Integrated Computer-Aided Engineering 17 (2010) 41–58
DOI 10.3233/ICA-2010-0328
IOS Press

Bypass methods for constructing robust automatic human tracking system

Hiroto Kakiuchi[a,*], Takao Kawamura[b], Tadaaki Shimizu[b] and Kazunori Sugahara[b]
[a]*Engineering Department, Melco Power Systems Co., Ltd. Hyogo-ku, Kobe, Japan*
[b]*Graduate School of Engineering, Tottori University, Tottori-city, Tottori, Japan*

Abstract. We have been proposing the automatic human tracking system to overcome limitations of existing video surveillance systems. This paper proposes two bypass methods to keep tracking a targeted person by the system constructed by using mobile agent technologies even if unexpected problems are occurred. The proposed methods are called as "Recalculation Bypass Method" and "Additional Calculation Bypass Method". These bypass methods utilize the algorithm which is elaborated in the paper, "An Algorithm to Determine Neighbor Nodes" [7]. The mobile agents are operating in automatic human tracking system and the bypass methods are utilized by the mobile agents. It is a very important ability for the mobile agent not to lose tracking in automatic human tracking systems. The bypass methods contribute to the realization of robust system and resolve two classifications of problem to ensure the tracking ability of mobile agents. The two classifications of problem are logical and physical problems. In the case of logical problems, they occur inside the system and they are called as "broken server". While in the case of physical problems, they occur outside the system and they are called as "pathway being modified". This system is able to track the targeted person continuously by utilizing the methods even if unexpected problems are occurred. We confirmed the effectiveness of the proposed bypass methods by the experiments that used more than one broken servers and pathway being modified.

1. Introduction

Due to the efficiency of video surveillance system to monitor a targeted person remotely, it was highly utilized in different institutions. Primarily, the video surveillance system is used as a security system because of its efficiency in tracking a particular person. If the function of the video surveillance system is extended to automatically tracked numerous number of people, the usage of the system will be extended to various fields. Such improved video surveillance system is to be called automatic human tracking system in this paper. The "automatic human tracking system" can be applied in various fields. For examples, to search for lost children, to gather/analyze consumers route pattern for marketing research of a retail establishment, and others. Our aim is to develop the automatic human tracking system by resolving the problems of conventional video surveillance system.

Currently existing video surveillance systems have limited capabilities, thus, the system is not applicable at any time. In one case, it is not be able to locate a numerous number of people located at several position at the same time and automatically locate a particular person. Furthermore, in a conventional video surveillance systems, users are involved in a heavy workload. Since users will be mostly involved in switching the view from one video camera to the other video camera, the number of targeted person that this system can locate is limited. Though an approach of identifying a particular person from a numerous surveillance positions is possible, it must increase the workload of the user. In such cases, it demands users to identify the person carefully.

Some researchers suggested the solutions that solve above problems. The first approach was to use an active camera [14,25] to track a person automatically, thus the

[*]Corresponding author: Hiroto Kakiuchi, Engineering Department, Melco Power Systems Co., Ltd. 1-1-2, Wadasaki-cho, Hyogo-ku, Kobe, Japan. Tel.: +81 78 682 6942; Fax: +81 78 682 6981; E-mail: Kakiuchi.Hiroto@zs.MitsubishiElectric.co.jp.

camera moves in a synchronized motion along with the movement of the targeted person. Since the method for correcting blurring image [9] is proposed, the active camera is available. And the approach is applicable to locating small number of people, but it is not efficient in locating numerous number of people. The second approach was to position the camera efficiently at strategic surveillance location [22]. But by this approach, the process will require large resources in order to thoroughly install the cameras. Such approach is not applicable to an establishment with limited resources. The third approach was implementing efficiently to track an object with multiple camera [26,27] using sensors. This research is efficient in the small area, but not in a large area. The forth approach was implementing 3D object tracking [11] with multiple sensors. We consider that the research can be used as a simulator of the human tracking system in the future.

To identify a numerous number of targeted people, the use of an image processing and installation of video cameras on any designated location will be a better approach. However it would be inappropriate to use a single server when locating numerous people, since the image processing increases server load. As such, a new type of system that is capable to identify and locate people is developed. The ratio of mobile agent and the number of person is equally proportionated on the mobile agent technologies [2,19–21] in this system. Agent-based approach is appropriate on distributed system and parallel processing according to many researches [5,6,16], since mobile agent can transfer copy agents to predetermined server in the system. And in cooperation working, multi-agent system is effective [4]. On distributed processing, the mobile agent technologies are more effective than the conventional video surveillance systems, assuming that a large number of servers with video camera are freely installed. If one mobile agent can track one person, then multiple mobile agents can track numerous number of people at the same time. And the server balances the load process of the operating mobile agent on each server with video camera. The video surveillance system enhanced with mobile agent technologies is called "Automatic Human Tracking System" [8,24]. In this system, a mobile agent tracks a person captured by a video camera and a server process the data. The video camera and the server are treated as a single entity since the video camera along with the server computer is deployed at the surveillance position. Before a person is tracked, a mobile agent is generated for a particular person. After verifying the feature of the person within the server [15], the mobile agent tracks the movement of the person by utilizing the neighbor camera/server location information. The destination neighbor camera/server location information is determined utilizing the "algorithm to determine neighbor nodes".

The system is effective in tracking a numerous number of people at the same time. However, this system has a problem in which the agent can not continuously track the person on particular situation. The situations in which the mobile agent can not transfer to the next server are based on the two problems, the logical and physical problems. In the case of a logical problem, it occurs inside the system like the instance of a "broken server". And in the case of a physical problem, it occurs outside the system like the instance of a "pathway being modified". We propose these bypass methods to resolve the above problems.

2. Automatic human tracking system

2.1. System features

The features of the system are shown in Fig. 1. Graphical user interface was developed in order to improve maintainability of the system configuration. The functions of the GUI are to create/edit graphical representation of a building layout, to deploy video cameras on the graphical layout, to monitor mobile agents, and to create data for simulation. The GUI utilizes the algorithm to determine neighbor nodes to compute the adjacency of video cameras, thus information is displayed graphically and the maintainability is improved. The algorithm is utilized not only to calculate the neighboring video cameras, but also to determine the mobile agent's next destination accurately. The algorithm also contributes in improving the reliability of the system by using minimal computing resources. Bypass methods is the subject of this paper, it utilizes the algorithm and improves the robustness of the automatic human tracking system. Since the mobile agents utilize the methods, continuous tracking of the target people is ensured. Thus, the reliability and persistency of the system are improved. The lost track re-detection methods are currently being researched, which can also improve the robustness of the system as well as the reliability and the persistency. Because the re-detection methods can ensure continuous tracking of the target people by re-detecting if the mobile agents lost track of the people. OSGi [18] and mobile agent technologies are adopted to improve the system scalability. OSGi

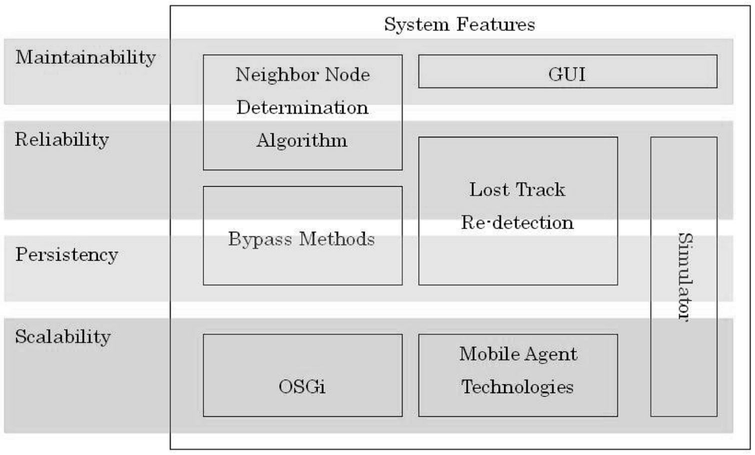

Fig. 1. System features.

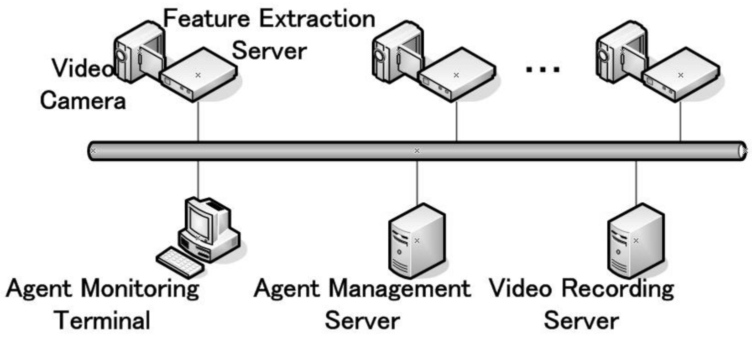

Fig. 2. System configuration.

is a framework developed using Java Language and a server software including web server function. In the OSGi server, the OSGi framework manages/controls a software which is treated as a single component called "bundle". Mobile agent server is a bundle on the OSGi server. In the simulator, a simulated image processing runs on the OSGi server by using a simulated information of the target person. And the simulator is utilized to improve the reliability and the scalability.

