Proceedings of the 21st IASTED International Conference
Parallel and Distributed Computing and Systems (PDCS 2009)
November 2 - 4, 2009 Cambridge, MA, USA

BYPASS METHOD BASED ON NEIGHBOR NODE DETERMINATION ALGORITHM FOR AUTOMATIC HUMAN TRACKING SYSTEM

H. Kakiuchi
Engineering Department
Melco Power Systems Co., Ltd.
1-1-2, Wadasaki-cho, Hyogo-ku
Kobe, Hyogo, Japan
email: Kakiuchi.Hiroto@zs.MitsubishiElectric.co.jp

S. Iwasaki, T. Kawamura, and K. Sugahara
Information Electronics Engineering
Graduate School of Tottori University
4-101, Minami, Koyama-cho
Tottori-city, Tottori, Japan
email: {kawamura, sugahara}@ike.tottori-u.ac.jp

ABSTRACT
This paper proposes two bypass methods to track a suspicious person by mobile agent technologies. The proposed methods are ”Recalculation Bypass Method” and ”Additional Calculation Bypass Method”. The proposed methods utilizes the algorithm which was elaborated in the paper, ”An Algorithm to Determine Neighbor Nodes”[1]. The algorithm is efficient in determining the next destination node of the mobile agents. The mobile agents is operating on automatic human tracking system which is enhanced by video surveillance system with mobile agent technologies. In the automatic human tracking system, the video camera and the server computer is treated as a single node. The video camera captures a suspicious person and pass the information to the server computer which it extracts the feature of the person. If a server computer is broken, then a mobile agent will loose track of the suspicious person. In the automatic human tracking system it is very important for the mobile agent not to lose track. On such case the bypass methods resolve two types of problem to ensure the tracking ability of the mobile agent. First problem is a logical problem which occurs inside the system an example would be a ”broken server” Second problem is a physical problem which occurs outside the system an example would be a ”pathway being modified”. By utilizing the methods, the system can continuously track the suspicious person even on the uncertain cases.

KEY WORDS
Bypass Method, Mobile Agent, Human Tracking System, Adjacent Matrix.

1 Introduction

In recent years, video surveillance system is utilized by companies and public offices. Existing video surveillance system is efficient in locating suspicious people. However existing systems still has limited capabilities thus enhancement is still required. On one case it can not be able to locate a numerous number of people at a time and automatically locate a particular person. In a conventional video surveillance system the users are involved in a heavy workload. Since the user will be mostly involved in switching from one video camera to the other, the number of suspicious people to locate is limited. Identifying a person from a numerous surveillance position also increases the workload of the user. In such cases it demands the user to carefully identify the person.

Some researches suggests solutions that solve the problems. On one approach, in order to track a person the use of a pan/tilt camera[2][3] is needed, thus the camera will have to move in a synchronized motion. However on this approach locating numerous people would be difficult. On another approach, efficiently positioning a camera at a particular surveillance station[4][5][6] was implemented. On this approach in order to thoroughly install the camera the process will need large resources. Such approach is not applicable to an establishment with limited resources.

On these situations, identifying numerous people through an image processing would be a better approach and video camera can be installed easily. However it would be inappropriate to use a single server when locating numerous people, since the image processing will increase server load. As such, a new type of system that will identify and locate people is proposed. On this system, the implementation of the mobile agent technologies[7][8] is realized as one mobile agent per person. The mobile agent technologies are suitable for distributed and parallel processing, since mobile agent can transfer the copy agents to predetermined server in the system. The mobile agent technologies are more effective than the conventional video surveillance system, 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 video cameras. The video surveillance system enhanced with mobile agent technologies is called ”Automatic Human Tracking System”[9][10]. In this system, a mobile agent tracks a person captured by a video camera. Since the video camera along with the server computer is deployed at the surveillance position, the video camera and the server are treated as a single entity. When a person is to be tracked, a mobile agent is generated for the particular person. The mobile agent tracks the movement of the person utilizing the neighbor camera/server location

information, after verifying the feature of the person within the server[11]. The neighbor camera/server location information is determined utilizing the ”Algorithm to Determine Neighbor Nodes”.

The system is effective in tracking numerous people at the same time. However this system has a problem in which the agent can not continuously track the person. There are two situations that agent can not transfer to the next location, these are the logical and the physical problem. On the case of a logical problem, this occurs inside the system an example would be a ”broken server” And on the case of a physical problem, this occurs outside the system an example would be a ”pathway being modified”. As such we propose these bypass methods to resolve the above problems.

