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

Norimichi Ukita [a,1] Takashi Matsuyama [a]

[a] *Yoshidahonmachi, Sakyo, Kyoto, Japan 606-8501.*

Abstract

This paper presents a real-time cooperative multi-target tracking system. The system consists of a group of Active Vision Agents (AVAs, in short) representing a logical model of a network-connected computer with an active camera. All AVAs track their target objects cooperatively by interacting dynamically 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 real-time cooperation among AVAs, we designed a three-layered interaction architecture. In each layer, parallel processes mutually exchange a variety of information for effective cooperation. We employed the dynamic memory architecture to achieve real-time information exchange. Experimental results demonstrated that AVAs track their target objects cooperatively in real-time while adaptively changing their roles.

Key words: Multi-target Tracking, Tracking using Multiple Active Camera, Cooperative Distributed Tracking, Active Vision Agent

1 Introduction

Object tracking technology allows the development of various real-world vision systems such as visual surveillance and monitoring systems[2], ITS (Intelligent Transport Systems)[3], and navigation of mobile robots[17].

To apply object tracking to these real-world systems, the object tracking method must be able to cope with complicated situations and conduct processing reactively in real time. However, most object tracking methods proposed

Email addresses: ukita@is.naist.jp (Norimichi Ukita), tm@i.kyoto-u.ac.jp (Takashi Matsuyama).

[1] Now working at Graduate School of Information Science, Nara Institute of Science and Technology, Japan.

to date, have some restrictions with regard to their functions and assumptions about the environment. For example, there have been a large number of studies of single-target tracking:

- Using a single fixed/active camera: [4]/[5].
- Using multiple fixed/active cameras: [6]/[1].

As a multi-target tracking system is required for application of the system to general purposes, researchers have recently concentrated on multi-target tracking and the number of studies in this area is increasing:

- Using a single fixed camera: [7,8,3]
- Using multiple fixed cameras: [9,15,11,21]

However, there are few *multi-target* tracking systems that use multiple cameras. In particular, there have been few reports concerning with multi-target tracking systems that employ *multiple active cameras*.

The limitations with regard to resources and functions (i.e., *single*-target, *single*-camera and *fixed*-camera tracking) reduce the effectiveness and generality of object tracking for real-world systems. In this paper, therefore, we propose a flexible multi-target tracking system with multiple active cameras, which can be applied to various real-world systems and can cope with dynamic complicated situations in the real world. In this system, pan, tilt and zoom parameters of multiple cameras have to be controlled to implement the following functions simultaneously: 1) wide-area observation for continuously tracking multiple target objects; and 2) local-area scrutiny for acquisition of detailed information (i.e., high-resolution images of the target objects). As these two principles for controlling cameras are opposed to each other, implementing these camera controls simultaneously in real time is an extremely difficult and complex problem. Thus, it is impossible to solve this problem with previous tracking systems, especially in cases in which a tracking system possesses a large number of active cameras. In [14], for example, pan-tilt-zoom cameras were employed for tracking target objects. The system used, however, at most two (non-active) panoramic cameras and two pan-tilt-zoom cameras. In addition, no elaborate strategies for dynamic role assignments among the cameras (i.e., which camera gazes at which object or where) were realized.

We solve the above problem (i.e., two opposed principles for controlling many cameras in real time) with cooperative communication among cameras. Our system consists of *Active Vision Agents* (AVAs), where an AVA is a logical model of an active vision system that is capable of communicating with each other through the network. Our objective is to realize a tracking system that can 1) change its behavior in an adaptive manner depending on the situation and the given task, and 2) keep tracking focused on the target objects.

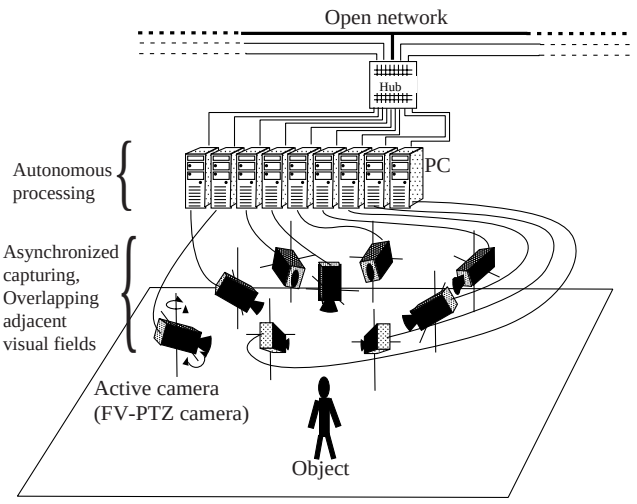


Fig. 1. System organization.

