Temporal Search Algorithm for Multiple-Pedestrian Tracking

Hye-Yeon Yu[1], Young-Nam Kim[1] and Moon-Hyun Kim[1]
[1] College of Software, Sungkyunkwan University
440-746, 2066 Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do, Republic of Korea
[e-mail: yu0529, hwarangjin, mhkim@skku.edu]
* Corresponding author: Moon-Hyun Kim

Received August 31, 2015; revised December 7, 2016; accepted April 8, 2016; published May 31, 2016

Abstract

In this paper, we provide a trajectory-generation algorithm that can identify pedestrians in real time. Typically, the contours for the extraction of pedestrians from the foreground of images are not clear due to factors including brightness and shade; furthermore, pedestrians move in different directions and interact with each other. These issues mean that the identification of pedestrians and the generation of trajectories are somewhat difficult. We propose a new method for trajectory generation regarding multiple pedestrians. The first stage of the method distinguishes between those pedestrian-blob situations that need to be merged and those that require splitting, followed by the use of trained decision trees to separate the pedestrians. The second stage generates the trajectories of each pedestrian by using the point-correspondence method; however, we introduce a new point-correspondence algorithm for which the A* search method has been modified. By using fuzzy membership functions, a heuristic evaluation of the correspondence between the blobs was also conducted. The proposed method was implemented and tested with the PETS 2009 dataset to show an effective multiple-pedestrian-tracking capability in a pedestrian-interaction environment.

Keywords: tracking, search algorithm, machine learning, point correspondence, decision tree

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (NRF-2014R1A2A1A11053902).

http://dx.doi.org/10.3837/tiis.2016.05.019 **ISSN : 1976-7277**

1. Introduction

For a long period of time, tracking has been an important theme in terms of computer vision, and it remains a challenging task. Computer-vision tracking involves the detection of the moving regions of images, the separation of foreground pedestrians from the background, and movement prediction. The tracking of a pedestrian is straightforward, as the pedestrian can easily be found in each frame, and the elapsed time can be used to determine the trajectory of the detected pedestrian; however, the generation of the trajectories of multiple tracked pedestrians is more complicated. During the tracking of multiple pedestrians, it is possible to mistakenly identify one of the other pedestrians as the target pedestrian, or to miss the detection of a pedestrian due to inter-pedestrian interactions such as *occlusions*.

Multi-object tracking is generally divided into the following three categories [1]: (1) Point tracking represents an object as a point and associates each point of a frame with a point in another frame. (2) Kernel tracking uses template- and density-based appearance models to track an object based on its shape and appearance. (3) Silhouette tracking extracts the contours of objects and tracks objects by matching silhouette shapes.

The point-tracking method involves the expression of the tracking target into a point, whereby the attribute of the target is extracted and attached to each point. This method requires a motion-correspondence problem, and in terms of classification, comprises statistical methods and heuristic methods. Multi Hypothesis Tracking (MHT) and Joint Probabilistic Data Association Filters (JPDAF) are examples of the statistical methods, and while they solve the interaction problem by finding jointly optimized trajectories, these methods can suffer from a combinatorial hypothesis space. Although global optimization can track a complete sequence, it is limited to a variety of assumptions in terms of an experiment. Globally, it is difficult to achieve success with optimum tracking because the combinatorial assignment problem is NP-complete. We propose a method that finds locally optimal trajectories using a heuristic function, which can be computed using consecutive 3 frames, i.e. $(t)_{\text{th}}$ frame, $(t-1)_{\text{th}}$ frame and $(t-2)_{\text{th}}$ frame. Even though this method does not guarantee to find globally optimal trajectories, it can be used for tracking pedestrians in real time by reducing search time. Further, while the local-tracking method that is used in the Kalman filter has a high accuracy regarding precision and localization, it needs to be modified to detect objects, despite the interactions between the objects, and so that the objects can be corresponded across frames for multi-object tracking [2].

The point-tracking method [1][3] requires an external mechanism to detect all of the pedestrians in every image frame. The tracking problem then requires an association of the detected blobs in the current frame with the tracked pedestrians of previous frames [4]. Our proposed earlier work presents a method for the generation of the trajectories of multiple pedestrians through the use of the motion-correspondence method [5], whereby each pedestrian is represented as a point. The experiment result of this earlier work shows an excellent trajectory-generation capability regarding detected positions in situations of added Gaussian noise; however, in a real-life tracking system, a pedestrian can sometimes appear as blob pieces due to detection-phase errors. Alternatively, several pedestrians can be detected as a merged blob due to their interactions with other pedestrians such as walking close together or crossing paths. It is therefore necessary to segment merged blobs that contain more than two pedestrians, and to also group separated pedestrian blobs. According to a recent method [8], the detected neighboring blobs can be represented as a graph, and the shortest-path algorithm

can be applied to the group blobs. The processed blobs are associated with the tracked pedestrians of previous frames; however, their work only allows for cases of merged blobs, whereby either a many-to-one or a one-to-many correspondence must occur between the blobs. A multi-objective path finding problem was the research that has been resolved by combines artificial immune system (AIS), chaos operator, and particle swarm optimization (PSO) [6]. The non-deterministic polynomial-time hard combinatorial optimization problem was solved by a research of the fitness-scaling adaptive genetic algorithm with local search [7].

In this paper, we propose a multi-pedestrian tracking system that assigns a blob to each pedestrian. For the first stage of our method, we detect the pedestrian blobs in each frame of the images. The detected blobs are then applied to decision trees that identify whether the blobs are to be merged or split, whereby the blobs are assigned to either a merged-blob decision tree or a split-blob decision tree. The merging and splitting processes regarding the identified blobs occurs next, and each blob is then assigned to a pedestrian. During the second stage, for each blob in the current frame, the A* algorithm is used to search for the corresponding blob in the previous frame. Fuzzy clustering is used to develop a heuristic function for the evaluation of the possibility of blob association, whereby membership is computed from the velocities of the blobs that are under a smooth-motion constraint.

The rest of this paper is organized as follows: section 2 describes the proposed system; section 3 explains the results of the experiment regarding the algorithm, including a performance analysis; and lastly, section 4 comprises the conclusion of the paper, as well as a discussion of future works.

2. Proposed Method

The block diagram of the proposed method is shown in **Fig. 1**. We separated the foreground from the background by using a Gaussian mixture model whereby the separated foreground consists of blobs. Each pedestrian blob is represented by features such as area, perimeter, aspect ratio, center of mass, average color, and the distance to the closest blob in the previous frame; based on training-set learning, these features are used for the construction of the decision trees. The resultant learned decision trees are used to classify the blobs as either “split” or “merged,” and the splitting and merging processes assign blobs to each pedestrian.

