Regular Paper

FlowScan: Estimating People Flows on Sidewalks Using Dashboard Cameras Based on Deep Learning

Yusuke Hara[1,a) Ryosuke Hasegawa[1,b) Akira Uchiyama[1,c) Takaaki Umedu[2,d)
Teruo Higashino[1,e)

Received: April 5, 2019, Accepted: October 3, 2019

Abstract: In this paper, we propose *FlowScan*: a pedestrian flow estimation technique based on a dashboard camera. Grasping flows of people is important for various purposes such as city planning and event detection. *FlowScan* can estimate pedestrian flows on sidewalks without taking much cost. Currently, dashboard cameras have been becoming so popular for preserving the evidence of traffic accidents and security reasons. *FlowScan* assumes that an application which analyzes video from the camera is installed on an on-board device. To realize such an application, we need to design a method for pedestrian recognition and occlusion-proof tracking of pedestrians. For pedestrian recognition, the application uses Deep Learning-based techniques; CNN (Convolutional Neural Networks) and LSTM (Long-Short-Term-Memory). In this process, the faces and backs of their heads are searched in the video separately to detect not only the number of pedestrians but also their directions. Then, a series of detected positions of heads are arranged into tracks depending on the similarity of locations and colors considering the knowledge about the movement of the vehicle and pedestrians. We have evaluated *FlowScan* using real video data recorded by a dashboard camera. The mean absolute error rate for people flow estimation of both directions was 18.5%, highlighting its effectiveness compared with the state-of-the-art.

Keywords: deep learning, CNN, LSTM, pedestrian tracking, dashboard cameras

1. Introduction

To realize effective and comfortable urban environment, grasping activity of people is a very essential matter. For example, by collecting the information of people flow, we can discover attracting spots and events. Also, we can find urgent situations promptly by comparing the current condition with average ones, and start evacuation process earlier.

Various methods have been proposed so far in order to grasp movement of people. For example, in the congestion degree map [1], people distribution is obtained by position information collected from GPS-enabled mobile phone users. However, these approaches estimate people distribution with rough granularity such as a 250 m × 250 m cell, which is obviously not enough to understand people movement with *spot-level*. For example, pedestrian traffic can be improved if we know the number of pedestrians heading to a station on the west side of a main street. Meanwhile, CCTVs are widely used [2], [3] for estimating people flow with spot-level continuously. The only limitation using CCTVs is high cost to deploy a large number of CCTVs over the target area.

Although there are many research works for pedestrian detection using cameras for driving support [4], detecting pedestrians on sidewalks is more challenging. This is because overlapping of pedestrians in images (occlusion) frequently occurs due to other pedestrians and objects such as poles and roadside trees/plants on sidewalks. For this reason, pedestrian detection for driving support may fail to estimate people flow. For people detection, Mask R-CNN [5] and OpenPose [6] are widely known for their notable performance. However, these works focus on *detection* of targets (e.g., people) which further requires analysis to estimate people flow (i.e., directions of their movement).

Therefore, estimation of people flow requires detection and tracking of pedestrians. One of the state-of-the-art methods is Detect-and-Track [7] which is ranked first in ICCV 2017 PoseTrack challenge. Detect-and-Track uses Mask R-CNN [5] for people detection and visual, location, and pose similarities for tracking of the detected people. Although such approaches based on deep learning are emerging, we may further enhance the performance by using domain knowledge.

In this paper, we propose *FlowScan* to estimate people flows with sidewalk level using a video of a dashboard-mounted camera (dashcam) which has become popular in recent years. By aggregating information from multiple dashcams of different vehicles, we try to achieve spatially high resolution at low cost even for a large urban area. *FlowScan* detects pedestrians using Stewart's method [8] based on deep learning, which is robust to occlusion. Note that we may use other methods such as Mask R-CNN [5]

[1] Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[2] Faculty of Data Science, Shiga University, Hikone, Shiga 522–8522, Japan
[a) y-hara@ist.osaka-u.ac.jp
[b) r-hasegawa@ist.osaka-u.ac.jp
[c) uchiyama@ist.osaka-u.ac.jp
[d) ta-umedu@biwako.shiga-u.ac.jp
[e) higashino@ist.osaka-u.ac.jp

and OpenPose [6] for pedestrian detection. Nevertheless, it is inherently impossible to detect pedestrians hidden by occlusion and to avoid false positives. We overcome this challenge by focusing on the characteristic of videos recorded by dashcams (i.e., *mobile cameras*). Since dashcams are mobile, pedestrians hidden by occlusion in a single frame are likely to be captured from different angles in the other time-consecutive frames. Moreover, we employ the following assumptions from domain specific knowledge. (1) In typical road environment, pedestrians generally move in the same or opposite direction of the observing vehicle. (2) The areas where assumption (1) does not hold like intersections can be recognized from given map data and the position measured by GPS. Such area will be excluded for pedestrian detection. (3) The regions in the captured frames where pedestrians can be seen are specified in advance by hand or some conventional method. (4) The moving speed of pedestrians is roughly known.

FlowScan is two-fold. First, we detect faces and backs of heads by applying Stewart's method [8] to each frame. Note that a dashcam captures *faces* of pedestrians going to the opposite direction against vehicle movement, and vice versa. Therefore, we roughly know moving directions of the pedestrians from the detection results. Then, the detected regions are tracked over time-consecutive frames based on position and color similarities for robust people flow estimation. We regard a sequence over multiple frames as a pedestrian to filter out false positives. For robustness against false negatives, *FlowScan* also allows tracking failure that can occur due to occlusion. The tracking results are directly used to count the number of pedestrians for each direction.

In order to evaluate the performance of *FlowScan*, we conducted an experiment using dashcam video recorded in Osaka downtown. The result highlights the effectiveness of *FlowScan*, showing 18.5% mean absolute error rate of bi-directional people flow estimation. Moreover, we confirmed *FlowScan* outperforms Detect-and-Track [7] which is one of the state-of-the-art tracking methods using Deep Learning.

In summary, our contributions are as below:

- We propose *FlowScan*, a Deep Learning-based method for sidewalk-level people flow estimation with knowledge about typical movement of vehicles and pedestrians specific to roads and sidewalks.
- We evaluate *FlowScan* by real video recorded by dashcams in downtown Osaka which includes 3,973 pedestrians over 20 minutes.
- To demonstrate the effectiveness of the above specific knowledge, we have compared *FlowScan* with Detect-and-Track [7], one of the state-of-the-art methods for people tracking.

