Real-time Multi-target Tracking by Cooperative Distributed Active Vision Agents

Norimichi Ukita and Takashi Matsuyama
Graduate School of Informatics, Kyoto University
Yoshidahonmachi, Sakyo, Kyoto, Japan
souhaku@vision.kuee.kyoto-u.ac.jp and tm@i.kyoto-u.ac.jp

ABSTRACT

This paper presents a real-time cooperative multi-target tracking system. The system consists of a group of Active Vision Agents (AVAs), where an AVA is a logical model of a network-connected computer with an active camera. All AVAs cooperatively track their target objects by dynamically interacting with each other. As a result, the system as a whole can track multiple moving objects simultaneously under complicated dynamic situations in the real world. To implement the real-time cooperation among AVAs, we designed a three-layered interaction architecture. In each layer, parallel processes mutually exchange various information for the effective cooperation. To realize the information exchange in real time, we employed the dynamic memory architecture[6]. Experimental results demonstrate that AVAs cooperatively track their target objects in real-time while adaptively changing their roles.

Categories and Subject Descriptors

I [**4**]: 8

General Terms

Design

1. INTRODUCTION

Object tracking is one of the most important and fundamental technologies for realizing real-world vision systems (e.g., visual surveillance systems, ITS (Intelligent Transport System) and so on). Although a large number of works about object tracking have been reported, we first propose a real-time *multi-target* tracking system with *multi-active* cameras.

We propose a real-time cooperative tracking system that gazes at multiple targets simultaneously. To realize real-time flexible tracking in a wide-spread area, the system consists of communicating *Active Vision Agents* (AVAs, in short), where an AVA is a logical model of a network-connected computer with an active camera. A group of spatially distributed AVAs enable continuous wide-area observation as well as detailed measurement of 3D object information. For real-time tracking by multiple AVAs, we have to solve many problems (e.g., how to design an active camera for dynamic object detection[1] and how to realize real-time object tracking with an active camera[6]). In this paper, we put our focus upon how to realize a real-time cooperation among AVAs.

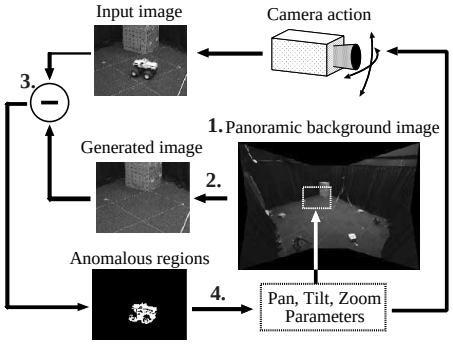


Figure 1: Object detection and tracking using an FV-PTZ camera.

In order to implement the real-time cooperation among AVAs, we propose a three-layered interaction architecture. In each layer, parallel processes exchange different kinds of information for effective cooperation. To realize a real-time information exchange and processing, we employ the dynamic memory architecture proposed in [6]. The dynamic interaction in each layer allows the total system to track multiple moving objects under complicated dynamic situations in the real world.

Experimental results demonstrate that the proposed real-time cooperation method enables the system to 1) successfully acquire the dynamic object information and 2) adaptively assign the appropriate role to each AVA.

2. COOPERATIVE OBJECT TRACKING

2.1 Architecture of AVA and Its Functions

Each AVA possesses a single *Fixed-Viewpoint Pan-Tilt-Zoom* (FV-PTZ) camera[1]: its projection center stays fixed irrespectively of any camera rotations and zoomings. With pan-tilt-zoom cameras, we aim at designing a system that can not only track trajectories of targets but also acquire their detailed information.

By employing the property of FV-PTZ camera, an AVA

AAMAS'02 July 15–19, 2002, Bologna, Italy.

can track a moving object as illustrated in Fig.1:

1. Generate a wide panoramic image of the scene; with the FV-PTZ camera, a wide panoramic image can be easily generated by mosaicing multiple images observed by changing pan, tilt and zoom parameters.
2. Extract a window image from the panoramic image according to the current pan-tilt-zoom parameters and regard it as the background image; the direct mapping exists between the position in the panoramic image and pan-tilt-zoom parameters of the camera.
3. Compute difference between the generated background image and an observed image.
4. If anomalous regions are detected in the difference image, select one and control the camera parameters to track the selected target. Otherwise, move the camera along the predefined trajectory to search for an object.

A camera is coupled to a network-connected computer. This network is not a special close network (e.g., PC cluster) but an unreliable open network.

In general, there are two kinds of agents:

Software agent is a virtual agent without any physical body in the real world. Each agent corresponds to a logical data (e.g., the information of the detected object) in the system (see [3], for example).

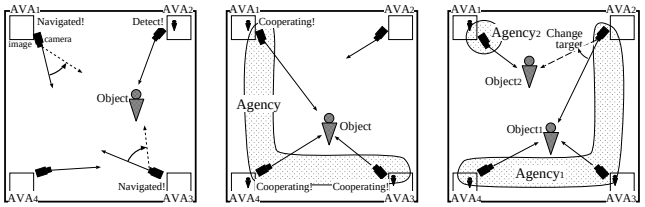
Real-world agent is an agent with its own physical body (e.g., an active camera and a mobile robot) that can be controlled by itself (see [4], for example).

We believe that an intelligent system has to possess its own body to mutually interact with the real world. We, therefore, define an *agent* to correspond to each physical body in the real world.

Although several object tracking systems with multi-camera and multi-agent systems are reported, most of them employ only software agents. In [2, 5], 1) a software agent is defined to correspond to the information of each object detected by the system and 2) all cameras are shared by software agents, each of which manages the information of each detected object. These definitions force each software agent to examine the object information detected by all cameras for tracking its target. Besides this technological problem, the above definitions have an essential limitation: multiple software agents may control a camera inconsistently in tracking their targets (i.e., controlling pan, tilt and zoom parameters), if the system employed active cameras. This results in difficulty for a camera to gaze at multiple objects simultaneously. Accordingly, the above essential limitation is fatal for realizing the system which we aim at.

In our system, on the other hand, an agent (i.e., AVA) corresponds to a single active camera. Each AVA can, therefore, control its own camera to gaze at its target. As we can see, our definition of the agent has the advantage in that it has the one-to-one correspondence between the agent and the camera.

In addition, in our system also, the information of each target should be managed intensively to 1) record the information of each object severally and 2) compare the information of different objects with each other. To realize these functions, a software agent, which has the one-to-one correspondence with an object, gathers its target information detected by AVAs. In our system, AVAs that track the same object form a group called an agency, and a software agent corresponding to each agency works as the entity of the agency (will be mentioned later).



