Sensors & Transducers
© 2015 by IFSA Publishing, S. L.
http://www.sensorsportal.com

Automatic Human Tracking System Using Localized Neighbor Node Calculation

[1, 2] Tappei Yotsumoto, [1] Kozo Tanigawa, [2] Miki Tsuji, [2] Kenichi Takahashi, [2] Takao Kawamura and [2] Kazunori Sugahara
[1] Melco Power Systems Co., Ltd., 1-1-2, Wadasaki-cho, Hyogo-ku, Kobe, 652-0855, Japan
[2] Tottori University, 4-101, Koyama-cho Minami, Tottori, 680-8550, Japan
[1] Tel.: +81-78-682-6942, fax: +81-78-682-6891
[1] E-mail: Yotsumoto.Tappei@zb.MitsubishiElectric.co.jp

Received: 31 August 2015 /Accepted: 15 October 2015 /Published: 30 November 2015

Abstract: A human tracking system based on mobile agent technologies has been proposed to achieve automatic human tracking function. In this system, each target person is tracked automatically by its mobile agent moving among cameras in which a person is detected. The current system utilizes an algorithm to predict which camera detects the target person. The algorithm needs information from all cameras about their monitoring ranges. If one monitoring range is updated by a pan / tilt / zoom operation or some other reason, the whole calculation to determine the relationship between camera nodes must be performed accordingly. In order to solve this problem, we propose in this paper an algorithm that uses only localized node information from each camera and its neighboring camera. With this proposed algorithm, each camera is able to calculate its neighbor nodes without obtaining the monitoring ranges of all cameras. This enables the construction of robust human tracking systems. *Copyright © 2015 IFSA Publishing, S. L.*

Keywords: Human tracking, Mobile agent, Pan/Tilt/Zoom, Neighbor relation, Localization.

1. Introduction

In recent years, in order to enhance public security, various kinds of systems such as entrance and exit management and detection of suspicious persons have been introduced. The most widely used system is a supervising system using cameras. In this system, operators must fix their eyes on two or more cameras to find a suspicious person. Considering the ability of the operators, the maximum number of cameras should be two or three for one operator. A number of operators are required for monitoring many cameras and tracking multiple people. Moreover, when an operator loses sight of a suspicious person, the operator must go over multiple cameras to find him/her. For this reason, systems that enable automatic human tracking using multiple cameras are proposed [1]. These systems, however, have two problems:

1) The computational cost for tracking a person is concentrated on one monitoring server;

2) Dealing with the change of monitoring range of cameras is not considered.

We have proposed an automatic human tracking system based on mobile agent technology. This system consists of cameras, tracking servers, mobile agents, and a monitoring terminal. In this system, a tracking server installed in each camera analyzes images received from the camera. Therefore, the computational cost of image analysis is distributed over each tracking server. A mobile agent is prepared for each person being tracked. The mobile agent migrates among tracking servers by detecting the physical data of a person being tracked. By checking the location of an agent at the monitoring terminal, the operator is able to know the location of the tracked persons.

Tracking all detected persons is possible if a number of cameras that monitor in all directions without blind spots are installed. However, this is an unrealistic idea and is very costly. A more realistic approach is to install a certain numbers of cameras at some specific points such as entrances of a building or rooms and passage crossings. In this case, there is a moment at which a tracked person is not displayed on any camera. When the human tracking system loses a tracked person, the system has to check every camera's view to find the lost person. A high computation cost for each camera is required for image analysis. Therefore, the algorithm was proposed to predict which camera would catch the tracked person next [1-3].

The algorithm calculates neighbor nodes of each camera based on the value of each camera's monitoring range, map of the floor, and the locations where cameras are installed. A node is defined as a location of a tracking server with a camera. If a tracked person goes out of the monitoring range of one camera, the person should appear in the neighbor nodes of the camera. Since the algorithm has already calculated each camera's neighbor nodes, computation cost for image analysis is charged only on its neighbor nodes. However, when the monitoring ranges of some camera are changed by panning / tilting / zooming operations or other reasons, we have to re-calculate neighbor nodes.

In this paper, we extend the current human tracking algorithm to localize the neighbor node calculation. The proposed algorithm utilizes only the monitoring range of neighbor nodes. By using the localization of neighbor node calculation, re-calculation cost of the neighbor node will be limited in some nodes even when many cameras change their monitoring ranges. Furthermore, this realizes a robust human tracking system that enables continuous tracking even when some nodes are down.

Section 2 of this paper mentions about the related work. Section 3 introduces the proposed human tracking system and describes the neighbor node calculation algorithm. Section 4 describes the localized neighbor node calculation algorithm. Section 5 shows the experimental result, and Section 6 concludes this paper.

2. Related Work

Y. Shirai researched about tracking multiple persons and proposed a technique for collaboration between cameras for tackling obstruction [4]. N. Kawashima proposed a tracking technique that eliminates noise such as shadow by using a dispersion matrix and by improving the background subtraction method [5]. These researches aimed at accuracy enhancement of persons' detection by using multiple cameras, but do not aim at tracking a target person across multiple cameras.

