{GID_i}$ and $f_t^{D_j}$ be the features extracted from the video trajectory of GID_i and device D_j, respectively, within a sliding window $[t - \triangle t, t]$. We will derive two features, activity type and moving direction. For the activity type of GID_i, there are four types, 0 for standing, 1 for walking, 2 for turning-left, and 3 for turning-right. To train a Support Vector Machine (SVM), we feed mean, standard deviation, upper and lower quarter, and median absolute deviation of a number of key sequences as inputs, where a time slot is a basic unit of a sliding window:

1. $\triangle$ *distance*: distance between each sampling point in a time slot.
2. $\triangle$ *angle*: angle between each sampling point in a time slot.
3. $\triangle$ *axis*: axis between each sampling point in a time slot.

The classification result is denoted as $f_t^{GID_i}[act]$. For the activity type of D_j, it is derived by the following key sequences:

1. Three-axis accelerometer reading.
2. Three-axis magnetometer reading.

In addition, by SVM, the classification result is denoted by $f_t^{D_j}[act]$.

The moving direction is a 2D normalized vector in the world space. The moving direction of GID_i is denoted by $f_t^{GID_i}[dir]$ and is calculated by the starting and ending sampling points in each time slot. The moving direction of D_j is denoted by $f_t^{D_j}$, which is converted directly from the magnetometer. The value is stabilized by linear regression.

Figure 5 illustrates the above procedure for video data and sensor data. GID_i's trajectory is taken from $T_t^{GID_i}$ of GT_t, while the input from device D_j is directly taken from its sensors.

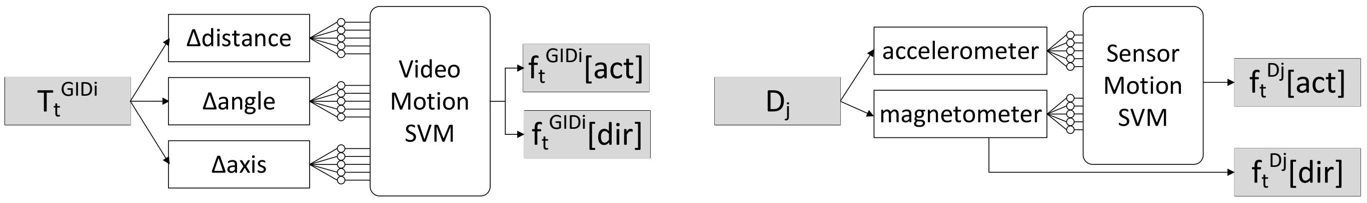


Figure 5. Feature extraction procedure for GID_i and D_j.

3.6. Device and Global Object Pairing

By comparing the similarity between $f_t^{D_j}$ and $f_t^{GID_i}$, we try to determine if GID_i and D_j are a pair. We will calculate two matrices: (i) short-term distance matrix δ_t and (ii) long-term weight matrix ω_t. These matrices include all GIDs and devices. By considering short- and long-term relations, we try to obtain more stable pairing results.

The matrix δ_t is formed by the distances between all $(f_t^{D_j}, f_t^{GID_i})$ pairs. Recall vectors $f_t^{GID_i}[act]$, $f_t^{GID_i}[dir]$, $f_t^{D_j}[act]$ and $f_t^{D_j}[dir]$. We define short-term distance as

$$\delta(GID_i, D_j) = \Big(d_{norm}(f_t^{GID_i}[act], f_t^{D_j}[act])^2 + d_{norm}(f_t^{GID_i}[dir], f_t^{D_j}[dir])^2 \Big)^{\frac{1}{2}}, \quad (1)$$

where d_{norm} is the distance between two vectors by normalizing to the range [0,1]. We then apply Hungarian matching on matrix δ_t to find the set of pairs denoted by $H(\delta_t)$. Note that this only reveals the result within the sliding window $[t- \triangle t, t]$. The long-term weight matrix ω_t is formed by considering a sequence of short-term results, namely $H(\delta_{t-\triangle}), H(\delta_{t-\triangle+1}), ..., H(\delta_t)$. For each (GID_i, D_j) pair, we define long-term distance as

$$\omega_t(GID_i, D_j) = 1 + (1-n) - \frac{2(1-n)}{e^{-5cnt(GID_i,D_j)/m}}, \quad (2)$$

where $cnt(GID_i, D_j)$ is the number of times that (GID_i, D_j) appears in $H(\delta_{t-\triangle}), H(\delta_{t-\triangle+1})$, ..., $H(\delta_t)$. Equation (2) is to simulate an inverse sigmoid function, where n is the convergence limit, m is the value of $cnt(GID_i, D_j)$ to meet n, and 5 is a fine-tuned value. Figure 6 shows $\omega_t(GID_i, D_j)$ under different n and m as opposed to a typical sigmoid function.