(a) (b) (c)

Figure 2: Basic scheme for cooperative tracking: (a) Gaze navigation, (b) Cooperative gazing, (c) Adaptive tracking.

2.2 Basic Scheme for Cooperative Tracking

In our system, many AVAs are embedded in the real world and observe a wide area. We impose the following constraints about the camera configuration on the system:

- Visual fields of cameras are overlapping with each other in order to keep tracking a target in the observation scene without a break.
- In addition, all the observation spaces can be observed by at least two cameras. This is because every space has to be observed by multiple cameras to reconstruct 3D information of an object.

Under this restriction, we can determine the camera configuration arbitrarily.

With these AVAs, we realize a multi-AVA system that cooperatively tracks multiple targets. Following are the tasks of the system:

1. Initially, each AVA independently searches for an object that comes into the observation scene.
2. If an AVA detects a target, it navigates the gazes of other AVAs towards the target (Fig.2 (a)).
3. AVAs, which gaze at the same object, track the focused target cooperatively (Fig.2 (b)). A group of AVAs that track the same object is called an *Agency*.
4. Depending on the target motion, each AVA dynamically changes its target (Fig.2 (c)).
5. When the target gets out of the scene, each AVA decides whether it searches for an object again or joins another agency depending on situations.

To realize the above cooperative tracking, we have to solve the following problems:

Multi-target identification: To gaze at each target, the system has to distinguish multiple objects.

Real-time and reactive processing: To cope with the dynamics in the scene (e.g., object motion), the system has to execute the process in real time and deal with the variations in the scene reactively.

Adaptive resource allocation: We have to implement a two phased dynamic resource (i.e., AVA) allocation: 1) To perform both object search and tracking simultaneously, the system has to preserve AVAs that search for new targets even while tracking targets, 2) For each target to be tracked by the AVA that is suitable for gazing at, the system has to adaptively assign AVAs to their targets.

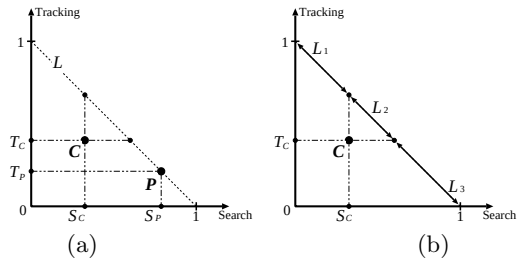


Figure 3: System state graph: (a) Current state and Task-constraint, (b) Three types of the system states.

We solve these problems with real-time cooperative communication among AVAs and agencies.

3. TASK SPECIFICATION

The tracking system need to search for an object in the scene. This role is called *Search*. Once the target is detected, the system gazes at it to obtain its information. This role is called *Tracking*. In addition, the system is also required to selectively gaze at the object whose information is necessary for the given task. We specify the task of the system by the following three parameters, namely a *Task-Constraint*, an *Object-Importance* and a *Utility-Function*.

3.1 Task-Constraint

An AVA that searches for an object is called a *Freelancer*-AVA. An AVA belonging to an agency for tracking its target is called a *Member*-AVA.

We realize various capabilities of the system, in terms of the combination of search and tracking as shown in Fig.3. We call this graph a *System State Graph*. We represent the state of the system as follows (Fig.3).

DEF. 1 (SEARCH-LEVEL, TRACKING-LEVEL). *The search-level (the horizontal axis) and the tracking-level (the vertical axis) indicate the rate of AVAs that perform search and tracking, respectively.*

$$\begin{aligned} 0 \leq \quad & \text{Search-level } (= N_F/N_A) \quad \leq 1 \\ 0 \leq \quad & \text{Tracking-level } (= N_M/N_A) \quad \leq 1, \end{aligned}$$

where N_F, N_M and N_A denote the numbers of freelancer-AVAs, member-AVAs and all AVAs, respectively.

We define the task-constraint and the current state of the system on the system state graph.

DEF. 2 (CURRENT STATE $\boldsymbol{P}(S_P, T_P)$). *This parameter ($\boldsymbol{P}$ in Fig.3) represents the search-level (S_P) and the tracking-level (T_P) at the present time. The range of $\boldsymbol{P}$ is on the line L in Fig.3 (a). That is, $(S_P + T_P)$ is always 1.*

DEF. 3 (TASK-CONSTRAINT $\boldsymbol{C}(S_C, T_C)$). *This parameter ($\boldsymbol{C}$ in Fig.3) represents the minimum search-level (S_C) and tracking-level (T_C), where $0 \leq (S_C + T_C) \leq 1$. That is, a combination of S_C and T_C is within a triangle determined by the horizontal and vertical axes and the line L in Fig.3. The system has to keep S_C and T_C while working. The system, therefore, adjusts its current state so that its current search-level and tracking-level (i.e., (S_P, T_P)) are not less than those of the task-constraint (i.e., (S_C, T_C)). The task-constraint is given by a user as a pair of constants (i.e., S_C and T_C) depending on the task of the system.*

Followings are the system states determined by the relations between the task-constraint and the current state.

Shortage of search-level: $\boldsymbol{P}$ is on L_1 in Fig.3 (b).

Task satisfaction: $\boldsymbol{P}$ is on L_2 in Fig.3 (b).

Shortage of tracking-level: $\boldsymbol{P}$ is on L_3 in Fig.3 (b).

Each AVA dynamically changes its own role between search and tracking in order to satisfy the task-constraint.

3.2 Object-Importance

The object-importance is given to each object's category that can be distinguished by the system.

DEF. 4 (OBJECT-IMPORTANCE I_P). *Let I_P denote the object-importance of the target of agency$_P$. The range of the object-importance is $0 \leq I_P \leq 1$.*

The number of the member-AVAs in agency$_P$ (denoted by M_P) is determined by the object-importance of the target: $M_P =$ (The total number of AVAs)$\times(I_P/S)$. Provided that S is the total sum of the object-importance $I_{1,\cdots,A}$, where A is the total number of the agencies. That is, the number of the member-AVAs is proportional to the object-importance of the target.

3.3 Utility-Function

Each AVA can freely change its role under the restrictions given by the task-constraint and object-importance. A guideline for the adaptive role assignment is represented by what we call the utility-function. Each AVA decides its role to increase the value of the utility-function while keeping the task-constraint and object-importance. The utility-function of our tracking system is the sum of the following search-value and tracking-value.

- **Search-value of a freelancer-AVA** is determined by a fitness of each freelancer-AVA for search.
- **Tracking-value of a member-AVA** is determined by a fitness of each member-AVA for tracking its target.