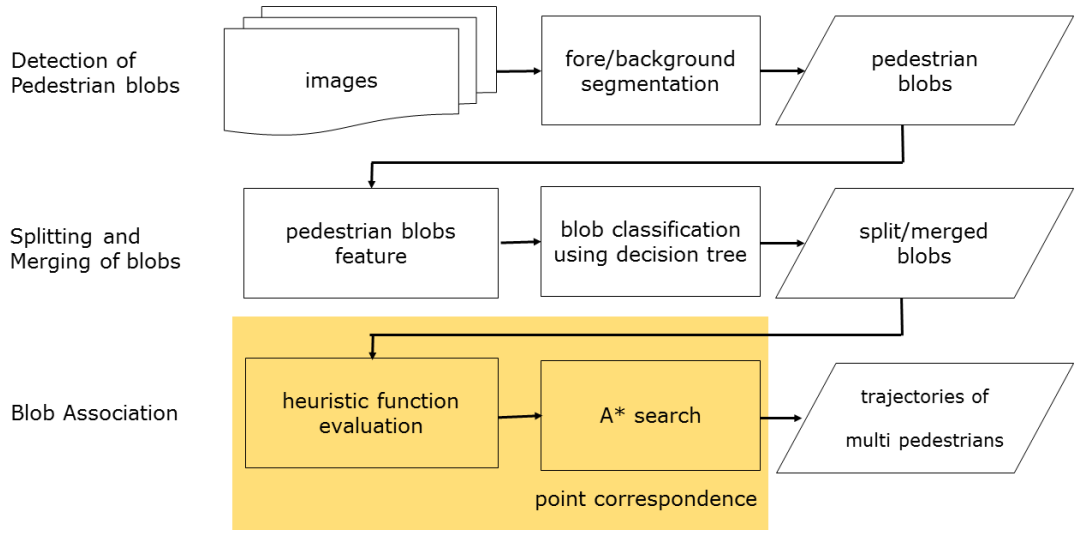


Fig. 1. Block diagram of the proposed method

To associate a blob in the current frame with a tracked pedestrian from a previous frame, we devised a modified A* heuristic search algorithm. For each blob, the search for the most-probable pedestrian from the previous frame is based on the Euclidean distance and the difference between the velocity angles of the blob and the pedestrian. The required heuristic function that computes the possibility that a tracked pedestrian corresponds with a blob was designed using a *fuzzy-C-means* (FCM) clustering algorithm. It is the association of the blobs with the pedestrians over the entire frame sequence that produces the trajectories, each of which is a trajectory for a single pedestrian.

2.1 Merging and splitting of blobs using Decision Trees

In images, the number of detected blobs is different from the number of pedestrians. During this stage, the multiple detected blobs of a pedestrian should be merged into one blob, and a single blob that consists of two pedestrians should be divided into two blobs; the decision trees are used for the performances of these merging and splitting processes. A decision-tree-based classification method was selected, since the process is relatively easy to understand and the effective features of the tree-structure-derived logical expression are easy to modify.

The first decision tree Merge tree T_M decides whether or not it will merge two adjacent blobs, whereas the second tree Split tree T_S decides whether it will split a blob or not; these decision trees are generated from an ID3 learning algorithm. A set of blob pairs $B_M = \{< b_{11}, b_{12} >, \dots, < b_{n1}, b_{n2} >\}$ is collected as a training set of T_M, and each pair is labeled as either a positive sample or a negative sample. For each pair $< b_{i1}, b_{i2} >$, the geometric features $f_M = \{f_{M1}, f_{Mn}\}$ are extracted and used as blob-pair descriptors, and it is these features that are used for the construction of T_M. A set of blobs $B_s = \{b_1, \dots, b_m\}$ is collected as a training set of T_S. Each blob is labeled as either a positive sample or a negative sample. For each blob, the geometric features $f_s = \{f_{s1}, f_{sm}\}$ are extracted to describe the blob and they serve as the features for T_S.

The ID3 decision tree is an information-theoretical algorithm invented by Ross Quinlan [10]. Each iteration of the ID3 involves the attainment of the largest information gain (or smallest entropy value), and a probability of p is determined from the occurrence frequency. The entropy of a given B_M is computed as follows:

$$E(B_M) = -\sum_{C_i \in C} p(C_i) \log_2 p(C_i) \tag{1}$$

where C is a set of classes in B_M that is either of a positive (merging) class or a negative (not-merging) class, and $\mathrm{p}(C_i)$ is the proportion of the number of pairs labeled as class C_i in B_M to $|B_M|$. The *information gain(IG)* from the partitioning of B_M according to the value of the feature f_M is computed using eq. (2), as follows:

$$IG(f_{M_i}, B_M) = E(B_M) - \sum_{v_j \in V_{M_i}} p(v_j) E(B_{v_j}^M) \tag{2}$$

In eq (2), $E(B_{v_j}^M)$ is the entropy of the subset $B_{v_j}^M$ that is constructed through the collection of pairs, each of which has a value of v_j for the feature f_{M_i}, from B_M. V_{M_i} denotes a set of values

for f_{M_i}; therefore, $B_M = \cup_{v_j \in M_i} B_{v_j}^M$. The probability $p(v_j)$ is computed using eq. (3), as follows:

$$p(v_j) = \frac{|B_{v_j}^M|}{|B^M|} \tag{3}$$

At each iteration of the construction of a decision tree, the feature that provides the maximum information gain is selected as a decision node. To deal with the numeric features, the C4.5 [11] ID3 algorithm is used.

The pair of blobs selected by T_M are merged, and the blob selected by T_S is split into two decision trees, whereby the merging and splitting processes are respectively used. A blob is split vertically in consideration of the shape of the person. After the blob-processing stage, each blob B_i is represented as $< x_i, y_i >$, as these are the coordinates of the center of mass.