H. Mori proposed a tracking technique in an environment where the monitoring ranges of multiple cameras are overlapped by unifying the monitoring images from several cameras [6]. A. Nakazawa proposed a mechanism for combining the physical data of multiple persons [7]. N. Ukita proposed a system to exchange monitoring images efficiently using an agent based framework [8]. These researches assume that the imaging ranges of cameras are overlapped.

N. Ukita and D. Makris proposed a method for estimating the migration path of a target based on entry-exit(in-out) information [9-10]. This method requires re-collection of the entry-exit (in-out) information when the monitoring ranges of cameras change. N. Takemura's research predicts possible routes of a person from his/her position and moving speed [11]. Y. Tanizawa proposed a mobile agent-based framework, called "FollowingSpace" [12]. In this system, when a user moves to another location in a physical space, a mobile agent attached to the user migrates to one of the nearest hosts from the current location of the user. T. Tanaka also proposed an agent-based approach to track a person [13]. However, a mechanism to predict in which camera a target would appear next was not discussed. K. Aoki proposed a cooperative surveillance system using active cameras [14]. In this system, each active camera adjusts its observation area to decrease blind spots. In these studies, cameras which a tracked person will appear next are predicted by movement of the person.

Our proposed algorithm enables us to calculate a camera which a target will appear next without considering uncertain movement of the person.

3. Human Tracking System Using Mobile Agent Technologies

An automatic human tracking system using mobile agent technologies has been developed [1]. In the system, there are multiple mobile agents, each of which tracks one person called a "*target*." Since all the targets are tracked automatically by each of the mobile agents, the location of each target can be known by monitoring the location of its corresponding mobile agent.

3.1. System Configuration

The structure of our automatic human tracking system is shown in Fig. 1. The system consists of cameras, tracking servers, mobile agents, and a monitoring terminal. Cameras are discretely installed in a monitoring area and have pan / tilt / zoom functions which change each camera’s monitoring range. A tracking server is connected to each camera and receives images from the camera. Tracking servers have the execution environment for a mobile agent and an image analysis function. Since the image analysis is performed in each tracking server, the computational cost of image analysis is distributed to each tracking server. The mobile agent migrates across tracking servers in accordance with the movement of a target. The locations of mobile agents are displayed in a monitoring terminal. The current positions of all targets can be seen through the location of mobile agents.

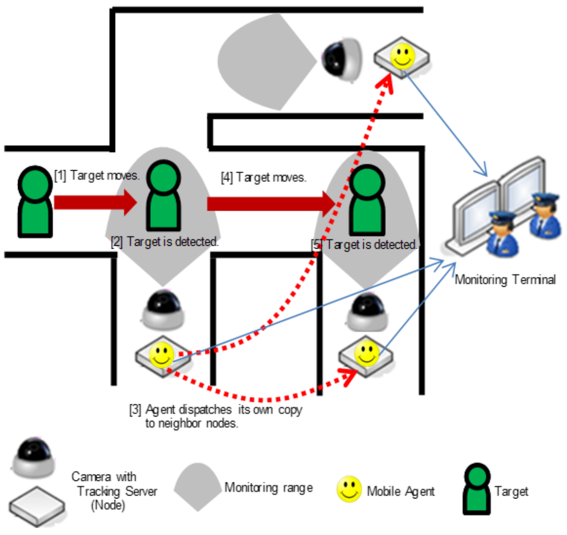

Fig. 1. Structure of the proposed system.

3.2. Tracking Flow

When a target comes into the monitoring range of each tracking server, the tracking server checks whether the target is being tracked by any agent in the server. If it is not, the tracking server generates a mobile agent containing the physical data of the target (e.g., facial features, color of attire). The mobile agent tracks the target based on the target’s physical data. At the same moment, the tracking server distributes copies of the tracking agent, called “*copy agents*”, to tracking servers of neighboring cameras where the target may pass. The calculation algorithm for neighbor nodes is described in the next subsection. Tracking servers of neighboring cameras analyze the camera image periodically based on the physical data which the copy agents have. If the target is detected, the copy agent in that camera becomes the new tracking agent and distributes new copies to tracking servers of neighboring cameras. The original tracking agent and copy agents are subsequently erased. Besides that, if an agent loses track of the target for a definite period of time, the agent removes itself.

3.3. Algorithm to Calculate Neighbor Nodes

Regarding camera locations and cost-efficiency, it is practical to install cameras only at specific places, such as building entrances, rooms, or passage crossings. In such an environment, each camera's monitoring range is not necessarily overlapped with other camera's monitoring range. Therefore, it becomes necessary to predict which camera a target will appear in next. In this case, it becomes necessary to predict which camera a target will appear in next.

In order to predict the next camera, we defined points that represent a route through which a target can pass as following.