This utility-function can be designed to be adapt itself to the task given by a user[1].

4. DYNAMIC INTERACTION FOR COOPERATIVE TRACKING

In our system, parallel processes cooperatively work by dynamically interacting with each other. As a result, the system as a whole works as a tracking system. By composing the system as a group of multiple processes, we can represent various complex behaviors of the total system through the interaction between processes. Designing the total system can be, therefore, reduced to designing each process. Furthermore, the states and those transitions of the system increase enormously by combining with each other. We believe that this property allows the system to cope with complicated situations in the real world.

For the system to engage in object tracking, object identification is significant. We, therefore, classify the system into three layers depending on the types of object information employed for identification. In each layer, object

[1]We give an example in Sec.6.1.

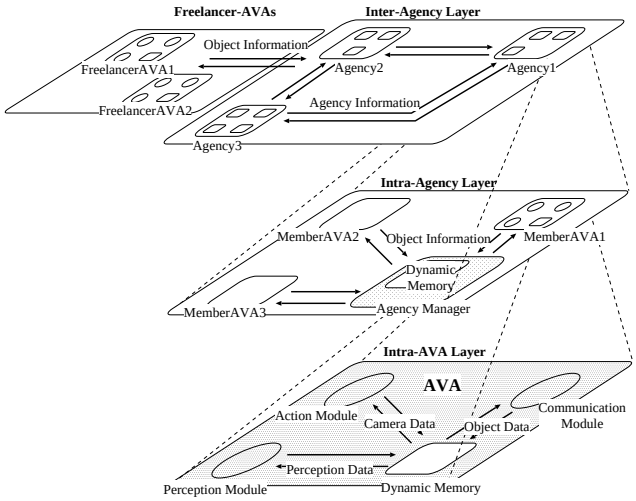


Figure 4: Three layers in the system: Intra-AVA (bottom), Intra-Agency (middle) and Inter-Agency (top) layers.

identification according to the type of exchanged information is established. Depending on whether or not object identification is successful, a dynamic interaction protocol for cooperative object tracking is activated. In what follows, we first introduce the general concept and functions for real-time interaction among processes, and address the interactions in each layer.

4.1 Dynamic Memory Architecture for Real-time Asynchronous Interaction

For real-time asynchronous interaction among parallel processes, Matsuyama *et al.*[6] proposed the *Dynamic Memory Architecture*. The dynamic memory architecture maintains not only temporal histories of state variables such as camera pan-tilt angles and target object locations but also their predicted values in the future.

In the dynamic memory architecture, multiple parallel processes share the dynamic memory. Each process writes its state variable such as pan-tilt angles of the camera and the target object location. This information is shared among all the processes through the dynamic memory. Since the shared information is written as a temporal history and shared among the processes, the time information in all the processes have to been synchronized with each other.

The read/write operations from/to the dynamic memory are defined as follows:

Write operation: When a process computes a value v of a variable at a certain moment t, it writes (v, t) into the dynamic memory. Since such computation is done repeatedly according to the dynamics of the process, a discrete temporal sequence of values is recorded for each variable in the dynamic memory.

Read operation: A reader process runs in parallel to the writer process and tries to read from the dynamic memory the value of the variable at a certain moment according to its own dynamics. Since the dynamic memory can interpolate a value at the specified moment from the recorded discrete values, the reader process can read a value at any temporal moment.

A reader process may run fast and require data which are not written yet by the writer process. In such case, the dynamic memory predicts an expected value in the future based on those data so far recorded.

4.2 Intra-AVA layer

In the bottom layer in Fig.4, perception, action and communication modules that compose an AVA exchange the time-series information with each other via the dynamic memory possessed by each AVA. The interaction among three modules materializes the functions of the AVA.

Perception: This module continues to capture images and detect objects in the observed image. Let the 3D view line L be determined by the projection center of the camera and the object region centroid in the observed image. When the module detects N objects at $t+1$, it computes and records into the dynamic memory the 3D view lines toward the objects (i.e., $L^1(t+1), \cdots, L^N(t+1)$). Then, the module compares them with the 3D view line toward its currently tracking target at $t+1$, $\widehat{L}(t+1)$. Note that $\widehat{L}(t+1)$ can be read from the dynamic memory whatever temporal moment $t+1$ specifies. Suppose $L^x(t+1)$ is closest to $\widehat{L}(t+1)$, where $x \in \{1, \cdots, N\}$. Then, the module regards $L^x(t+1)$ as denoting the newest target view line and records it into the dynamic memory.

Action: When an active camera is ready to accept a control command, the action module reads the 3D view line toward the target (i.e., $\widehat{L}(now)$) from the dynamic memory and controls the camera to gaze at the target. As will be described later, when an agency with multiple AVAs tracks the target, it measures the 3D position of the target (i.e., $\widehat{P}(t)$) and sends it to all member AVAs, which then is written into the dynamic memory by the communication module. If such information is available, the action module controls the camera based on $\widehat{P}(now)$ in stead of $\widehat{L}(now)$.

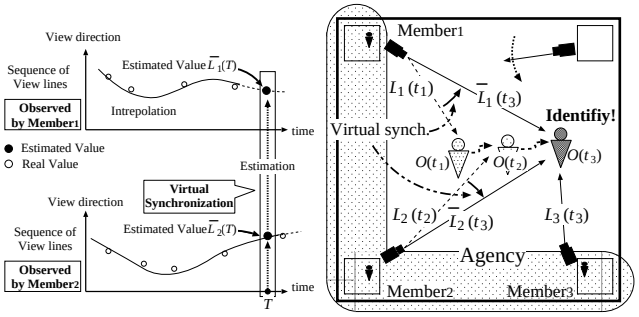
Communication: Data exchanged by the communication module over the network can be classified into two types: detected object data (e.g., $\widehat{L}(t)$ and $\widehat{P}(t)$) and messages for various communication protocols which will be described later.

To cooperatively work as stated above, perception, action and communication modules provide the following time-series information managed by each module; perception: 3D view lines of detected objects, action: camera parameters, communication: received information of objects.

4.3 Intra-Agency layer

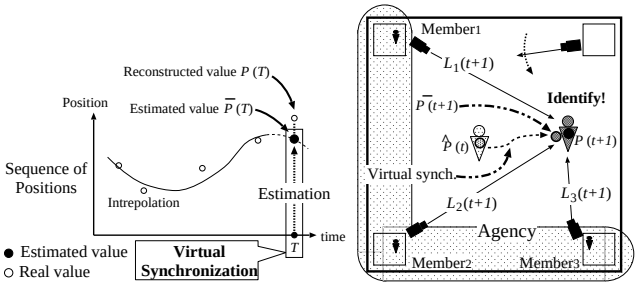
In the middle layer in Fig.4, member-AVAs belonging to the same agency exchange the information of the detected objects for object identification. In our system, an agency should correspond one-to-one to a target. To make this correspondence dynamically established and persistently maintained, the following two kinds of object identification are required in the intra-agency layer.

