Reduction of Neighbor Node Calculations for Automatic Human Tracking System

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

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

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

Abstract— **Construction method of human tracking systems by using mobile agent technologies is described in this paper. The human tracking system by mobile agent technologies achieves automatic human tracking functions by migrations of mobile agents among camera nodes according with the movements of target persons. The authors have developed the method to decide the destination nodes of mobile agents by calculating neighbor cameras that will catch the tracked person next. In this paper, the method for reducing the amount of calculation to decide the destination of the agent is proposed. In the proposed method, only the data of neighboring cameras which are selected by localization is required. By utilizing this method, it is possible to realize highly robust human tracking systems without having the information of all cameras in the systems.**

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

I. INTRODUCTION

In recent years, in order to protect security of our society, various kinds of systems, such as entrance and exit management and discovery of trespasser, were introduced. The most widely used one is a supervising system using cameras. In the supervising system using cameras, operators must fix their eyes on a number of cameras to find a suspicious person. However, considering the ability of the operators, the maximum number of cameras should be limited. In case of tracking multiple person by more than one monitor equipment, some operators should cooperate with each other for monitoring. Moreover, when an operator loses sight of a suspicious person, the operator must go over multiple cameras to find the suspicious person. Considering these points, systems which enable to track a person automatically among multiple cameras are proposed.

Until now, various kinds of human tracking system have been proposed to reduce the load of the operators. Shiroi [2] surveys researches tracking multiple persons and proposes a technique to make multiple cameras cooperate with one another.. Kawashima [3] proposes the method to raise the detection accuracy of tracking persons with many cameras that eliminates noises such as shadow by using a dispersion matrix and the background subtraction. These researches aim at improvement of persons' detection accuracy by using multiple cameras, and do not consider the case of tracking targets across multiple cameras.

As the research for tracking across multiple cameras, Mori [4] proposes the tracking technique to unify the monitoring image of multiple cameras. Nakazawa [5] proposes the mechanism for combining feature information of multiple persons. Ukita [6] proposes the system to exchange the monitoring images efficiently by an agent-based framework. Aoki [7] proposes a cooperative surveillance system which realizes surveillance of a whole monitoring area by using active cameras. In this system, each active camera adjusts its monitoring range to decrease blind spots by overlapping with the monitoring range of other active cameras. These researches supposes that the monitoring range of cameras are overlapped.

Some mobile agent-based frameworks have been proposed. Tanizawa [8] proposes a mobile agent-based framework "Following Space". 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. Tanaka [9] also proposes an agent-based approach to track a person. However, in these proposals, a mechanism to predict which camera will track a target next is not explained enough.

We proposed a mobile agent-based system for automatic human tracking. Here, the codes which can migrate among tracking servers are called mobile agents. The feature data of a target person that help distinguish him/her from other persons is extracted from camera images in tracking servers and is stored in the mobile agent. Each agent achieves human tracking based on this feature data. The agents migrate among nodes only with small size feature data and it is not necessary to send camera images. For this reason, the amount of communication traffics of proposed system is very small. This system consists of cameras, tracking servers, mobile agents, and a monitoring terminal. In our system, a tracking server installed in each camera analyzes images received from the camera. A mobile agent is prepared for each person being tracked. A mobile agent migrates among tracking servers by detecting the feature data of a target person. The locations of mobile agents are displayed on a monitoring terminal. The operator is able to know the current locations of all the tracked persons by checking the locations of the mobile agents.

In our system, a certain number of cameras are installed at some specific points such as entrances of a building, of rooms and passage crossing. In such an environment, it sometimes happens that a tracked person disappears from any camera's monitoring view. Therefore, we propose the algorithm [1] to predict camera's view in which the tracked person will appear in next. The algorithm calculates neighbor nodes of each camera based on the value of each camera's monitoring range, the map of the floor and the locations where cameras are installed. Hereafter, term *node* means tracking server and/or camera. By using the algorithm, our system predicts cameras which will catch the tracked person next. However, the algorithm needs every camera's current monitoring range for re-calculation of its neighbor nodes. It is difficult when many cameras change their monitoring ranges frequently by pan / tilt / zoom operation. In this paper, we extend the human tracking algorithm for localizing the neighbor node calculation. By using the localization of neighbor node calculation, re-calculation cost of the neighbor node will be low even when many cameras change their monitoring ranges.