2. Related Works

2.1 Pedestrian Detection by Dashcams

With the rapid growth of autonomous car technologies, pedestrian detection by dashcams has been widely studied for safety driving support. Reference [4] detects pedestrians by using HOG (Histogram Of Gradient) feature and SVM (Support Vector Machine). Another approach [9] uses a gradient feature with AdaBoost. Reference [10] also detects pedestrians by using a body parts feature to mitigate the effect of occlusion. Reference [11] combines HOG, self-similarity, and motion features. However, these approaches inherently suffer from limited accuracy due to occlusion and/or false positives since they focus on safety driving support, targeting pedestrians approaching roadways instead of sidewalks.

2.2 Object Detection by CNN

Object detection by CNN (Convolutional Neural Network) has been attracting a vast number of researchers since the impressive achievement of ImageNet [12]. A problem to detect multiple objects with multiple classes in an image is called Localization and Classification which is one of the challenging problems. R-CNN (Recurrent CNN) [13] is a well-known method for this problem, where classification by CNN is applied after extracting candidate areas of target objects by Selective Search [14]. This helps faster detection compared to methods using a sliding window that tries to classify all possible areas. Nevertheless, Selective Search does not work well due to frequent occlusion.

Faster R-CNN [15] improves R-CNN by introducing RPN (Region Proposal Network) instead of Selective Search to accelerate computation time. Mask R-CNN [5] further extends Faster R-CNN to achieve image segmentation, which means pixel-level segmentation rather than object detection based on bounding boxes. Also, Stewart et al. [8] proposed a method based on Long Short Term Memory (LSTM) [16] for robustness against occlusion. Stewart's method does not require extraction of candidate areas. Instead, it achieves robustness to occlusion by detecting the region where a pedestrian most likely exists one by one. The other advantage is that it detects pedestrians even from some parts of a body such as an upper body, an arm, and a leg.

Among the above sophisticated methods for pedestrian detection, in this paper, we use Stewart's method for pedestrian detection in each frame. However, we note that our contribution is to employ the knowledge about specific movement of vehicles and pedestrians in the tracking after the detection. This means *FlowScan* can also employ other detection methods instead of Stewart's method.

2.3 Pedestrian Tracking

Pedestrian tracking is an essential technique to understand people flow. For object tracking, the Correlation Filter (CF) and Deep Learning are often used. CF [17], [18], [19] returns the center position of a given target object after correlation computation, which is processed by fast Fourier transform. However, it is vulnerable to occlusion since CF tries to find a region with high correlation with a tracking target in a previous frame. For pedestrian tracking by dashcams, occlusion is one of the challenging problems since those cameras are installed on relatively low positions such as a dashboard.

Reference [20] uses CNN for object tracking by a binary classifier (target or others) and iterates classification around the estimated target location in the previous frame. Nevertheless, for videos captured by dashcams, tracking may often fail since especially leaving pedestrians (back of heads) look very similar. Reference [21] combines a deep learning-based object detector called YOLO [22] and Long-Short-Term-Memory (LSTM) for robustness against occlusion. Also, Detect-and-Track [7] combines Mask R-CNN [5] with tracking based on visual, location and pose similarities. Different from these approaches, *FlowScan* leverages the knowledge about typical movement of vehicles and pedestrians, which is specific to roads and sidewalks.

3. *FlowScan*

3.1 Assumptions

We assume videos are recorded by dashcams of some cooperative drivers or a few workers. The recorded videos are transmitted via a gateway such as a smartphone and processed for people flow estimation at a server. However, to reduce the amount of mobile data transmission, in-vehicle processing is also possible by using processors with mobile GPU. In this case, each vehicle sends only the estimated people flow rather than videos.

We define people flow as the numbers of pedestrians going to the same/opposite directions as the moving direction of the vehicle. We call a pedestrian going to the same (opposite) direction as the vehicle as a *leaving* (*approaching*) *pedestrian*. The estimated people flow is visualized onto a map as depicted in **Fig. 1** based on the location information obtained by GPS, etc. Such information about the people flow is useful for various purposes such as marketing, city planning, evacuation planning, and so on.

Here, we assume that each dashcam records the view through a vehicle's front windscreen. We also assume parameters of the camera are given, such as the mounted position, the angle of view, and the resolution. To apply our method to the environments where they are mounted in some slanted direction, we must insert a perspective transformation process after pedestrian detection process into our method. For people flow estimation, we assume that the regions containing sidewalks in the captured videos are roughly known. Under this assumption, we use a region of 640×480 pixels that are manually defined beforehand. Note that the target regions can be also estimated by some techniques although it is out of scope of this paper. For example, image recognition of lane lines is useful to estimate possible regions of sidewalks. It is also useful to combine vehicle positions obtained by GPS, camera install settings, and road information (e.g., the number of lanes). An alternative method is simply applying pedestrian detection to the whole part of time-consecutive frames, and roughly identifying the target region from the position distribution of the detected pedestrians. We note that the design of *FlowScan* in this paper focuses on straight streets without any junctions to know the regions containing sidewalks. However, the above alternatives may enable us to automatically identify such regions.

We also assume that almost all pedestrians are either leaving or approaching pedestrians. This assumption is natural since most pedestrians move toward their destinations along with roads. However, we manually excluded videos while many pedestrians were waiting to cross the road around intersections. Note that this is possible by using position information obtained by GPS.

3.2 Overview

As shown in **Fig. 2**, *FlowScan* is two-fold: (i) pedestrian detection for each frame, and (ii) pedestrian tracking for multiple time-consecutive frames. In pedestrian detection, pedestrians are detected with their directions that are either faces or backs of heads. For this purpose, we utilize Stewart's method [8] which combines CNN and LSTM for robust detection. To further mitigate the effect of false negatives and false positives, pedestrian tracking is conducted for multiple time-consecutive frames.

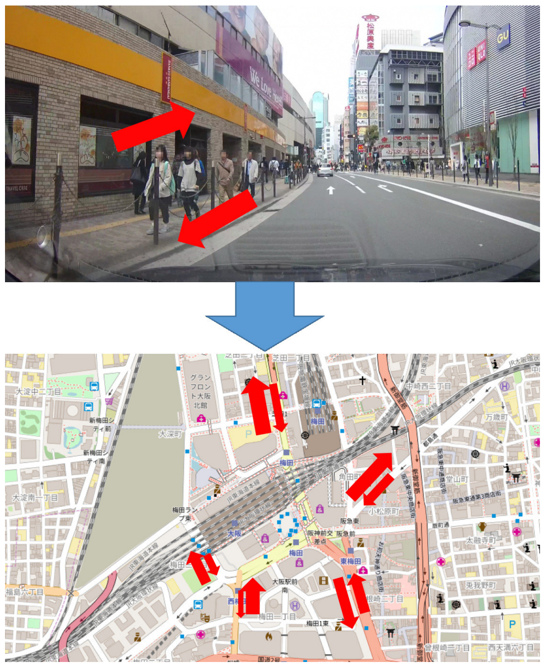


Fig. 1 People flow estimation example (Copyright © OpenStreetMap contributors).