2.2. *System architecture and process flow*

The system configuration of the automatic human tracking system is shown in Fig. 2. It is assumed that the system is applied in a certain building. Before a person is granted access inside the building, the person's information is registered into the system. Through a camera the person's image of the face and body is captured and registered into the system. Any person who is not registered or not recognized by the system is not allowed to roam inside the building.

This system is composed of agent monitoring terminal, agent management server, video recording server, and feature extraction server with video camera. The agent monitoring terminal is used for registering the target person's information, retrieving and displaying the information of the initiated mobile agents, and displaying video of the target entity. The agent management server records mobile agents' tracking information history, and provides the information to the agent monitoring terminal. The video recording server records all video images and provides the images to the agent monitoring terminal via request. The feature extraction server along with the video camera analyzes the entity image and extracts the feature information from the image.

A mobile agent tracks a target entity by using the feature information and the neighbor nodes information. The number of mobile agents is in proportion to the number of the target entities. A mobile agent is initialized at the agent monitoring terminal and launched into the feature extraction server. The mobile agent

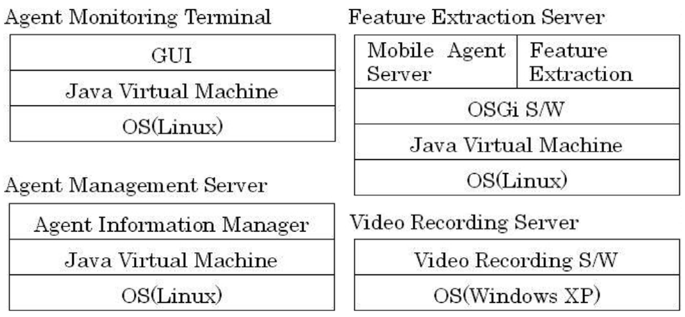

Fig. 3. System architecture.

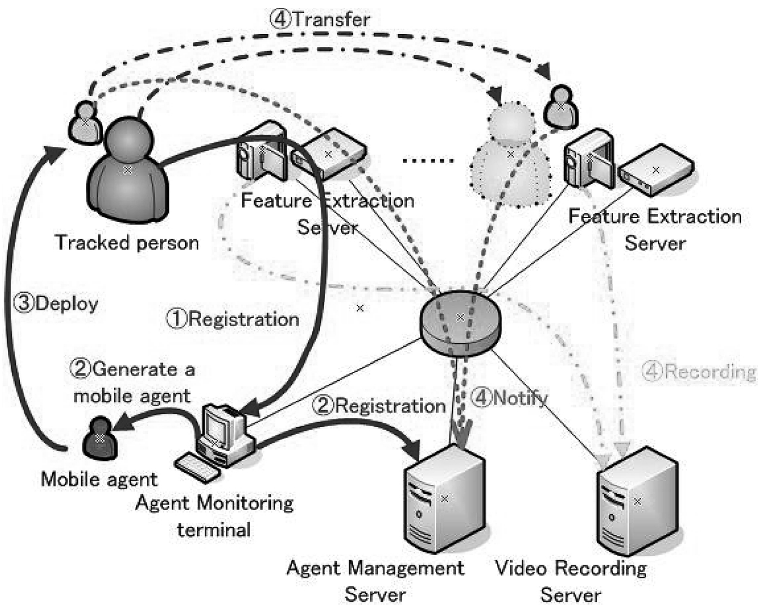

Fig. 4. Processing flow.

extracts the feature of a captured entity and compares it with the feature which the agent already has. If the feature is equivalent, the entity is located by the mobile agent.

The system architecture is shown in Fig. 3. GUI is operated only on the agent monitoring terminal. GUI is able to register images of the entities and monitor all the mobile agents' status. Mobile agent server is executed on the feature extraction server and allow the mobile agents to execute. Feature extraction function is able to extract the feature of the captured entities, and the feature information is utilized for tracking of mobile agents. OSGi S/W connects these software, and the software is able to utilize each other. Agent information manager manages all mobile agent information and provide the information to agent monitoring terminal. Video recording S/W records all video, and provides the video movie to agent monitoring terminal. Each PC is equipped with Intel Pentium IV 2.0 GHz and 1 GBytes memory. The required system has to satisfy the conditions that maximum execution time of feature judgment is 1 second and maximum execution time of mobile agent transfer is 200 milliseconds.

The processing flow of the proposed system is shown in Fig. 4. First, a system user selects the entity on the screen of the agent monitoring terminal, and extracts the tracked entitiy's feature information as electronic data. Next, a mobile agent is generated per person and registered as the mobile agent information including the feature information into the agent management server. Then the mobile agent is launched from the terminal to the first feature extraction server. When the mobile agent catches the target entity on the feature extraction server, the mobile agent notifies the agent management

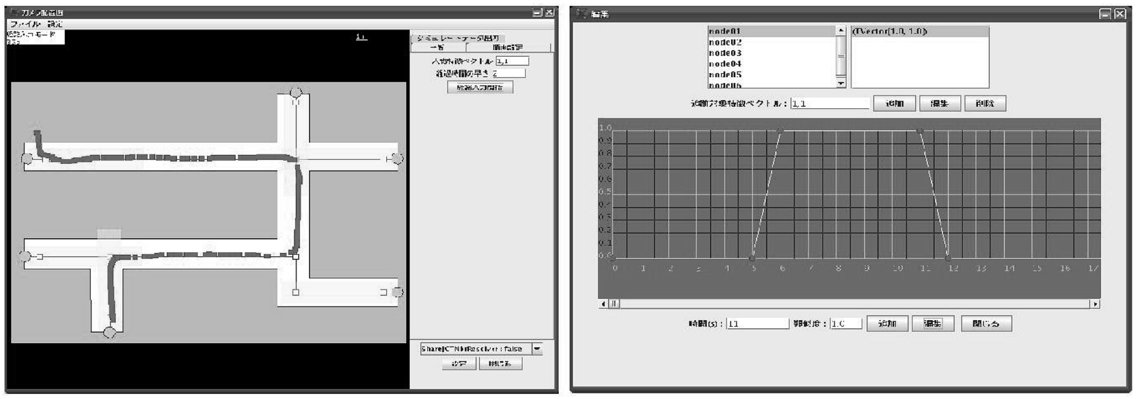
Fig. 5. Editor of simulation route to follow and creator of simulation feature data.

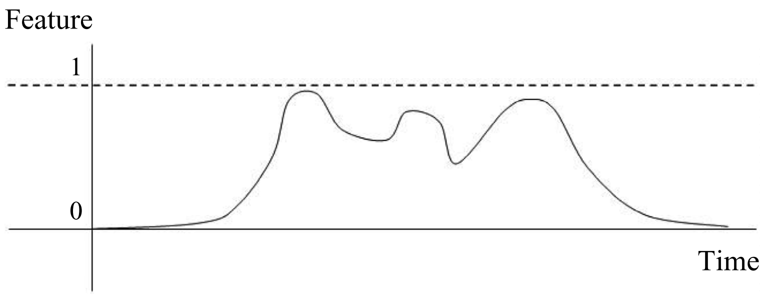

Fig. 6. Feature data.

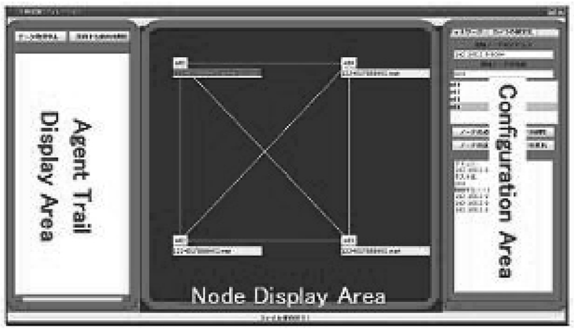

Fig. 7. Graphical User Interface.

server of the information such as the video camera number, the discovery time and the mobile agent identifier. Finally the mobile agent deploys the copy of himself to the neighbor feature extraction servers and waits for the person to appear. If the mobile agent catches the person, the mobile agent notifies the agent management server of the information, removes original agent and other copy agents, and deploys the copy of himself to neighbor feature extraction servers again. Continuous tracking is realized by repeating the above flow.