In the following sections, Section II introduces our human tracking system and describes the neighbor node calculation algorithm. Section III describes about the localization of neighbor node calculation. Section IV shows the experimental results, and we conclude the paper in Section V.

II. HUMAN TRACKING SYSTEM UTILIZING MOBILE AGENTS

An automatic human tracking system using mobile agent technologies has been developed. 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 mobile agent, the location of each target can be known by monitoring the location of its corresponding mobile agent.

A. *System configuration*

The structure of the system is shown in Figure 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 also the execution environment for a mobile agent and 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 server 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.

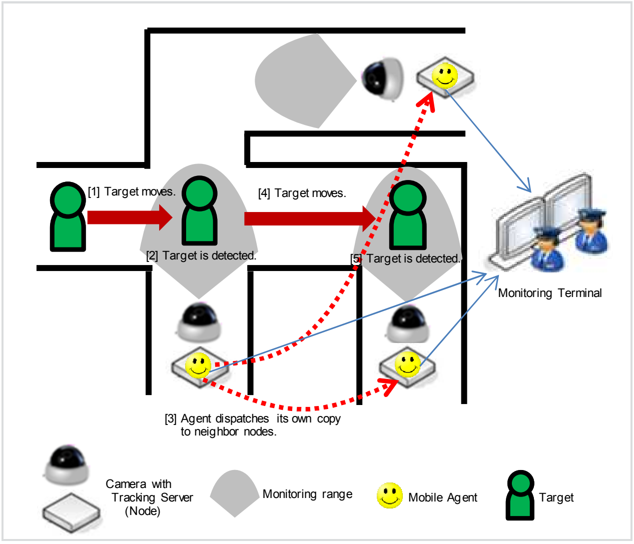

Figure 1. Structure of the proposed system.

B. *Tracking flow*

When a target comes into the monitoring range of a certain tracking server, the tracking server checks whether an agent tracking the target exists 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 to the tracking servers will distribute copies of the active agent, that 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.

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][5][6][7] for tracking a target by using overlapping ranges are not applicable. Thus, it is necessary to predict which camera will catch the target next.

In order to predict the next camera, we first define the points to represent a route on which a target can move:

- Branch point which is a passage crossing,
- Camera point where a camera is installed, and
- Viewing point which is prepared 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.

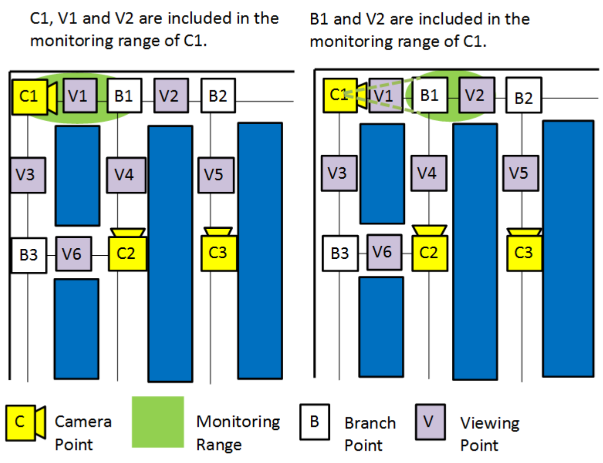
C1, V1 and V2 are included in the monitoring range of C1.
B1 and V2 are included in the monitoring range of C1.
C1
V1
B1
V2
B2
V3
V4
V5
B3
V6
C2
C3
C
Camera Point
Monitoring Range
B
Branch Point
V
Viewing Point

Figure 2. Photographing ranges of camera C1.

The monitoring range of each camera changes by pan, tilt and/or zoom. For example, when the monitoring range of camera C1 is as in the left map of Figure 2, the monitoring range of C1 becomes [V1, B1]. When the monitoring range of C1 is as in the right map of Figure 2, the monitoring range of C1 becomes [B1, V2]. We define matrix X of $|C|\times|P|$ from the monitoring range of all cameras. Here, C is a set of camera points and P is a set of branch, camera, and viewing points. Element X_{ij} of matrix X is defined as (1).

