Automatic Human Tracking using Localization of Neighbor Node Calculation

Tappei Yotsumoto[†‡], Kozo Tanigawa[†], Miki Tsuji[‡],
Kenichi Takahashi[‡], Takao Kawamura[‡], Kazunori Sugahara[‡]

[†]System Engineering Department,
Melco Power Systems Co. Ltd.
Kobe, Japan
email: {Yotsumoto.Tappei@zb, Tanigawa.Kozo@zx}.MitsubishiElectric.co.jp

[‡]Graduate School of Engineering,
Tottori University
Tottori, Japan
email: {s112033, takahashi, kawamura, sugahara}@eecs.tottori-u.ac.jp

***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 a lot of information from all cameras about their monitoring ranges of the cameras. 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.**

Keywords-Human tracking; Mobile agent; Pan/Tilt/Zoom; Neighbor relation; Localization.

I. 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. However, 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. In order to monitor many cameras or track multiple people, employing more operators required. Moreover, when an operator loses sight of a suspicious person, the operator must go over multiple cameras to find the suspicious person. For this reason, systems that enable automatic human tracking using multiple cameras are proposed. These systems, however, have two problems: (A) the computational cost for tracking a person is concentrated on one monitoring server; and (B) loss of tracking ability due to a change in camera monitoring range.

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 between 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, it 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, occasionally a tracked person disappears from a camera's monitoring range. 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 display the tracked person next [1].

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 that camera's neighbor nodes. In this case, the algorithm calculates only some nodes, which are from the neighboring camera and the calculation cost for image analysis will be low. Still, there is another case when the monitoring ranges of some camera changes by panning / tilting / zooming operations or other reasons. In this case, the algorithm will require every camera's current monitoring range, and must re-calculate all the neighbor nodes. However, it is not practical to change the monitoring ranges frequently.

In this paper, the current human tracking algorithm is extended to localize the neighbor node calculation. The proposed algorithm utilizes only the monitoring range of neighbor nodes. Furthermore, this realizes a robust human tracking system that enables continuous tracking even when some nodes are down.

Section II of this paper describes the related research. Section III introduces the proposed human tracking system and describes the neighbor node calculation algorithm. Section IV describes the localized neighbor node calculation algorithm. Section V shows the experimental result, and Section VI contains the conclusion of this paper.

II. RELATED RESEARCH

There are some studies for human tracking between plural cameras.

Y. Shirai researched about tracking multiple persons and proposed a technique for collaboration between cameras for tackling obstruction [2]. N. Kawashima proposed a tracking technique that eliminates noise such as shadow by using a dispersion matrix and by improving the background subtraction method [3]. This research was aimed at accuracy enhancement of persons' detection by using multiple cameras, but was unrelated to track a target person across multiple cameras. Since the image recognition has a possibility that an error occurs, we took an approach that does not rely on image recognition.

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 [4]. A. Nakazawa proposed a mechanism for combining the physical data of multiple persons [5]. N. Ukita proposed a system to exchange monitoring images efficiently using an agent based framework [6]. This research assumed 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 [7][8]. These methods require re-collection of the entry-exit (in-out) information when the monitoring ranges of cameras change. N. Takemura's research predicted possible routes which a person could take from one position and speed of the person [9]. Since the information of the appearance and the moving speed of the person are affected by uncertain movement of the person, we took an approach to predict which camera would display the tracked person next from the information of the equipment (e.g., a monitoring range of camera).

Y. Tanizawa proposed a mobile agent-based framework, called "FollowingSpace" [10]. 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 [11]. However, a mechanism to predict in which camera a target would appear next was not explored. K. Aoki proposed a cooperative surveillance system using active cameras [12]. In this system, each active camera adjusts its observation area to decrease blind spots. P. Ibach et al. proposed an algorithm that employs clustering of mobile nodes [13]. This algorithm is combined with techniques for position based routing. An approach using stochastic locking and semi-hierarchical grouping for a peer-to-peer shared memory system was proposed in [14]. In these studies, a way to cooperate with the node located near is shown. However, we emphasize connection relationship of nodes than their location. Even if distance between nodes is far, it is possible to obtain plural cameras would display the tracked person next by following the connection relationship.