2.3. *Simulator and GUI*

A simulator is currently being developed in Java Language. The simulator consists of 3 functions, the image processing simulator, the editor of simulation route to follow and the creator of simulation feature data. The simulator tools are shown in Fig. 5. The genuine image processing function is also currently being developed by the feature extraction method based on SIFT [3,23]. Since it is actually difficult to place a lot of cameras, the image processing simulator is performed on the feature extraction server instead of the genuine image processing function. In addition, this simulator changes a target entity's feature to a walking target entity by using a simulation agent. The simulation agent [12] is also mobile agent that simulates the movement of a target entity and changes a target entity feature as shown in Fig. 6. The movement of the target entity is digitized by the editor of simulation route to follow and the target entity feature is digitized by the creator of simulation feature data. If the system executes multiple simulation agents, numerous number of target entities are able to be simulated.

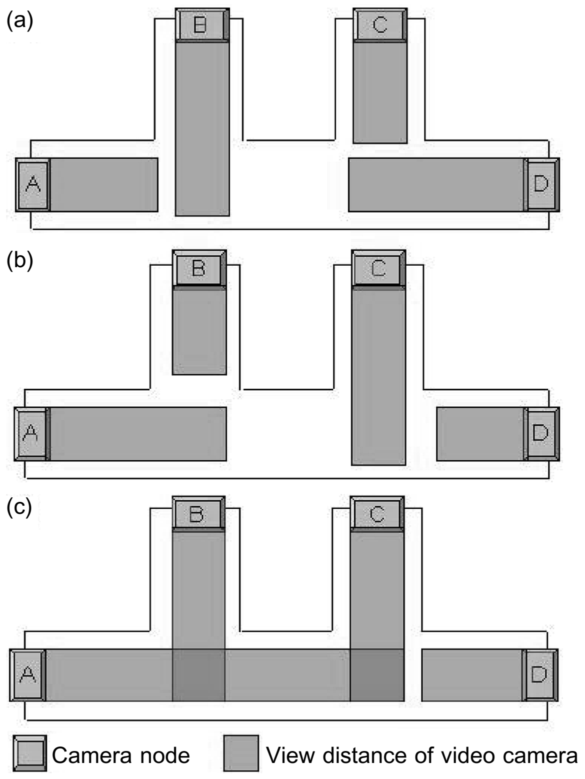

Fig. 8. Example that view distance influences.

Graphical user interface is shown in Fig. 7. Agent trail information is displayed on the left side area of the GUI. A map of running agents and each agent's status are displayed on the center side area. A user sets the configuration, IP address of feature extraction server, accuracy level of feature extraction server, etc on the right side. The GUI can simulate various layouts of the feature extraction server, set the accuracy level of the feature extraction server, and confirm the movement of the agents. Test of the generator was confirmed with data from 20 cameras. Currently the simulator needs more improvement especially when using more than 20 cameras. This simulator will be adding other necessary functions. And the simulator will be also utilized to verify the performance of a system. As such, the simulator is utilized in the examination.

3. An algorithm to determine neighbor node

3.1. A problem on determining neighbor video cameras

If a mobile agent tracks a target entity, the mobile agent has to understand the deployed location of the video cameras. However the neighbor cameras are determined by the view distance of the video cameras. Then the problem influenced by the difference of the view distances occurs. The problem consists of the difference of an overlap of view, an interrupt of view and a clear view.

A scenario in which neighbor video camera location is influenced by the view distances is as shown in Fig. 8. The figure shows 3 diagrams portraying a floor plan with 4 video cameras each, considering the view distances of each video camera are different and assuming that the target entity to be tracked moves from the location of video camera A to video camera D.

The neighbor of video camera A in object (a) of Fig. 8 is video camera B but not C and D. In object (a) of Fig. 8, video camera C and D are not considered as the neighbors of video camera A, because video camera B blocks the view of video camera C and D. And the target entity can be captured at an earlier time on video camera B.

But in the case of object (b) of Fig. 8, the neighbors of video camera A are video camera B and C but not camera D. Thus neighbor video camera's location indicates the difference of view distances of video cameras. The case of object (c) in Fig. 8 is more complicated.

The neighbors of video camera A in object (c) of Fig. 8 are video camera B, C, and D. And video camera B is not considered as the neighbor of video camera C. It is because video camera A exists as the neighbor between video camera B and C. When it is assumed that a target entity moves from A to D, the target entity is sure to be captured by video camera A, B, A, and C in that order.

This scenario indicates that the definition of "neighbor" cannot be determined clearly because the determination of the neighbor definition is influenced by the change of view distances and it becomes more complicated as the number of video cameras increases.

3.2. Overview of the algorithm

The developed algorithm can easily determine the neighbor video camera's location without the influence of view distances and modification of the information of the currently installed cameras. The modification information is set on the system to compute neighbor video cameras on the diagram, which is expressed as graph. Nodes are used to compute neighbor video camera's information in this algorithm. The nodes are defined as follows:

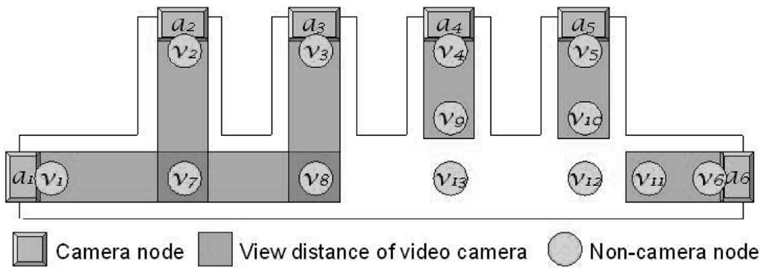

Fig. 9. Figure that sets non-camera nodes.

Camera Node: the location of video camera is labeled as camera node and the nodes are defined as $A = \{a_1, a_2, ..., a_p\}$. This node is also a server with video camera.

Non-camera Node: the nodes are defined as $V = \{v_1, v_2, ..., v_q\}$. Conditions of a non-camera node are stated below:

– Either of crossover, corner, terminal of pathway.
– The position where video camera is installed.
– The end point of the view distance of video camera.

In addition, the point where the above conditions are overlapped is treated as one node. When the view distance of the video camera reaches non-camera node, the non-camera node is defined as the neighbor of the camera node. When two non-camera nodes are next to each other on a course, it is defined that those nodes are neighbors. Figure 9 shows the example of applying these definitions and shows view distances of video cameras.

The algorithm uses adjacent matrix. Two kinds of the adjacent matrix are needed in the algorithm. One is adjacent matrix X made from camera nodes' locations as rows and non-camera nodes' locations as columns. Another one is adjacent matrix Y made from non-camera nodes' locations as rows and columns. The neighbor information of video camera is calculated from the connection information of non-camera node by using adjacent matrix X and Y . However, neighbor information is miscalculated in a particular condition. Therefore, when the non-camera nodes that cross each other the view distance of two or more video camera exist mutually, it is necessary to break the connection between those nodes. It explains details in the definition of adjacent matrix Y .

The below is the algorithm to determine the neighbor nodes:

1. Set camera nodes and non-camera nodes on the diagram.
2. Generate adjacent matrix X from camera nodes' locations and non-camera nodes' locations, and generate adjacent matrix Y from non-camera nodes' location. Adjacent matrix X indicates that rows are camera nodes and columns are non-camera nodes. Adjacent matrix Y indicates that rows and columns are non-camera nodes, which generate adjacent matrix Y to resolve an overlap problem of view distances between video cameras.
3. Calculate adjacent matrix X' and Y' by excluding unnecessary non-camera nodes from adjacent matrix X and Y .
4. Calculate neighbor's location matrix by multiplying adjacent matrix and transposed matrix X'^T . This neighbor's location matrix is the neighbor's nodes information.

Unnecessary non-camera node is a non-camera node in which it has no camera node as a neighbor. Adjacent matrix X' and Y' are computed without unnecessary nodes, and using the procedure shown later. It might be better to include the unnecessary nodes in the diagram from the beginning. Since the risk of committing an error will be higher as the diagram becomes larger, we include the unnecessary nodes from the beginning and remove them at the end.

3.3. Algorithm to determine neighbor nodes

The diagram of Fig. 9 is expressed in Fig. 10 as graph. In Fig. 10, the adjoining nodes are connected with the line. Table 1 is the adjacent matrix X which is a table representation based on Fig. 10, and Table 2 is the temporary adjacent matrix Y which is a table representation based on Fig. 10 before satisfying all conditions. Table 3 is the adjacent matrix Y whose conditions are satisfied after all.

Table 1 is the adjacent matrix X consisted of p rows q columns, in which $|A| = p$ and $|V| = q$. Element x_{ij} is defined as Eq. (1) in adjacent matrix X .