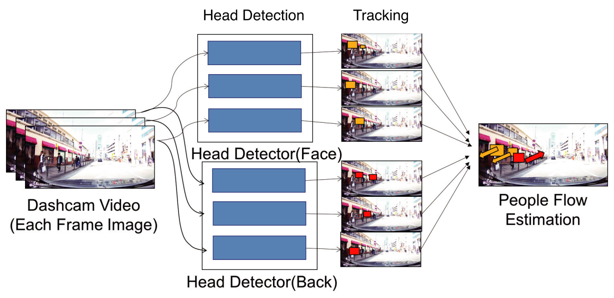


Fig. 2 *FlowScan* overview.

3.3 CNN-based Pedestrian Detection

Figure 3 illustrates the overview of pedestrian detection. For a target region of 640×480 pixels, 1024-dimensional vectors for each 20×15 cell are obtained by GoogLeNet as a feature. Each feature vector represents a summary of the corresponding region, including positions of objects. Therefore, we input features of each cell to LSTM which outputs a bounding box with reliability representing the position of the target (i.e., a face or a back of a head). LSTM outputs a bounding box with the highest reliability which is also an input to the next LSTM unit. We use the same loss function as Ref. [8]. By doing so, multiple detection of the same target is avoided. Finally, LSTM outputs all bounding boxes with reliabilities over threshold T for aggregating them to obtain the detection result of a frame. We tune the threshold T in Section 4.2.1.

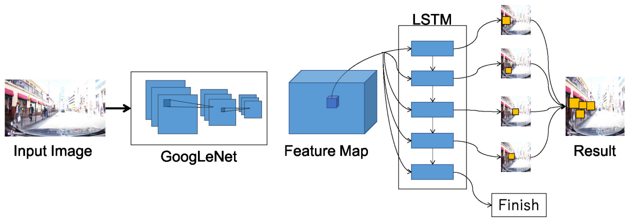


Fig. 3 Pedestrian detection by CNN.

3.4 People Flow Estimation by Pedestrian Tracking

For people flow estimation, we aggregate pedestrian detection results of multiple time-consecutive frames to track movement of each pedestrian. Since pedestrian detection focuses on the features in a single frame (at the moment), we cannot avoid false positives and false negatives. To overcome this problem, we fully utilize the detection results of the time-consecutive frames.

An overview of the tracking algorithm is described in Algorithm 1. We denote the i-th bounding box at frame t as $b_i^t \in B^t$. B^t is a set of bounding boxes detected at frame t.

We define a similarity $\mathrm{sim}(b_i^t, b_j^u)$ between b_i^t and b_j^u for pedestrian tracking as below.

$$\mathrm{sim}(b_i^t, b_j^u) = w l(b_i^t, b_j^u) + (1 - w) v(b_i^t, b_j^u), \tag{1}$$

where $l(b_i^t, b_j^u)$ and $v(b_i^t, b_j^u)$ are similarities according to position and color features. w $(0 < w < 1)$ is a weight. The details of these two similarities are described in Sections 3.4.1 and 3.4.2.

We have to identify bounding boxes representing the same target over multiple time-consecutive frames for pedestrian tracking. This is processed as follows. We define a set $y(b_i^t)$ of bounding boxes regarded as the same pedestrian. First, we initialize $y(b_i^t) = \{b_i^t\}$ for each b_i^t. Then, for each pair of bounding boxes with the highest similarity $\mathrm{sim}(b_i^t, b_j^{t+1})$ in frames t and $t+1$, we update $y(b_i^t)$ as $y(b_i^t) = y(b_j^{t+1}) = y(b_i^t) \cup y(b_j^{t+1})$ if $\mathrm{sim}(b_i^t, b_j^{t+1})$ exceeds the threshold S, regarding b_i^t and b_j^{t+1} are the same pedestrian. Then, b_i^t is removed from B^t. If such a bounding box is not found, the same process is repeated for the next frame $t+2$. This process is iterated until any corresponding bounding box is found up to $t+\Delta t$ frame, which enables *FlowScan* to tolerate false negatives in some frames. Similarly, the same process is iterated between frames $t+1$ and $t+2, t+3, \ldots, t+\Delta t+1$. In this paper, we empirically set $\Delta t = 10$. By applying the above process to all the frames, we obtain multiple sets of bounding boxes each of which represents a trajectory of a pedestrian.

Finally, we introduce a parameter W, which represents the minimum number of frames for the series of bounding boxes to be considered as a track. That is, we ignore $y(b_i^t)$ whose size is less than W since it is likely to be a false positive. W can be set for each direction (face or back) independently. The result of pedestrian tracking directly indicates people flow, representing the number of pedestrians going to each direction.

3.4.1 Location Similarity

Location similarity $l(b_i^t, b_j^u)$ is defined for the center position p_i^t of bounding box b_i^t at frame t and p_j^u at frame u $(t < u)$. Intuitively, for a pedestrian detected at frame t, we predict his position in the latter frame based on the relative speed of the vehicle and the pedestrian. The vehicle speed is obtained by GPS while

Algorithm 1 Pedestrian Tracking Algorithm

```
Input: Set $B^t$ of detected bounding boxes at all frames
Output: Set $T$ of estimated trajectories
  for each frame $t$ of a recorded video do
      for each $b_i^t \in B^t$ do
          $y(b_i^t) = \{b_i^t\}$
      end for
  end for
  for each frame $t$ of a recorded video do
      for $k = 1$ to $\Delta t$ do
          for each $b_i^t \in B^t$ do
              $b_j^u = \underset{b_j^{t+k} \in B^{t+k} \| |y(b^{t+k})| == 1}{\arg\max} sim(b_i^t, b_j^{t+k})$
              if $sim(b_i^t, b_j^{t+k}) > S$ then
                  $y(b_i^t) = y(b_j^{t+k}) = y(b_i^t) \cup y(b_j^{t+k})$
                  remove $b_i^t$ from $B^t$
              end if
          end for
      end for
  end for
  for each $y(b_i^t)$ do
      if $|y(b^t)| > W$ and $y(b^t) \notin T$ then
          add $y(b^t)$ to $T$
      end if
  end for
```

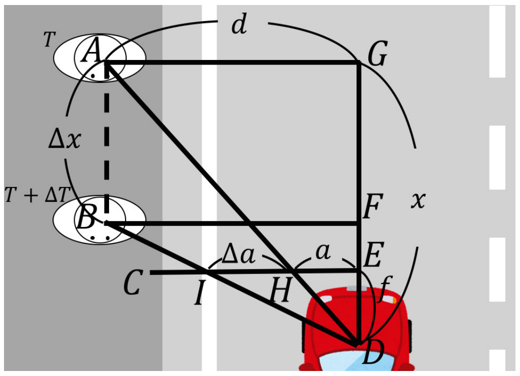


Fig. 4 Camera position and perspective pedestrian location.

we assume the pedestrian walks at the average walking speed (1.11 m/s) of them. For a detected pedestrian (bounding box) at p_i^t, the predicted position $p_i^{t\to u}$ at frame u is obtained.