- Branch points (passage crossings): B_i
- Camera points (camera locations): C_i
- Viewing points (between two branch points, between two camera points, and between a branch point and a camera point): V_i

The monitoring range of each camera is determined from these points, as shown in Fig. 2.

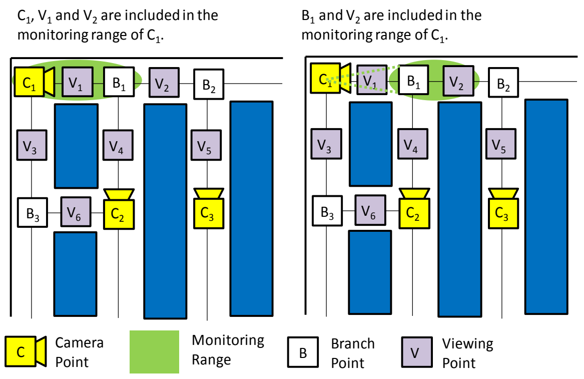
C1, V1 and V2 are included in the monitoring range of C1.
B1 and V2 are included in the monitoring range of C1.
C1
V1
B1
V2
B2
V3
V4
V5
B3
V6
C2
C3
C
Camera Point
Monitoring Range
B
Branch Point
V
Viewing Point

Fig. 2. Representation of a route.

The monitoring range of each camera changes by pan, tilt and / or zoom operations. For example, when the monitoring range of camera C_1 is as in the left part of Fig. 2, the monitoring range of C_1 becomes [C_1, V_1, B_1]. When the monitoring range of C_1 is as in the right map of Fig. 2, the monitoring range of C_1 becomes [B_1, V_2]. Matrix X of $|C| \times |P|$ is defined from the monitoring range of all cameras. Element X_{ij} of matrix X is defined as (1).

$$X_{ij} = \begin{cases} 0, & \begin{array}{c} \mathit{where\ the\ monitoring\ range} \\ \mathit{of\ the\ camera}\ C_i\ \mathit{does\ not} \\ \mathit{include\ the\ point}\ P_j. \end{array} \\ 1, & \begin{array}{c} \mathit{where\ the\ monitoring\ range} \\ \mathit{of\ the\ camera}\ C_i \\ \mathit{includes\ the\ point}\ P_j. \end{array} \end{cases} \tag{1}$$

where C is a set of camera points and P is a set of branch points, camera points and viewing points. Cameras that have overlapping monitoring ranges (neighboring cameras) are identified using (2). The monitoring range of Camera C_i and C_j are overlapped if $D_{ij} \geq 1$.

$$D = X \cdot X^T \tag{2}$$

Next, the adjacency matrix Y of $|P| \times |P|$ is defined. Element Y_{ij} of matrix Y is defined as (3).

$$Y_{ij} = \begin{cases} 0, & \begin{array}{c} \textit{where the point } P_i \textit{ and} \\ \textit{the point } P_j \textit{ are not} \\ \textit{neighboring each other.} \end{array} \\ 1, & \begin{array}{c} \textit{where the point } P_i \textit{ and} \\ \textit{the point } P_j \textit{ are} \\ \textit{neighboring each other.} \end{array} \end{cases} \tag{3}$$

When $E_{ij} \geq 1$ in (4), the neighboring camera is overlapped with $(n-1)$ points away from the monitoring range of the camera C$_i$.

$$E = X \cdot Y^n \cdot X^T \tag{4}$$

Even if the neighboring cameras can be identified by (4), the number of points between the monitoring ranges of two cameras is unknown. In other words, n is unknown. Therefore, points which are not included in the monitoring range of all camera from matrix X and Y are eliminated. Matrix X' is generated from matrix X by eliminating all the points in column j that satisfy (5).

$$\sum_{k=i}^{m} X_{kj} = 0 \tag{5}$$

Similary, matrix Y' is generated from matrix Y by eliminating all the points in column j and row j. Furthermore, X_{ik} is set to 1 if $X_{ij} = 1$ and $X_{jk} = 1$. This prevents a route from being cut off by the elimination of a point. The neighbor camera can be predicted by calculating (6) from matrix X' and Y'.

$$E' = X' \cdot Y' \cdot X'^T \tag{6}$$

4. Localization of Neighbor Node Calculation

The neighbor nodes can be calculated by the algorithm described in Section 3.3. The algorithm, however, needs matrix X that contains the monitoring ranges of all cameras. Since matrix X consists of all points, it is necessary to update matrix X each time a monitoring range of any camera in the system is changed by pan / tilt / zoom functions. Therefore, we localize the algorithm for decreasing the number of points for the calculation. Localization achieves neighbor node calculation without using all points in the system.