2.2 Blob association

2.2.1 Heuristic-function evaluation

After the blobs have been processed, an A* search algorithm is used to associate the blobs in $(t-1)_{\text{th}}$ frame $B^{t-1} = \{B_1^{t-1}, \ldots, B_n^{t-1}\}$ with the blobs in $(t)_{\text{th}}$ frame $B^t = \{B_1^t, \ldots, B_n^t\}$. To develop a heuristic-evaluation function for the association of the pair $< B_i^{t-1}, B_j^t >$ for the A* algorithm, we must assume that each blob in $(t)_{\text{th}}$ frame is a cluster center. Using FCM clustering, we measure the confidence degree of the blob B_i^{t-1} in the $(t-1)_{\text{th}}$ frame that belongs to a cluster wherein the center of B_j^t is in $(t)_{\text{th}}$ frame. FCM clustering is a method whereby classifications are made according to the membership degree that is commensurate with the distance of the clusters and the input data [12]. The same process is repeated for the calculation of the cluster center until the change is within the acceptable range of the threshold value. The first step of FCM clustering is an establishment of the number of clusters and an initialization according to the exponential weight. The second step is the calculation of the center point of the fuzzy clustering. The third step is a calculation of the new membership function for the center of the cluster. Lastly, if the change in the cluster is less than the threshold value, the repeated process ends; however, if the change in the cluster is greater than or equal to the threshold value, the process is repeated from the second step onward.

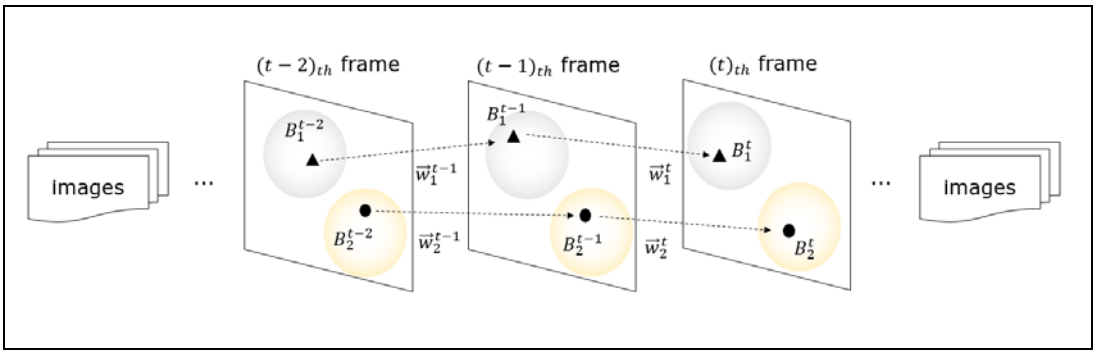


Fig. 2. $(t-1)_{\text{th}}$ frame and $(t)_{\text{th}}$ frame for blob matching of each blob

We defined the joint feature $\vec{w}_j = [d_j, \theta_j]$, where $d_j = (d_j \cdot x, d_j \cdot y) \in R^2$ represents the distance vector of a point and θ_j represents the orientation of this vector. A set of joint features can be represented as the following $\vec{W}^t$:

$$\begin{aligned} &\vec{W}^t = \{\vec{w}_1^t, \vec{w}_2^t, ..., \vec{w}_{m_t}^t\}, \vec{w}_j^t = [d_j^t, \theta_j^t], j = 1, ..., m_t, \\ &d_j^t = \left|B_j^t - B_i^{t-1}\right|, \theta_j^t = \arctan\left(\frac{d_j^t.y}{d_j^t.x}\right), i = 1, ..., m_{t-1}, \end{aligned} \tag{4}$$

where B_j^t is the *j th* blob in the $(t)_{\text{th}}$ frame and B_i^{t-1} is an *i th* blob in the $(t-1)_{\text{th}}$ frame.

A set of the joint features of a blob center can be represented as the following $\vec{W}^{t-1}$:

$$\begin{aligned} &\vec{W}^{t-1} = \{\vec{w}_1^{t-1}, \vec{w}_2^{t-1}, ..., \vec{w}_{m_{t-1}}^{t-1}\}, \\ &\vec{w}_i^{t-1} = [d_i^{t-1}, \theta_i^{t-1}], i = 1, ..., m_{t-1}, \\ &d_i = \left|B_i^{t-1} - B_{a_i^{t-2}}^{t-2}\right|, \theta_i = \arctan\left(\frac{d_i.y}{d_i.x}\right), , \end{aligned} \tag{5}$$

where a_i^{t-2} is an index of a blob in the $(t-2)_{\text{th}}$ frame, which is connected to the *i th* blob in the $(t-1)_{\text{th}}$ frame, B_i^{t-1}. Thus, $\vec{w}_i{}^{t-1}$ denotes velocity vector of $(i)_{th}$ blob of $(t-1)_{\text{th}}$ frame in polar coordinates.

As shown in **Fig. 2**, g_{ij}^{t-1} is the confidence degree for blob B_i^{t-1} to associate with blob B_j^t. This confidence degree is estimated using motion constraints that are based on the velocity vector and orientation angle of each blob. The confidence-degree functions for blobs between the $(t-1)_{\text{th}}$ frame and the $(t)_{\text{th}}$ frame satisfies the following constraints:

$$\begin{aligned} &\sum_j g_{ij}^{t-1} = 1, \forall i, g_{ij}^{t-1} = [0,1], \\ &1 \le i \le m_{t-1}, 1 \le j \le m_t. \end{aligned} \tag{6}$$

If we assume that each blob in the $(t)_{\text{th}}$ frame is a cluster center, then we can estimate the confidence degree by minimizing the following function Z:

$$\text{minimize } Z(g, \vec{w}) = \sum_{i,j} (g_{ij}^{t-1})^q \left\|\vec{w}_i^{t-1} - \vec{w}_j^t\right\|^2, \tag{7}$$

where q is a parameter that controls the fuzziness. It is a smoothness constraint of the velocity. It minimizes total differences between velocity of each blob at $(t-1)_{th}$ frame and velocity of the corresponding blob at $(t)_{th}$ frame.