In the target environment, horizontal movements are dominant. That is, pedestrians and vehicles move mainly horizontally, and the horizontal movement of a pedestrian in the video are larger than the vertical one. So we focus only on the geometry with a view from above as shown in **Fig. 4** to calculate the location similarity. The figure illustrates the position relationship between a dashcam and a pedestrian from above. Suppose pedestrian P is at position A at frame T and a camera is at D facing toward E which is parallel with the sidewalk. The distance between the sidewalk and the direction vector of the camera is denoted as d, which is manually calculated from the video [*1]. The segment CE is the projection plane of the camera. In the perspective view, the pedestrian is positioned at H, which is a pixels apart from the

[*1] In actual use scenarios, we need to estimate d for each frame, which can be achieved by a combination of several methods such as line detection and pedestrian detection. However, in this paper, we focus on the main concept of people flow estimation and leave it out of scope.

center of the image. The pedestrian P moves to B in the perspective view after ΔT sec. The moving distance of P denoted as Δx, which is the length of the segment AB, is estimated as $\widehat{\Delta x}$ from the relative speed between the pedestrian and the vehicle. Suppose P is at I in the image, I is $a + \Delta a$ pixels apart from the center of the image. Here Δa can be estimated as $\widehat{\Delta a}$, which can be calculated from $\widehat{\Delta x}$ based on geometry. Using f, the focal point of the camera in pixel, $p_i^{t\to u}$ can be calculated as:

$$p_i^{t\to u} = a + \widehat{\Delta a} = a + \frac{a^2\widehat{\Delta x}}{fd - a\widehat{\Delta x}}. \tag{2}$$

Then location similarity can be defined by the difference between $\widehat{\Delta a} = p_i^{t\to u} - p_j^u$ and Δa, which is the moving distance in pixels in perspective view (video images). We have defined location similarity as follows using a margin parameter M, empirically defined as 50 pixels.

$$l(b_i^t, b_j^u) = max\left(0, 1 - \frac{|\widehat{\Delta a}|}{|\Delta a| + M}\right), \tag{3}$$

3.4.2 Color Similarity

Colors of hair, faces, clothes, etc. are useful for tracking. Therefore, we define the color similarity as the correlation of color histograms for a pair of bounding boxes as below.

$$v(b_i^t, b_j^u) = \frac{1}{|c|}\sum_c \frac{\sum_I (H_i^c(I) - \bar{H}_i^c)(H_j^c(I) - \bar{H}_j^c)}{\sqrt{\sum_I (H_i^c(I) - \bar{H}_i^c)^2 \sum_I (H_j^c(I) - \bar{H}_j^c)^2}}$$

In the above definition, H_i^c is the histogram of color channel c of i-th bounding box b_i^t. The average color $\bar{H}_i^c$ of b_i^t is defined as $\frac{1}{N}\sum_I H_i^c(I)$ where N is the number of bins of the histogram and $H_i^c(I)$ is the frequency of the I-th bin. In our pedestrian detection, a bounding box indicates the region of a face or back of a head. To reflect the color of clothes on our color similarity, we expand the detected bounding box as 3 times longer toward the direction of the ground and use the color histogram of the expanded bounding box for the color similarity. For the color channel c, we use the 3 channels of HSV (Hue, Saturation, and Value).

3.5 Aggregation of Estimated People Flows

In Section 3.4, we described *FlowScan* which estimates people flows by a single dashcam. However, more dashcams of multiple participants may be available in practice. For example, we may implement *FlowScan* as a participatory sensing system where cooperative users upload the estimated people flows to a cloud server. Therefore, aggregation of the estimated people flows is required to obtain the estimation results as shown in Fig. 2. Actually, the aggregation design is non-trivial because we need to consider the reliability of the estimated people flows such as freshness, trust of the upload user, and the estimation quality.

Although we leave the aggregation design as a future work due to the challenges, one of the reasonable aggregation designs is averaging. We denote an estimated people flow by a user from an intersection i and the next intersection j at time t as $f_i^t j(u)$. Then, we obtain the aggregated people flow $\hat{f}_i^t j$ at time t from i to j by averaging the uploaded people flows from $t - W_a$ to $t + W_a$ where W_a is a time window size.

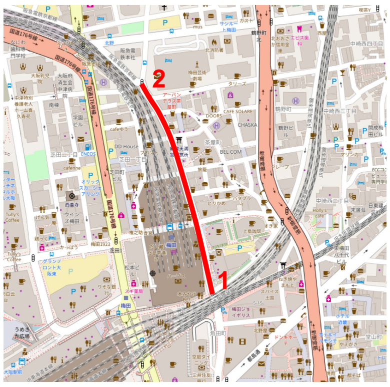


Fig. 5 Experiment area (Chayamachi, Osaka) (Copyright © OpenStreetMap contributors).

4. Evaluation

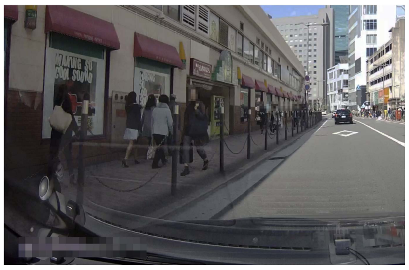
4.1 Experiment Environment

For evaluation, we recorded videos in Chayamachi area, which is a downtown area near Osaka Station in Umeda district, Japan. The driving route is illustrated in **Fig. 5** with a red line between points 1 and 2. We placed a dashcam (DRY-WiFiV5c, Yupiteru Corporation) onto a vehicle's dashboard and drove the route several times around noon on holidays when many pedestrians are walking.