Table 1
Adjacent matrix X with camera nodes and non-camera nodes

X	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
a_1	1	0	0	0	0	0	1	1	0	0	0	0	0
a_2	0	1	0	0	0	0	1	0	0	0	0	0	0
a_3	0	0	1	0	0	0	0	1	0	0	0	0	0
a_4	0	0	0	1	0	0	0	0	1	0	0	0	0
a_5	0	0	0	0	1	0	0	0	0	1	0	0	0
a_6	0	0	0	0	0	1	0	0	0	0	1	0	0

Table 2
Adjacent matrix Y with non-camera nodes and non-camera nodes

Y	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
v_1	0	0	0	0	0	0	1	0	0	0	0	0	0
v_2	0	0	0	0	0	0	1	0	0	0	0	0	0
v_3	0	0	0	0	0	0	0	1	0	0	0	0	0
v_4	0	0	0	0	0	0	0	0	1	0	0	0	0
v_5	0	0	0	0	0	0	0	0	0	1	0	0	0
v_6	0	0	0	0	0	0	0	0	0	0	1	0	0
v_7	1	1	0	0	0	0	0	1	0	0	0	0	0
v_8	0	0	1	0	0	0	1	0	0	0	0	0	1
v_9	0	0	0	1	0	0	0	0	0	0	0	0	1
v_{10}	0	0	0	0	1	0	0	0	0	0	0	1	0
v_{11}	0	0	0	0	0	1	0	0	0	0	0	1	0
v_{12}	0	0	0	0	0	0	0	0	0	1	1	0	1
v_{13}	0	0	0	0	0	0	0	1	1	0	0	1	0

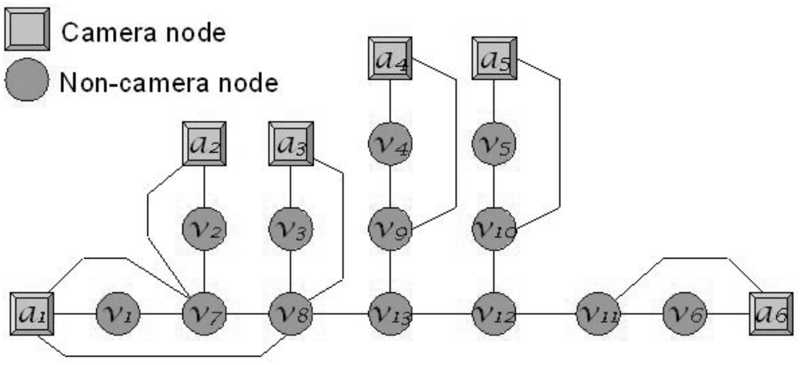

Fig. 10. Graph of Fig. 9.

$$x_{ij} = \begin{cases} 1 \textit{ There is the line which links camera} \\ \quad \textit{node } a_i \\ \quad \textit{and non-camera node } v_j. \\ 0 \textit{ Other.} \end{cases} \qquad (1)$$

Table 2 is the adjacent matrix Y which consists of q rows q columns, in which $|V| = q$. Element y_{ij} is defined as Eq. (2) in adjacent matrix Y before satisfying all conditions.

$$y_{ij} = \begin{cases} 1 \textit{ There is the line which links camera} \\ \quad \textit{node } v_i \\ \quad \textit{and non-camera node } v_j. \\ 0 \textit{ Other.} \end{cases} \qquad (2)$$

Consider the problem of (c) in Fig. 8 with Fig. 9. Video cameras, a_2 and a_3, have the same case of overlapped view problem in Fig. 9. Carefully observing the non-camera nodes, v_7 and v_8, the nodes are adjacent with more than one cameras. The summation of column v_7 in X and the summation of column v_8 in X are larger than 1, if the adjacent matrix X is used. If the summation is larger than 1, the number of cameras around the non-camera node is more than 1. In addition, $y_{78} = y_{87} = 1$ also indicates that v_7 is a neighbor of v_8, if the adjacent matrix Y is used. In this case, the neighbor relationship of non-camera nodes v_7 and v_8 can be separated to resolve the overlapped view

Table 3
Adjacent matrix Y with non-camera nodes and non-camera nodes

Y	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
v_1	0	0	0	0	0	0	1	0	0	0	0	0	0
v_2	0	0	0	0	0	0	1	0	0	0	0	0	0
v_3	0	0	0	0	0	0	0	1	0	0	0	0	0
v_4	0	0	0	0	0	0	0	0	1	0	0	0	0
v_5	0	0	0	0	0	0	0	0	0	1	0	0	0
v_6	0	0	0	0	0	0	0	0	0	0	1	0	0
v_7	1	1	0	0	0	0	0	0	0	0	0	0	0
v_8	0	0	1	0	0	0	0	0	0	0	0	0	1
v_9	0	0	0	1	0	0	0	0	0	0	0	0	1
v_{10}	0	0	0	0	1	0	0	0	0	0	0	1	0
v_{11}	0	0	0	0	0	1	0	0	0	0	0	1	0
v_{12}	0	0	0	0	0	0	0	0	0	1	1	0	1
v_{13}	0	0	0	0	0	0	0	1	1	0	0	1	0

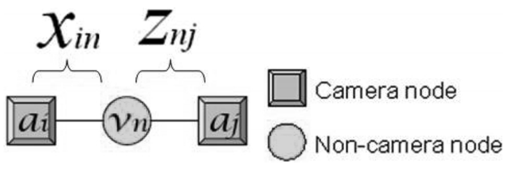

Fig. 11. Two camera nodes via one non-camera node.

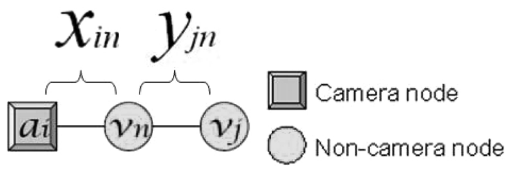

Fig. 12. Camera node and non-camera node via one non-camera node.

problem. Non-camera nodes v_7 and v_8 can be separated if it is assumed $y_{78} = y_{87} = 0$.

If the conditions Eq. (3) are satisfied, then element y_{ij} and element y_{ji} are replaced as $y_{ij} = y_{ji} = 0$.

$$\begin{cases} \sum_{n=1}^{m} x_{ni} > 1 \\ \sum_{n=1}^{m} x_{nj} > 1 \\ y_{ij} = y_{ji} = 1 \end{cases} \tag{3}$$

Table 3 is the adjacent matrix Y that satisfies the conditions Eq. (3) and resolves the problem of (c) in Fig. 8. Summation of column of v_7 and summation of column of v_8 in adjacent matrix X are lager than 1. Therefore y_{78} and y_{87} are replaced by 0.

Matrix Z is a transposed matrix X . It is possible to consider that x_{ij} indicates neighbor from camera node a_i to non-camera node v_j, and z_{ji} indicates neighbor from non-camera node v_j to camera node a_i.

Next, considering the case as shown in Fig. 11, the camera node a_i is neighbor to camera node a_j via non-camera node v_n. The relation between camera node a_i and non-camera node v_n is treated as $x_{in} = 1$, and the relation between non-camera node v_n and camera node a_i is treated as $z_{nj} = 1$. In addition, considering camera node a_i can reach camera node a_j via non-camera node v_n, then $x_{in} \times z_{nj} = 1$ can be derived. Therefore, the arithmetic expression indicates the relation between camera a_i and camera a_j, and it is possible to state that camera node a_i is neighbor to camera node a_j. If it is assumed that the element b_{ij} of adjacent matrix B indicates the relation between camera node a_i and camera node a_j, the arithmetic expression via non-camera node $v_n (n = 1, ..., m)$ is possible to be expressed as Eq. (4). However, if the values $i = j$, it makes $b_{ij} = 0$, because neighbor relation as $i = j$ indicates neighbor to camera itself.

$$b_{ij} = \sum_{n=1}^{m} x_{in} z_{nj} \begin{cases} \geqslant 1 & a_i \textit{ is adjacent to } a_j \\ = 0 & a_i \textit{ is not adjacent to } a_j \end{cases} \tag{4}$$

Considering the case in Fig. 12, camera node a_i is neighbor to non-camera node v_j via non-camera node v_n. The relation between camera node a_i and v_n is treated as $x_{in} = 1$, and the relation between non-camera node v_n and non-camera node v_j is treated as $y_{nj} = 1$. And y_{jn}, an element of transposed matrix Y can be represented as $y_{nj} = y_{jn} = 1$. In addition, considering camera node a_i can reach non-camera node v_j via non-camera node v_n, $x_{in} \times y_{nj} = x_{in} \times y_{jn} = 1$ can be derived. Therefore, the arithmetic expression indicates the relation between camera node a_i and non-camera node v_j, and it is possible to define that camera node a_i is a neighbor to non-camera node v_j. If it is assumed that adjacent matrix element c_{ij} indicates relation between camera node a_i and non-camera node v_j, the arithmetic expression via non-camera node $v_n (n = 1, ..., m)$ is possible to be expressed as Eq. (5).