III. 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.

A. System configuration

The structure of our automatic human tracking system is shown in Figure 1.

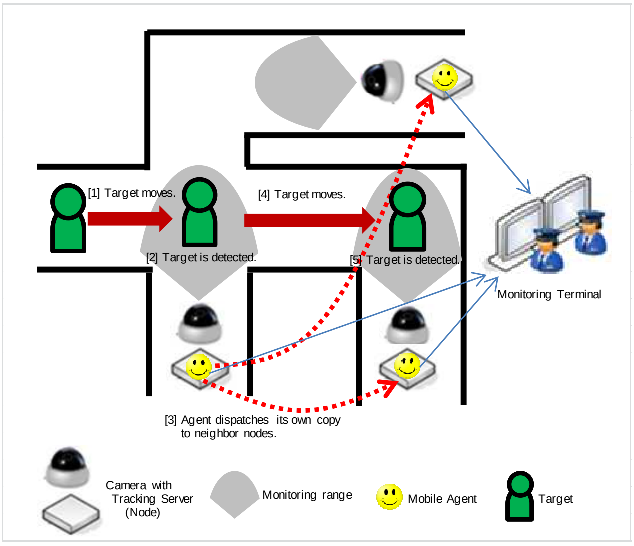

Figure 1. Structure of the proposed system.

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 known through the location of mobile agents.

B. *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 will distribute copies of the active agent to tracking servers of neighboring cameras located in areas 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 in the copy agents to check if the target is in sight. If the target is detected by a tracking server in a neighboring camera, the copy agent of that camera becomes the new active agent and distributes new copies to tracking servers of neighboring cameras. The original active 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. The agent exists in the last known position until it is removed.

C. *Algorithm to calculate neighbor nodes*

Regarding camera location, it is practical to install cameras only at specific places, such as building entrances, rooms, or passage crossings. In such an environment, the techniques [4] for tracking a target by using overlapping ranges are not applicable when a target disappears from the monitoring range of a camera. In this case, it becomes necessary to predict which camera a target will appear in next.

In order to predict the next camera, the points that represent a route through which a target can pass should be defined:

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

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

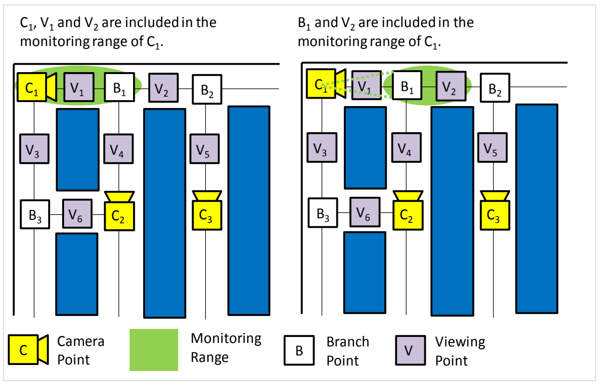
C_1, V_1 and V_2 are included in the monitoring range of C_1.
B_1 and V_2 are included in the monitoring range of C_1.
C_1
V_1
B_1
V_2
B_2
V_3
V_4
V_5
B_3
V_6
C_2
C_3
C
Camera Point
Monitoring Range
B
Branch Point
V
Viewing Point

Figure 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 Figure 2, the monitoring range of C_1 becomes [V_1, B_1]. When the monitoring range of C_1 is as in the right map of Figure 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).

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

Here, 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).

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

The monitoring range of Camera C_i and C_j are overlapped if $D_{ij} \geq 1$.

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

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

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.

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

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, and by connecting two points i and k if $X_{ij} = 1$ and $X_{jk} = 1$. By directly connecting two points which connected through an eliminated point, it prevents a route from being cut off by the elimination of a point. The next camera can be predicted by calculating (6) from matrix X' and Y'.

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

IV. Localization of neighbor node calculation

The neighbor nodes can be calculated by the algorithm described in Section III-C. The algorithm, however, needs matrix X that contains the monitoring ranges of all cameras. Next, it is necessary to identify new branch and viewing points due to changes of monitoring ranges as a result of pan / tilt / zoom functions. The more cameras there are in the system, the higher the number of calculation points must increase. 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 calculation, the neighbor nodes of one camera are calculated by the camera itself. All of the points in the system are not required to calculate neighbor nodes of each camera. Each camera manages only determined points within its monitoring range and those located between its monitoring range and the monitoring range of its neighbor nodes. Let us define a matrix Yc. The elements of Yc are defined as (7).