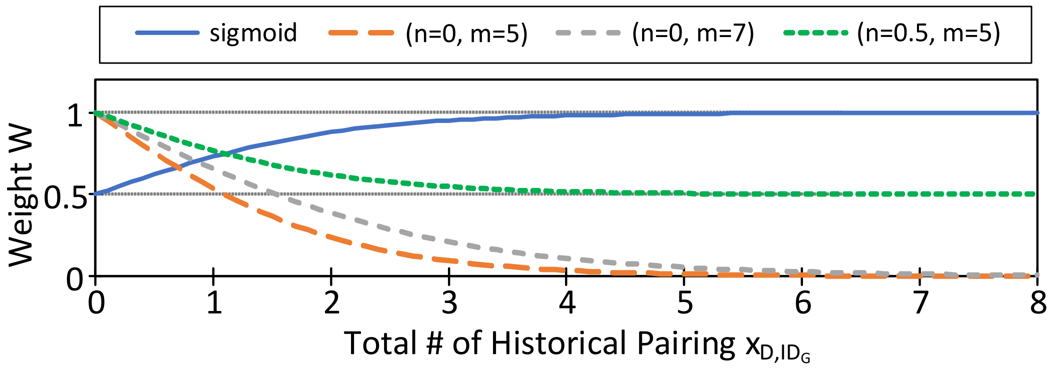


Figure 6. Equation (2) under different parameters.

Based on δ_t and ω_t, we propose two pairing models in Figure 7. The one-layer model uses only one Hungarian matching, which takes accumulated $H(\delta_t * \omega_t)$ as inputs, where $*$ means pairwise multiplication. However, we find it to be weak in handling the ID exchange problem during tracking. Therefore, we propose the two-layer model by applying Hungarian matching twice, the first time on δ_t and the second time on ω_t. We will validate this claim by experiments.

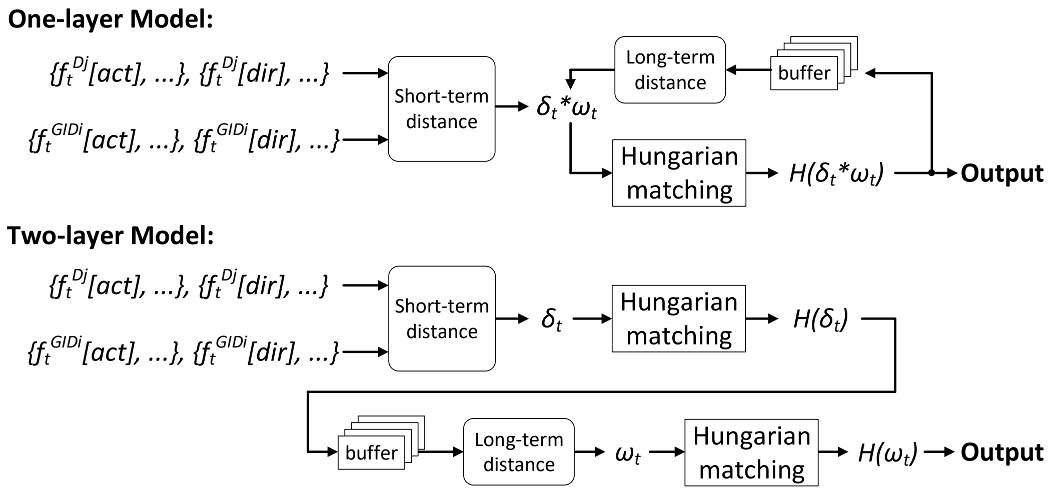


Figure 7. The proposed one-layer and two-layer pairing models.

4. Performance Evaluation and Discussion

4.1. Experimental Setting

We built a prototype, which consists of two Nokia 8.1 smartphones to simulate IP cameras. We set their height to six meters, view angle to 50 degrees down, and resolution to 600 $\times$ 800 pixels. These phones connect to an IEEE 802.11ac access point and stream videos at 20 FPS to our fusion server. Wearable devices are simulated by Android smartphones, which also connect to our fusion server and transmit sensor data at a rate of 50 Hz. The wireless router is ASUS RT-AC86U with a Cortex A53 1.8 GHz dual-core processor and 256 MB RAM. Its claimed rate is 1734 Mbps under 802.11ac and 450 Mbps under 802.11n. Our fusion server has an Intel i7-9750H CPU with six cores, 32 GB RAM, and a RTX 2080 MAX-Q GPU. To speed up the detection speed of YOLOv3-608 (which resizes images to 608 $\times$ 608), we added an external RTX 2080 eGPU. Table 1 shows the respective processing time. Processing one frame takes 0.081 s, which approximates to 12 FPS in real-time. This value is calculated from a 5-min clip with processing around 6000 frames. Table 2 shows the activity recognition results of our SVM model, where V-SVM is video motion SVM, and S-SVM is sensor motion SVM. It is trained by a 12-min video by taking 80% data for training and 20% for testing. The accuracies are 97.5% and 99.6% for classifying the four motions from video and sensor data, respectively. The precision and recall are around 95% to 98% for four types of motions. The results show that SVM models are reliable for obtaining motion features.