This paper is organized in 6 sections. Overviews of automatic human tracking system and neighbor node determination algorithm are introduced in Section 2 and 3. The bypass method is described in Section 4. Then examination for the method is described in Section 5. Finally, in Section 6, concluding remarks is described.

2 Automatic Human Tracking System

The system configuration of the automatic human tracking system is shown in Fig.1. It assumes that the system is used in a building, the user captures an image of the face and the body of a person by the video camera, the user registers the image into the system, and roaming person with unregistered image can be recognized 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 person, for confirming the current location staying the agent, and for displaying video of the captured person. The Agent Management server records the agent’s tracking information, provides the information to the terminal and requests video image to the Video Recording server by an operation from the terminal. The Video Recording server records all video images and provides the images to the terminal by request from the Agent Management Server. The Feature Extraction server sets up by the video camera, analyzes the person image and extracts the feature information from the image. An agent tracks the person by using the feature information and the neighbor nodes information.

The processing flow of the proposed system is the following:

1. The Video Recording server records video images from all video cameras at all time.
2. The user selects the person on the screen of the Agent Monitoring terminal, and extracts the tracked person’s feature information as electronic.
3. A mobile agent is generated for the person and registered as the mobile agent information including the feature information into the system.
4. The mobile agent deployed on the first Feature Extraction server begins pursuing the person.
5. When the mobile agent finds the person, the mobile agent notifies the Agent Management server of the information such as the video camera number, the discovery time, and the mobile agent identifiers first to the Agent Management server.
6. When the person moves to the area of the next video camera, the mobile agent transfers to the next video camera near the person.
7. If the mobile agent finds the person, the mobile agent notifies the Agent Management server of the information as stated on step (5).

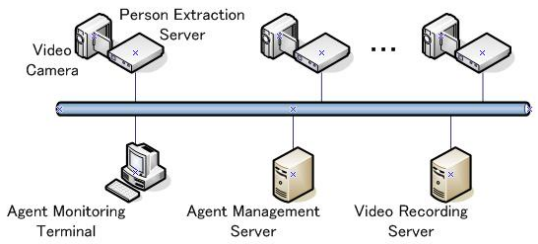

Figure 1. System Configuration

The mobile agent repeats the above process from step (5) to (6) until the person goes out of a building.

3 Neighbor Node Determination Algorithm

3.1 A Problem on Determining Neighbor Video Cameras

A scenario in which neighbor video camera location is influenced by view distance is described in Fig.2. Fig.2 shows 3 diagrams portraying a floor plan with 4 video cameras each. And view distances of each video camera are different. It assumes that a person to be pursued starts to move from the location of video camera A. The neighbor of video camera A in object (a) of Fig.2 is video camera B and, not C and D. In object (a) of Fig.2, 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 pursued person can be captured at an earlier time on video camera B. But in the case of object (b) of Fig.2, 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 by view distance of video camera. The case of object (c) in Fig.2 is more complicated. The neighbors of video camera A in object (c) of Fig.2 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

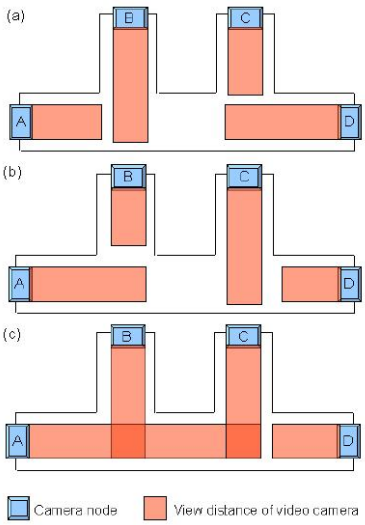

Figure 2. Example that view distance influences

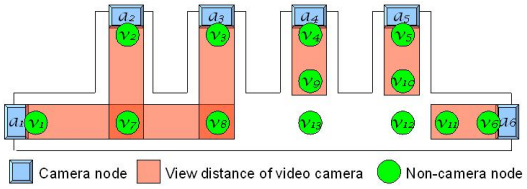

Figure 3. Figure that sets non-camera nodes

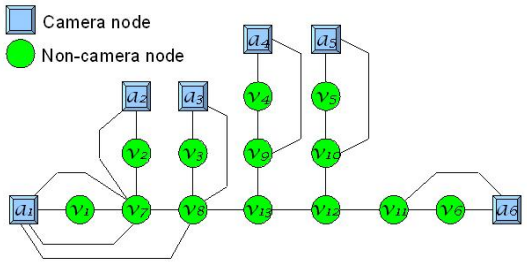

Figure 4. Graph of Fig.3

camera B and C. When it is assumed that a person move to D from A, the person is sure to be captured by video cameras in order of video camera A, B, A, and C.