$$c_{ij} = \sum_{n=1}^{m} x_{in} y_{nj} \begin{cases} \geqslant 1 & a_i \textit{ is adjacent to } v_j \\ = 0 & a_i \textit{ is not adjacent to } v_j \end{cases} \tag{5}$$

Table 4
Unnecessary nodes in adjacent matrix X

X	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
a_1	1	0	0	0	0	0	1	1	0	0	0	**0**	**0**
a_2	0	1	0	0	0	0	1	0	0	0	0	**0**	**0**
a_3	0	0	1	0	0	0	0	1	0	0	0	**0**	**0**
a_4	0	0	0	1	0	0	0	0	1	0	0	**0**	**0**
a_5	0	0	0	0	1	0	0	0	0	1	0	**0**	**0**
a_6	0	0	0	0	0	1	0	0	0	0	1	**0**	**0**

Table 5
Adjacent matrix X'

X'	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
a_1	1	0	0	0	0	0	1	1	0	0	0
a_2	0	1	0	0	0	0	1	0	0	0	0
a_3	0	0	1	0	0	0	0	1	0	0	0
a_4	0	0	0	1	0	0	0	0	1	0	0
a_5	0	0	0	0	1	0	0	0	0	1	0
a_6	0	0	0	0	0	1	0	0	0	0	1

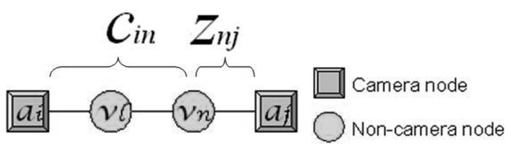

Fig. 13. Two camera nodes via two non-camera nodes.

Considering the case in Fig. 13, the camera node a_i is a neighbor to camera node a_j via two non-camera nodes, v_l and v_n. If it is assumed that the element of adjacent matrix D is d_{ij}, it is possible to derive the expression (6) applying the result of Fig. 12.

$$d_{ij} = \sum_{n=1}^{m} c_{in} z_{nj} \begin{cases} \geqslant 1 & a_i \textit{ is adjacent to } a_j \\ = 0 & a_i \textit{ is not adjacent to } a_j \end{cases} \tag{6}$$

From the above results, when it calculates adjacent matrix E via n or more nodes, it can use the expression (7).

$$E = X(Y)^{n-1}X^T \begin{cases} \geqslant 1 & a_i \textit{ is adjacent to } a_j \\ = 0 & a_i \textit{ is not adjacent to } a_j \end{cases} \tag{7}$$

When It is considered the value of n on the expression (7), it is difficult to decide whether or not the non-camera nodes are n, especially when it is in a loop, as shown in Fig. 14. If there is no existence of a loop of non-camera nodes, the problem will not occur. Then adjacent matrix X' and Y' are computed so that a diagram may be constituted from camera nodes and non-camera nodes without unnecessary non-camera nodes. Figure 15 is the graph re-calculated from Fig. 9.

3.4. Eliminating of unnecessary non-camera node

Consider the adjacent matrix X' and Y' computed without unnecessary non-camera nodes. Unnecessary non-camera nodes are nodes that are not connected from any camera nodes. The matrix is calculated with the following procedure.

In the case of the adjacent matrix X , the procedure is stated below:

- Search unnecessary non-camera node v_n in which camera node is not neighbor.
- Remove the column of the node v_n.

The adjacent matrix X' is computed from the adjacent matrix X without the unnecessary nodes.

When the data on Table 1 is considered as an example, the unnecessary non-camera nodes are highlighted as shown on Table 4. The columns v_{12} and v_{13} represent the unnecessary non-camera nodes, and adjacent matrix X' becomes as Table 5.

In the case of adjacent matrix Y , non-camera node v_n is extracted to compute adjacent matrix X' . The procedure is stated below:

- Determine the column of unnecessary node v_n from the adjacent matrix X .
- Perform an OR operation on the column of unnecessary node v_n and the columns of all the neighbor nodes of v_n on adjacent matrix Y .
- Perform an OR operation on the row of unnecessary node v_n and the rows of all the neighbor nodes of v_n on adjacent matrix Y .
- Remove the row and the column of the unnecessary node v_n from the adjacent matrix Y .

Table 6
Unnecessary nodes in adjacent matrix Y

Y	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
v_1	0	0	0	0	0	0	1	0	0	0	0	**0**	**0**
v_2	0	0	0	0	0	0	1	0	0	0	0	**0**	**0**
v_3	0	0	0	0	0	0	0	1	0	0	0	**0**	**0**
v_4	0	0	0	0	0	0	0	0	1	0	0	**0**	**0**
v_5	0	0	0	0	0	0	0	0	0	1	0	**0**	**0**
v_6	0	0	0	0	0	0	0	0	0	0	1	**0**	**0**
v_7	1	1	0	0	0	0	0	0	0	0	0	**0**	**0**
v_8	0	0	1	0	0	0	0	0	0	0	0	**0**	**1**
v_9	0	0	0	1	0	0	0	0	0	0	0	**0**	**1**
v_{10}	0	0	0	0	1	0	0	0	0	0	0	**1**	**0**
v_{11}	0	0	0	0	0	1	0	0	0	0	0	**1**	**0**
$\mathbf{v}_{12}$	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**
$\mathbf{v}_{13}$	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**

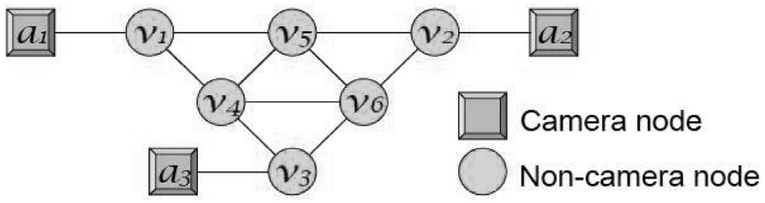

Fig. 14. Graph that non-camera nodes are looped.

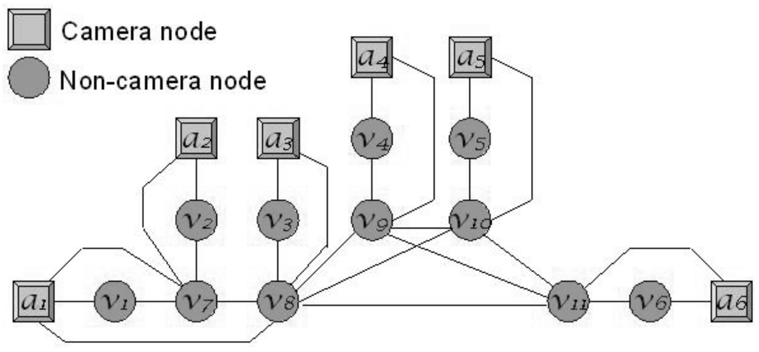

Fig. 15. Graph without unnecessary non-camera nodes.

The adjacent matrix Y' is computed from the adjacent matrix Y without the unnecessary nodes.

When Table 3 is considered as an example, the unnecessary non-camera nodes are highlighted as shown on Table 6. The inherited result is Table 7. The rows and columns of v_{13} and v_{14} represent the unnecessary non-camera nodes. And the adjacent matrix Y' becomes Table 8. It makes $y'_{ij}(i = j)$ to 0, because neighbor relation as $i = j$ indicates neighbor to itself.

3.5. *Neighbor node matrix*

It is possible to derive neighbor node matrix using expression (8) from the result of the foregoing paragraphs. However, if $i = j$ then the value of e_{ij} in matrix E is replaced as 0 because neighbor relation $i = j$ indicates being neighbor to camera itself. The matrix is the neighbor node information.

$$E = X'(Y')^{n-1}X'^T \begin{cases} \geqslant 1 & a_i \textit{ is adjacent to } a_j \\ = 0 & a_i \textit{ is not adjacent to } a_j \end{cases} \quad (8)$$

4. Bypass methods

Improving the robustness of the system is one of the important factors to consider when designing a system. Though increasing the numbers of hardware is

Table 7
Inherited result from Table 6

Y	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
v_1	0	0	0	0	0	0	1	0	0	0	0	**0**	**0**
v_2	0	0	0	0	0	0	1	0	0	0	0	**0**	**0**
v_3	0	0	0	0	0	0	0	1	0	0	0	**0**	**0**
v_4	0	0	0	0	0	0	0	0	1	0	0	**0**	**0**
v_5	0	0	0	0	0	0	0	0	0	1	0	**0**	**0**
v_6	0	0	0	0	0	0	0	0	0	0	1	**0**	**0**
v_7	1	1	0	0	0	0	0	0	0	0	0	**0**	**0**
v_8	0	0	1	0	0	0	0	**1**	**1**	**1**	**1**	**1**	1
v_9	0	0	0	1	0	0	0	**1**	**1**	**1**	**1**	**1**	1
v_{10}	0	0	0	0	1	0	0	**1**	**1**	**1**	**1**	1	**1**
v_{11}	0	0	0	0	0	1	0	**1**	**1**	**1**	**1**	1	**1**
$\mathbf{v}_{12}$	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	1	1	**1**	1
$\mathbf{v}_{13}$	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	1	**1**