In the localized algorithm, all of the points in the system are not required to calculate neighbor nodes of each camera. Each camera manages only points that are included in the monitoring range of itself and its neighbor nodes. Let us define a matrix Yc. The elements of Yc are defined as (7).

$$Yc_{ij} = \begin{cases} 0, & \begin{array}{c} \textit{where the point } Pc_i \textit{ and} \\ \textit{the point } Pc_j \textit{ are not} \\ \textit{neighboring each other.} \end{array} \\ 1, & \begin{array}{c} \textit{where the point } Pc_i \textit{ and} \\ \textit{the point } Pc_j \textit{ are} \\ \textit{neighboring each other.} \end{array} \end{cases} \tag{7}$$

where Pc$_i$ and Pc$_j$ are the points included in the route located between the monitoring range of camera C and the monitoring range of its neighboring cameras. Similarly, we define matrix Xc by (8).

$$Xc_{ij} = \begin{cases} 0, & \begin{array}{c} \textit{where the monitoring range} \\ \textit{of the camera } C_i \textit{ does not} \\ \textit{include the point } Pc_j. \end{array} \\ 1, & \begin{array}{c} \textit{where the monitoring range} \\ \textit{of the camera } C_i \\ \textit{includes the point } Pc_j. \end{array} \end{cases} \tag{8}$$

Xc and Yc are prepared in each camera. All the points are not required in Xc and Yc. Next, matrix Xc' and Yc' are derived from matrix Xc and Yc using the method described in Section 3.3. Then, the neighbor nodes are calculated by:

$$Ec' = Xc' \cdot Yc' \cdot Xc'^T \tag{9}$$

The localized algorithm achieves a robust human tracking system because all points in the system are not required. If the monitoring range of a camera changes, the localized algorithm requires the updated matrixes of that camera and its neighboring cameras.

4.1. An Example of Reduced Points

By the localization, the number of points constituting the matrix decreases, and as a result, the calculation cost is reduced. We show an example of localization by using a map in Fig. 3.

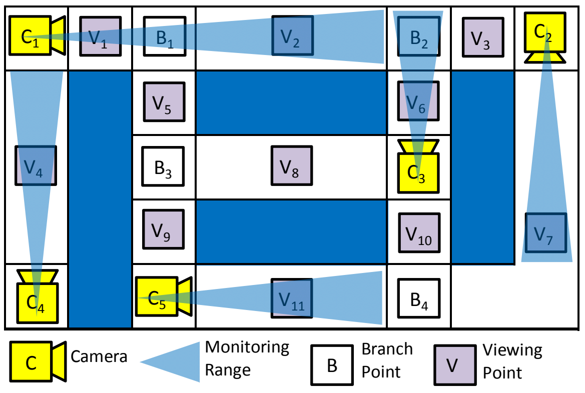

Fig. 3. Example map.

In Fig. 3, C$_i$ represents a camera point. B$_i$ represents a branch point, and V$_i$ represents a viewing point. There are 5 camera points, 4 branch points and 10 viewing points in the map. Since the total number of the points is 19, the size of matrix X becomes 5×19 ($|the\ number\ of\ camera\ points| \times |the\ total\ number\ of\ the\ all\ points|$), and the size of matrix Y becomes 19×19. The monitoring range of each camera is represented by a triangular range.

Here, we focus on camera C$_1$. Yc_1 consists of points included in the monitoring range of camera C$_1$ and points located between the monitoring range of C$_1$ and its neighbor nodes. Therefore, Yc_1 consists of C$_1$, C$_3$, C$_5$, B$_1$, B$_2$, B$_3$, V$_1$, V$_2$, V$_4$, V$_5$, V$_8$ and V$_9$. Thus, the size of matrix Yc_1 becomes 12×12 as shown in (10).

$$Yc_1 = \begin{array}{c} \\ \mathrm{C_1} \\ \mathrm{C_3} \\ \mathrm{C_5} \\ \mathrm{B_1} \\ \mathrm{B_2} \\ \mathrm{B_3} \\ \mathrm{V_1} \\ \mathrm{V_2} \\ \mathrm{V_4} \\ \mathrm{V_5} \\ \mathrm{V_8} \\ \mathrm{V_9} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} \mathrm{C_1} & \mathrm{C_3} & \mathrm{C_5} & \mathrm{B_1} & \mathrm{B_2} & \mathrm{B_3} & \mathrm{V_1} & \mathrm{V_2} & \mathrm{V_4} & \mathrm{V_5} & \mathrm{V_8} & \mathrm{V} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad (10)$$

Similarly, the size of Xc_1 becomes 4×12 as shown in (11).

$$Xc_1 = \begin{array}{c} \\ \mathrm{C_1} \\ \mathrm{C_3} \\ \mathrm{C_4} \\ \mathrm{C_5} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} \mathrm{C_1} & \mathrm{C_3} & \mathrm{C_5} & \mathrm{B_1} & \mathrm{B_2} & \mathrm{B_3} & \mathrm{V_1} & \mathrm{V_2} & \mathrm{V_4} & \mathrm{V_5} & \mathrm{V_8} & \mathrm{V_9} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad (11)$$

Then, matrix Xc_1' is generated from Xc_1 by eliminating column B$_3$, V$_5$, V$_8$ and V$_9$ because their columns satisfy (8). Thus, Xc_1' becomes 4×8 matrix as shown in (12).