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

3.2 Overview of the Algorithm

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

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. Fig.3 shows the example of applying these definitions and shows view distance of video camera.

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' location as rows and non-camera nodes' location as columns. Another one is adjacent matrix Y made from non-camera nodes' location 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 special condition. Therefore, when the non-camera node that crosses each other the view distance of two or more video camera exists 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 neighbor nodes:

1. Set camera nodes and non-camera nodes on the diagram.

2. Generate adjacent matrix X from camera nodes' location and non-camera nodes' location, 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 distance between video cameras.

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	0	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

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 would 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 Determination Algorithm

The diagram of Fig.3 is expressed in Fig.4 as graph. In Fig.4, adjoining nodes are connected with the line. Table 1 is the adjacent matrix X that was digitalized based on Fig.4 and Table 2 is the temporary adjacent matrix Y that was digitalized based on Fig.4 before satisfying all conditions. Table 3 is the adjacent matrix Y which 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 (1) in adjacent matrix X .

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

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

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

Consider the problem of (c) in Fig.2 with Fig.3. Video cameras, a_2 and a_3, have same problem in Fig.3. When paying attention to non-camera nodes, v_7 and v_8, here, these nodes are adjacent with plural cameras. This consists equations that summation of column v_7 in X and summation of column v_8 in X are larger than 1, if the adjacent matrix X is used. And $y_{78} = y_{87} = 1$ also consists because neighbor of v_7 is v_8, if the adjacent matrix Y . In this case, the neighbor relationship of non-camera nodes, v_7 and v_8, can be cut, and the neighbor relationship of camera nodes, from a_1 to a_3, and non-camera nodes, v_7 and v_8, can be left if it is assumed $y_{78} = y_{87} = 0$.

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

$$\begin{cases} \sum_{n=1}^{m} x_{ni} \geq 1 \\ \sum_{n=1}^{m} x_{nj} \geq 1 \\ y_{ij} = y_{ji} = 1 \end{cases} \quad (3)$$

Table 3 is the adjacent matrix Y that satisfies the conditions (3) and resolved the problem of (c) in Fig.2. Summation of column of v_7 and summation of column of v_8 each other in adjacent matrix X are lager than 1. Therefore y_{78} and y_{87} are replaced by 0.

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

Figure 5. Two camera nodes via one non-camera node

Figure 6. Camera node and non-camera node via one non-camera node

Matrix Z is made from 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. Therefore, $x_{ij} \times z_{ji} = 1$.

Next considering the case as shown on Fig.5, 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 consists of $x_{in} = 1$ and the relation between non-camera node v_n and camera node a_i consists of $z_{nj} = 1$. In addition, it is considered that $x_{in} \times z_{nj} = 1$ consists, if camera node a_i can reach camera node a_j via non-camera node v_n. Therefore, it is possible to derive the relation between camera a_i and camera a_j with the above arithmetic expression, and it is possible to define 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 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 (4). However, if $i = j$ consists, then it makes $b_{ij} = 0$ because neighbor relation as $i = j$ indicates neighbor to camera itself.

$$b_{ij} = \sum_{n=1}^{m} x_{in} z_{nj} \begin{cases} \geq 1 & a_i \text{ is adjacent to } a_j \\ = 0 & a_i \text{ is not adjacent to } a_j \end{cases} \quad (4)$$

Considering the case in Fig.6, 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 consists of $x_{in} = 1$ and the relation between non-camera node v_n and non-camera node v_j consists of $y_{jn} = 1$ and transposed y_{jn} is also 1. In addition, it is considered that $x_{in} \times y_{jn} = 1$ consists, if camera node a_i can reach non-camera node v_j via non-camera node v_n. Therefore, it is possible to derive the relation between camera node a_i and non-camera node v_j with the above arithmetic expression, 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

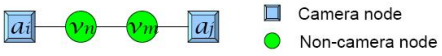

Figure 7. Two camera nodes via two non-camera nodes

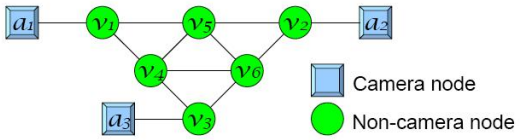

Figure 8. Graph that non-camera nodes are looped

as (5).

$$c_{ij} = \sum_{n=1}^{m} x_{in} y_{jn} \begin{cases} \geq 1 & a_i \text{ is adjacent to } v_j \\ = 0 & a_i \text{ is not adjacent to } v_j \end{cases} \quad (5)$$

Considering the case on Fig.7, the camera node a_i is a neighbor to camera node a_j via two non-camera nodes, v_n and v_m.