The confidence degrees can be derived from the following Lagrange function:

$$L = \sum_{i,j} (g_{ij}^{t-1})^q \left\| \vec{w}_i^{t-1} - \vec{w}_j^t \right\|^2 + \sum_i \lambda_i \left(\sum_j g_{ij}^{t-1} - 1 \right), \tag{8}$$

where λ_i are the Lagrange multipliers. By differentiating L with respect to g_{ij}^{t-1} and applying normalization constraints.

$$\frac{\partial L}{\partial g_{ij}^{t-1}} = 0, \quad i = 1, \ldots, m_{t-1}, j = 1, \ldots, m_t \tag{9}$$

Then the confidence degree can be obtained as the following function:

$$g_{ih}^{t-1} = \frac{1}{\left(\dfrac{\left\| \vec{w}_i^{t-1} - \vec{w}_h^t \right\|^2}{\sum_{j=1}^{m} \left\| \vec{w}_i^{t-1} - \vec{w}_j^t \right\|^2} \right)^{1/(q-1)}}, 1 \le i \le m_{t-1}, 1 \le h \le m_t. \tag{10}$$

For each blob in the $(t)_{\text{th}}$ frame, the computed confidence degree is used as the value of the heuristic function in the A* search algorithm. This heuristic function is represented as the following $(m+1) \times (m+1)$ cost matrix *CM*. CM_{ij} is an element of the *i* th row and *j* th column of the *CM*, and it represents the heuristic values for the B_i^t and B_j^{t-1} blob correspondence that is g_{ij}^{t-1}. If the element g_{ij}^{t-1} is less than the threshold ε, it is set to zero. The last row of the *CM* defines e^{t-1} so that the sum of the confidence degrees for each blob in the $(t-1)_{\text{th}}$ frame is 1.

$$CM = \begin{array}{c} \\ B_1^t \\ B_2^t \\ B_3^t \\ B_4^t \\ \cdots \\ B_{m_t}^t \\ e^{t-1} \end{array} \begin{array}{c} \begin{array}{cccccc} B_1^{t-1} & B_2^{t-1} & B_3^{t-1} & B_4^{t-1} & \cdots & B_{m_{t-1}}^{t-1} \end{array} \\ \begin{pmatrix} g_{11}^{t-1} & 0 & 0 & 0 & \cdots & 0 \\ g_{12}^{t-1} & g_{22}^{t-1} & g_{32}^{t-1} & 0 & \cdots & 0 \\ g_{13}^{t-1} & g_{23}^{t-1} & g_{33}^{t-1} & g_{43}^{t-1} & \cdots & 0 \\ g_{14}^{t-1} & 0 & g_{34}^{t-1} & g_{44}^{t-1} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & g_{m_{t-1}m_t}^{t-1} \\ e_1^{t-1} & e_2^{t-1} & e_3^{t-1} & e_4^{t-1} & \cdots & e_{m_{t-1}}^{t-1} \end{pmatrix} \end{array} \tag{11}$$

$$e_i^{t-1} = \sum_{j=1}^{m_t} I(g_{ij}^{t-1} < \varepsilon) * g_{ij}^{t-1}, \forall i \tag{12}$$

2.2.2 A* search

The search tree is constructed as the nodes of depth i that are the candidate nodes for the i th blob in the $(t-1)_{\text{th}}$ frame. Firstly, the root node S becomes the start node that is a null node, and S is placed into a priority queue. S is selected from the priority queue and the successor nodes $B_1^t, B_2^t,$ and B_3^t are subsequently generated; these are the blobs in the $(t)_{\text{th}}$ frame with a non-zero confidence degree of association with B_1^{t-1} in eq. (11), and they are inserted in the priority queue and sorted according to the confidence degree. From the priority queue, a node with the maximum confidence degree B_1^t is selected and expanded. In level 2 of the search tree, nodes with a non-zero confidence degree regarding an association with B_2^{t-1} will appear. After B_1^t is selected, the successors of B_2^t and B_3^t from eq. (11) are generated and inserted into the priority queue; from this queue, a node with a maximum confidence degree B_3^t is selected. The successors B_2^t and B_4^t are generated, since they have non-zero confidence degrees for an association with B_3^{t-1}. Note that a node that appears in the path from the root node to the current selected node is not generated as successor. This process is iterated until the goal node is selected from the queue. The goal node is the node in depth m of the search tree if, through this iterative process, it finds the best matching pairs between the blobs in the $(t-1)_{\text{th}}$ frame and the blobs in the $(t)_{\text{th}}$ frame. After the search is finished, the algorithm traces back from the goal node to the starting node by following the path to obtain association pairs. From the optimal path found in the search graph of **Fig. 3**, the following association pairs are derived:

$$E^t = \{< B_1^t, B_2^t >, < B_2^t, B_3^t >, < B_3^t, B_4^t >, ..., < B_{m_t-1}^t, B_{m_t}^t >\} \tag{13}$$

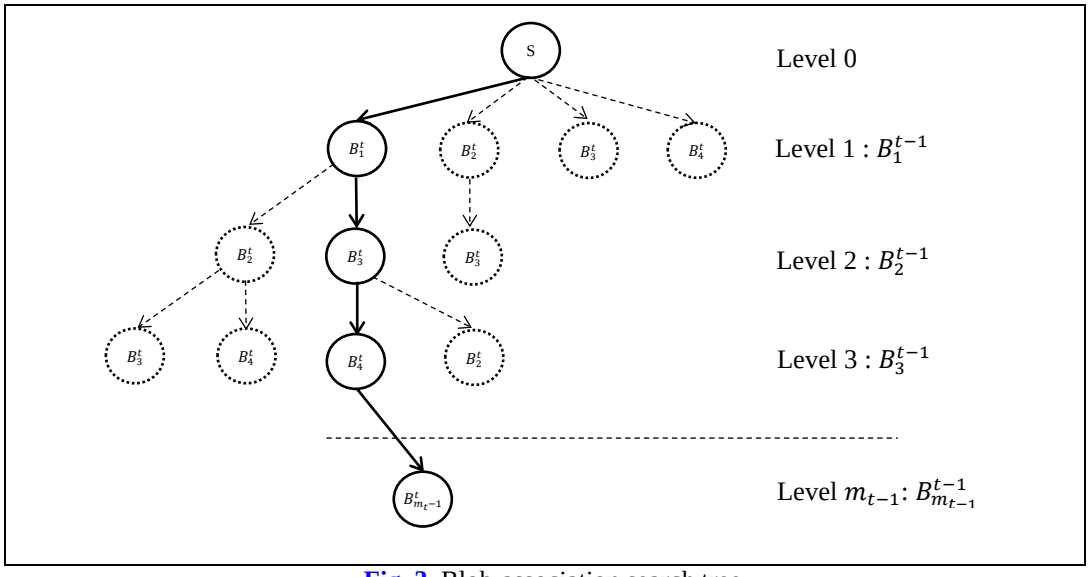


Fig. 3. Blob-association search tree

In the search graph of **Fig. 3**, the dotted line represents the associations of the blobs along the found path at each level of the search tree.