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

Here, $\mathrm{Pc_i}$ and $\mathrm{Pc_j}$ are the points included in the monitoring range of camera C or the points located between the monitoring range of camera C and the monitoring range of its neighbor nodes. Similarly, we define matrix Xc by (8).

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

Xc and Yc consist of a set number of points for each camera. The total number of points in the system does not need to be known. Next, matrix Xc' and Yc' are derived from matrix Xc and Yc using the method described in Section III-C. Then, the neighbor nodes are calculated by:

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

The localized calculation 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. The following subsection describes update flows when a monitoring range changes.

A. *Support for 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. Cameras that receive this notice update their Xc matrix with the updated monitoring range. Furthermore, in the event that the monitoring range of a camera crosses the monitoring camera of a neighbor node, the matrix Xc and Yc of the camera will not have some points required for calculating neighbor nodes because some new points are generated by the crossing monitoring ranges. For example, in the left part of Figure 3, camera C_1 has monitoring range information for the neighbor node C_2, but does not have the information for C_3. Then points C_1 and V_3 of matrix $Xc1$ become 1. When the monitoring range of C_1 changes as seen in the right part of Figure 3, C_1 and C_3 become neighbors. However, $Xc1$ and $Yc1$ of C_1 have no information about the monitoring range of C_3. Therefore, C_1 gets $Xc2$ and $Yc2$ of C_2, and combines $Xc1$ and $Yc1$ with $Xc2$ and $Yc2$. Then, C_1 can update its neighbor nodes by running a localized neighbor node calculation because $Xc2$ and $Yc2$ include information about points between C_2 and C_3. The points V_2, B_2, V_5 and C_3 on a route to C_3 are added to $Xc1$ and $Yc1$. The point C_1, B_1 and V_2 of matrix $Xc1$ become 1.

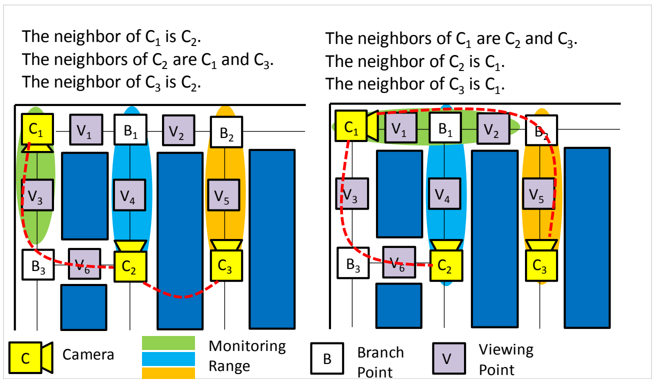
The neighbor of C1 is C2.
The neighbors of C2 are C1 and C3.
The neighbor of C3 is C2.
The neighbors of C1 are C2 and C3.
The neighbor of C2 is C1.
The neighbor of C3 is C1.
Camera
Monitoring Range
Branch Point
Viewing Point

Figure 3. Change of monitoring range.
Dashed line shows that two cameras are neighbors.

B. *Support for additional camera*

If a new camera is added to the system, it will not have matrixes Xc and Yc. Therefore, when a new camera is installed, we give access information (e.g., IP address and authentication information) about neighbor nodes to the new camera. Hereafter, C_{new} represents a new camera and C_1 represents its neighbor node. C_{new} receives the matrix $Xc1$ and $Yc1$ from C_1. Since C_1 is adjacent to C_{new}, the location of C_{new} installed is included in $Xc1$ and $Yc1$. Therefore, C_{new} can calculate its neighbor nodes by using $Xc1$ and $Yc1$.

For example, in the left part of Figure 4, $Xc1$ and $Yc1$ have information about the points from C_1 to C_2, and C_1 to C_3.