$$Xc_1' = \begin{array}{c} \\ \mathrm{C_1} \\ \mathrm{C_3} \\ \mathrm{C_4} \\ \mathrm{C_5} \end{array} \begin{array}{c} \begin{array}{cccccccc} \mathrm{C_1} & \mathrm{C_3} & \mathrm{C_5} & \mathrm{B_1} & \mathrm{B_2} & \mathrm{V_1} & \mathrm{V_2} & \mathrm{V_4} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad (12)$$

For the generation of matrix Yc_1', at first, columns and rows of B$_3$, V$_5$, V$_8$ and V$_9$ are deleted from Yc_1. After that, B$_3$, V$_5$, V$_8$ and V$_9$, which satisfy $Yc_{ik} = 1$ and $Yc_{kj} = 1$ are set to 1 in Yc_1'. For example, the values of $[i, j]$ = [B$_1$, B$_3$] and [B$_3$, B$_1$] are set to 1 by the deletion of V$_5$. This prevents cutting off the route between B$_1$ and B$_3$. Similarly, the values of $[i, j]$ = [B$_1$, V$_8$], [V$_8$, V$_9$], [B$_1$, V$_9$] and [V$_9$, B$_1$] are set to 1 by the deletion of B$_3$; [B$_1$, C$_3$], [C$_3$, B$_1$], [C$_3$, V$_9$] and [V$_9$, C$_3$] are set to 1 by V$_8$; [B$_1$, C$_5$], [C$_5$, B$_1$], [C$_3$, C$_5$] and [C$_5$, C$_3$] are set to 1 by V$_9$. Thus, the matrix Yc_1' is generated as shown in (13).

$$Yc_1' = \begin{array}{c} \\ \mathrm{C_1} \\ \mathrm{C_3} \\ \mathrm{C_5} \\ \mathrm{B_1} \\ \mathrm{B_2} \\ \mathrm{V_1} \\ \mathrm{V_2} \\ \mathrm{V_4} \end{array} \begin{array}{c} \begin{array}{cccccccc} \mathrm{C_1} & \mathrm{C_3} & \mathrm{C_5} & \mathrm{B_1} & \mathrm{B_2} & \mathrm{V_1} & \mathrm{V_2} & \mathrm{V_4} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad (13)$$

4.2. Dealing with the Change of Monitoring Range

A change of monitoring range of a camera may cause a change of its neighbor node. If that happens, the camera notifies its neighbor cameras that the monitoring range has changed. Neighbor cameras update their Xc with the updated monitoring range included in the received notification. When the monitoring range of a camera crosses the monitoring camera of a neighbor node, the matrix Xc and Yc of the camera does not have any points required for calculating neighbor nodes. For example, in the left part of Fig. 4, camera C$_1$ has monitoring range information for the neighbor node C$_2$. However, camera C1 does not need to have the information of C3 because a tracked person moving from C1 to C3 always passes a monitoring range of camera C2. When the monitoring range of C$_1$ changes as seen in the right part of Fig. 4, C$_1$ and C$_3$ become neighbors. However, Xc_1 and Yc_1 of C$_1$ have no information about the monitoring range of C$_3$. Therefore, C$_1$ gets Xc_2 and Yc_2 of C$_2$, and combines Xc_1 and Yc_1 with Xc_2 and Yc_2. Then, C$_1$ can update its neighbor nodes by running a localized neighbor node calculation because Xc_2 and Yc_2 include information about points between C$_2$ and C$_3$.

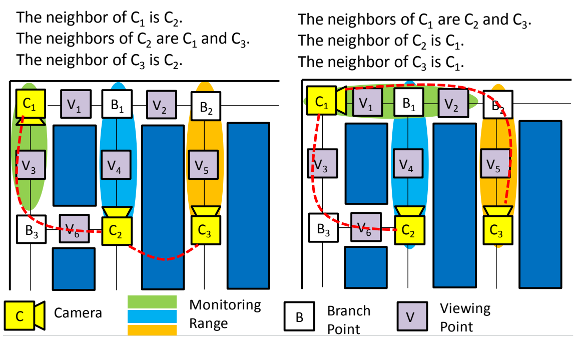
The neighbor of C$_1$ is C$_2$.
The neighbors of C$_2$ are C$_1$ and C$_3$.
The neighbor of C$_3$ is C$_2$.
The neighbors of C$_1$ are C$_2$ and C$_3$.
The neighbor of C$_2$ is C$_1$.
The neighbor of C$_3$ is C$_1$.
C$_1$
V$_1$
B$_1$
V$_2$
B$_2$
V$_3$
V$_4$
V$_5$
B$_3$
V$_6$
C$_2$
C$_3$
C
Camera
Monitoring Range
B
Branch Point
V
Viewing Point

Fig. 4. Change of monitoring range.
Dashed line shows that two cameras are neighbors.