3. Experiment Results

We used the Caviar "Crowd of four people meet, walk and split" sequence , PETS 2009 dataset S2 from the L1 sequence view 001 and the L2 sequence view 001 as shown in Fig. 5. The Caviar sequence is used only for tracking in clean environment. WEKA (Waikato Environment for Knowledge Analysis) version 3.7, written in JAVA, is an experimental tool used for machine learning [27] that we used for ID3 machine learning and the visualization of the decision trees. Blob detection and a trajectory-generation algorithm were implemented in OpenCV 2.4.2 with Visual Studio 2010.

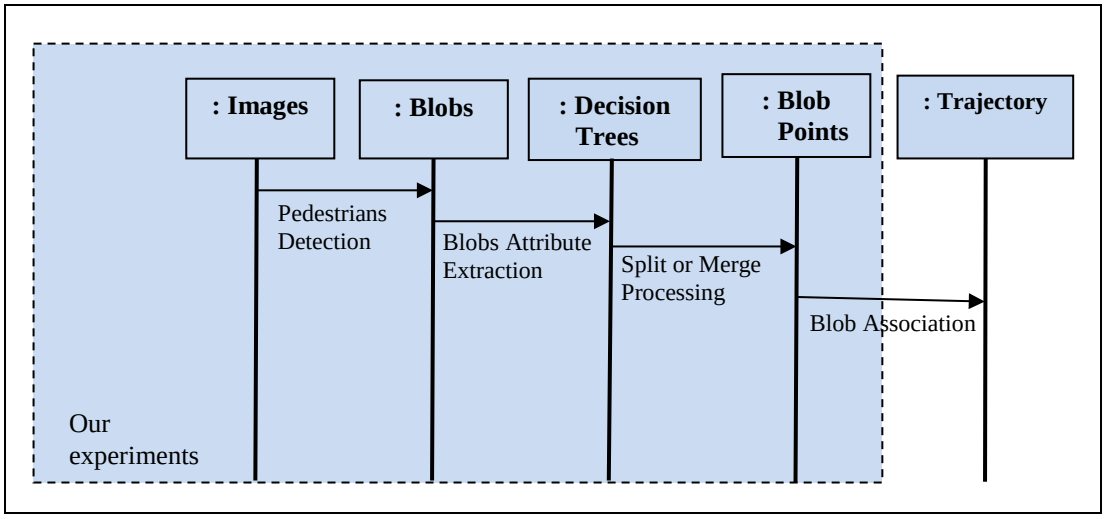


Fig. 4. Sequence diagram of the proposed method

The sequence diagram of the proposed method of this paper is shown in **Fig. 4**, whereby the data flow of the system is shown from left to right. In **Fig. 4**, the image sequence is provided as an input to the system from the left side. The detected pedestrian blobs are the data that are generated next, followed by the decision trees that are the next output data for the ID3 learning algorithm. After the merging and splitting of the blobs, each blob is represented as a center of the mass point. Lastly, the trajectories of the pedestrians are generated as the output data of the blob-association process.

(a) CAVIAR dataset

(b) PETS 2009. S2. L1.

(b) PETS 2009. S2. L2.

Fig. 5. data set used in the experiment

3.1 Dataset and Condition Description

The video footage used in the experiments features multiple walking pedestrians whose motions are generally meeting and breakup situations. In our study, one pedestrian is detected as one blob so that a trajectory can be generated using the point-correspondence method.

(a) In the merging situation (b) In the splitting situation

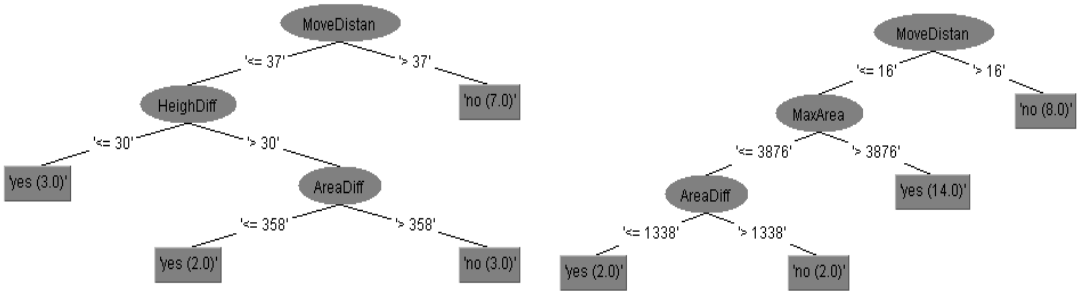
Fig. 6. Merge or split situation

Fig. 6 shows splitting or merging situations for the detected blobs in the image. Two pedestrians are detected as one blob, so the blob number 0 of **Fig. 6** (a) needs to be split; furthermore, one pedestrian is detected as two blobs, so the blob number 0 and the blob number 1 of **Fig. 6** (b) need to be merged.

3.2 Trajectory Generation

3.2.1 ID3 Decision Tree

The decision trees T_S and T_M are generated in terms of the blob attributes that are extracted from the training-input image sequence. The ID3 decision tree that is obtained from the calculation of the entropy of the divided sets is shown in **Fig. 7**. **Fig. 7** a) shows the split decision tree where "MoveDistance" is an attribute denoting the distance from the closest blob in the $(t-1)_{\text{th}}$ frame. The "HeightDiff" attribute represents the difference between the blob height and the height of the closest blob in the $(t)_{\text{th}}$ frame. "AreaDiff" represents the difference between the areas of two blobs. In **Fig. 7** b), "MaxArea" denotes the larger area of two adjacent blobs, and "AreaDiff" denotes the area difference between two adjacent blobs.



a) the split decision tree, T_S, for merging situations

b) the merge decision tree, T_M, for splitting situations

Fig. 7. Blob-merge and blob-split decision trees

In T_S, the threshold of the moving distance is 37 and the threshold of the area difference is 358. If it is decided that blobs are to be split from T_S, the blob is divided into two blobs using the splitting procedure. If it is decided that two blobs are to be merged from T_M, the two blobs are merged into a single blob. After the splitting and merging processes, the center of the mass of each blob that is calculated is used for the confidence-degree computation.

3.2.2 Multiple-Pedestrian Trajectory in Clean Environment

In the first scenario, we tested the proposed method in a clean environment, whereby we assumed that interferences between pedestrians such as occlusion do not occur; however, in cases of occlusion, we manually assigned a blob for each pedestrian. **Table 1** shows the cost matrix that was obtained from the calculation of the confidence degree of the association of two blobs that were selected from the $(t)_{\text{th}}$ frame and the $(t-1)_{\text{th}}$ frame for eight sequential frames of the L1 sequence. For the computation of the confidence degree, ε = 0.05, and if the confidence degree is less than 0.05, it is set to 0. The number in $(i)_{th}$ row and $(j)_{th}$ column of each table represents confidence degree g_{ij}^{t-1} of each frame. For example, the second table in top row is a cost matrix for corresponding blobs in the 2_{nd} frame to blobs in the 1_{nd} frame. 0.22 in 2_{nd} row and 3^{rd} column of this table denotes confidence degree for corresponding 2_{nd} blob of 2_{nd} frame to the 3^{rd} blob of 1_{st} frame, i.e. g_{32}^{1}.