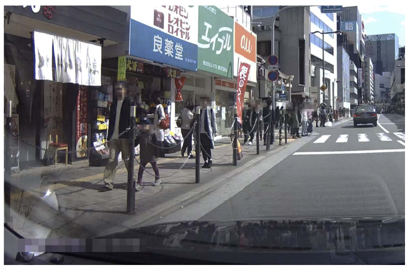
For training of Stewart's method, we used the same learning parameter settings as described in Ref. [8] and the software library published by the authors [*2]. For deep learning, we used a workstation with Intel Xeon CPU E5-1680 v3 @ 3.20 GHz, 128 GB of memory and GeForce GTX 1080 GPU.

We used four datasets for evaluation: (1) the *head dataset* for pedestrian detection experiment, (2) the *people flow dataset A* consisting of 104 segments on one side of the street, (3) the *people flow dataset B* consisting of 15 segments on the other side of the street, and (4) the *speed dataset* consisting of 5 segments on the same side of the street as the *people flow dataset A*. **Figures 6** (a) and 6 (b) show captured pictures from dataset A and B, respectively. The videos in each dataset contain a lot of similar backgrounds, while the backgrounds are different between datasets A and B as shown in the pictures. We use dataset A for evaluation mainly. Dataset B is used to check if the detector works well with pictures that have different backgrounds from training data. To build the *head dataset*, we have picked every 10 frames (images) from the recorded video and manually labeled 2,560 faces and 2,695 backs of heads of pedestrians for training data. For the *people flow datasets A* and *B*, each segment consists of a video recorded while the vehicle was driving from an intersection to the next one. The speed data of the vehicle are recorded by a GPS receiver together with the video. According to the data, the vehicle drove mainly at speeds of less than 30 km/h. Note that we excluded some parts recorded while the vehicle stopped at intersections. In total, the numbers of approaching and leaving pedestrians are 2,280 and 1,693 for the *dataset A*, respectively. Also, the numbers of approaching and leaving pedestrians are 629 and 444 for the *dataset B*, respectively. We randomly chose

[*2] https://github.com/Russell91/ReInspect



(a) Dataset A (main side)



(b) Dataset B (opposite side)

Fig. 6 Photos of the target side and the opposite side.

5 segments from the *dataset A* to build the *speed dataset*.

The similarity weight in Eq. (1) is set as $w = 0.5$ for all the experiments. We note that the parameter settings are different in each of the following sections because we use different datasets for parameter tuning (i.e., training). Especially, in Sections 4.2.4 and 4.2.5, we conducted 10-fold cross validation for the *dataset A*. The details of the parameter settings in the other evaluation are described later.

For an evaluation metric, we use an absolute error rate which is defined as:

$$\frac{|\tilde{N}_i^s - N_i^s|}{N_i^s}, \tag{4}$$

where $\tilde{N}_i^s$ is the estimated number of pedestrians and N_i^s is the ground truth in a segment s. The absolute error rate is calculated independently for approaching and leaving pedestrians. In the following evaluation, we use the average of the absolute error rate (i.e., mean absolute error (MAE) rate) of all the segments in the test data unless otherwise stated.

4.2 Evaluation Results

4.2.1 Pedestrian Detection Tuning

For the tuning of Stewart's method, we have used 100 images randomly chosen from the head dataset. **Figure 7** shows precision and recall for different threshold T of Stewart's method in pedestrian detection. For comparison with a simple method, Fig. 7 also shows the head detection result by a cascade classifier based on Haar-like feature proposed by Ref. [23]. We used its implementation provided by OpenCV. For training the cascade classifier, we cropped the head images from the learning data and built the classifiers of faces and backs. Then, we also changed a threshold parameter of the cascade classifier to detect a target.

Precision and recall of the cascade classifier range from 0.1 to 0.7 and 0.7 to 0.3, respectively. Specifically, in the case of the highest precision, recall decreased to 0.3, which is obviously insufficient. On the other hand, precision and recall of Stewart's

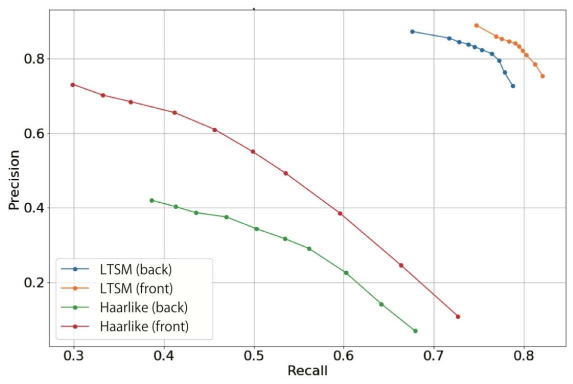


Fig. 7 Effect of threshold (precision and recall).

method are within the range from 0.7 to 0.85 and 0.85 to 0.68 for both directions, respectively.

Meanwhile, in both cases of the cascade classifier and Stewart's method, the performance of leaving pedestrians is lower than approaching pedestrians. This is because we can see faces of approaching pedestrians. A face image contains various parts such as eyes, nose, and mouth while images of leaving pedestrians contain much less features.

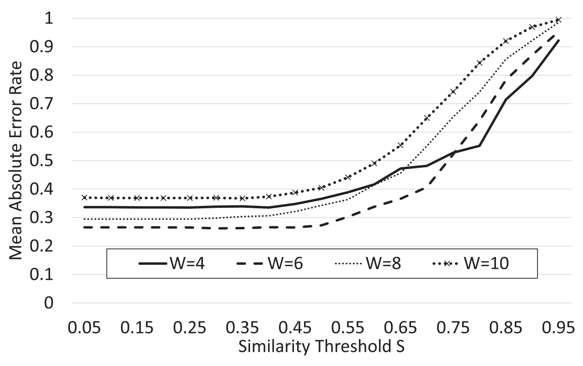
In *FlowScan*, people flow estimation is performed by combining head detection in multiple frames. However, the performance of head detection also affects the other parameters W and S since the reliability of the head detection could vary. Therefore, we add the parameter T to the parameter set for optimization in the following sections.