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

Then, cameras whose monitoring ranges overlap each other are calculated by (2).

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

The monitoring ranges of camera C_i and C_j are overlapped if $D_{ij} \geqq 1$. Next, we define adjacency matrix Y of $|P|\times|P|$. Element Y_{ij} of matrix Y is defined as (3).

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

When $E_{ij} \geqq 1$ in (4), the monitoring range of the camera C_j overlaps with *(n-1)* points away from the monitoring range of the camera C_i.

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

III. LOCALIZATION OF NEIGHBOR NODE CALCULATION

The neighbor nodes can be calculated by the algorithm described in Section II-C. Matrices X and Y are composed of all points. Therefore, the larger the monitoring area is and/or the larger number of the cameras increases, the larger X and Y become. Therefore, we localize the calculation of neighbor nodes. Localization allows calculation of the neighbor nodes from limited points without considering all points in the system.

A. Localization of matrix X and Y

In the localized algorithm, each camera manages the points included in the monitoring range of the camera and the points located between the monitoring range of the camera and that of its neighbor nodes. Therefore, we define a matrix Yc. The elements of Yc camera C manages are

$$Yc_{ij} = \begin{cases} 0, & \text{In the case where the point } Pc_i \text{ and the point } Pc_j \text{ are not neighboring each other.} \\ 1, & \text{In the case where the point } Pc_i \text{ and the point } Pc_j \text{ are neighboring each other.} \end{cases} \tag{5}$$

Here, Pc_i and 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 (6).

$$Xc_{ij} = \begin{cases} 0, & \text{In the case where the monitoring range of the camera } C_i \text{ does not include the point } Pc_j. \\ 1, & \text{In the case where the monitoring range of the camera } C_i \text{ does include the point } Pc_j. \end{cases} \tag{6}$$

Since Xc and Yc consist of limited points depending on each camera, each camera does not need to manage all points in the system any more.

B. Calculation of neighbor nodes

From the matrices Xc and Yc, we calculate

$$Ec = Xc \bullet Yc^{n} \bullet Xc^{T} \qquad (7)$$

When $Ec_{ij} \geq 1$ in (7), the monitoring range of the camera C_j overlaps with *(n-1)* points away from the monitoring range of the camera C_i. Since *Xc* and *Yc* consist of points around the camera, the system does not need to collect every camera's current monitoring range. However, we do not know a number of points between the monitoring ranges of two cameras; in other words, we do not know *n*. Therefore, we eliminate points which are not included in the monitoring range of all cameras from matrix *Xc* and *Yc*. Matrix *Xc'* is generated from matrix *Xc* by eliminating all the column *j* which satisfy (8).

$$\sum_{k=i}^{m} Xc_{kj} = 0 \qquad (8)$$

Matrix *Yc'* is also generated from matrix *Yc* by eliminating all the column *j* and row *j* which satisfy (8). At this time, two points connected through an eliminated point should be connected, which prevents a route from being cut off by elimination of a point. Therefore, Xc'_{ik} is set to 1, if Xc_{ij} = 1 and Xc_{jk} = 1. Then, the next camera is found by calculating (9) from matrix *Xc'* and *Yc'*.

$$Ec' = Xc' \bullet Yc' \bullet Xc'^{T} \qquad (9)$$

C. An example of localization

We show an example of localization by using a map in Figure 3. In Figure 3, *Cx* represents a camera point. *Bx* represents a branch point, and *Vx* 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*| × |*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.

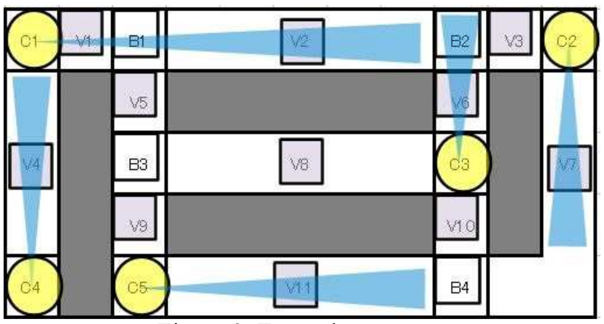

Figure 3. Example map.