Table 1. Processing time of major components.

	YOLOv3 (2 Cameras)	**SORT**	**GID Tracking**	**Device Pairing**	**Overall**
Average Time Consumption (s)	0.058	0.003	0.014	0.002	0.081

Table 2. Evaluation of SVM activity recognition models.

	Train#	Test#	Accuracy
V-SVM	11,020	2,756	0.975
S-SVM	143,164	35,792	0.996

	Stop		Straight		Turn Left		Turn Right	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
V-SVM	0.997	0.997	0.948	0.981	0.964	0.981	0.991	0.968
S-SVM	0.989	0.999	0.994	0.989	0.999	0.996	0.999	0.996

Figure 8 shows our experimental scenario. We invited four testers to walk under the cameras. Three of them wear a smartphone on their chests, as shown in Figure 8b. The last one is a guest without any device. In Figure 8a, an identified person will be tagged as a green bounding box with its corresponding profile. The yellow bounding box indicates the result of recovered missed detection, which will be discussed in Section 4.3. We designed three mobility patterns, as shown in Figure 9. Pattern 1 is a random walk to observe tracking ability when people are walking freely inside the area. We generated two trials. Case 1a and 1b are two different random trajectories. Pattern 2 is following with multiple interleaving, which may cause an ID switch due to SORT [13], because SORT does not consider the target's visual features on frames. We also generated two trials (2a and 2b) for pattern 2 to observe the ID switch issue. Our system can solve this issue by recovering the lost bounding box. Pattern 3 is trajectory mimicking to simulate that people have similar moving patterns. Since the moving patterns are similar, we generated one trial (case 3) for pattern 3. In each case, video clips and sensor data were collected for 5 min.

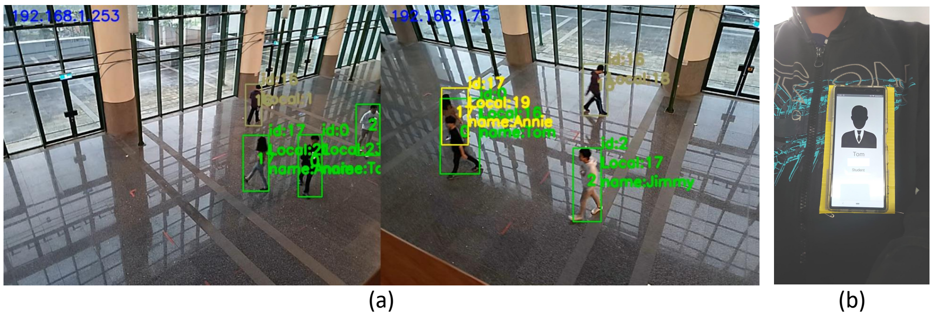


Figure 8. (**a**) Experiment scene. (**b**) Smartphone setup on chest.

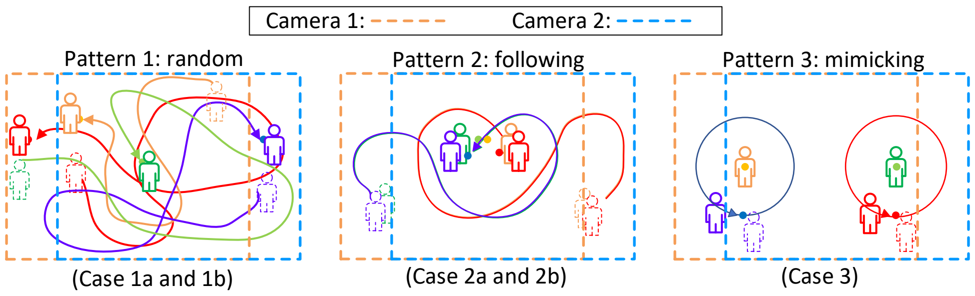


Figure 9. Mobility patterns for experiments.