Table 8
Adjacent matrix Y'

Y'	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
v_1	0	0	0	0	0	0	1	0	0	0	0
v_2	0	0	0	0	0	0	1	0	0	0	0
v_3	0	0	0	0	0	0	0	1	0	0	0
v_4	0	0	0	0	0	0	0	0	1	0	0
v_5	0	0	0	0	0	0	0	0	0	1	0
v_6	0	0	0	0	0	0	0	0	0	0	1
v_7	1	1	0	0	0	0	0	0	0	0	0
v_8	0	0	1	0	0	0	0	1	1	1	1
v_9	0	0	0	1	0	0	0	1	1	1	1
v_{10}	0	0	0	0	1	0	0	1	1	1	1
v_{11}	0	0	0	0	0	1	0	1	1	1	1

an effective approach like a duplex system, the cost of implementation will become a burden of expense. An approach utilizing bypass methods is an efficient approach to improve the robustness of the automatic human tracking system. Considering the case when there is a sudden beakdown on the feature extraction server, the operation of the mobile agent still has to continue to operate. The bypass methods provide such ability to the system, thus the robustness of the system is improved.

"Bypass" is an act to avoid an obstacle and reach the destination. When a mobile agent bypasses an obstacle, the mobile agent utilizes neighbor node information. Therefore, the "algorithm to determine neighbor nodes" is utilized by the mobile agent. There are two kinds of problems on where to consider bypass in the automatic human tracking system, ie. logical problems and physical ones. In the case of logical problem, the problem occurs inside the system such as "broken server". The physical problem is a problem that occurs outside the system like the instance of "pathway being modified".

Two types of bypass methods which utilize the "algorithm to determine neighbor nodes" are described.

"Recalculation Bypass Method", a method on where the mobile agent recalculates neighbor node information except broken camera nodes. The method's feature is that recalculated neighbor node information is simple and mobile agent can bypass broken camera nodes utilizing the information.

"Additional Calculation Bypass Method", a method on where a mobile agent searches next neighbor nodes with a coefficient as $n \geqslant 3$ in the Eq. (8). The method's feature is that the mobile agent can find next neighboring camera nodes via n non-camera nodes and mobile agent is possible to utilize the information when a pathway is being modified.

4.1. Recalculation bypass method

The "algorithm to determine neighbor nodes" is utilized in Recalculation Bypass Method. The broken camera node problem is the situation when a mobile agent can not transfer to the destination camera node. In this case the mobile agent will recalculate surrounding neighbor nodes except the broken camera nodes.

Considering the case that the mobile agent transfers when a broken camera node occurs, there are two situations on this kind of problem. One situation is when a camera node is already broken before a mobile agent transfers. The other situation is when a camera node is broken during the transfer of the mobile agent. However, such stated problems have no difference on the side of the mobile agent. It is considered as failures of transfer on the mobile agent. Therefore these should be treated as one single situation.

Then, consider the method of increasing reliability. In order to increase the reliability of a system, the broken camera node is to be separated from the system and the system is to be reconstructed dynamically. If agent management server controls the reconstruction of the automatic human tracking system, the mobile agent will inquire the system status to the agent management server. In other words, if a mobile agent can not communicate with the agent management server, the mobile agent will not transfer. As such the mobile agent will resolve this problem independently.

Finally, consider the necessity of the broken camera node information. It is a waste that a mobile agent is transferred to the broken camera node. If the broken camera node information is excepted from adjacent matrix, the calculation time becomes shorter. Therefore the broken camera node information should be removed from the adjacent matrix.

Due to the above statements, the recalculation bypass method was developed to resolve the broken camera node problems. Procedure is as follows.

1. Generate new adjacent matrix X' .
   (a) Remove the row of the broken camera node from the adjacent matrix X .
   (b) Remove the column of the unnecessary node v_n from the adjacent matrix X if the total sum of column v_n is 0.
2. Generate new adjacent matrix Y' .
   (a) Determine the column of unnecessary node v_n from the adjacent matrix X .
   (b) Perform an OR operation on the column of unnecessary node v_n and the columns of all the neighbor nodes of v_n on adjacent matrix Y .
   (c) Perform an OR operation on the row of unnecessary node v_n and the rows of all the neighbor nodes of v_n on adjacent matrix Y .
   (d) Remove the row and the column of the unnecessary node v_n from the adjacent matrix Y .
3. Calculate neighbor node information with a coefficient of $n = 2$ utilizing the Eq. (8)
4. Transfer the mobile agents to the neighbor nodes utilizing the result of the above calculation.

4.2. *Additional calculation bypass method*

The “algorithm to determine neighbor nodes” is utilized in Additional Calculation Bypass Method. A mobile agent calculates the neighbor node with a coefficient of $n \geqslant 3$ in Eq. (8).

It is assumed that a temporary detour was prepared for the modification of the pathway. For instance, a temporary pathway through a room was established. When the remodeled area does not obstruct the view of cameras or the temporary pathway is covered by the view of the camera at both ends, the problem of mobile agent transfer will not occur.

A mobile agent will fail to track the target entity if the pathway is not between the camera nodes to which the mobile agent transfers next. In such a situation when the pathway of the detour is clearly recognized in advance, Recalculation Bypass Method can be utilized. However, if a pathway of the detour is suddenly established as shown in Fig. 16 and no system user is aware of it, Additional Calculation Bypass Method will be utilized to resolve the problem. Object (a) of Fig. 16 shows a common situation. Object (b) of Fig. 16 shows the situation that a temporary detour is prepared. When the view of the camera is interrupted by the remodeled area, the same problem will occur.

Therefore, Additional Calculation Bypass Method is developed to resolve the above problems. This method can compute neighbor camera node via n next non-camera nodes. The procedure is as follows.

1. Calculate the neighbor nodes with a coefficient of $n \geqslant 3$ utilizing the Eq. (8)
2. Transfer the mobile agents to the neighbor nodes except the broken nodes utilizing the result of the above calculation.

5. Examination

Through the examination of the two methods, it was verified whether the mobile agent can track a target entity.

Detailed results of the examination are described in the following sub sections.

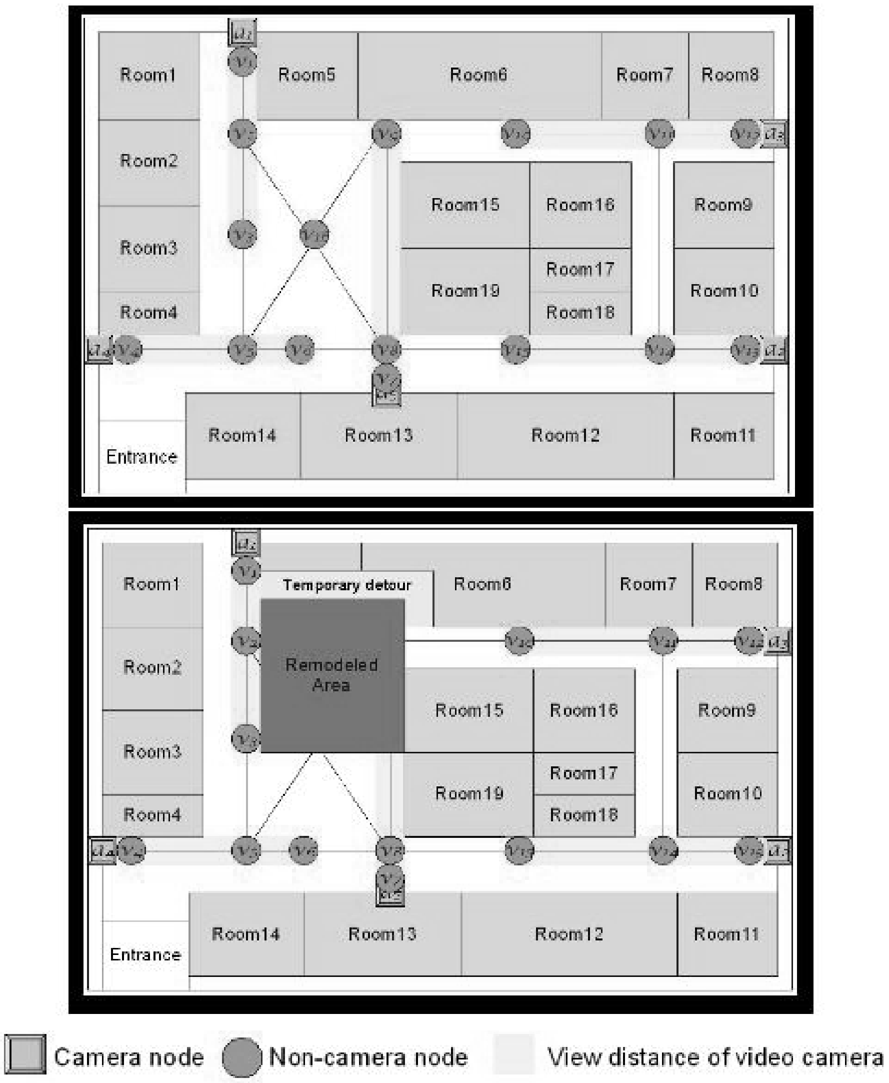

Fig. 16. Temporary detour.