4.2.2 Effect of Threshold Parameters

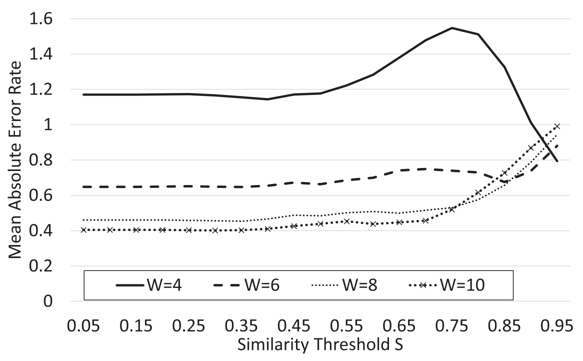
We evaluate the effect of the similarity threshold S and the minimum number of frames W for the *dataset A*. **Figures 8** (a) and 8 (b) show the MAE rate of approaching and leaving pedestrians for different S, respectively. We see that the MAE rate increases with the increase of S in most cases. This is because the correct trajectories are wrongly filtered if S is extremely high. We also see the decrease of MAE rate when S is greater than 0.75 for $W = 4, 6$ in leaving pedestrians. In addition, the MAE rate of $W = 4, 6$ is relatively higher than the others. Therefore, the estimated trajectories contain many wrong trajectories for S less than 0.75, which are filtered by S greater than 0.75. Overall, the performance does not change largely for S less than 0.5. This indicates a good performance of the head detection.

From the results of different W shown in Figs. 8 (c) and 8 (d), we see similar trends in both directions: the MAE rate decreases with the increase of W. This is because *FlowScan* allows short trajectories to detect pedestrians when W is small, which leads to a large number of wrong detections. On the other hand, large W can deteriorate the MAE rate as shown in Fig. 8 (c). This is because correct trajectories are wrongly filtered if W is too large. However, we do not see such deterioration for the leaving pedestrians shown in Fig. 8 (d). This is because backs of heads are harder to detect due to few image features. If we increase W, *FlowScan* requires a longer trajectory to detect a pedestrian. Therefore, larger W is effective for leaving pedestrians (backs of heads) to filter false positives out.

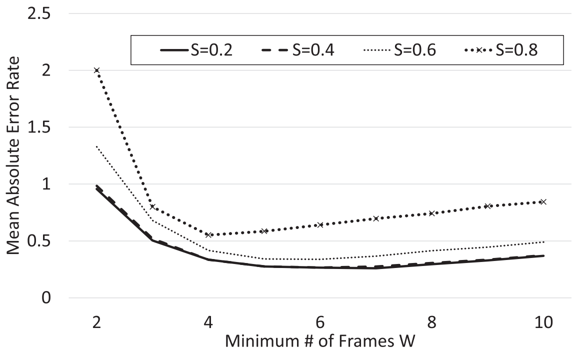
The above results indicate that we need to set the threshold



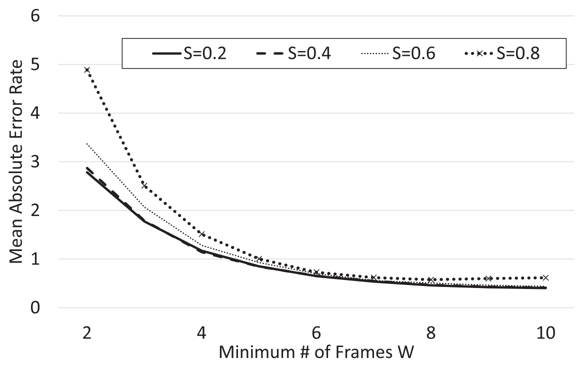
(a) Effect of S for approaching pedestrians



(b) Effect of S for leaving pedestrians



(c) Effect of W for approaching pedestrians



(d) Effect of W for leaving pedestrians

Fig. 8 Effect of similarity threshold S and minimum number of frames W.

parameters appropriately depending on the performance of head detection. To do this, we may prepare several settings adjusted for some typical environments, one of which is chosen for a target environment. Another option is to collect a small dataset for parameter tuning for a target environment. Also, autonomous parameter adjustment may be possible by learning the relationship between parameter settings and head detection results. Since the goal of this paper is to present the basic design of people flow estimation, we leave it as our future work.

Table 1 MAE rate for different vehicle speed.

Direction	*FlowScan*	Fixed Speed
Approaching	24.8%	22.9%
Leaving	26.0%	27.1%

4.2.3 Effect of Vehicle Speed Variation

To see the effect of vehicle speed variation, we evaluated *FlowScan* for videos during stops and moves separately for the speed dataset. For comparison, we have also applied *FlowScan* with the fixed speed ($p_i^{t \to u}$) set by the average speed of the vehicle during the test data. For the parameter setting, we used $T = 0, W = 7$, and $S = 0.5$ for approaching and $T = 0, W = 4$, and $S = 0.8$ for leaving pedestrians which are optimized for the speed dataset.

From the result shown in **Table 1**, we can see the MAE of *FlowScan* for leaving pedestrians is 26.0%, which is better than that with the fixed speed. However, for approaching pedestrians, the MAE of *FlowScan* is 1.9% worse than the fixed speed. This is because the speed did not change greatly and became close to the average speed used as the fixed speed. On the other hand, *FlowScan* relies on the speed calculated by GPS, which actually has some error affecting the tracking performance. To overcome GPS error, we may use speed estimation by a smartphone inertial sensor or speed information retrieval by OBD-II (On-Board Diagnostics).

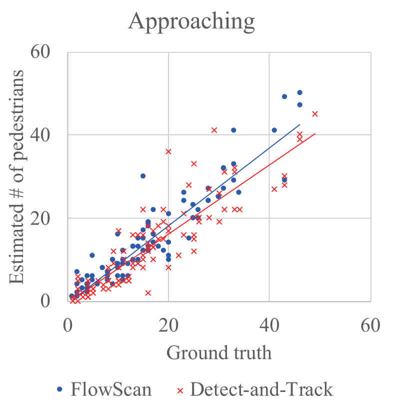
4.2.4 People Flow Estimation

To investigate the performance of people flow estimation, we compared *FlowScan* with Detect-and-Track [7] which is one of the state-of-the-art methods for estimating and tracking human body key points in multi-person videos. We conducted 10-fold cross validation for the people flow dataset A.

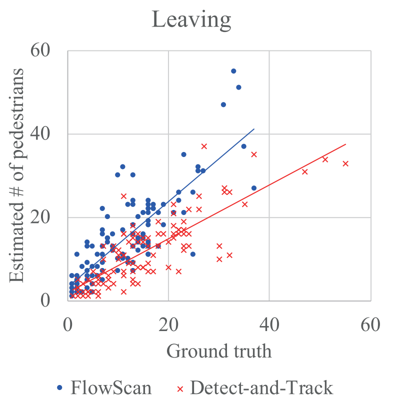
For the implementation of Detect-and-Track, we used the source code available online [*3]. Since Detect-and-Track does not directly estimate the people flow, we applied the following algorithm for people flow estimation based on Detect-and-Track. First, we obtain trajectories of pedestrians detected by Detect-and-Track. Then, for each trajectory, we determine the direction of the trajectory (i.e., forward or backward) based on the positions of left and right shoulders estimated by Detect-and-Track. If a right shoulder is on the right side of the position of the left shoulder in a frame, the pedestrian is supposed to be a leaving pedestrian, and vice versa. Finally, the direction of a trajectory is determined by majority voting from pose estimation results of each frame [*4].