4.3. Dealing with the Additional Camera

If a new camera is added to the system, it does not have matrixes Xc and Yc. Hereafter, C$_{new}$ represents a new camera and C$_1$ represents its neighbor node. When a new camera is installed, we give access information (e.g., IP address and authentication information) about neighbor nodes to the new camera. C$_{new}$ receives the matrix Xc_1 and Yc_1 from C$_1$. Since C$_1$ is adjacent to C$_{new}$, the location of C$_{new}$ installed is included in Xc_1 and Yc_1. Therefore, C$_{new}$ can calculate its neighbor nodes by using Xc_1 and Yc_1.

For example, in the left part of Fig. 5, Xc_1 and Yc_1 have information about the points from C$_1$ to C$_2$, and C$_1$ to C$_3$.

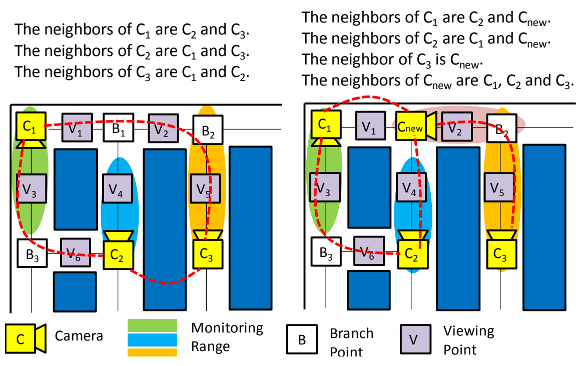
The neighbors of C$_1$ are C$_2$ and C$_3$.
The neighbors of C$_2$ are C$_1$ and C$_3$.
The neighbors of C$_3$ are C$_1$ and C$_2$.
The neighbors of C$_1$ are C$_2$ and C$_{new}$.
The neighbors of C$_2$ are C$_1$ and C$_{new}$.
The neighbor of C$_3$ is C$_{new}$.
The neighbors of C$_{new}$ are C$_1$, C$_2$ and C$_3$.
C$_1$
V$_1$
B$_1$
V$_2$
B$_2$
V$_3$
V$_4$
V$_5$
B$_3$
V$_6$
C$_2$
C$_3$
C$_{new}$
C
Camera
Monitoring Range
B
Branch Point
V
Viewing Point

Fig. 5. Addition of a camera.
Dashed line shows that two cameras are neighbors.

When C$_{new}$ is installed as shown in the right part of Fig. 5, C$_{new}$ receives Xc_1 and Yc_1 from C$_1$. Since Xc_1 and Yc_1 contain all the information required to calculate neighbor nodes of C$_{new}$, C$_{new}$ can calculate its neighbor nodes C$_1$, C$_2$ and C$_3$.

Furthermore, the addition of C$_{new}$ may accompany with the updating of points. Fig. 6 shows an example of update policy.

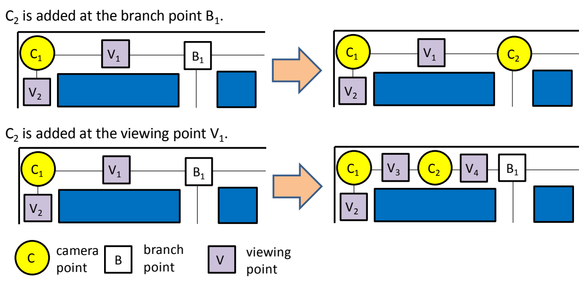
C$_2$ is added at the branch point B$_1$.
C$_1$
V$_1$
B$_1$
V$_2$
C$_2$
C$_2$ is added at the viewing point V$_1$.
V$_3$
V$_4$
C
camera point
B
branch point
V
viewing point

Fig. 6. Change of the points.

When a camera is added at a branch point, the branch point changes to a camera point. When a camera is added at a viewing point, the viewing point is divided into two viewing points on the both sides of added camera point.

Updated points are added to Xc_1 and Yc_1, and C$_{new}$ updates Xc_1 with its monitoring range. The updated matrix Xc_1 and Yc_1 are matrixes Xc_{new} and Yc_{new}. C$_{new}$ calculates neighbor nodes using Xc_{new} and Yc_{new}. Then, C$_{new}$ notifies its neighbor nodes of the updated points and its monitoring range. Cameras that receive this notification update their Xc and Yc and recalculate their neighbor nodes. In this way, the system can effectively handle the change of neighbor nodes as a result of the addition of a camera.

4.4. Dealing with the Removing Camera

As for removing cameras, two cases must be considered; intentional removal and unintentional removal. Because unintentional removal means the sudden loss of camera C$_{rem}$, it is more difficult to handle than intentional removal. Therefore, we discuss only the case of unintentional removal.