If it is assumed that the element of adjacent matrix D is d_{ij}, it is possible to derive (6) under applying the result of Fig.6.

$$d_{ij} = \sum_{n=1}^{m} c_{in} z_{nj} \begin{cases} \geq 1 & a_i \text{ is adjacent to } a_j \\ = 0 & a_i \text{ is not adjacent to } a_j \end{cases} \quad (6)$$

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

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

When It is considered the value of n on the above (7), it is difficult to decide whether or not the non-camera nodes are n, and in the case where it belongs to a loop, as shown in Fig.8. If there is no existence of a loop, 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. Fig.9 is the graph re-calculated from Fig.3.

3.4 Removing of Unnecessary Non-Camera Node

Consider the adjacent matrix X' , 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:

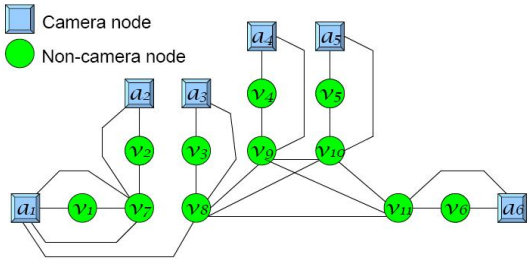

Figure 9. Graph without unnecessary non-camera nodes

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

- Searches unnecessary non-camera node v_n in which camera node is not neighbor.
- Removes 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 will be highlighted as shown on Table 4. The columns v_{12} and v_{13} represent the unnecessary non-camera nodes, and adjacent matrix X' become 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 unnecessary non-camera node v_n from adjacent matrix X .
- Perform an OR operation on adjacent matrix Y using the column of unnecessary non-camera node v_n from adjacent matrix X and the columns of the neighbor nodes v_n.
- Perform an OR operation on adjacent matrix Y using the row of unnecessary non-camera node v_n from adjacent matrix X and the row of the neighbor nodes v_n.

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

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
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

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
v_{12}	0	0	0	0	0	0	0	1	1	1	1	1	1
v_{13}	0	0	0	0	0	0	0	1	1	1	1	1	1

- Remove the row and column of the node v_n.

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 will be 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.

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

3.5 Neighbor Node Matrix

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

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

4 Bypass Methods

Two types of bypass methods are described. First method is called "Recalculation Bypass Method", mobile agent recalculates neighbor node information excepting broken camera nodes and utilizing the "Algorithm to Determine Neighbor Nodes". The features of this method are: recalculated neighbor node information which is very simple and mobile agent can bypass broken camera nodes utilizing the information. Second method is called "Additional Calculation Bypass Method", mobile agent searches next neighbor nodes with a coefficient as $n \geq 3$ in the equation(8) utilizing the "Algorithm to Determine Neighbor Nodes". The features of this method are: 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.

It is important to consider situations on where these methods can be used efficiently. Then, consider the situation where mobile agent need a bypass. First case is on the shutdown event or the failure event of a camera node and network. And on second case, the situation of the pathway being modified.

Considering the first case. It is separated in two cases: camera node problems and network problems. In the case of camera node problems, there are two kinds of situation.

- Camera node is broken before agent will transfer.
- Camera node is broken while agent is transferring.

However, as for mobile agent, the above situations seems to be the same. Because the difference between the above situation is not understood by the agent during the failure of transfer. Therefore the situation on where the camera node failed while the agent is transferring is considered to be the same as in the situation of a failed camera node.

In the case of network problems, it is one important element to determine how mobile agent re-detect a lost track, when the failed network returns normally. Because mobile agent can not transfer anywhere in the case of failed network. When the failed network returns normally, mobile agent can continue tracking the person on the case the mobile agent can detect the person on the camera node. However, on the case that the mobile agent cannot detect the person it has to re-detect at the previous camera node to continue tracking. Therefore, in this case bypass methods is not applicable because it is considered as a topic of other research and research unto it is being done.

Then, Recalculation Bypass Method is applied only in the first sub-case as described in the following.

Next, considering the second case. It is assumed that a temporary detour was prepared for the modification of the pathway. For instance, it is assumed that a pathway through a room is prepared temporarily. When the pathway is between the camera nodes to where a mobile agent transfer, a problem will not occur. A mobile agent will fail to track the suspicious person if the pathway is not between the camera nodes unto which the mobile agent transfers next. In such a situation, when the pathway of the detour is clearly recognized, Recalculation Bypass Method can be utilized. However, if a pathway of the detour is suddenly established and no system user is aware about it, then Additional Calculation Bypass Method is utilized. On this method it can compute neighbor camera node via n next non-camera nodes.