Figures 9 (a) and 9 (b) show the comparison result of *FlowScan* and Detect-and-Track. Here, we have used the number of pedestrians manually labelled as the ground truth. Each marker in the graphs represents the result for each video and the lines represent the regression lines. We thought how crowded the road might affect the performance, however the influence of the number of pedestrians seem relatively small. We note that the esti-

[*3] https://github.com/facebookresearch/DetectAndTrack/
[*4] In the case of a tie, approaching pedestrians have priority in our implementation.



(a) Estimated number of approaching pedestrian and ground truth



(b) Estimated number of leaving pedestrian and ground truth

Fig. 9 Comparison with Detect-and-Track.

mated numbers of pedestrians shown in Fig. 9 are not always correct because it may contain double counting. Therefore, we further evaluate both methods in terms of precision and recall later to see the correctness of the estimation results. For the approaching pedestrians, the MAE rate of *FlowScan* is 8% while Detect-and-Track achieves 19%. Meanwhile, for the leaving pedestrians, the MAE rate of *FlowScan* and Detect-and-Track are 29% and 72%, respectively. The difference between *FlowScan* and Detect-and-Track is much clearer for the leaving pedestrians where the number of features for tracking is less than the approaching pedestrians. Overall, *FlowScan* achieves 18.5% MAE rate while Detect-and-Track achieves 45.5% MAE rate on average. Furthermore, we compared *FlowScan* and Detect-and-Track in terms of precision and recall. We define the that an estimated trajectory is correct if and only if it tracks the same target more than 50% of its associated bounding boxes. Then, we manually counted the numbers of correct/incorrect trajectories to calculate precision and recall. From the result shown in **Table 2**, we have confirmed that *FlowScan* outperforms Detect-and-Track. This is mainly because *FlowScan* leverages the prior knowledge about typical pedestrian movement on sidewalks.

4.2.5 Robustness to Occlusion

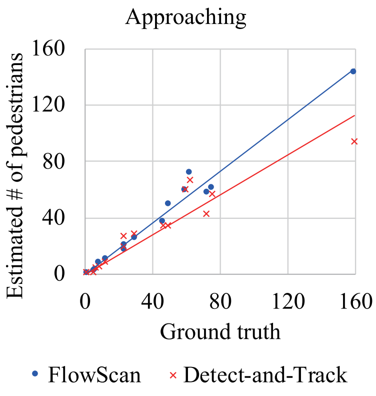
To see the robustness of *FlowScan* to occlusion, we compared the MAE rate between $\Delta t = 0$ and $\Delta t = 10$. As described in Section 3.4, Δt is the parameter to allow false negatives (i.e., occlusion). $\Delta t = 0$ does not allow any false negatives to track the pedestrians.

Table 2 Precision and recall comparison.

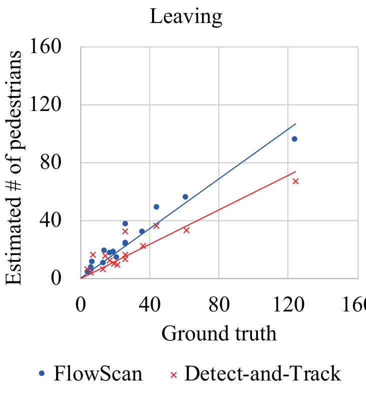
Direction	*FlowScan*	Detect-and-Track
Approaching (Precision, Recall)	(87.7%, 81.1%)	(58.7%, 46.3%)
Leaving (Precision, Recall)	(84.9%, 75.4%)	(51.9%, 35.5%)

Table 3 MAE rate for different Δt.

	$\Delta t = 0$	$\Delta t = 10$
Approaching	35.3%	27.1%
Leaving	42.0%	37.6%



(a) Estimated number of approaching pedestrians and ground truth



(b) Estimated number of leaving pedestrians and ground truth

Fig. 10 Effect of different background.

Table 3 shows the results. We see that $\Delta t = 10$ outperforms $\Delta t = 0$. This is because *FlowScan* allows time-consecutive false negatives up to Δt frames, which prevents accuracy degradation due to occlusion.

4.2.6 Effect of Different Background

In order to see the effect of different backgrounds, we also evaluated *FlowScan* for the people flow dataset B as the test data while the dataset A as the training data. The parameter setting is $T = 0.5$, $W = 6$, and $S = 0.1$ for leaving and $T = 0.5$, $W = 8$, and $S = 0.1$ for approaching pedestrians. Photographs of the dataset A and B are shown in Figs. 6 (a) and 6 (b), respectively.

Figures 10 (a) and 10 (b) show the similar plots to Figs. 9 (a) and 9 (b) of the results for dataset B. For the approaching pedestrians, the MAE rates of *FlowScan* and Detect-and-Track are 14.6% and 28.3%, respectively. Also, for the leaving pedestrians, *FlowScan* achieves 28.3% MAE rate which is better than 44.5% by Detect-and-Track. **Table 4** shows precision and recall. We see

Table 4 Precision and recall comparison (different background).

Direction	*FlowScan*	Detect-and-Track
Approaching (Precision, Recall)	(92.5%, 78.9%)	(55.6%, 43.4%)
Leaving (Precision, Recall)	(84.1%, 77.7%)	(50.2%, 33.6%)

that *FlowScan* achieves more than 84.1% precision and 77.7% recall for both of the approaching and leaving pedestrians, which is much higher than the compared method. The above results indicate that the correct pedestrian tracking by *FlowScan* results in the better estimated numbers of pedestrians shown in Fig. 10. On the other hand, the lower precision and recall of Detect-and-Track indicates the estimated numbers in Fig. 10 may often contain double counting. Overall, we have confirmed that, even for a different background, *FlowScan* still achieves better results than the state-of-the-art.

5. Conclusion

In this paper, we proposed *FlowScan*: a method to estimate sidewalk-level people flow by using dashcams. Our method combines Deep Learning-based pedestrian detection and model-based tracking to overcome the challenges of frequent occlusion and false positives. In pedestrian detection, faces and backs of heads are separately detected to understand moving directions of pedestrians as well as their existence by applying CNN and LSTM. Then, the trajectories of the detected bounding boxes are estimated based on location and color similarities with the knowledge about typical movement of vehicles and pedestrians.