Our evaluations are based on four metrics: (i) PITA (Person Identification and Tracking Accuracy), (ii) IDP (Identification Precision), (iii) IDR (Identification Recall) and (iv) IDF1 (Identification F-score 1).

$$\text{PITA} = 1 - \frac{\sum_t^N (ObjMiss + IncPair)_t}{\sum_t^N (truth\#)_t} \tag{3}$$

$$\text{IDP} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4}$$

$$\text{IDR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5}$$

$$\text{IDF1} = 2\frac{\text{IDP} * \text{IDR}}{\text{IDP} + \text{IDR}} \tag{6}$$

In the above, ObjMiss means missing human object detection (including false positive and negative), and IncPair means incorrect pairing of devices and human objects. TP is the number of true positive pairs, FP is the number of false positive pairs, and FN is the number of false negative pairs.

4.2. Pairing Accuracy

First, we compare single-camera versus multi-camera cases with the two-layer method. Figure 10a shows PITA, IDP, IDR and IDF1 under different mobility patterns. PITA is the accuracy of a whole video clip and sensor data. The value of IDP, IDR and IDF1 is calculated for each person. When more than two people appear in the scene, several pairing results are obtained. We point out the maximum and minimum values in Figure 10 to examine the robustness of our model. Using multiple cameras gives higher accuracy in all cases because it provides more view angles on the environment, thus improving vision recognition capability. On average, accuracy is increased by 10%. The difference between the maximum and minimum value of IDP, IDR, and IDF1 under two cameras is lower than one camera, which means using multiple cameras is more robust.

Next, we compare the two proposed pairing methods. Here, 1L and 2L mean 1-layer and 2-layer methods, respectively. TSync means using time synchronization; otherwise, Dynamic Time Warping (DTW [37]) will be applied. For Equation (2), n is set to 0.9 for DTW1L and TSync1L, and n is set to 0 for TSync2L. The number m of historical records is set to 10, and the fusion frequency is 2 Hz. Figure 10b shows the results using different pairing methods, where there are always two cameras. We can see that TSync2L performs the best in most cases. For mobility pattern 2, false negative detection occurs due to obstacles. Therefore, TSync1L and DTW1L have much lower accuracy than TSync2L. On average, TSync2L leads by 10% in PITA and IDF1. Considering the difference between the maximum and minimum value, TSync2L is the most robust, and TSync1L outperforms DTW1L.

When DTW is applied, short-term data misalignment may be recoverable. However, it is hard to recover longer misalignment. Therefore, clock synchronization is more important when there are more devices and cameras.

When investigating object tracking, it is common to use an F-score of 1 for comparison. Table 3 shows the detail IDF1 of each person in each case. When the system can not determine the GID of an object, we set it as “unknown”. This would make IDF1 decrease significantly. From Table 3, we can see that IDF1 is greater than 90% under pattern 1. In pattern 2, there are more detection failures, ID switches/exchanges, and occlusions for the computer vision task. Even so, our system can still track people with IDF1 of 80%. Unfortunately, pattern 3 is a challenge to our system because people have similar movements. Our current movement features are unable to distinguish such cases. A possible solution is to explore more detailed features, such as skeleton data, which can be a direction of future work.