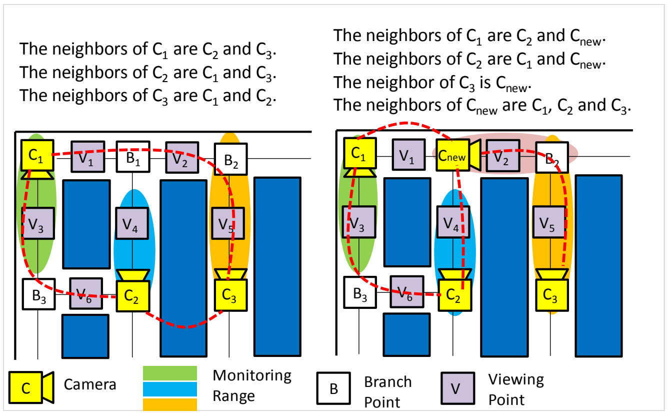
The neighbors of C1 are C2 and C3.
The neighbors of C2 are C1 and C3.
The neighbors of C3 are C1 and C2.
The neighbors of C1 are C2 and Cnew.
The neighbors of C2 are C1 and Cnew.
The neighbor of C3 is Cnew.
The neighbors of Cnew are C1, C2 and C3.
Camera
Monitoring Range
Branch Point
Viewing Point

Figure 4. Addition of a camera.
Dashed line shows that two cameras are neighbors.

When C_{new} is installed as seen in the right part of Figure 4, C_{new} receives $Xc1$ and $Yc1$ from C_1. Since $Xc1$ and $Yc1$ 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.

Moreover, the addition of C_{new} is accompanied with the updating of points. Figure 5 shows an example of this update.

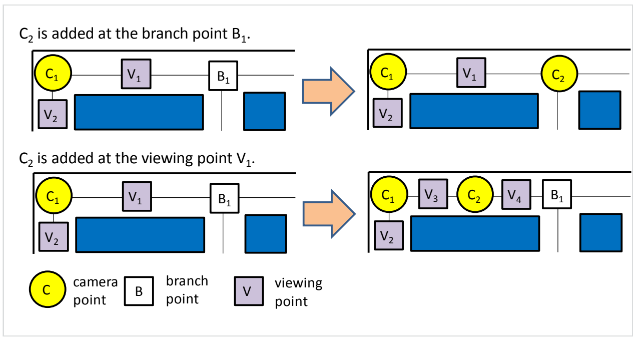
C2 is added at the branch point B1.
C2 is added at the viewing point V1.
camera point
branch point
viewing point

Figure 5. Change of the points.

When a camera is added at a branch point, the branch point is changed to a camera point. When a camera is added at a viewing point, the viewing point becomes a camera point and the viewing point is divided into two points, which are on the both sides of the camera point.

Updated points are added to $Xc1$ and $Yc1$, and C_{new} updates $Xc1$ with its new monitoring range. The updated matrix $Xc1$ and $Yc1$ are matrixes $Xcnew$ and $Ycnew$. Then, C_{new} calculates neighbor nodes using $Xcnew$ and $Ycnew$. Then, C_{new} notifies its neighbor nodes of the updated points and its monitoring range. Cameras that receive this notice update their Xc and Yc matrixes 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.

C. *Support for removing a camera*

As for removing cameras, two cases must be considered; intentional removal and unintentional removal. Because unintentional removal results in the sudden loss of camera C_{rem}'s $Xcrem$ and $Ycrem$ matrixes, it is more difficult to handle than intentional removal. Therefore, only the case of unintentional removal will be discussed.

To prevent sudden loss of matrixes $Xcrem$ and $Ycrem$ when unintentional removal occurs, each camera exchanges its Xc and Yc matrixes with its neighbor node automatically. Each camera always monitors its neighboring cameras to see if they are accessible or not. If any one of the cameras becomes inaccessible, the camera is assumed to have been removed unintentionally. Suppose camera C_1 detects removal of C_{rem}, C_1 combines the $Xcrem$ and $Ycrem$ with its $Xc1$ and $Yc1$ matrixes. In the combined $Xc1$ matrix, 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 0. By calculating using the method in Section IV-A, C_1 can calculate its neighbor node even if C_{rem} is intentionally 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 III-C.