5.1. Examination results of recalculation bypass method

When one camera node is broken, its node becomes invalid. In such case, the influence by the view distance of the camera changes. This is an influence problem by the difference of view distance of the camera. The problem consists of an overlap of view, an interrupt of view, and a clear view. In the case of that the view of a camera node is clear, even if the camera node is broken, the camera node should be eleminated from neighbor node information. But in the other two cases, the problem become complicated so that the neighbor node information changes. It is the same even if more than one server are broken at a time. The recalculation bypass method is examined in order to verify the effectiveness of the method in these cases.

The conditions of the examination are described below.

- More than one camera nodes, a_1 and a_3 as shown in Fig. 17, are broken at a time.
- A camera node with the overlapping view is broken and a camera node with the interrupted view is broken.
- A target entity is moving from camera node a_6 to camera node a_2

Adjacent matrix X' and Y' are calculated from the above conditions. The calculated execution time is

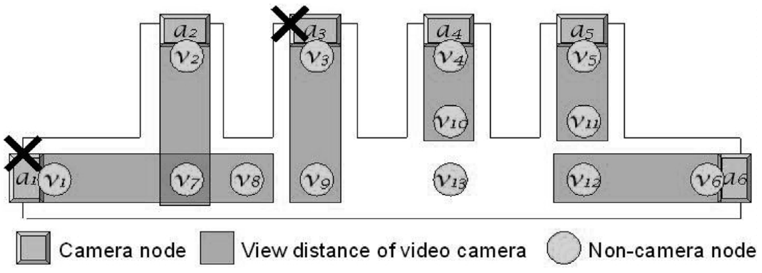

Fig. 17. The case of that plural cameras are broken.

Table 9
Adjacent matrix X'

X'	v_2	v_4	v_5	v_6	v_7	v_{10}	v_{11}	v_{12}
a_2	1	0	0	0	1	0	0	0
a_4	0	1	0	0	0	1	0	0
a_5	0	0	1	0	0	0	1	0
a_6	0	0	0	1	0	0	0	1

Table 10
Adjacent matrix Y'

Y'	v_2	v_4	v_5	v_6	v_7	v_{10}	v_{11}	v_{12}
v_2	0	0	0	0	1	0	0	0
v_4	0	0	0	0	0	1	0	0
v_5	0	0	0	0	0	0	1	0
v_6	0	0	0	0	0	0	0	1
v_7	1	0	0	0	0	1	0	1
v_{10}	0	1	0	0	1	0	0	1
v_{11}	0	0	1	0	0	0	0	1
v_{12}	0	0	0	1	1	1	1	0

Table 11
Neighbor node information E

E	a_2	a_4	a_5	a_6
a_2	0	1	0	1
a_4	1	0	0	1
a_5	0	0	0	1
a_6	1	1	1	0

within 5 milliseconds. Result of the computation is shown in Tables 9 and Table 10. Result of the computation of neighbor node information, E except a_1 and a_3 is shown in Table 11. a_2, a_4 and a_5 are candidate destinations except a_1 and a_3 from a_6 to a_2.

The result shows that neighbor node information is derived correctly, even if more than one camera nodes are broken. After examining the results, the mobile agent can continuously track a target entity in the automatic human tracking system.

5.2. *Examination results of additional calculation bypass method*

If a pathway of the detour is suddenly established as shown in Fig. 16, the environment to track a target entity is changed. The case is the same when the view of the camera is blocked by an obstacle. It is inefficient to change the system setting for such sudden events. An influence problem by the difference of view distances consists of an overlap of view, an interrupt of view, and a clear view as described before. In the case that the view of a camera node is clear, there will be no influence even if a remodeling occurs in front of the camera node. But in the other cases, a problem occurs so that the view distance of camera changes. It is the same even if the remodeling occurs when more than one cameras are influenced. The additional calculation bypass method is examined in order to verify the effectiveness of the method in these cases.

The conditions of the examination are described below.

– Remodeling area to influence more than one camera nodes, a_7, a_8 and a_9 occurs as shown in Fig. 18.
– The pathway between non-camera node v_1 and v_{13} is modified as shown in Fig. 18.
– A temporary detour is established near the pathway.
– A target entity is moving from camera node a_6 to camera node a_2

Adjacent matrix X' and Y' are calculated from the above conditions. The calculated execution time is within 5 milliseconds. Result of the computation is shown in Tables 12 and Table 13. Result of the computation of neighbor node information, E is shown in Table 14 utilizing the Eq. (8). From E, neighbor nodes of a_1 are a_2, a_3, a_4 and a_6. Neighbor nodes of a_2 are a_1 and a_3. Neighbor nodes of a_3 are all nodes. Neighbor nodes of a_4 are a_1, a_3, a_5 and a_6. Neighbor nodes of a_5 are a_3, a_4 and a_6. Neighbor nodes of a_6 are a_1, a_3, a_4 and a_5.

The result shows that neighbor node information is derived correctly, even if the view distances of cameras are changed by the remodeled area. After examining the results, a mobile agent can continuously track a target entity in the automatic human tracking system.

Table 12
Adjacent matrix X'

X'	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
a_1	1	0	0	0	0	0	1	1	0	0	0	0
a_2	0	1	0	0	0	0	1	0	0	0	0	0
a_3	0	0	1	0	0	0	0	0	1	0	0	0
a_4	0	0	0	1	0	0	0	0	0	1	0	0
a_5	0	0	0	0	1	0	0	0	0	0	1	0
a_6	0	0	0	0	0	1	0	0	0	0	0	1

Table 13
Adjacent matrix Y'

Y'	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
v_1	0	0	0	0	0	0	1	0	0	0	0	0
v_2	0	0	0	0	0	0	1	0	0	0	0	0
v_3	0	0	0	0	0	0	0	0	1	0	0	0
v_4	0	0	0	0	0	0	0	0	0	1	0	0
v_5	0	0	0	0	0	0	0	0	0	0	1	0
v_6	0	0	0	0	0	0	0	0	0	0	0	1
v_7	1	1	0	0	0	0	0	1	0	0	0	0
v_8	0	0	0	0	0	0	1	0	1	0	0	0
v_9	0	0	1	0	0	0	0	1	0	1	0	1
v_{10}	0	0	0	1	0	0	0	0	1	0	0	1
v_{11}	0	0	0	0	1	0	0	0	0	0	0	1
v_{12}	0	0	0	0	0	1	0	0	1	1	1	0

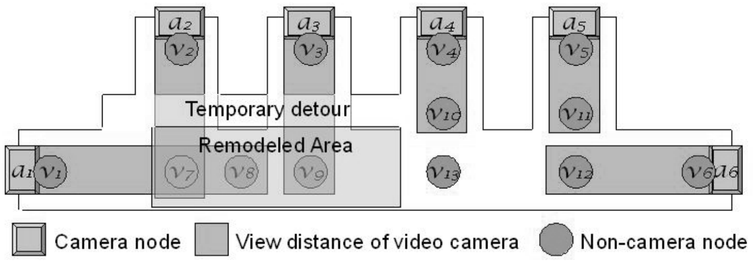

Fig. 18. The case of construction.

Table 14
Neighbor node information E

E	a_1	a_2	a_3	a_4	a_5	a_6
a_1	0	5	2	1	0	1
a_2	5	0	1	0	0	0
a_3	2	1	0	3	1	3
a_4	1	0	3	0	1	3
a_5	0	0	1	1	0	2
a_6	1	0	3	3	2	0

6. Conclusion

The algorithm utilized in this paper is simple and calculable at high speed considering an execution time within 5 milliseconds for the adjacent matrix X as 6×13 and the adjacent matrix Y as 13×13. This algorithm was able to compute the neighbor node information at high-speed in the automatic human tracking system. And the algorithm contributed to improve the processing performance and reliability of the system. If the algorithm is utilized in the case that neighbor node information is influenced by some events, the bypass methods are efficient.