Table 1. Cost Matrix (CM)

	t-1 / t	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9
1th	B_1	0.86	0	0	0	0	0	0	0	0
	B_2	0	0.74	0.15	0	0	0	0	0	0
	B_3	0	0.13	0.67	0	0	0	0	0	0
	B_4	0	0	0	0.71	0.17	0	0	0	0
	B_5	0	0	0	0.12	0.62	0.05	0	0	0
	B_6	0	0	0	0	0	0.67	0.10	0	0
	B_7	0	0	0	0	0	0.12	0.73	0.05	0
	B_8	0	0	0	0	0	0	0	0.75	0
	B_9	0	0	0	0	0	0	0	0	0.94

		B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9
2th		0.58	0	0	0	0	0	0.08	0.05	0
		0.05	0.66	0.22	0	0	0	0.12	0.06	0
		0.06	0.13	0.48	0	0	0	0.12	0.06	0
		0.05	0	0	0.64	0.10	0	0.12	0.06	0
		0.05	0	0	0.09	0.62	0	0.12	0.06	0
		0.05	0	0	0	0	0.71	0.25	0	0
		0.05	0	0	0	0	0.08	1	0.06	0
		0.06	0	0	0	0	0	0.11	0.56	0
		0.05	0	0	0	0	0	0.08	0	0.70

3th	B_1	0.82	0	0	0	0.06	0	0	0.08	0
	B_2	0	0.45	0.25	0.07	0.08	0.06	0.06	0.09	0
	B_3	0	0.21	0.36	0.06	0.08	0.05	0.06	0.09	0
	B_4	0	0.06	0.06	0.49	0.13	0.06	0.06	0.08	0
	B_5	0	0.06	0.07	0.11	0.35	0.06	0.07	0.09	0
	B_6	0	0	0.05	0.06	0.07	0.52	0.11	0.06	0
	B_7	0	0.06	0.07	0.07	0.10	0.11	0.52	0.09	0
	B_8	0	0.05	0.06	0.06	0.08	0	0.05	0.37	0
	B_9	0	0	0	0	0.05	0	0	0.06	0.65

4th		0.81	0	0	0	0	0.09	0	0	0
		0	0.79	0.15	0	0	0.12	0	0	0
		0	0.06	0.53	0	0	0.11	0		0
		0	0	0	0.64	0.06	0.13	0	0	0
		0	0	0.05	0.06	0.62	0.12	0.07	0	0
		0	0	0	0	0	1	0.09	0	0.05
		0	0	0.05	0	0.07	0.23	0.57	0	0
		0	0	0.05	0	0	0.11	0	0.75	0
		0	0	0	0	0	0.10	0	0	0.63

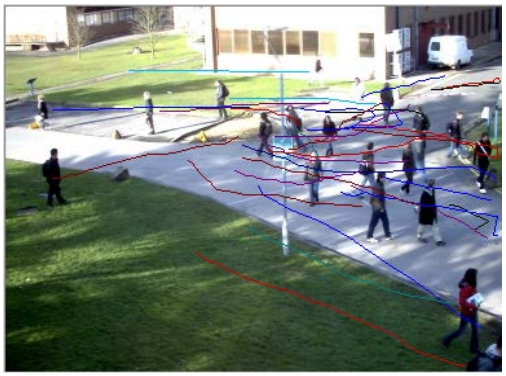
5th	B_1	0.57	0	0	0	0	0	0	0	0.12
	B_2	0.05	0.66	0.12	0	0	0	0	0	0.12
	B_3	0.06	0.07	0.54	0	0	0	0	0	0.12
	B_4	0	0	0.05	0.66	0	0	0	0	0.12
	B_5	0.06	0	0.06	0	0.64	0	0.09	0	0.12

6th		0.80	0	0	0	0	0	0	0	0
		0	0.59	0.06	0.06	0	0	0	0	0
		0	0.06	0.69	0	0	0	0	0	0
		0	0.07	0	0.66	0	0	0	0	0
		0	0.05	0	0	0.61	0	0.14	0	0

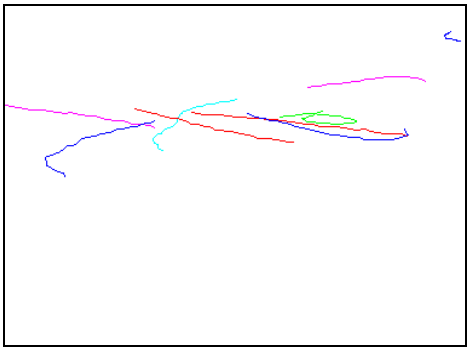
	B_6	0	0	0	0	0	0.72	0.07	0	0.13		0	0	0	0	0	0.76	0.06	0	0.05
	B_7	0.06	0	0.06	0	0.08	0.05	0.60	0	0.12		0	0.05	0	0	0.12	0	0.57	0	0
	B_8	0.06	0	0.05	0	0	0	0	0.83	0.12		0	0.05	0	0	0	0	0	0.76	0
	B_9	0	0	0	0	0	0	0	0	1		0	0	0	0	0	0	0	0	0.63
7th	B_1	0.71	0	0	0	0	0	0	0	0	8th	0.66	0	0.06	0	0	0	0	0.05	0
	B_2	0	0.74	0	0.09	0	0	0.05	0.05	0		0	0.68	0.07	0.11	0	0	0	0.06	0
	B_3	0	0	0.67	0	0	0	0.05	0.05	0		0.05	0	0.51	0	0	0	0	0.06	0
	B_4	0	0.06	0	0.63	0	0	0.05	0.05	0		0	0.09	0.07	0.61	0	0.05	0	0.05	0
	B_5	0	0	0	0	0.71	0	0.30	0.05	0		0	0	0.07	0	0.51	0.05	0.24	0.06	0
	B_6	0	0	0	0	0	0.74	0.05	0.06	0.05		0	0	0.06	0	0	0.58	0	0.06	0.5
	B_7	0	0	0	0	0.1	0	0.35	0.06	0		0	0	0.07	0	0.19	0.07	0.46	0.09	0
	B_8	0	0	0	0	0	0	0.06	0.58	0		0	0	0.07	0	0	0.07	0.06	0.53	0
	B_9	0	0	0	0	0	0	0	0	0.63		0	0	0	0	0	0	0	0	0.62

Fig. 8 shows the generated trajectories for the S2 views of the L1 and L2 sequences from the PETS 2009 dataset. The pedestrian group consists of nine people in the (a) L1 sequence and 33 people in the (b) L2 sequence. The pedestrians of (a) frequently change their moving direction in comparison with those of (b). The L1 and L2 sequences generated trajectories of 30 frames and 26 frames, respectively. **Fig. 8** (c) shows the trajectories that were traced using the proposed method for the L1 sequence, and **Fig. 8** (d) shows the generated trajectories for the L2 sequence. For both cases, the generated trajectories are exactly matched with the ground truth.