Spatial object identification The agency has to establish object identification between the 3D view lines detected by its member-AVAs$_{1,\cdots,M}$. $\{L^i_m(t_m)|i = 1, \cdots, N_m\}$ denotes the 3D view lines$_{1,\cdots,N_m}$ that are



(a) (b)

Figure 5: Virtual synchronization for spatial object identification: (a) Read values from the dynamic memory, (b) Spatial identification.



(a) (b)

Figure 6: Virtual synchronization for temporal object identification: (a) Read values from the dynamic memory, (b) Temporal identification.

detected by member-AVA$_m$ at t_m. If the distance between the 3D view lines detected by different AVAs is less than the threshold, these 3D view lines are considered as the information of the same object. In addition, the intersection of the identified 3D view lines is regarded as the 3D position of the object.

Temporal object identification To gaze at the target continuously, the agency compares the 3D position of the target at t ($\widehat{P}(t)$) with the 3D positions of the objects observed at $t+1$ ($\{P_i(t+1)|i = 1, \cdots, N\}$). Let $P_x(t+1)$ have the shortest distance between $\widehat{P}(t)$, where $x \in \{1, \cdots, N\}$. The agency then considers $P_x(t+1)$ as the 3D position of the target at $t+1$. Consequently, $\widehat{P}(t+1) = P_x(t+1)$.

4.3.1 Virtual Synchronization for Spatial Identification

Since AVAs capture images autonomously, member-AVAs observe the object information at different times. Furthermore, message delay via network makes an interval between capturing timings of different AVAs larger. The result of object identification is, therefore, unreliable because asynchronized object information from each camera is compared with each other.

Other distributed systems which consist of autonomous cameras coped with this problem as follows: In [2], the newest information gathered from each camera is considered to be observed at the same time. In [5, 7], the system regards the detected information observed at t_i and t_j, where $|t_i - t_j|$ is small enough, as simultaneous information. These approximate methods break down under complicated situations and network congestion.

To solve this problem, we put the dynamic memory in the intra-agency layer as we do in the intra-AVA layer. A 3D view line at any time can be estimated from time-series data recorded in the dynamic memory. We can, therefore, estimate a 3D view line observed by each AVA at the same time. We call this procedure a *Virtual Synchronization*.

Figure 5 (a) shows the virtual synchronization. In this example, the object information (i.e., the 3D view line) detected by member-AVA$_1$ and member-AVA$_2$ is written in the same dynamic memory (indicated by white points in the figure). To establish spatial object identification at T, the agency can obtain the 3D view lines detected by both member AVAs at T (denoted by $\bar{L}_1(T)$ and $\bar{L}_2(T)$, both of which are indicated by black points in the figure) by estimating the values from the dynamic memory.

In our system, spatial object identification is practically realized as follows. When an agency is formed (mentioned in Sec.*4.3.3*), an *Agency Manager* is generated at the same time. An agency manager is an autonomous module [2] independent of AVAs, and performs the following tasks as a delegate of an agency:

- Two types of object identification: identification among member-AVAs and identification with other agencies.
- Management of the dynamic memory in each agency.
- Communication with agencies and freelancer-AVAs.

That is, an agency is a conceptual group, and an agency manager is an entity of the agency. There exists one-to-one correspondence between the target (and its information) and the agency (and its agency manager). For each agency to manage the information of its target intensively, the system handle the object information as follows: 1) all the member-AVAs send the information of the detected objects to its agency manager, 2) the agency manager records its object information into the database when the agency is eliminated, and this information will be read by the newly generated agency if their targets are the same object.

When member-AVA$_m$ sends the information of the detected object (i.e., $\{L^i_m(t_m)|i = 1, \cdots, N_m\}$) to the agency manager in the same agency, the agency manager writes the received value in its dynamic memory. When spatial object identification is required, the agency manager reads the observed object information of all member-AVAs$_{1,\cdots,M}$, which are obtained by the virtual synchronization (i.e., $\{\bar{L}^i_1(T)|i = 1, \cdots, N_1\}, \cdots, \{\bar{L}^i_M(T)|i = 1, \cdots, N_M\}$). If the distance between $\bar{L}^i_p(T)$ and $\bar{L}^j_q(T)$ is small enough, the agency manager regards i-th detected result of AVA$_p$ and j-th detected result of AVA$_q$ as the 3D view lines both of which go towards the same object.

In an example illustrated in Fig.5 (b), by comparing virtually synchronized values $\bar{L}_1(t_3)$, $\bar{L}_2(t_3)$ and $L_3(t_3)$ with each other, reliable spatial object identification can be realized.

4.3.2 Virtual Synchronization for Temporal Identification

For temporal object identification, an agency manager has to compare the 3D position of its target at t with the 3D positions of the detected objects$_{1,\cdots,N}$ at $t+1$. The result of object identification is, however, unreliable because the

[2] In our system, an agency manager is implemented by a UNIX process on a PC.

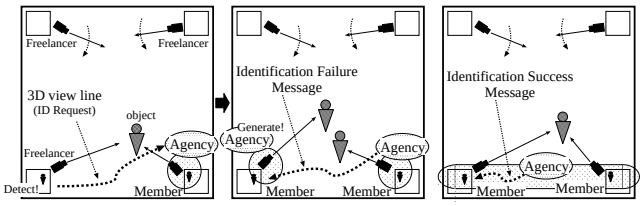


Figure 7: Agency formation protocol.

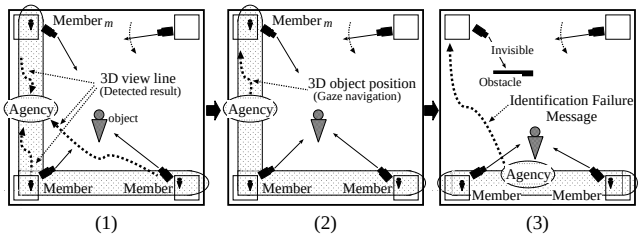


Figure 8: Agency maintenance protocol.