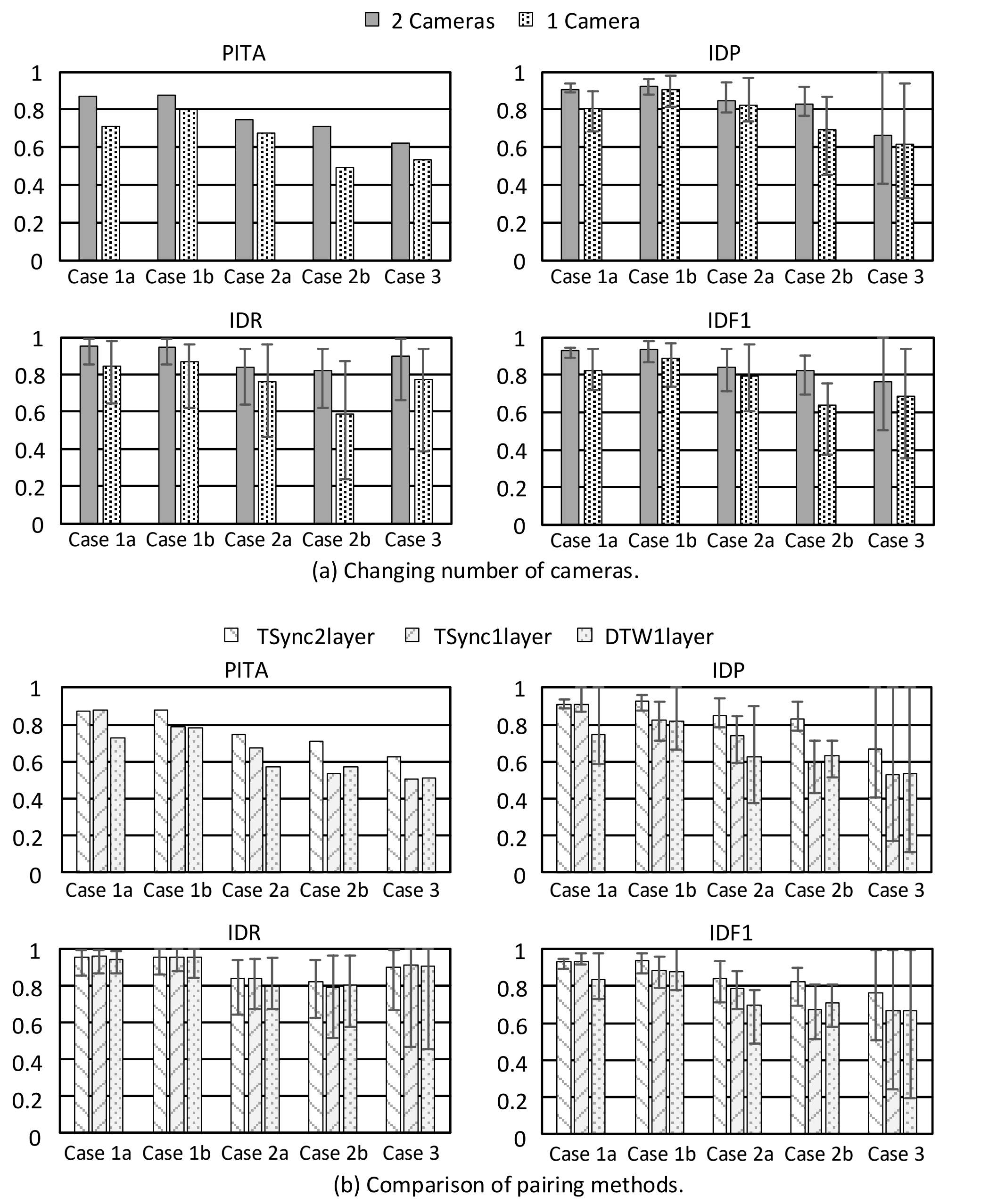


Figure 10. Pairing accuracy.

Table 3. Comparison of Per-person IDF1.

	Case 1a	**Case 1b**	**Case 2a**	**Case 2b**	**Case 3**
User 1	0.937	0.972	0.932	0.88	0.992
User 2	0.915	0.910	0.857	0.855	0.867
User 3	0.940	0.975	0.837	0.853	0.557
Unknown	0.927	0.882	0.725	0.71	0.508

4.3. False Negative Recovery Capability

False negative detection is inevitable in image object detection. As shown earlier, this may be recovered by our system in camera intersection regions. In order to observe the recovery capability, we collected 1886 pieces of data in the overlapping area $R_1 \cap R_2$ when a person moves in. Ideally, both cameras C_1 and C_2 should be able to detect and track them. We treat the tracking results of C_1 as ground truth. When solving the device–object pairing problem, we assume that C_1 cannot detect the person, and then we use the information from C_2 to recover the missed detection of C_1. The recovered detection is regarded as a new bounding box of C_1, and then we compare these new bounding boxes against those detected by the same C_1 through YOLO. Figure 11 shows a scenario. A person in black clothes is in the overlapping area $R_1 \cap R_2$. Although both C_1 and C_2 may detect the person, we purposefully remove C_1's detection. During the data fusion procedure, we use the information from C_2 to recover the lost bounding box on C_1. Then the predicted results for C_1 are compared against the ground truth (i.e., the one detected by YOLO). The outcomes are measured by Intersection over Union (IoU), $IoU = (Area\,of\,Overlap)/(Area\,of\,Union)$, of these two bounding boxes. An IoU value closer to 1 indicates a better recovery effect.