From the experimental results, two kinds of bypass methods were effective in bypassing a broken camera node and a pathway being modified, even if the broken camera nodes influenced the neighbor node information and the view distance of camera was changed by the remodeling. A mobile agent was able to continue tracking a target entity by using the methods even if these events occurred. The robustness of the automatic human tracking system is improved by the efficient bypass methods aimed to solve the problems caused by the change of view distances. The algorithm will be utilizable to a similar problem as a view distance influence problem.

In addition, the simulator was effective to simulate a target entity. In other examination, mobile agents was able to track numerous number of entities at the same time by using the bypass methods in the automatic human tracking system with the simulator mentioned before. But at the present, the simulator needs more improvement especially when using more than 20 cameras. We will strive hard to improve the simulator.

An agent's transfer rate was from 600 milliseconds to 700 milliseconds by using HTTP protocol. This result was slightly bad in consideration to the speed of the entity. If TCP protocol was applied to the agent transfer, the rate was within 100 milliseconds and it was enough to satisfy the consideration. HTTP will be better than TCP, because HTTP will be convenient and flexible in the internet world. So, the agent transfer rate on the HTTP will be improved.

If a mobile agent understands the movement of a person, the number of the agent copy can decrease and the system resource will be reduced. In order to realize that, it is necessary to calculate the distance or the direction from the image of a video camera and there is an effective research [17]. In addition, if feature extraction function is not influenced against illuminance changes using an effective research [10], a tracking function will become more robust. Furthermore, if trouble shooting function is added in the system by using an effective research [13], the maintainability of the system will be improved. Therefore, we are considering the research to improve the automatic human tracking system. And we are considering to use the system to action analysis by adding facial expression recognition method [1].

We are researching the lost track re-detection function and quick feature extraction of the entity captured by video camera to build more advanced tracking systems. If the lost track re-detection function is combined with the bypass methods, the bypass methods will be more efficient. A lost track re-detection function will detect the target entity in the case the mobile agent loses track of the target entity. With such function, the automatic human tracking system will be more stable. As the future problem, the lost track re-detection function by utilizing the algorithm will be considered.

Acknowledgements

This work was supported in Melco Power Systems Co., Ltd. and in Tottori University. The authors would also like to acknowledge Mr. Yasuo Matsumiya, Mr. Kiyoshi Tsutsui, Mr. Masayuki Shibamoto, Mr. Mitsuo Hoshino, Mr. Michio Kojima, Mr. Kensuke Fukumoto, Ms. Mayu Tanaka, Mr. Hubert Ambales, Mr. Yusuke Hamada, and Mr. Shinya Iwasaki.

References

[1] D.-T. Lin and D.C. Pan, SIntegrating a mixed-feature model and multiclass support vector machine for facial expression recognition, *Integrated Computer-Aided Engineering* **16**(1) (2009), 61–74.

[2] D.B. Lange and M. Oshima, Seven good reasons for mobile agents, *Communications of the ACM* **42**(3) (1999), 88–89.

[3] D.G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision* **60**(2) (2004), 91–110.

[4] D. Monticolov, V. Hilaire, S. Gomes and A. Koukam, A Multi-agent System for Building Project Memories to Facilitate Design Process, *Integrated Computer-Aided Engineering* **15**(1) (2008), 3–20.

[5] G. Cabri, L. Leonardi and F. Zambonelli, Mobile-Agent Coordination Models for Internet Applications, *Computer* **33**(2) (February 2000), 82–89.

[6] G. Valetto, G. Kaiser and Gaurav S. Kc, A Mobile Agent Approach to Process-based Dynamic Adaptation of Complex Software Systems, *Lecture Notes in Computer Science* **2077/2001** (2001), 102–116.

[7] H. Kakiuchi, T. Kawamura, T. Shimizu and K. Sugahara, An Algorithm to Determine Neighbor Nodes for Automatic Human Tracking System, *2009 IEEE INTERNATIONAL CONFERENCE on ELECTRO/INFORMATION TECHNOLOGY*, Windsor, Canada, 2009, pp. 96–102.

[8] H. Kakiuchi, Y. Hamada, T. Kawamura, T. Shimizu and K. Sugahara, To realize Automatic Human Tracking System based on Mobile Agent Technologies, *Proceedings of the 59th Chugoku branch union convention of the Institute of Electrical Engineers of Japan and Information Processing Society of Japan*, Tottori, Japan, 2008, p. 485.

[9] H. Yin and I. Hussain, Independent component analysis and nongaussianity for blind image deconvolution and deblurring, *Integrated Computer-Aided Engineering* **15**(3) (2008), 219–228.

[10] Iyad Jafar and Hao Ying, New algorithms for contrast enhancement in grayscale images based on the variational definition of histogram equalization, *Integrated Computer-Aided Engineering* **15**(2) (2008), 131–147.

[11] J.C. Noyer, L. Patrick and B. Mohammed, Correlation-based particle filter for 3D object tracking, *Integrated Computer-Aided Engineering* **16**(2) (2009), 165–177.

[12] J. Pettre, T. Simeon and J.P. Laumond, Planning human walk in virtual environments, *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems, Lausanne*, Switzerland, Sept. 2002, pp. 3048–3053.

[13] K. Perusich, Using Fuzzy Cognitive Maps to Identify Multiple Causes in Troubleshooting Systems, *Integrated Computer-Aided Engineering* **15**(2) (2008), 197–206.

[14] K. Terashita, N. Ukita and M. Kidode, Efficiency Improvement of Probabilistic-Topological Calibration of Widely Distributed Active Cameras, IPSJ SIG Technical Report, 2009–CVIM–166, 2009, pp. 241–248.

[15] N. Ishida, Y. Hamada, H. Kakiuchi, T. Shimizu, T. Kawamura and K. Sugahara, Feature extraction method for Automatic Human Tracking System based on Mobile Agent Technolo-

gies, *Proceedings of the 59th Chugoku branch union convention of the Institute of Electrical Engineers of Japan and Information Processing Society of Japan*, Tottori, Japan, 2008, p. 418.

[16] N.R. Jennings, An agent-based approach for building complex software systems, *Communications of the ACM* **44**(4) (April 2001), 35–41.

[17] O. Millnert, T. Goedeme, T. Tuytelaars, L.V. Gool, A. Huntemann and M. Nuttin, Range determination for mobile robots using an omnidirectional camera, *Integrated Computer-Aided Engineering* **14**(1) (2007), 63–72.

[18] OSGi Alliance, OSGi Alliance Specifications OSGi Service Platform Release 1, http://www.osgi.org/Specifications/HomePage, 2008.

[19] R.S. Gray, G. Cybenko, D. Kotz, R.A. Peterson and D. Rus, D'Agents: Applications and performance of a mobile-agent system, *Software: Practice and Experience* **32**(6) (2002), 543–573.

[20] S. Motomura, T. Kawamura and K. Sugahara, Maglog: A Mobile Agent Framework for Distributed Models, *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, Phoenix, Arizona, USA, 2005, pp. 414–420.

[21] T. Kawamura, S. Motomura and K. Sugahara, Implementation of a Logic-based Multi Agent Framework on Java Environment, *Proceedings of IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems* (*Henry Hexmoor* (*eds.*)), Waltham, Massachusetts, USA, 2005, pp. 486–491.

[22] U.M. Erdem and S. Sclaroff, Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements, *CVIO2006* **103**(3) (2006), 156–169.

[23] Y. Cui, N. Hasler, T. Thormaehlen and H.-P. Seidel, Scale Invariant Feature Transform with Irregular Orientation Histogram Binning, *Proceedings of the International Conference on Image Analysis and Recognition*, ICIAR 2009, Halifax, Canada: Springer.

[24] Y. Hamada, S. Iwasaki, H. Kakiuchi, T. Kawamura and K. Sugahara, Pursuit methods for Automatic Human Tracking System based on Mobile Agent Technologies, *Proceedings of the 59th Chugoku branch union convention of the Institute of Electrical Engineers of Japan and Information Processing Society of Japan*, Tottori, Japan, 2008, p. 486.

[25] Y. Kawaguchi, A. Shimada, D. Arita and R. Taniguchi, Object Trajectory Acquisition with an Active Camera for Wide Area Scene Surveillance, IPSJ SIG Technical Report, 2008–CVIM–163, 2008, pp. 1306–1311.

[26] Y. Yao, C.-H Chen, B. Abidi, D. Page, A. Koschan and M. Abidi, Sensor Planning for Automated and Persistent Object Tracking with Multiple Cameras, *CVPR2008* (2008).

[27] Y. Yao, C.-H. Chen, B. Abidi, D. Page, A. Koschan and M. Abidi, Sensor Planning for PTZ Cameras Using the Probability of Camera Overload, *ICPR2008* (2008).