Matrixes Xc_{rem} and Yc_{rem} are lost when unintentional removal occurs. To tackle it, each camera exchanges its Xc and Yc matrixes with its neighbor node. Each camera periodically monitors its neighboring cameras to check if they are accessible or not. If one of the cameras becomes inaccessible, the camera is regarded to have been removed unintentionally. Suppose camera C$_1$ detects removal of C$_{rem}$, C$_1$ combines Xc_{rem} and Yc_{rem} with its Xc_1 and Yc_1. In the combined Xc_1, C$_1$ sets the rows corresponding to the camera point of C$_{rem}$ to 0. This means that the monitoring range of C$_{rem}$ becomes nothing. By calculating using the method in Section 4.2, C$_1$ can calculate its neighbor node even if C$_{rem}$ is unintentionally removed. Note that the update of points is not required because unnecessary points (e.g., a camera point corresponding to C$_{rem}$) are deleted automatically using Equation (5) in Section 3.3.

5. Experiments

To verify the effectiveness of the proposed method, we performed a simulation. We created a map of the surveillance area and assumed that the plurality of cameras are installed in experiments.

This experiment was conducted by installing 18 cameras in 124.5 m × 51 m area as shown in Fig. 7. In this experiment, three targets enter the monitoring area from the entrance, walk randomly at a speed of 1.5 m/s to 3.0 m/s, and then exit the area. When one target exits the area, a new target enters from the entrance. Pan / tilt / zoom of each camera occurs randomly once every 30 seconds. Removal and addition of a camera occurs once every 8 hours. In addition, it was assumed that each camera detects a target accurately because there is a focus on localization of neighbor node calculation in this experiment.

This simulation lasted 72 hours. The simulation result is shown in Fig. 8. The camera number is displayed on the vertical axis and elapsed time on the horizontal axis. Movements of the targets are shown by dotted lines, and the locations of mobile agents are shown by solid lines. The occurrence times of pan, tilt, and zoom are shown as ◇. Camera addition is shown as ○. Camera removal is shown as ×.

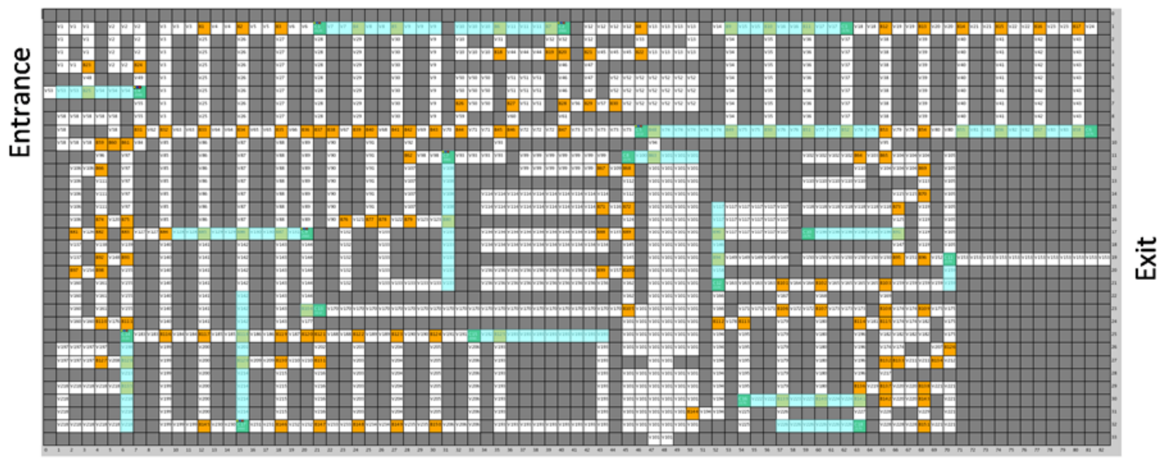

Fig. 7. Simulation map.

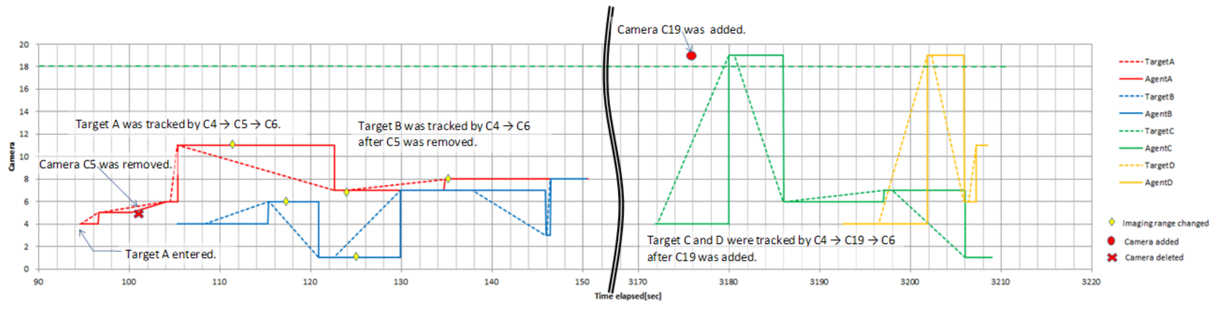

Fig. 8. Result of tracking simulation.

Initially, camera C_5 is adjacent to camera C4, and C_6 is adjacent to C_5. In Fig. 8, target A was detected by C_4, then detected by C_5 and finally detected by C_6. C_5 was deleted after 102 seconds had elapsed. Next, target B was detected by C_4, and then by C_6. This shows that neighbor nodes were correctly updated when C_5 was deleted.