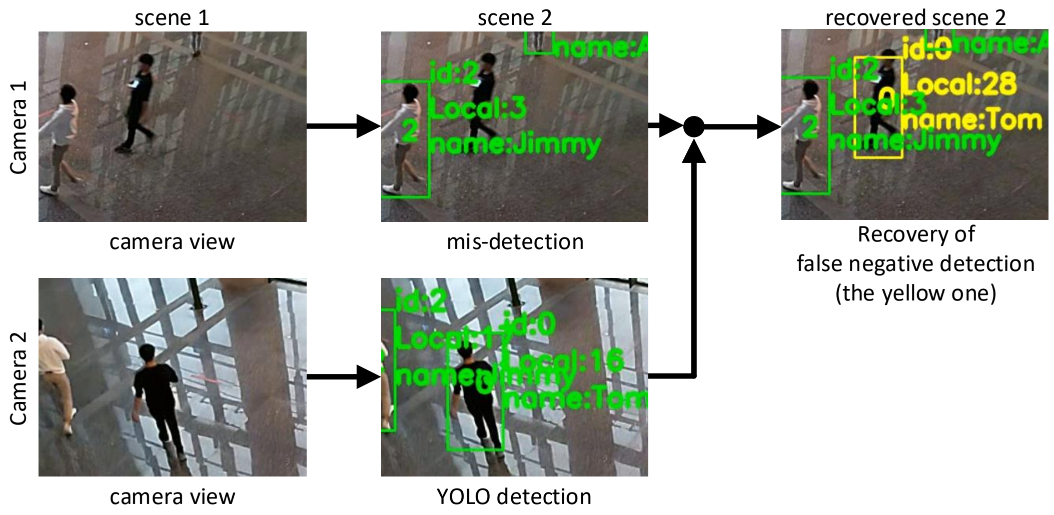


Figure 11. False negative recovery scenario.

Figure 12a shows the histogram of IoU of these 1886 tests. We see that the top three IoU ranges fall in the (0.6, 0.9] interval, and 50% of results are in the (0.6, 1] interval. Figure 12b shows the histogram of the central pixel distance of two bounding boxes. We see that over 75% of the distances fall in the (0, 15] interval. Even though the view angles of C_1 and C_2 are different, most of our predicted bounding boxes by C_2 can achieve an IoU above 0.7 and a pixel distance of less than 20. On average, the mean IoU is 0.64, and the mean distance is 12 pixels. For surveillance applications, such driftings are acceptable, and thus the solving device–object pairing problem helps us to identify and track people or things from different angles and visualize their sensor data easily.

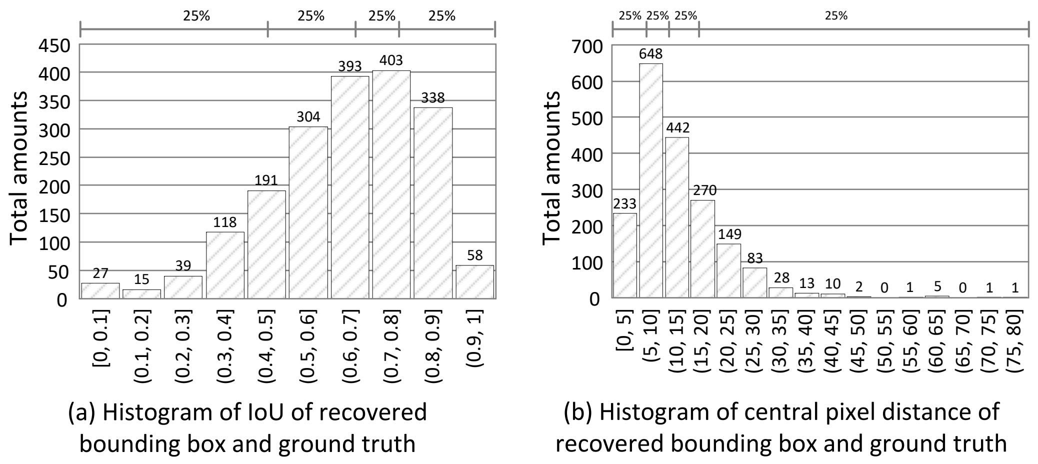


Figure 12. Evaluation results for false negative recovery.

5. Conclusions

The device–object pairing problem arises as an essential issue in the IoT world when we concurrently track a group of things with both cameras and sensors. Correctly pairing visual objects with their sensor data may enable lots of new applications. For example, in high-value livestock farming, the visual objects can be defined as animals. In an automatic warehouse, we can treat the robots as visual objects. Based on our system, we can interpret the status of visual objects with sensor information. This work proposes a device–object pairing system consisting of multiple cameras and wearable devices. The overlapping area between cameras can be used for detecting identical objects. When a missed detection occurs, we can use the information from another camera to recover the missing bounding box. We design a lightweight human-assisted process to estimate a homography matrix for each camera that transforms each camera pixel to a ground coordination. The procedure can be finished within a few minutes. To find the relationship between wearable devices and visual objects, we extract motion features from them and then design one-layer and two-layer algorithms to predict possible pairing. A prototype has been built to test the feasibility of our system under several actual scenarios. It also demonstrates the ability to recover the missing bounding box. Future work may be directed to considering larger scales, which involve a more efficient deployment procedure and an IoT platform for device management. In order to improve device–object pairing results, more features, such as sub-meter indoor localization or skeleton recognition, can be taken into consideration.