To confirm the effectiveness of *FlowScan*, we have collected real data in Osaka downtown. The results have shown that *FlowScan* achieves 18.5% mean absolute error rate for people flow estimation which is promising to enrich future smart cities. One of our future works is implementation of a real system based on mobile devices such as smartphones and embedded AI computing devices. Also parameters of cameras such as color balance or exposure values vary depending on the cameras and are also automatically adjusted to the environment. So to make our method robust, we should append color and brightness adjustment process before the detection process.

Acknowledgments This work was supported in part by JSPS KAKENHI JP26220001, JP15H02690 and JP26700006.

References

[1] Zenrin-Datacom Company, Limited: Density map, available from ⟨http://lab.its-mo.com/densitymap/⟩.

[2] Silveira Jacques Junior, J., Musse, S. and Jung, C.: Crowd Analysis Using Computer Vision Techniques, *IEEE Signal Processing Magazine*, Vol.27, No.5, pp.66–77 (2010).

[3] Wu, Z., Thangali, A., Sclaroff, S. and Betke, M.: Coupling detection and data association for multiple object tracking, *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pp.1948–1955 (2012).

[4] Zhang, Y., Wang, G., Gu, X., Zhang, S. and Hu, J.: Real-time pedestrian detection with the videos of car camera, *Advances in Mechanical Engineering*, Vol.7, No.12, p.1687814015622903 (2015).

[5] Abdulla, W.: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow (2017), available from ⟨https://github.com/matterport/Mask_RCNN⟩.

[6] Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E. and Sheikh, Y.: OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields, arXiv preprint arXiv:1812.08008 (2018).

[7] Girdhar, R., Gkioxari, G., Torresani, L., Paluri, M. and Tran, D.: Detect-and-Track: Efficient Pose Estimation in Videos, *CVPR* (2018).

[8] Stewart, R., Andriluka, M. and Ng, A.Y.: End-To-End People Detection in Crowded Scenes, *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pp.2325–2333 (2016).

[9] Sabzmeydani, P. and Mori, G.: Detecting Pedestrians by Learning Shapelet Features, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp.1–8 (online), DOI: 10.1109/CVPR.2007.383134 (2007).

[10] Bar-Hillel, A., Levi, D., Krupka, E. and Goldberg, C.: Part-based Feature Synthesis for Human Detection, *Proc. European Conference on Computer Vision: Part IV*, *ECCV'10*, pp.127–142 (2010).

[11] Walk, S., Majer, N., Schindler, K. and Schiele, B.: New features and insights for pedestrian detection, *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.1030–1037 (online), DOI: 10.1109/CVPR.2010.5540102 (2010).

[12] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp.248–255 (2009).

[13] Girshick, R., Donahue, J., Darrell, T. and Malik, J.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp.580–587 (online), DOI: 10.1109/CVPR.2014.81 (2014).

[14] Uijlings, J.R., Sande, K.E., Gevers, T. and Smeulders, A.W.: Selective Search for Object Recognition, *International Journal of Computer Vision*, Vol.104, No.2, pp.154–171 (online), DOI: 10.1007/s11263-013-0620-5 (2013).

[15] Ren, S., He, K., Girshick, R. and Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, *Advances in NeuralInformation Processing Systems* (*NIPS*) (2015).

[16] Hochreiter, S. and Schmidhuber, J.: Long Short-term Memory, *Neural Computation*, Vol.9, No.8, pp.1735–1780 (1997).

[17] Galoogahi, H.K., Sim, T. and Lucey, S.: Correlation filters with limited boundaries, *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pp.4630–4638 (2015).

[18] Ma, C., Huang, J.-B., Yang, X. and Yang, M.-H.: Hierarchical Convolutional Features for Visual Tracking, *Proc. IEEE International Conference on Computer Vision* (*ICCV*), pp.3074–3082 (2015).

[19] Valmadre, J., Bertinetto, L., Henriques, J.F., Vedaldi, A. and Torr, P.H.: End-to-end representation learning for Correlation Filter based tracking, *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pp.5000–5008 (2017).

[20] Nam, H. and Han, B.: Learning Multi-domain Convolutional Neural Networks for Visual Tracking, *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pp.4293–4302 (2016).

[21] Ning, G., Zhang, Z., Huang, C., He, Z., Ren, X. and Wang, H.: Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking, *Proc. IEEE International Symposium on Circuits and Systems* (*ISCAS*), pp.1–4 (2016).

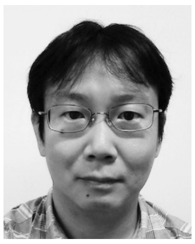
[22] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection, *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), pp.779–788 (online), DOI: 10.1109/CVPR.2016.91 (2016).

[23] Viola, P. and Jones, M.: Rapid object detection using a boosted cascade of simple features, *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol.1, pp.I-511–I-518 (2001).

Yusuke Hara received his M.E. degree in Information and Computer Science from Osaka University, Japan in 2018. He works at Panasonic since 2018.

Ryosuke Hasegawa received his M.E. degree in Information and Computer Science from Osaka University, Japan in 2019. He is a Ph.D. student at Graduate School of Information Science and Technology, Osaka University, Japan. His current research interests include sensing and data analytics for sports. He is a member of IPSJ.

Akira Uchiyama received his M.E. and Ph.D. degrees in Information and Computer Science from Osaka University in 2005 and 2008, respectively. He is an Assistant Professor in Mobile Computing Laboratory in Department of Information Networking, Graduate School of Information Science and Technology, Osaka University. He was a visiting scholar in University of Illinois at Urbana-Champaign in 2008 and a research fellow of the Japan Society for the Promotion of Science from 2007 to 2009. His current research interests include mobile sensing and applications in mobile networks and pervasive computing. He is a member of IEEE, ACM, IEICE and IPSJ.



Takaaki Umedu received his M.E. degree in Informatics and Mathematical Sciences and his Ph.D. degree in Information Science from Osaka University, Japan in 2001 and 2004, respectively. He is an Associate Professor at Shiga University. His current research interests include design and implementation of distributed systems and mobile systems. He is a member of IPSJ.

Teruo Higashino received his B.S., M.S. and Ph.D. degrees in Information and Computer Sciences from Osaka University, Japan in 1979, 1981 and 1984, respectively. He joined the faculty of Osaka University in 1984. Since 2002, he has been a Professor in Graduate School of Information Science and Technology at Osaka University. His current research interests include design and analysis of distributed systems, communication protocol and mobile computing. He is a senior member of IEEE, a fellow of IPSJ, and a member of ACM and IEICE of Japan.