Additionally, camera C_{19} was added between C_4 and C_6 after 3176 seconds had elapsed. Until C_{19} was added, targets were detected by C_6 after being detected by C_4. After C_{19} was added, the target was detected by C_{19} after C_4, and then by C_6 after C_{19}. This means that the neighbor node of C_4 was updated correctly to C_{19}, and that the neighbor node of C_{19} was updated correctly to C_6. Also, even if a monitoring range changed due to pan / tilt / zoom, neighbor nodes were calculated correctly.

There was no failure in the neighbor node calculation, and targets continued to be tracked by mobile agents accurately.

6. Conclusions

We propose the automatic human tracking system using mobile agent technologies. To track a person, we need to find the neighbor node which catches the target person. The proposed algorithm is able to calculate these neighbor nodes with only localized node information. By using the algorithm, it is possible to calculate neighbor node of each camera without the monitoring ranges of all cameras in the system even when monitoring ranges of cameras are changed or cameras are added / removed. The proposed algorithm provides continuous tracking ability even if some nodes are down. We confirmed the effectiveness of the proposed system in simulation experiments.

References

[1]. K. Tanigawa, T. Yotsumoto, K. Takahashi, T. Kawamura, K. Sugahara, Determination of neighbor node in consideration of the photographing range of cameras in human tracking system, *IEICE Transactions on Communications*, Vol. J97-B, No. 10, October 2014, pp. 914-918.

[2]. T. Yotsumoto, K. Tanigawa, M. Tsuji, K. Takahashi, T. Kawamura, K. Sugahara, Automatic Human Tracking using Localization of Neighbor Node Calculation, in *Proceedings of the 9th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'15)*, Venice, Italy, 23-28 August 2015, pp. 139-145.

[3]. M. Tsuji, T. Yotsumoto, K. Tanigawa, K. Takahashi, T. Kawamura, K. Sugahara, Reduction of Neighbor Node Calculations for Automatic Human Tracking System, in *Proceedings of the 9th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE'15),*

Venice, Italy, 23-28 August 2015, pp. 154-159.

[4]. Y. Shirai, J. Miura, Human Tracking Complex Environment, *IPSJ Journal. Computer Vision and Image Media*, Vol. 43, SIG 4(CVIM 4), June 2002, pp. 33-42.

[5]. N. Kawashima, N. Nakamura, R. Hagiwara, H. Hanaizumi, An Improved Method for Background Sub and Its Application to Tracking of Moving Objects, *IPSJ SIG Technical Report. Computer Vision and Image Media*, 2007, 87, September 2007, pp. 11-16.

[6]. H. Mori, A. Utsumi, J. Ohya, M. Yachida, Human Motion Tracking Using Non-synchronous Multiple Observations, *IEICE Transactions on Information and Systems*, Vol. J84-D-II, No. 1, January 2001, pp. 102-110.

[7]. A. Nakazawa, S. Hiura, H. Kato, S. Inokuchi, Tracking Multi Persons Using Distributed Vision Systems, *IPSJ Journal*, Vol. 42, No. 11, November 2001, pp. 2669-2710.

[8]. N. Ukita, Real-Time Cooperative Multi-Target Tracking by Dense Communication among Active Vision Agent, *IEICE Transactions on Information and Systems*, Vol. J88-D-I, September 2005, pp. 1438-1447.

[9]. N. Ukita, Probabilistic-Topological Calibration of Widely Distributed Cameras, *IEICE Transactions on Information and Systems*, Vol. J89-D, No. 7, July 2006, pp. 1523-1533.

[10]. D. Makris, T. Ellis, J. Black, Bridging the Gaps between Cameras, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, Vol. 2, 2004, pp. 205-210.

[11]. N. Takemura, Y. Nakamura, Y. Matsumoto, H. Ishiguro, A Path Planning Method for Human Tracking Agents using Variable-term Prediction, in *Proceedings of the International Conference on Artificial Neural Networks (ICANN'10)*, 2010, pp. 407-410.

[12]. Y. Tanizawa, I. Satoh, Y. Anzai, A User Tracking Mobile Agent Framework "FollowingSpace", *IPSJ Journal*, Vol. 43, No. 12, December 2002, pp. 3775-3784.

[13]. T. Tanaka, T. Ogawa, S. Numata, T. Itao, M. Tsukamoto, S. Nishio, Design and Implementation of a human Tracking System Using Mobile Agents in Camera and Sensor Networks, in *Proceedings of the IPSJ Transaction on Groupware and Network Services Workshop*, November 2004, pp. 15-20.

[14]. K. Aoki, A. Yoshida, S. Arai, N. Ukita, M. Kidode, Functional Assessment of Surveillance of Whole Observation Area by Active Cameras, *IPSJ Journal*, Vol. 48, No. SIG17, 2007, pp. 65-77.