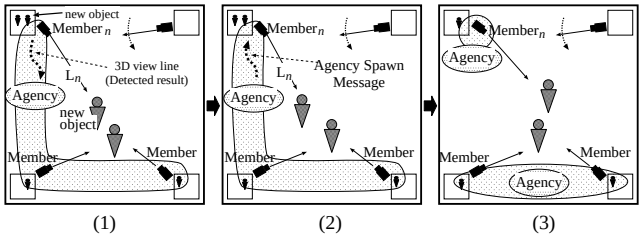


Figure 9: Agency spawning protocol.

object information obtained at different times is compared with each other.

This problem can be also solved with the dynamic memory. That is, an agency manager records the 3D position of the target (i.e., $\hat{P}(t)$) into the dynamic memory as time-series data. The agency manager can, therefore, estimate the 3D position of the target at $t+1$ (i.e., $\bar{P}(t+1)$), and compare $\bar{P}(t+1)$ with the 3D positions of the detected objects (i.e., $\{P_i(t+1)|i=1,\cdots,N\}$). $P_x(t+1)$ which has the shortest distance between $\bar{P}(t+1)$ is considered as the 3D position of the target at $t+1$. Thus, the result of temporal object identification becomes reliable.

Figure 6 shows an example of temporal object identification with the virtual synchronization. The 3D position $P(t+1)$ is reconstructed at $t+1$. The agency manager then estimates the 3D position of the target at $t+1$ (i.e., $\bar{P}(t+1)$), and compares $\bar{P}(t+1)$ with $P(t+1)$.

Depending on whether or not spatial and temporal object identifications are successful, the dynamic interactions in the intra-agency layer are activated. These dynamic interactions are defined by the following three protocols.

4.3.3 Agency Formation Protocol

An *Agency Formation* protocol defines 1) the new agency generation by a freelancer-AVA and 2) the participation of a freelancer-AVA in an existing agency.

Initially, each AVA independently searches for an object. When a freelancer-AVA finds a new object, it requests from the existing agencies object identification between the newly detected object and the target of each agency (Fig.7, (1)). Depending on whether or not the result of object identification is successful, the freelancer-AVA works as follows:

- **When no agency established a successful identification,** the freelancer-AVA that finds the new object starts a new agency manager and joins this agency (Fig.7, (2-a)).
- **When an agency established a successful identification,** the freelancer-AVA joins the agency that has made successful identification, if the system can keep the task-constraint (Fig.7, (2-b)).

4.3.4 Agency Maintenance Protocol

An *Agency Maintenance* protocol defines 1) the cooperative tracking, 2) the continuous maintenance of an agency and 3) the elimination of an agency.

After an agency is generated, the agency manager continues spatial and temporal object identifications for cooperative tracking (Fig.8, (1)). If temporal object identification between the target of the agency and the object detected by member-AVA$_m$ fails, the agency manager reports the 3D position of the target to member-AVA$_m$. This information navigates the gaze of member-AVA$_m$ towards the target (Fig.8, (2)). Nevertheless, if the failure of identification continues for a long time, the agency manager puts member-AVA$_m$ out of the agency (Fig.8, (3)).

If all member-AVAs are unable to observe the target, the agency manager eliminates the agency. All the member-AVAs then return to freelancer-AVAs.

4.3.5 Agency Spawning Protocol

An *Agency Spawning* protocol defines the new agency generation from an existing agency.

After spatial and temporal object identifications, the agency manager may find such a 3D view line(s) that does not correspond to the target. Let L_n denote such 3D view line detected by member-AVA$_n$ (Fig.9, (1)). The agency manager requires other agencies to compare L_n with their own targets for object identification. If none of the identification is successful (namely, it seems that there is not an agency that tracks the newly detected object in the system), the agency manager orders member-AVA$_n$ to generate a new agency (Fig.9, (2)). Member-AVA$_n$ then joins a new agency (Fig.9, (3)).

4.4 Inter-Agency layer

The fundamental task of an agency is to keep tracking its own target. To keep tracking the target in the complicated wide area, agencies need to adaptively exchange their member-AVAs with each other. To realize the adaptive reconstruction of the agency, the information about targets and member-AVAs are exchanged between agencies (the top layer in Fig.4). An agency that has received this information from another agency (agency$_i$) compares the 3D position of its own target with that of agency$_i$'s target. This object identification is not reliable if these 3D positions are observed at different times. This problem can be solved with the virtual synchronization in the same way as temporal object identification in the intra-agency layer. With the 3D positions of its target recorded as time-series data in the dynamic memory, the agency manager can synchronize the 3D position of its target with the received 3D position of another object.

Depending on the result of object identification between agencies, the following two protocols are activated.

4.4.1 Agency Unification Protocol

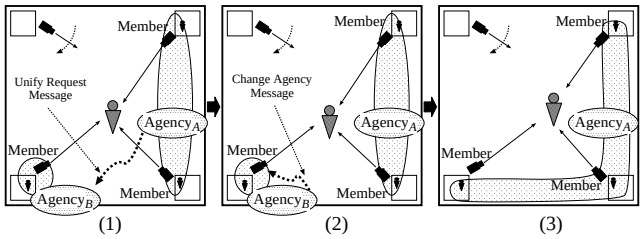


Figure 10: Agency unification protocol.

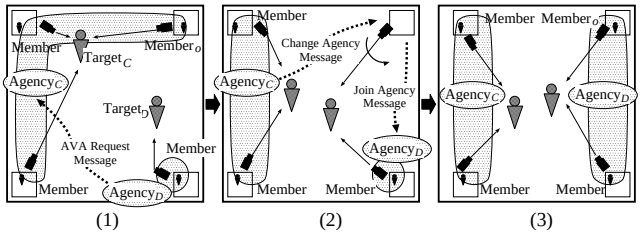


Figure 11: Agency restructuring protocol.

An *Agency Unification* protocol defines the merging procedure of the agencies, both of which happen to track the same object. This protocol is achieved when the result of object identification between the agencies is successful.

Followings are actual examples of situations that cause the agency unification.

- When the agency considers multiple objects in the scene as a single object because of the identification failure (e.g., when different objects become close enough to be identified as the same object).
- When a single object is regarded as multiple objects because of the identification failure, and then multiple agencies are formed for the same object.

That is, this protocol is required to cope with failures of object identification and discrimination.

Figure 10 shows an example. Agency manager$_A$, which has made successful object identification with agency$_B$, requests agency$_B$ to join agency$_A$ (Fig.10, (1)). Agency manager$_B$ then orders its member-AVAs to transfer to agency$_A$ (Fig.10, (2)). Agency manager$_B$ then eliminates itself. Thus, two agencies merge together (Fig.10, (3)).