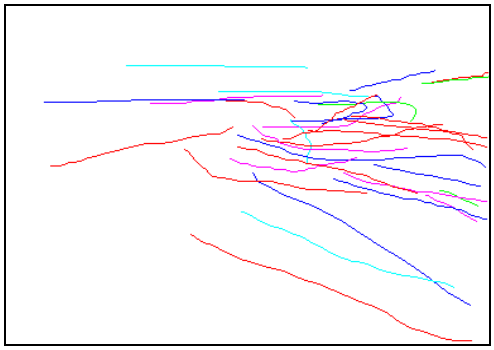
(a) L1 ground-truth trajectories



(b) L2 ground-truth trajectories



(c) L1 trajectories of the proposed method



(d) L2 trajectories of the proposed method

Fig. 8. Multiple trajectories of pedestrians

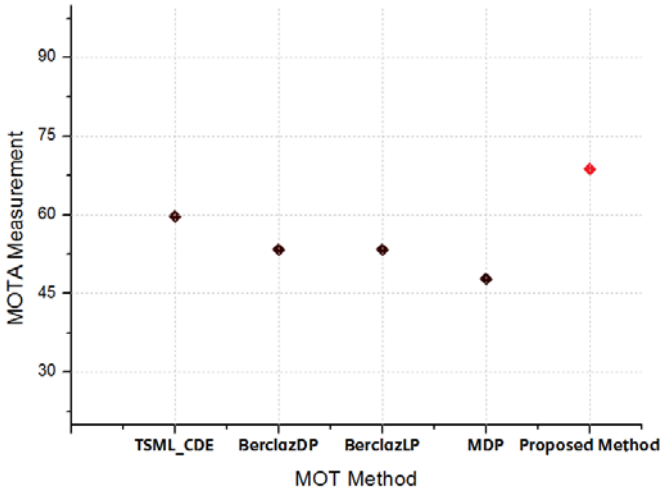
Table 2. Tracking performance in clean environment

	Noise	CCPR	CDTR	AIDC
CAVIAR data set	0 %	100 %	100 %	0 %
	50 %	98 %	100 %	0.1 %
PETS data set	0 %	100 %	100 %	0.0 %
	9 %	99 %	100 %	0.0 %
	18 %	99 %	100 %	0.0 %
	27 %	98 %	100 %	0.1 %

Table 2 shows the tracking performance for the trajectory generation of **Fig. 8** (d) and CAVIAR data set. The generated trajectories of the L2 sequence show the ratio of the 100 % correctly detected tracks (CDTR), the ratio of the 100 % correctly corresponding pairs (CCPR), and the 0 % average ID change (AIDC) in the first row, where there is no added noise. We added noise for the blob locations, and for each added noise, we measured the CCPR, CDPR, and AIDC [5].

3.2.3 Tracking in Real Environment

In the second scenario, we tested the proposed method in a real environment where interferences between pedestrians were considered. In this scenario, the generated decision trees are used for splitting a blob and merging blobs. For the quantitative performance evaluation for the experiment in this scenario, we use the MOTA (Multiple Object Tracking Accuracy) that is provided by the Classification of Events, Activities, and Relationships (CLEAR) consortium. The MOTA is a performance index for the measurement of tracking accuracy according to the sum of the following three errors: ratio of missed tracking target, false detection rate, and ratio of mismatched blobs. A comparison between the tracking results obtained from the experiment and the GT trajectories yielded the following findings: The miss rate is 3.42 % and the false positive rate is 27.82 %. As shown in **Fig. 9**, our proposed method shows a MOTA of 68.75 %. We also compared the performance indexes of other participants to the PETS 2009. Regarding the 2D MOT benchmark features, the tracking results were compared for the PETS 2009 S2 L2 images. The index used in **Fig. 9** is taken from the state-of-the-art methods of the MOT Challenge Benchmark [24]. Our method produced favorable results in comparison with those of the MOT Challenge Benchmark; for the comparison, our method-exclusion process is based on the multi-view images.



Matrix	Value
Precision	72.17 %
Miss rate	3.42 %
False pos. rate	27.82 %
MOTA	68.75 %

Fig. 9. Tracking-performance comparison with other methods

4. Conclusion

During the formulation of a method for the generation of trajectories for multiple pedestrians, numerous difficulties arise. Our point-correspondence algorithm for multiple-object tracking that was previously proposed has been modified and expanded in this work. To track multiple pedestrians in a real environment, we consider the interactions between pedestrians as the most important issue. We therefore developed a decision method for the splitting and merging of blobs, whereby decision trees learned from the ID3 algorithm are used to resolve the occlusion problem of pedestrians. For the identified situations, the blob-splitting or blob-merging processes are applied for the redefinition of a blob for each pedestrian and to calculate the association costs. We used the A* algorithm to successfully generate the pedestrian trajectories, whereby a smooth constraint is used to devise the heuristic function for the association of a blob in the $(t)_{\text{th}}$ frame with a blob in the $(t-1)_{\text{th}}$ frame. In our experiment, we used the MOTA index for a performance evaluation regarding multiple-pedestrian tracking. In a comparison between the proposed method and the state-of-the-art methods of the 2009 MOT Challenge Benchmark, a favorable performance was observed. We did, however, restrict the compared methods to the tracking systems from single-view image sequences, but we are going to improve the system so that it can handle more-frequent pedestrian interactions.