Author Contributions: Conceptualization, K.-L.T., K.-R.W. and Y.-C.T.; methodology, K.-L.T.; software, K.-L.T.; validation, K.-L.T., K.-R.W. and Y.-C.T.; formal analysis, K.-L.T.; investigation, K.-L.T., K.-R.W. and Y.-C.T.; resources, K.-R.W., Y.-C.T.; data curation, K.-L.T., K.-R.W.; writing—original draft preparation, K.-L.T., K.-R.W. and Y.-C.T.; writing—review and editing, K.-L.T., K.-R.W. and Y.-C.T.; visualization, K.-L.T., K.-R.W.; supervision, Y.-C.T.; project administration, K.-R.W., Y.-C.T.; funding acquisition, Y.-C.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by ITRI, Pervasive Artificial Intelligence Research (PAIR) Labs, and Ministry of Science and Technology (MoST) under Grant 110-2221-E-A49-033-MY3 and 110-2634-F-009-019. This work is also financially supported by "Center for Open Intelligent Connectivity" of "Higher Education Sprout Project" of NYCU and MOE, Taiwan.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

YOLO	You Only Look Once
RMPE	Regional Multi-person Pose Estimation
FPS	Frames Per Second
MOT	Multiple Object Tracking
RSSI	Received Signal Strength Indicator
IMU	Inertial Measurement Unit
JSON	JavaScript Object Notation
SORT	Simple Online and Realtime Tracking
GT	Global Tracking table
LID	Local ID
GID	Global ID
TGT	Tentative Global Tracking table
SVM	Support Vector Machine
V-SVM	video motion SVM

S-SVM	sensor motion SVM
PITA	Person Identification and Tracking Accuracy
IDP	Identification Precision
IDR	Identification Recall
IDF1	Identification F-score 1
TSync	Time Synchronization
DTW	Dynamic Time Warping
IoU	Intersection over Union

References

1. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [CrossRef] [PubMed]
2. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
3. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *43*, 172–186. [CrossRef]
4. Fang, H.; Xie, S.; Tai, Y.; Lu, C. RMPE: Regional Multi-person Pose Estimation. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2353–2362.
5. Ceccato, V. Eyes and Apps on the Streets: From Surveillance to Sousveillance Using Smartphones. *Crim. Justice Rev.* **2019**, *44*, 25–41. [CrossRef]
6. Kostikis, N.; Hristu-Varsakelis, D.; Arnaoutoglou, M.; Kotsavasiloglou, C.; Baloyiannis, S. Towards remote evaluation of movement disorders via smartphones. In Proceedings of the International Conference of the IEEE Engineering in Medicine and Biology Society, Boston, MA, USA, 30 August–3 September 2011; pp. 5240–5243.
7. Bramberger, M.; Doblander, A.; Maier, A.; Rinner, B.; Schwabach, H. Distributed Embedded Smart Cameras for Surveillance Applications. *IEEE Comput.* **2006**, *39*, 68–75. [CrossRef]
8. Vasuhi, S.; Vaidehi, V. Target Tracking using Interactive Multiple Model for Wireless Sensor Network. *Inf. Fusion* **2016**, *27*, 41–53. [CrossRef]
9. Dai, J.; Zhang, P.; Wang, D.; Lu, H.; Wang, H. Video Person Re-Identification by Temporal Residual Learning. *IEEE Trans. Image Process.* **2019**, *28*, 1366–1377. [CrossRef]
10. Rathod, V.; Katragadda, R.; Ghanekar, S.; Raj, S.; Kollipara, P.; Rani, I.A.; Vadivel, A. Smart Surveillance and Real-Time Human Action Recognition Using OpenPose. In *ICDSMLA 2019*; Springer: Singapore, 2019; pp. 504–509.
11. Lienhart, R.; Maydt, J. An Extended Set of Haar-like Features for Rapid Object Detection. *Proc. Int. Conf. Image Process.* **2002**, *1*, 900–903.
12. Berclaz, J.; Fleuret, F.; Turetken, E.; Fua, P. Multiple Object Tracking Using K-Shortest Paths Optimization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 1806–1819 [CrossRef] [PubMed]
13. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple Online and Realtime Tracking. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468.
14. YuEmail, F.; Li, W.; Li, Q.; Liu, Y.; Yan, X.S.J. POI: Multiple Object Tracking with High Performance Detection and Appearance Feature. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 36–42.
15. Eshel, R.; Moses, Y. Homography based Multiple Camera Detection and Tracking of People in a Dense Crowd. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Anchorage, AK, USA, 23–28 June 2008.
16. Spanhel, J.; Bartl, V.; Juranek, R.; Herout, A. Vehicle Re-Identifiation and Multi-Camera Tracking in Challenging City-Scale Environment. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Long Beach, CA, USA, 16–20 June 2019.
17. Neff, C.; Mendieta, M.; Mohan, S.; Baharani, M.; Rogers, S.; Tabkhi, H. REVAMP^{2}T: Real-Time Edge Video Analytics for Multicamera Privacy-Aware Pedestrian Tracking. *IEEE Internet Things J.* **2020**, *7*, 2591–2602. [CrossRef]
18. Kokkonis, G.; Psannis, K.E.; Roumeliotis, M.; Schonfeld, D. Real-time Wireless Multisensory Smart Surveillance with 3D-HEVC Streams for Internet-of-Things (IoT). *J. Supercomput. Vol.* **2017**, *73*, 1044–1062 [CrossRef]
19. Zhang, T.; Chowdhery, A.; Bahl, P.; Jamieson, K.; Banerjee, S. The Design and Implementation of a Wireless Video Surveillance System. In Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom), Paris, France, 7–11 September 2015; pp. 426–438.
20. Tsai, R.Y.C.; Ke, H.T.Y.; Lin, K.C.J.; Tseng, Y.C. Enabling Identity-Aware Tracking via Fusion of Visual and Inertial Features. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019.
21. Kao, H.W.; Ke, T.Y.; Lin, K.C.J.; Tseng, Y.C. Who Takes What: Using RGB-D Camera and Inertial Sensor for Unmanned Monitor. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019.
22. Ruiz, C.; Pan, S.; Bannis, A.; Chang, M.P.; Noh, H.Y.; Zhang, P. IDIoT: Towards Ubiquitous Identification of IoT Devices through Visual and Inertial Orientation Matching During Human Activity. In Proceedings of the IEEE/ACM International Conference on Internet-of-Things Design and Implementation (IoTDI), Sydney, NSW, Australia, 21–24 April 2020; pp. 40–52.