4.4.2 Agency Restructuring Protocol

An *Agency Restructuring* protocol defines the dynamic interchange of member-AVAs between agencies. This protocol is achieved when the result of object identification between the agencies is unsuccessful. The agency manager performs this protocol taking into account the following two factors:

- The number of the member-AVAs is determined by the object-importance of the target.
- Under the restriction about the number, each agency is attended by AVAs, which are suitable for gazing at the target, based on the utility-function.

We have various factors in determining the fitness of each AVA for tracking, namely the criterion for the agency restructuring. A user can settle down this criterion depending on the task given to the system.

In an example illustrated in Fig.11, agency$_D$ requests a member-AVA from agency$_C$, and then agency$_C$ transfers member-AVA$_o$ to agency$_D$.

4.4.3 Communication with Freelancer-AVAs

An agency manager communicates with freelancer-AVAs as well as with other managers (the top layer in Fig.4). As described in the agency formation protocol in Sec.*4.3.3*, a freelancer-AVA activates the communication with agency managers when it detects an object. To determine whether or not generate a new agency based on the agency formation protocol, a freelancer-AVA communicates with agency managers when it detects an object. An agency manager, on the other hand, sends to freelancer-AVAs its target position when the new data are obtained. Then, each freelancer-AVA decides whether it continues to be a freelancer-AVA or joins into the agency depending on the task specification and the current state of the system. Note that in our system a user can specify the number of freelancer-AVAs to be preserved while tracking targets.

5. COMPLETENESS OF THE SYSTEM

5.1 Completeness for Persistent Tracking

We define an agency as a representation of a target in the system. Because of this definition, the maximum number of targets is equal to that of agencies. An agency has to be attended by at least one member-AVA for tracking its target. The maximum number of agencies, therefore, is the total number of AVAs in the system. In the proposed system, however, an agency reconstructs 3D information of its target from 2D information of the object observed by multiple member-AVAs. The 3D information of the target greatly assists the agency to keep tracking the target. Accordingly, each agency should have at least two member-AVAs for reliable object tracking.

Based on the above discussion, we summarize the relations between the tracking ability of the system and the numbers of targets and AVAs (denoted by n_t and n_a, respectively) as follows:

Case 1: $n_t \leq \lceil \frac{n_a}{2} \rceil$: The system can stably track all targets while obtaining their 3D information.

Case 2: $\lceil \frac{n_a}{2} \rceil < n_t \leq n_a$: Although the system can track all targets, $(n_t - \lceil \frac{n_a}{2} \rceil)$ or more targets are tracked by a single AVA.

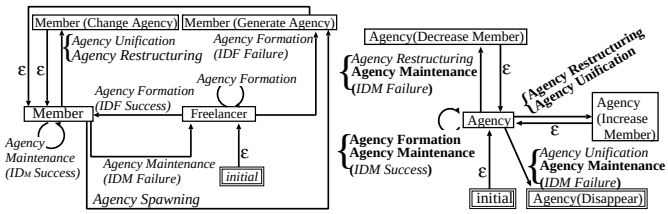
Case 3: $n_a < n_t$: $(n_t - n_a)$ or more objects cannot be tracked by the system simultaneously.

Note that $\lceil n \rceil$ denotes the maximum integer less than n.

The limitation about the number of targets results because 1) an agency receives the information of objects only from its member-AVAs and 2) an agency has to be attended by at least one member-AVA. To design alternative methods, we could modify the system as follows:

Broadcast for 3D reconstruction: If an agency receives the object information from all AVAs, it can possibly reconstruct the 3D information of the target even when it has only one member-AVA. In this case, however, each AVA has to send the information of the detected objects to all agency managers. This increases a network-load.

Camera control for vacuous agencies: If an agency can exist without any member-AVA, the system can track all targets even in the above case 3. To obtain the information of the target, a vacuous agency has to gather



(a) AVA (b) Agency

Figure 12: State transition networks of the AVA and agency. Each box and arrow indicate a state and state transition, respectively. A with each arrow causes a state transition. Protocols shown by bold and italic fonts are caused by object identification and a message that reports the result of object identification established by another agency, respectively. An automatic state transition (denoted by ϵ) occurs immediately. ID_F and ID_M denote object identification of the agency with the freelancer-AVA and member-AVA, respectively.

the object information from non-member-AVAs. We have to, therefore, solve the problem about increasing the network-load mentioned above. In addition, a vacuous agency has essential problems: since a vacuous agency cannot control any camera, it is not guaranteed that the vacuous agency 1) keeps tracking the target by controlling pan-tilt parameters of a camera(s) and 2) acquires its high-resolution image by adjusting a zoom parameter of a camera(s).

Thus, to avoid these problems, we designed the system as proposed above.

5.2 Completeness of Cooperative-tracking Protocols

In the proposed system, all events happened in the real world are characterized by the results of object identification. Therefore, by verifying the types of the protocols that are executed depending on the result of each object identification, we can confirm the necessity and sufficiency of the protocols for multi-target tracking.

All the protocols are activated by an agency, and object identification is established when the agency received the object information from freelancer-AVAs, member-AVAs and other agencies. Table 1 shows the types of the protocols that are activated according to the relations between the type of the received object information and the result of object identification. As we can see, the protocols are designed just enough in accordance with the situations in the real world.

5.3 Soundness of Communication and State Transition

In each layer, multiple parallel processes 1) dynamically exchange their information with each other for cooperation and 2) adaptively change their states. These dynamic interaction and state transition have to be realized without causing deadlock.

Intra-AVA layer: The perception, action and communication modules exchange their information through the dynamic memory in the intra-AVA layer. The dynamic memory enables the modules to asynchronously obtain the information of another process at any time.

In our system, since all modules in an AVA are implemented by threads in a single PC, band-width among modules is enough high.

Intra-agency layer: Since each agency manager has its dynamic memory, 1) asynchronous message transmission from a member-AVA to its agency manager is guaranteed and 2) reliable object identification in the intra-agency layer is realized[3].

In addition, several information (e.g., 3D position of the target and messages based on the cooperative-tracking protocols) is reported from the agency manager to its member-AVAs by the message transmission. A member-AVA accepts only the message from its agency manager not to be affected inconsistently by multiple agencies: for example, a message delay incurs the invalid communication.

Fig.12 (a) shows the state transition of the AVA. All the state transitions of the AVA are caused by the cooperative-tracking protocols except for ϵ.

Inter-agency layer: Depending on the result of inter-agency object identification, various messages are exchanged between agencies based on the inter-agency cooperative-tracking protocol. To avoid a conflict of different interactions between agencies, 1) each agency activates a protocol only with a single agency simultaneously and 2) a timeout process is utilized to cope with message delays, dynamic agency generation and elimination, and other unpredictable factors.