Here, we focus on camera C1. Yc_1 consists of points included in the monitoring range of camera C1 and points located between the monitoring range of C1 and its neighbor nodes. Therefore, Yc_1 consists of C1, C3, C5, B1, B2, B3, V1, V2, V4, V5, V8 and V9. Thus, the size of matrix Yc_1 becomes 12 × 12 as shown in (10).

$$Yc_1 = \begin{array}{c|cccccccccccc} & C1 & C3 & C5 & B1 & B2 & B3 & V1 & V2 & V4 & V5 & V8 & V9 \\ \hline C1 & 0&0&0&0&0&0&1&0&1&0&0&0 \\ C3 & 0&0&0&0&0&0&0&0&0&0&1&0 \\ C5 & 0&0&0&0&0&0&0&0&0&0&0&1 \\ B1 & 0&0&0&0&0&0&1&1&0&1&0&0 \\ B2 & 0&0&0&0&0&0&0&1&0&0&0&0 \\ B3 & 0&0&0&0&0&0&0&0&0&1&1&1 \\ V1 & 1&0&0&1&0&0&0&0&0&0&0&0 \\ V2 & 0&0&0&1&1&0&0&0&0&0&0&0 \\ V4 & 1&0&0&0&0&0&0&0&0&0&0&0 \\ V5 & 0&0&0&1&0&1&0&0&0&0&0&0 \\ V8 & 0&1&0&0&0&1&0&0&0&0&0&0 \\ V9 & 0&0&1&0&0&1&0&0&0&0&0&0 \end{array} \qquad (10)$$

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

$$Xc_1 = \begin{array}{c|cccccccccccc} & C1 & C3 & C5 & B1 & B2 & B3 & V1 & V2 & V4 & V5 & V8 & V9 \\ \hline C1 & 1&0&0&1&0&0&1&1&0&0&0&0 \\ C3 & 0&1&0&0&1&0&0&0&0&0&0&0 \\ C4 & 0&0&0&0&0&0&0&0&1&0&0&0 \\ C5 & 0&0&1&0&0&0&0&0&0&0&0&0 \end{array} \qquad (11)$$

Then, matrix Xc_1' is generated from Xc_1 by eliminating column B3, V5, V8 and V9 because their columns satisfy (8). Thus, Xc_1' becomes 4 × 8 matrix as shown in (12).

$$Xc_1' = \begin{array}{c|cccccccc} & C1 & C3 & C5 & B1 & B2 & V1 & V2 & V4 \\ \hline C1 & 1&0&0&1&0&1&1&0 \\ C3 & 0&1&0&0&1&0&0&0 \\ C4 & 0&0&0&0&0&0&0&1 \\ C5 & 0&0&1&0&0&0&0&0 \end{array} \qquad (12)$$

For the generation of matrix Yc_1', at first, columns and rows of B3, V5, V8 and V9 are deleted from Yc_1. After that, B3, V5, V8 and V9, which satisfy Yc_{ik} = 1 and Yc_{kj} = 1 is set to 1 in Yc_1'. For example, the values of [i, j] = [B1, B3] and [B3, B1] are set to 1 by the deletion of V5. This prevents cutting off the route between B1 and B3. Similarly, the values of [i, j] = [B1, V8] , [V8, V9], [B1, V9] and [V9, B1] are set to 1 by the deletion of B3; [B1, C3], [C3, B1], [C3, V9]and [V9, C3] is set to 1 by V8; [B1, C5], [C5, B1], [C3, C5]and [C5, C3] is set to 1 by V9. Thus, the matrix *Yc₁'* is generated as shown in (13).

$$Yc_1' = \begin{array}{c|cccccccc} & C1 & C3 & C5 & B1 & B2 & V1 & V2 & V4 \\ \hline C1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ C3 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ C5 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ B1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ B2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ V1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ V2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ V4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \qquad (13)$$

IV. EXPERIMENT

To confirm effectiveness of the proposed method, the simulation experiment using Java is conducted. In the experiment, we prepared seven maps from 30 × 30 [m²] to 210 × 210 [m²]. The number of points in each map is shown in Table I.