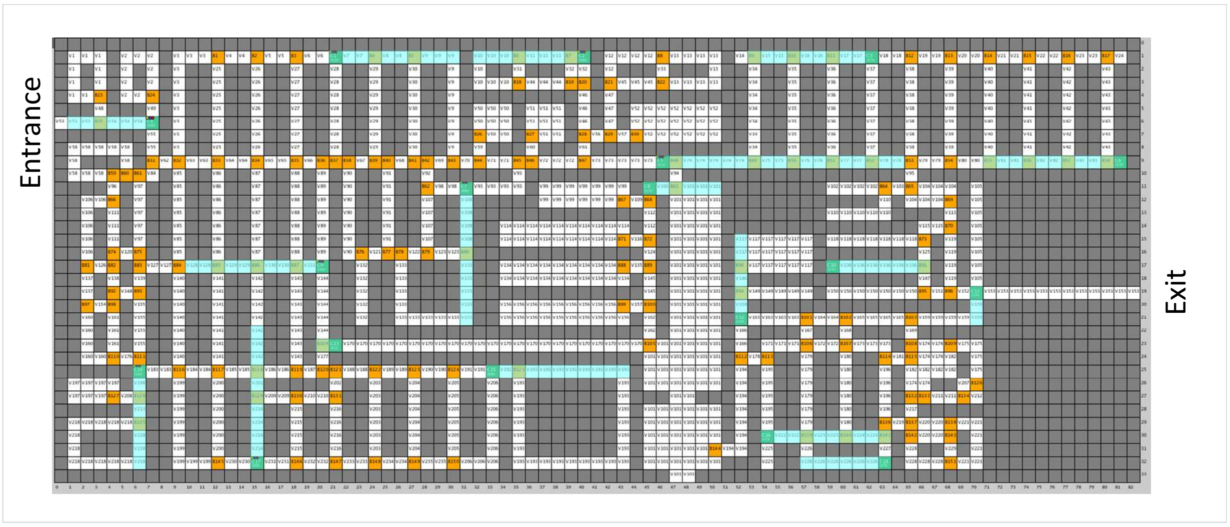

Figure 6. Simulation map.

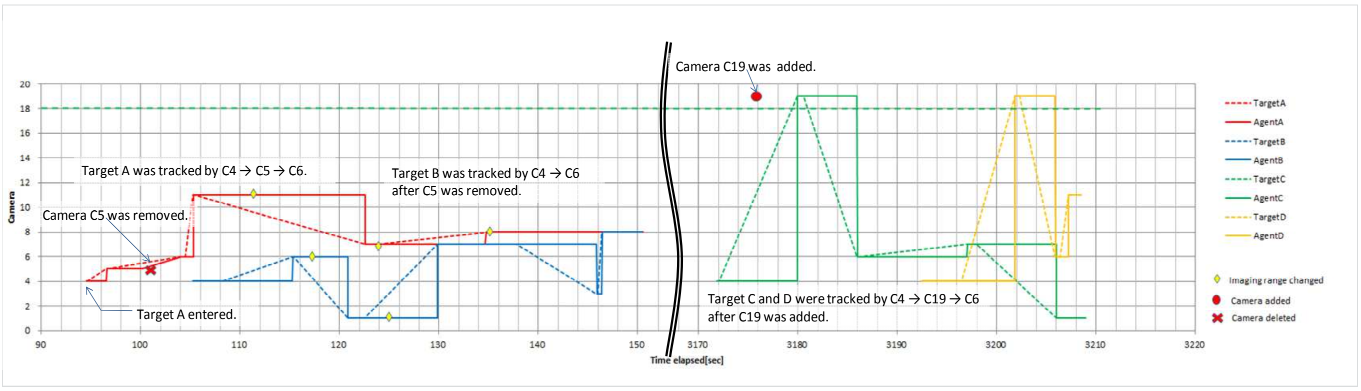

Figure 7. Result of Tracking simulation.

V. EXPERIMENT

The effectiveness of the proposed method was tested using a simulation experiment. The experiment was conducted by installing 18 cameras in 124.5m x 51m area as shown in Figure 6.