4.1 Recalculation Bypass Method

Recalculation Bypass Method is the method being utilized by the mobile agent to recalculate the neighbor node without the broken camera node utilizing the "Algorithm to Determine Neighbor Nodes". The 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 adjacent matrix Y using the column of unnecessary node v_n from the adjacent matrix X and the columns of the neighbor nodes of v_n.
   (c) Perform an OR operation on adjacent matrix Y using the row of unnecessary node v_n from the adjacent matrix X and the rows of the neighbor nodes of v_n.
   (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 equation as shown in (8)
4. Transfer the mobile agents to the neighbor nodes utilizing the result of the above calculation.

4.2 Additional Calculation Bypass Method

Additional Calculation Bypass Method is the method being utilized by the mobile agent to search next neighbor nodes with a coefficient of $n \geq 3$ as shown in the equation (8) and

Table 9. Adjacent Matrix X'

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

Table 10. Adjacent Matrix Y'

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

utilizing the "Algorithm to Determine Neighbor Nodes". The procedure is as follows.

1. Calculate the neighbor nodes with a coefficient of $n \geq 3$ utilizing the equation as shown in (8)
2. Transfer the mobile agents to the neighbor nodes except the broken node utilizing the result of the above calculation.

5 Examination

In the examination of the two kinds of method, it was verified whether the mobile agent can track a suspicious person.

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

5.1 Examination Results of Recalculation Bypass Method

The condition of the examination is described below.

- Camera node a_1 in Fig.3 is broken.
- A suspicious person is moving from camera node a_6 to camera node a_2

Adjacent matrix X' and Y' is calculated from the above conditions. Result of the computation is shown in Table 9 and Table 10.

Result of the computation of neighbor node information E excepting a_1 is shown in Table 11.

a_3, a_4 and a_5 are candidate destinations except a_1 from a_6 to a_2 based from the result. Examining the results a mobile agent can continuously track a suspicious person in the automatic human tracking system.

Table 11. Neighbor Node Information E

E	a_2	a_3	a_4	a_5	a_6
a_2	0	1	0	0	0
a_3	1	0	1	1	1
a_4	0	1	0	1	1
a_5	0	1	1	0	1
a_6	0	1	1	1	0

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}
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

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_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

5.2 Examination Results of Additional Calculation Bypass Method

The condition of the examination is described below.

- The pathway between non-camera node v_7 and v_8 in Fig.3 is a pathway being modified.
- A temporary detour is established near the pathway.
- A suspicious person is moving from camera node a_6 to camera node a_2

Adjacent matrix X' and Y' is calculated from the above conditions. Result of the computation is shown in Table 12 and Table 13.

Result of the computation of neighbor node information E is shown in Table 14 utilizing the equation as shown in (8) .

a_1, a_3, a_4 and a_5 are candidate destinations from a_6 to a_2 base from the result. It is possible to assume that a_1 is adjacent to a_6 when a detour is established between v_7 and v_8. Examining the results a mobile agent can continuously track a suspicious person in the automatic human tracking system.

Table 14. Neighbor Node Information E

E	a_1	a_2	a_3	a_4	a_5	a_6
a_1	0	3	4	3	3	3
a_2	3	0	0	0	0	0
a_3	4	0	0	4	4	4
a_4	3	0	4	0	4	4
a_5	3	0	4	4	0	4
a_6	3	0	4	4	4	0

6 Conclusion

On the situations where a camera node is broken and a pathway is being modified, a mobile agent can continuously track the suspicious person utilizing the bypass methods.

Furthermore on the case of more than two broken camera nodes and tracking of more than two suspicious people was simulated in automatic human tracking system simulator software which was developed. The simulator software was developed and intended to simulate large count of camera nodes in the automatic human tracking system.

Based from the test result, two kinds of bypass methods are efficient in bypassing a broken camera node and a pathway being modified. In addition mobile agents can track numerous number of people at the same time by using the bypass method in the automatic human tracking system.

We are researching the lost track re-detection function and quick feature extraction of the person captured by video camera as a consideration in a more advanced tracking system. 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 suspicious person being tracked on the case the mobile agent lost track of the person. With such function the automatic human tracking system will be more stable.

In the near future, we will propose the lost track re-detection function.

References

[1] 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.

[2] 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.

[3] Y. Kawaguchi, A. Shimada, D. Arita, 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.

[4] 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.

[5] 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.

[6] U.M. Erdem, S. Sclaroff, Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements, *CVIO2006*, Vol.103, No.3, 2006, pp. 156–169.

[7] 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.

[8] 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.

[9] 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, pp. 485.

[10] 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, pp. 486.

[11] 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 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, pp. 418.