Table I. NUMBER OF POINTS IN THE SIMULATION MAP

Map Size [m²]	Number of points			
	Camera points	Branch points	Viewing points	total
30×30	8	18	44	70
60×60	32	80	184	296
90×90	72	189	418	679
120×120	128	328	740	1196
150×150	200	518	1160	1878
180×180	288	751	1674	2713
210×210	392	1030	2286	3708

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 go out from the exit. When a target leaves the monitoring area, a new target enters from the entrance. Pan / tilt / zoom of each camera occur irregularly in the ratio once in 30 seconds. Deletion and addition of cameras occur in the ratio once in 8 hours. Table II shows the number of points in *Xc* and *Yc*.

Table II. NUMBER OF POINTS

Map size [m²]	# of camera points managed in *Xc*			# of all points managed in *Xc*/*Yc*		
	AVE	MAX	MIN	AVE	MAX	MIN
30×30	6	8	5	34	35	52
60×60	12	20	5	75	138	25
90×90	12	26	4	77	162	25
120×120	16	33	3	99	213	19
150×150	16	40	3	100	262	17
180×180	16	41	4	95	256	21
210×210	16	53	4	95	330	25

The size of matrix *X* becomes |*camera points*|× |*total points*| shown in Table I, and the size of matrix *Y* becomes |*total points*|× |*total points*|. For example, in 120 × 120 [m²] map, the size of *X* is 128 × 1196, and the size of *Y* is 1196 × 1196. When we apply the localization, the average size of matrix which each camera manages is dramatically reduced in all maps. For example, in 120 × 120 [m²] map, the average size of *Xc* is 16 × 99, the average size of *Yc* is 99 × 99 shown in Table II. In the way, comparing the Table I and Table II, the number of points having each camera is reduced. This means the change of monitoring range of a camera effects on limited nodes. According to the size of the monitoring area, the number of the points which is expected to be reduced becomes large at accelerated pace.

We also measured the processing time to calculate neighbor nodes. The results are summarized in Table III.

Table III. CALCULATION TIME

Map size [m²]	Average with localization [ms]	Average without localization [ms]
30×30	0.054	0.192
60×60	0.064	0.735
90×90	0.091	9.268
120×120	0.128	54.815
150×150	0.117	216.471
180×180	0.130	1299.056
210×210	0.192	7496.471

If the monitoring range of the camera is changed by pan/tilt/zoom, the re-calculation of the neighbor node is needed by (4) each time. We show the calculation time for several map sizes in Table III. We can see in Table III that without localization, when the monitoring area is large, the re-calculation of the neighbor time is very large. In contrast, with localization, the re-calculation of the neighbor time is not larger than without localization. For example, when we do not apply the localization, the system spends 7496 [ms] for neighbor node calculations in 210 × 210 [m^2] map. In this experiment, since pan / tilt / zoom of each camera occurs irregularly in the ratio once in 30 second, which is already overloaded, the system fails to track targets. On the other hand, when we apply the localization in 210 × 210 [m^2] map, the system can calculate neighbor nodes in 0.192 [ms] in average. That is, the localization enables the system to calculate neighbor nodes even when the size of the map is bigger. In addition to that, the calculation time is almost the same.

V. CONCLUSION

The automatic human tracking system using mobile agent technologies is proposed. For tracking a person by using mobile agent technologies, we need to find the neighbor node which catches the target person. Therefore, we propose the localization algorithm to calculate neighbor nodes. This realizes continuous tracking abilities even if the monitoring ranges of the cameras frequently change. The effectiveness of the proposed system was confirmed in the simulation. The simulation result shows the calculation cost of neighbor nodes is suppressed even when the number of cameras installed in a system increases. The future task is to conduct large scale experiments using the proposed algorithm 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," ISPJ 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] K. Aoki, A. Yoshida, S. Arai, N. Ukita, and M. Kidode, "Functional Assessment of Surveillance of Whole Observation Area by Active Cameras," ISPJ Journal. Vol.48, No.SIG17, 2007, pp.65-77.

[8] 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.

[9] 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.