Fig.12 (b) shows the state transition of the agency.

6. EXPERIMENTS

We employed ten AVAs. Each AVA is implemented on a network-connected PC (PentiumIII 600MHz × 2) with an active camera (SONY EVI-G20), where perception, action, and communication modules are realized as threads. The communication module exchanges information by unreliable UDP messages. In addition, the internal clocks of all the PCs are synchronized by Network Time Protocol. With this architecture, the perception module can capture images and detect objects in the observed image at about 0.1[sec] intervals on average.

Figure 15 (a) illustrates the camera layout in the room. Camera_9 and camera_{10} are placed about 1.5m above the floor. All other cameras are placed about 2.5m above the floor. The external camera parameters are calibrated.

6.1 Designing Utility-Function

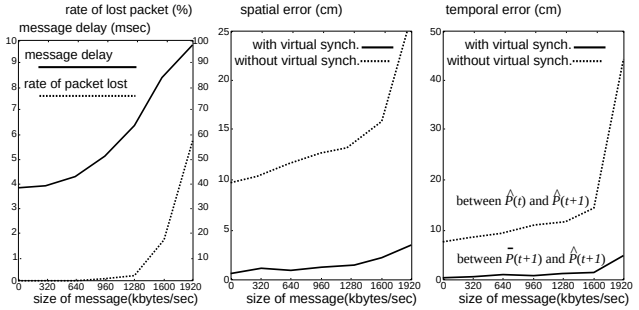
We designed the utility-function as follows:

Search-value of freelancer-AVA$_f$: Let W_f denote the area size of the floor that is visible from AVA_f. In this experiment, W_f was computed from the external parameters of camera_f (i.e., the 3D position and view direction of the camera). The search-value of AVA_f (denoted by V_{S_f}) is determined as follows: $V_{S_f} = \alpha_S \times W_f$, where α_s is a constant that is determined so that V_{S_f} is well-balanced with the tracking-value.

[3]Depending on the band-width and network-load among AVAs and agency managers, the performance of the dynamic memory changes (described in Sec.6.2).

Table 1: Protocols activated depending on the result of object identification.

Received object information	Identification success	Identification failure
3D view lines of detected objects from a freelancer-AVA	Agency Formation	Agency Formation
3D view line of the target object from a member-AVA	Agency Maintenance	Agency Maintenance and Spawning
3D view lines of non-target objects from a member-AVA	Agency Maintenance	Agency Spawning
3D point of the target object from an agency	Agency Unification	Agency Restructuring



(a) (b) (c)

Figure 13: (a) Delay of the message (solid line) and Rate of lost packet (dotted line), (b) Error in spatial object identification, (c) Error in temporal object identification.

Tracking-value of member-AVA$_m$: Let D^n_m denote the 3D distance between the camera of AVA$_m$ and the target of agency$_n$, and A^n_m denote the angle between the central direction of AVA$_m$'s view angle and the direction from the camera to the target object. The tracking-value of AVA$_m$ (denoted by $V_{T^n_m}$) is determined as follows: $V_{T^n_m} = (1)/(D^n_m) \times (1)(A^n_m)$.

6.2 Performance Evaluation

We conducted experiments with the systems with/without the virtual synchronization. To verify the effectiveness of the virtual synchronization against not only the asynchronized observations but also the network congestion, we broadcasted vain packets over the network to adjust the network load.

The system tracked two computer-controlled mobile robots. Both the robots repeated a straight-line motion at a speed of 50[cm/sec] in the observation scene.

Figure 13 (a) shows variations of network conditions when the size of the vain messages is changed. The error of spatial identification in Fig.13 (b) denotes the average distance between the reconstructed 3D position and the 3D view lines detected by member-AVAs. The error of temporal identification in Fig.13 (c) denotes the average distance between the 3D positions of the same target, each of which are reconstructed/estimated at different times (i.e., $\widehat{P}(t)/\bar{P}(t+1)$ and $\widehat{P}(t+1)$).

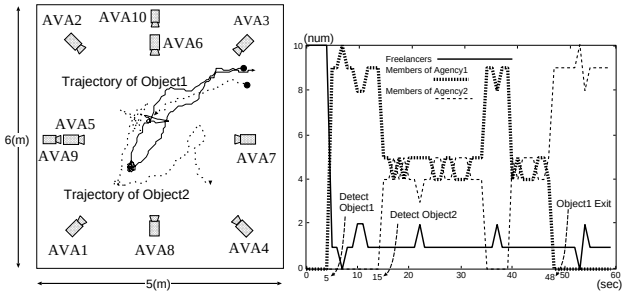
As we can see, the virtual synchronization helps both spatial and temporal object identifications, especially in the case of bad network conditions.

6.3 Verification of Cooperative Tracking Protocols

In the next experiment, the system tracked two people. Object$_1$ first came into the scene. Next, object$_2$ came into the scene. Both objects then moved freely.

Followings are the task specification of this experiment.

Task-constraint: Search-level=0.1. Tracking-level=0.9.



(a) (b)

Figure 15: (a) Trajectories of the targets, (b) The number of AVAs that performed each role.

Object-importance: The values for all objects were 1.0.

The upper part of Fig.14 shows the partial image sequences observed by AVA$_2$, AVA$_5$ and AVA$_9$. The images on the same column were taken at almost the same time. The regions enclosed by black and gray lines in the images show the detected regions of object$_1$ and object$_2$, respectively. Each figure in the bottom of Fig.14 shows the role of each AVA and the agency organization at such a moment when the same column of images in the upper part were observed. White circles denote freelancer AVAs, while black and gray circles indicate member AVAs belonging to agency$_1$ and agency$_2$, respectively. Black and gray squares indicate computed locations of target$_1$ and target$_2$ respectively.

The system worked as follows.

a: Initially, each AVA searched for an object independently.

b: AVA$_5$ first detected object$_1$, and agency$_1$ was formed.

c: All AVAs except for AVA$_5$ were tracking object$_1$ as the member-AVAs of agency$_1$, while AVA$_5$ was searching for a new object as a freelancer-AVA.

d: AVA$_5$ detected object$_2$ and generated agency$_2$.

e: The agency restructuring balanced the numbers of member-AVAs in agency$_1$ and agency$_2$.

f: Since no AVA could distinguish two objects, the agency unification protocol merged agency$_2$ into agency$_1$.

g: When the targets got apart, agency$_1$ detected a 'new' target. Then, it activated the agency spawning protocol to generate agency$_2$ again for target$_2$.

h: Object$_1$ was going out of the scene.

i: After agency$_1$ dissolved, all the AVAs except for AVA$_4$ tracked object$_2$ as the member-AVAs of agency$_2$.