References

[1] A. Yilmaz, O. Javed and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, pp. 13, 2006. Article (CrossRef Link)

[2] X. Li, K. Wang, W. Wang and Y. Li, "A multiple object tracking method using Kalman filter," in *Proc. of 2010 IEEE International Conference on Information and Automation (ICIA)*, pp. 1862-1866, 2010. Article (CrossRef Link)

[3] K. Shafique and M. Shah, "A Noniterative Greedy Algorithm for Multiframe Point Correspondence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 51-65, Jan. 2005. Article (CrossRef Link)

[4] P. Foggia, G. Percannella, A. Saggese and M. Vento, "Real-time tracking of single people and groups simultaneously by contextual graph-based reasoning dealing complex occlusions," in *Proc. of 2013 IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, pp. 29-36. 2013. Article (CrossRef Link)
[5] K. Eom, J. Jung and M. Kim, "A heuristic search-based motion correspondence algorithm using fuzzy clustering," *International Journal of Control, Automation and Systems (IJCAS)*, vol. 10, no. 3, pp. 594-602, 2012. Article (CrossRef Link)
[6] Y. Zhang, Y. Jun, G. Wei and L. Wu, "Find multi-objective paths in stochastic networks via chaotic immune PSO," *Expert Systems with Applications*, vol. 37, no. 3, pp. 1911-1919, 2010. Article (CrossRef Link)
[7] S. Wang, Z. Lu, L. Wei, G. Ji and J. Yang, "Fitness-scaling adaptive genetic algorithm with local search for solving the Multiple Depot Vehicle Routing Problem," *Simulation*, 2015. Article (CrossRef Link)
[8] J. Berclaz, F. Fleuret, E. Turetken and P. Fua, "Multiple Object Tracking Using K-Shortest Paths Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1806-1819, sept. 2011. Article (CrossRef Link)
[9] G. Kim, T. An and M. Kim, "Estimation of Crowd Density in Public Areas Based on Neural Network," *KSII Transactions on Internet and Information Systems(TIIS)*, vol. 6, pp. 2170-2190, 2012. Article (CrossRef Link)
[10] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol.1, pp.81-106, 1986. Article (CrossRef Link)
[11] J. R. Quinlan, "C4.5: Programs for Machine Learning," *Morgan Kaufmann Publishers*, San Mateo, CA, 1993.
[12] J. Bezdek, R. Ehrlich and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191-203, 1984. Article (CrossRef Link)
[13] R. Yager and D. Filev, "Approximate clustering via the mountain method," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 8, pp. 1279-1284, 1994. Article (CrossRef Link)
[14] A. Ellis and J. Ferryman, "PETS2010 and PETS2009 evaluation of results using individual ground truthed single views," in *Proc. of 2010 Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 135-142, sept, 2010. Article (CrossRef Link)
[15] S. Guo, C. Hsu, P. Wu and J. S. Tsai, "A Trajectory-Based Point Tracker Using Chaos Evolutionary Programming," *Next-Generation Applied Intelligence*, Springer, pp. 212-220, 2009. Article (CrossRef Link)
[16] T. Fasciano, H. Nguyen, A. Dornhaus and M. C. Shin, "Tracking multiple ants in a colony," in *Proc. of 2013 IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 534-540, 2013. Article (CrossRef Link)
[17] Y. Caspi, D. Simakov and M. Irani, "Feature-Based Sequence-to-Sequence Matching," *International Journal of Computer Vision*, vol. 68, no. 1, pp. 53-64, 2006. Article (CrossRef Link)
[18] K. Shafique and M. Shah, "A noniterative greedy algorithm for multiframe point correspondence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 51-65, 2005. Article (CrossRef Link)
[19] C. Toth, D. Grejner-Brzezinska and S. Moafipoor, "Pedestrian tracking and navigation using neural networks and fuzzy logic," in *Proc. of IEEE International Symposium on Intelligent Signal Processing*, WISP 2007, pp. 1-6, 2007. Article (CrossRef Link)
[20] A. Francois, "Real-Time Multi-Resolution Blob Tracking," *Institute for Robotics and Intelligent Systems(IRIS)* Technical Report IRIS-04-422, University of Southern California, Los Angeles, California, 2004.
[21] Z. Yang and B. Yuan, "Vision based multi-pedestrian tracking using adaptive detection and clustering," *Intelligent Data Engineering and Automated Learning-IDEAL 2013*, Springer Berlin Heidelberg, pp. 58-66, 2013. Article (CrossRef Link)
[22] I. O. Sebe, S. You and U. Neumann, "Globally optimum multiple object tracking," *SPIE Defense and Security Symposium*, vol. 5810, pp.82 -93, 2005. Article (CrossRef Link)

[23] L. Leal-Taixe, A. Milan, I. Reid, S. Roth and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *ArXiv Preprint arXiv:1504.01942*, 2015.
[24] L. Milan, A. Leal-Taix, K. Schindler, S. Roth, and I. Reid, MOT Challenge, 2015. http://www.motchallenge.net
[25] Ferryman, J., Shahrokni, A.: PETS 2009 (2009), http://www.cvg.rdg.ac.uk/PETS2009
[26] Homepages.inf.ed.ac.uk, CAVIAR Test Case Scenarios, 2015. http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/
[27] Cs.waikato.ac.nz, "Weka 3 - Data Mining with Open Source Machine Learning Software in Java," http://www.cs.waikato.ac.nz/ml/weka/

Hye-Yeon Yu received the M.S. degree in Information and Communications from Sungkyunkwan University, Seoul, Korea, in 2013. She is currently a Ph.D. candidate in Sungkyunkwan University, Suwon, Korea. Her research interests include artificial intelligence, machine learning, and computer vision.

Young-Nam Kim received the M.S. degree in College of Information and Communication Engineering from Sungkyunkwan University, Suwon, Korea, in 2015. He is currently a Ph.D. candidate in Sungkyunkwan University, Suwon, Korea. His research interests include artificial intelligence, machine learning, and computer vision.

Moon-Hyun Kim received the B.S. degree in Electronic Engineering from Seoul National University in 1978, the M.S. degree in Electrical Engineering from Korea Advanced Institute of Science and Technology, Korea, in 1980, and the Ph.D. degree in Computer Engineering from the University of Southern California in 1988. From 1980 to 1983, he was a Research Engineer at the Daewoo Heavy Industries Co., Seoul. He joined College of Software, Sungkyunkwan University, Seoul, Korea in 1988, where he is currently a Professor. In 1995, he was a Visiting Scientist at the IBM Almaden Research Center, San Jose, California. In 1997, he was a Visiting Professor at the Signal Processing Laboratory of Princeton University, Princeton, New Jersey. His research interests include artificial intelligence, machine learning, and image recognition.