Three targets entered the monitoring area from the entrance, walked randomly at a speed of 1.5m/s to 3.0m/s, and then exited the area. When one target exited the area, a new target entered from the entrance. Pan / tilt / zoom of each camera occured randomly once every 30 seconds. Removal and addition of a camera occured once every 8 hours. In this experiment, it was assumed that each camera detects a target accurately because there was a focus on localization of neighbor node calculation. The simulation result is shown in Figure 7. 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 ×.

Initially, camera C_5 is adjacent to camera C_4, and C_6 is adjacent to C_5. In Figure 7, 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.

The simulation lasted 72 hours. There was no failure in the neighbor node calculation, and targets continued to be tracked by mobile agents accurately.

VI. CONCLUSION

We propose the automatic human tracking system using mobile agent technologies. To track a person using mobile agents, a node that is used to detect a target must first be calculated. The proposed algorithm is able to obtain 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. The effectiveness of the proposed system was confirmed in a simulation and experiments. The next step will be large scale experiments using the proposed algorithm to test continuous automatic human tracking in an actual environment.

REFERENCES

[1] K. Tanigawa, T. Yotsumoto, K. Takahashi, T. Kawamura, and K. Sugahara "Determination of neighbor node in consideration of the photographing range of cameras in human tracking system," The IEICE Transactions on Communications, vol. J97-B, no. 10, Oct.2014, pp. 914-918.

[2] Y. Shirai, and J. Miura, "Human Tracking Complex Environment," IPSJ Journal. Computer Vision and Image Media, vol. 43, SIG 4(CVIM 4), Jun. 2002, pp. 33–42.

[3] N. Kawashima, N. Nakamura, R. Hagiwara, and 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), Sep. 2007, pp. 11–16.

[4] H. Mori, A. Utsumi, J. Ohya, and M. Yachida, "Human Motion Tracking Using Non-synchronous Multiple Observations," The IEICE Transactions on Information and Systems, vol. J84-D-II, Jan. 2001, pp. 102–110.

[5] A. Nakazawa, S. Hiura, H. Kato, and S. Inokuchi, "Tracking Multi Persons Using Distributed Vision Systems,"IPSJ Journal. vol. 42, no.11, Nov. 2001, pp. 2669–2710.

[6] N. Ukita, "Real-Time Cooperative Multi-Target Tracking by Dense Communication among Active Vision Agent," The IEICE Transactions on Information and Systems, vol. J88-D-I, Sep. 2005, pp. 1438–1447.

[7] N. Ukita, "Probabilistic-Topological Calibration of Widely Distributed Cameras," The IEICE Transactions on Information and Systems, vol. J89-D(7), July 2006, pp. 1523-1533.

[8] D. Makris, T. Ellis, and J. Black , "Bridging the Gaps between Cameras," CVPR2004, vol.2, 2004, pp. 205-210.

[9] N. Takemura, Y. Nakamura, Y. Matsumoto, and H. Ishiguro, "A Path Planning Method for Human Tracking Agents using Variable-term Prediction", International Conference on Artificial Neural Networks (ICANN), 2010, pp. 407-410.

[10] Y. Tanizawa, I. Satoh, and Y. Anzai, "A User Tracking Mobile Agent Framework "FollowingSpace","IPSJ Journal. vol. 43, no. 12, Dec. 2002, pp. 3775–3784.

[11] T. Tanaka, T. Ogawa, S. Numata, T. Itao, M. Tsukamoto, and S. Nishio, "Design and Implementation of a human Tracking System Using Mobile Agents in Camera and Sensor Networks,"IPSJ Transaction on Groupware and Network Services Workshop 2004 , Nov. 2004, pp. 15–20.

[12] K. Aoki, A. Yoshida, S. Arai, N. Ukita, and M. Kidode, "Functional Assessment of Surveillance of Whole Observation Area by Active Cameras," IPSJ Journal. vol.48, no.SIG17, 2007, pp. 65-77.

[13] P. Ibach, N. Milanovic, J. Richling, V. Stantchev, A. Wiesner, and M. Malek, "CERO: CE robots community." IEE Proceedings-Software, 152(5), 2005, pp. 210-214.

[14] P. Ibach, V. Stantchev, and C. Keller. "DAEDALUS–A Peer-to-Peer Shared Memory System for Ubiquitous Computing." Euro-Par 2006 Parallel Processing. Springer Berlin Heidelberg, 2006, pp. 961-970.