Figure 15 (a) shows the trajectories of the targets reconstructed by the agencies. Figure (b) shows the dynamic population changes of freelancer AVAs, AVAs tracking target$_1$ and those tracking target$_2$.

As we can see, the dynamic cooperations among AVAs and agencies worked well and enabled the system to keep tracking multiple targets taking into account the given task.

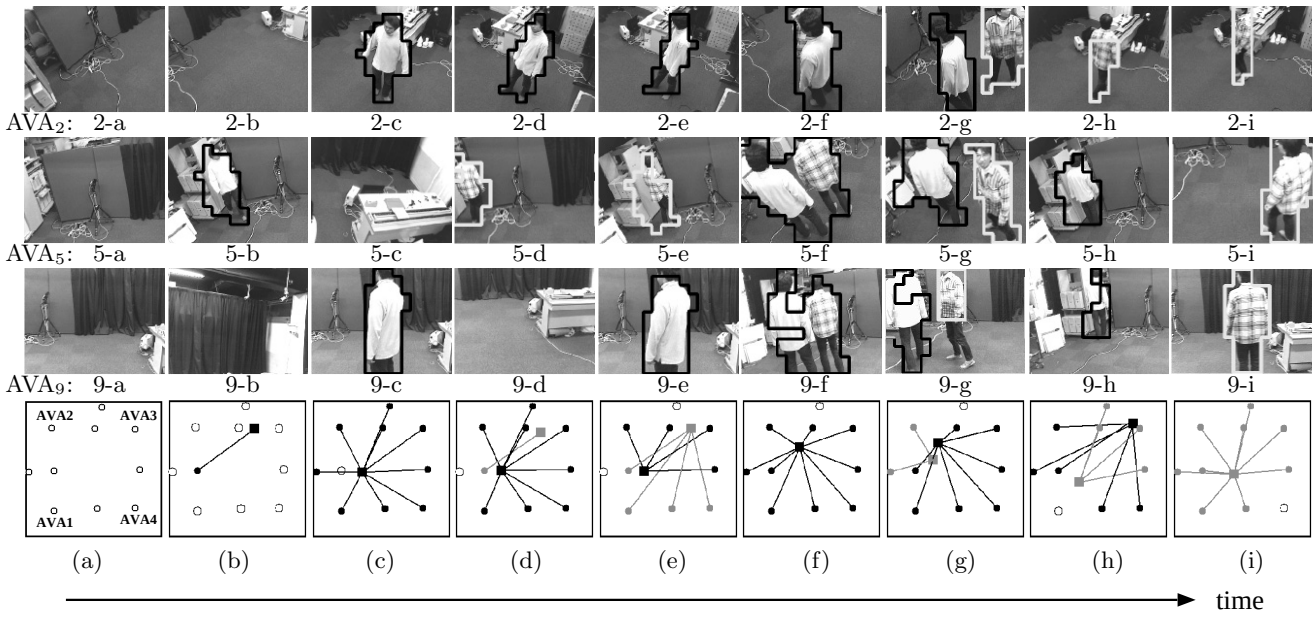


Figure 14: Upper: Partial image sequences, Lower: The role of each AVA and the agency organization

7. CONCLUDING REMARKS

This paper proposed a real-time cooperative multi-target tracking system with multiple active cameras. The system has the following properties:

- Parallel processes dynamically interact with each other, which results in the system that works as a whole for cooperative tracking.
- The system is classified into three layers to efficiently establish various types of object identification.

  Intra-AVA: Perception, action and communication modules work together as a single AVA by dynamically interacting with each other.

  Intra-Agency: AVAs in the same agency exchange object information to track the target.

  Inter-Agency: In order to adaptively restructure agencies taking into account targets' motions, agencies mutually exchange their information.
- Employing the dynamic memory realized the dynamic interactions in each layer without synchronization. The system is endowed with a high reactiveness.

These properties allow the system to be adaptable to complicated dynamic situations in the real world.

To practically apply our system to real-world vision systems, the following issues should be discussed

The number of trackable targets: To enable the system to track more targets than the number of AVAs, we can modify the system so that an agency without any member-AVAs can be generated. For an AVA to report the object information to agencies without increasing the network-load, the message should be sent only to agencies that require the information. Such a member-AVA that sends the information to other agencies is called a *Supporter-AVA*. While a member-AVA can be a supporter-AVA for multiple agencies, it has to belong to a single agency as a member-AVA for avoiding inconsistent camera-control from different agencies.

Tracking in isolated camera configuration: In the proposed system, visual fields of all AVAs are overlapping with each other. To keep tracking a target even if cameras are embedded sparsely, the system has to employ not only the 3D trajectory of the target but also other information for object identification: e.g., 1) appearance information of objects is useful, and 2) constraints on the route and lapse assist object identification[8].

This work was supported by the Grant-in-Aid for Scientific Research(No.13224051).

8. REFERENCES

[1] T. Matsuyama, "Cooperative Distributed Vision - Dynamic Integration of Visual Perception, Action and Communication -", *Proc. of Image Understanding Workshop*, pp.365–384, 1998.

[2] A. Nakazawa, H. Kato, S. Hiura and S. Inokuchi, "Tracking multiple people using distributed vision systems", *Proc. of ICRA*, 2002.

[3] N. Yoshida and T. Fuki, "Target Tracking Using Tuple-Space-Based Mobile Agents", *Proc. of 19th IASTED ICAI*, pp.389–393, 2001.

[4] H. Ishiguro, "Distributed Vision System: A Perceptual Information Infrastructure for Robot Navigation", *Proc. of IJCAI-97*, Vol.1, pp.36–41, 1997.

[5] B. Horling, *et al*, "Distributed Sensor Network for Real Time Tracking", *Proc. of the 5th ICAA*, pp. 417–424, 2001.

[6] T. Matsuyama, *et al*, "Dynamic Memory: Architecture for Real Time Integration of Visual Perception, Camera Action, and Network Communication", *Proc. of CVPR*, pp.728–735, 2000.

[7] G. P. Stein, "Tracking from Multiple View Points: Self-calibration of Space and Time", *Proc. of CVPR*, Vol. I, pp.521–527, 1999.

[8] V. Kettnaker and R. Zabih. "Bayesian multi-camera surveillance", in *Proc. of Computer Vision and Pattern Recognition*, pp.253–259, 1999.