23. Cabrera-Quiros, L.; Tax, D.M.J.; Hung, H. Gestures In-The-Wild: Detecting Conversational Hand Gestures in Crowded Scenes Using a Multimodal Fusion of Bags of Video Trajectories and Body Worn Acceleration. *IEEE Trans. Multimed.* **2020**, *22*, 138–147. [CrossRef]
24. Nguyen, L.T.; Kim, Y.S.; Tague, P.; Zhang, J. IdentityLink: User-Device Linking through Visual and RF-Signal Cues. In Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing, Seattle, WA, USA, 13–17 September 2014; pp. 529–539.
25. Van, L.D.; Zhang, L.Y.; Chang, C.H.; Tong, K.L.; Wu, K.R.; Tseng, Y.C. Things in the air: Tagging wearable IoT information on drone videos. *Discov. Internet Things* **2021**, *1*, 6. [CrossRef]
26. Tseng, Y.Y.; Hsu, P.M.; Chen, J.J.; Tseng, Y.C. Computer Vision-Assisted Instant Alerts in 5G. In Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 3–6 August 2020; pp. 1–9.
27. Zhang, L.Y.; Lin, H.C.; Wu, K.R.; Lin, Y.B.; Tseng, Y.C. FusionTalk: An IoT-Based Reconfigurable Object Identification System. *IEEE Internet Things J.* **2021**, *8*, 7333–7345. [CrossRef]
28. MQTT Version 5.0. Available online: http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (accessed on 3 August 2021).
29. Standard ECMA-404 The JSON Data Interchange Syntax. Available online: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/ (accessed on 3 August 2021).
30. Romero, A. Keyframes, InterFrame and Video Compression. Available online: https://blog.video.ibm.com/streaming-video-tips/keyframes-interframe-video-compression/ (accessed on 3 August 2021).
31. Camera Calibration with OpenCV. Available online: https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html (accessed on 3 August 2021).
32. Basic Concepts of the Homography Explained with Code. Available online: https://docs.opencv.org/master/d9/dab/tutorial_homography.html (accessed on 3 August 2021).
33. The Shapely User Manual. Available online: https://shapely.readthedocs.io/en/latest/manual.html (accessed on 3 August 2021).
34. Clementini, E.; Felice, P.D.; van Ooster, P. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *International Symposium on Spatial Databases*; Springer: Berlin/Heidelberg, Germany, 1993; pp. 277–295.
35. Moore, K.; Landman, N.; Khim, J. Hungarian Maximum Matching Algorithm. Available online: https://brilliant.org/wiki/hungarian-matching/ (accessed on 3 August 2021).
36. Kuhn, H.W. The Hungarian Method for the Assignment Problem. *Nav. Res. Logist. Q.* **1955**, *10*, 83–97. [CrossRef]
37. Berndt, D.J.; Clifford, J. Using Dynamic Time Warping to Find Patterns in Time Series. In Proceedings of the KDD Workshop, Seattle, WA, USA, 31 July–1 August 1994; pp. 359–370.