For real-time object tracking by multiple AVAs, we have solved the following problems:

- The design of an active camera for dynamic object detection[1].
- Realization of real-time object tracking with an active camera[20].

In this paper, we focus on how to realize real-time cooperation among AVAs.

To implement real-time cooperation among AVAs, we propose a three-layered interaction architecture. In each layer, parallel processes exchange different kinds of object information for effective cooperation. For real-time information exchange and processing, we employed the dynamic memory architecture[20]. 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 demonstrated that the proposed real-time cooperation method enables a system with ten AVAs to: 1) successfully acquire dynamic object information and 2) adaptively assign an appropriate role to each AVA.

1.1 System Organization

Our system consists of a group of network-connected computers, each of which possesses an active camera, as illustrated in Fig. 1. A group of spatially distributed active cameras enables continuous wide-area observation as well as detailed measurement of 3D object information. We imposed the following constraints regarding the camera configuration on the system:

- Visual fields of the cameras overlap with each other to keep tracking a target in the observation field without a break. That is, in our system, the area of

observation is determined by the number of cameras and their visual fields.
- In addition, all the observation spaces can be observed by at least two cameras because every space must be observed by multiple cameras to reconstruct 3D information of an object.

By employing multiple pan-tilt-zoom cameras, we designed a system that can not only simply track the trajectories of target objects but also acquire their high-resolution images taken from various directions [2].

With the above architecture and functions, each AVA works autonomously while maintaining its own intrinsic dynamics and cooperates with the other AVAs by exchanging information through the network. The network is not a special closed network (e.g., high-speed PC cluster) but a common open network.

Each AVA captures images asynchronously because it works autonomously maintaining its own dynamics. That is, the system is not in need of any synchronization mechanism (e.g., sync-pulse generator and gen-lockable camera). In [12] and [13], on the other hand, fully synchronized multi-camera systems, with all cameras synchronized by a common sync-generator, were proposed. However, all cameras should be controlled independently and observe the scene asynchronously because each camera must keep its intrinsic dynamics to adapt itself reactively to dynamic situations in the scene.

To allow each AVA to compare its observed information with that of another AVA as time-series data, we suppose that the internal clocks of all AVAs are synchronized. For example, by comparing time stamps of images captured by different AVAs with each other, the system can identify images taken at almost same time.

1.2 Architecture of AVA and its Functions

Each AVA possesses a single *Fixed-Viewpoint Pan-Tilt-Zoom* (FV-PTZ) camera[1]: its projection center remains fixed irrespective of any camera rotation and zooming. With a pan-tilt camera, it is difficult to detect arbitrary objects from an observed image. In [14], a moving object is detected by simply analyzing flows estimated in consecutive images. The system described in [5] can detect moving objects by subtracting consecutive images even if the camera is rotated during the observation. The implementation of background image compensation allows the system to apply motion detection techniques

[2] The high-resolution images of an object are very useful for application of the system to face and gesture recognition, volume reconstruction, and some other vision algorithms.

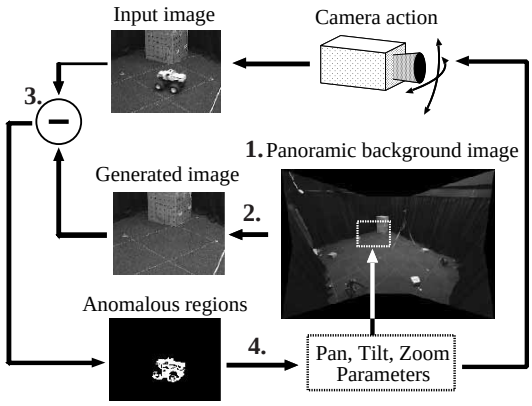


Fig. 2. Object detection and tracking using an FV-PTZ camera.

for the fixed camera to images taken with different camera rotations. With these methods, however, stationary objects cannot be detected without other methods that require the object information given in advance, such as color detection and face recognition.

In our system, on the other hand, an AVA can easily detect and track an arbitrary moving object as illustrated in Fig. 2:

(1) Generate a wide panoramic image of the background scene in advance. With the FV-PTZ camera, a wide panoramic image can be generated easily by mosaicing multiple images observed by changing pan, tilt, and zoom parameters. 1) The pan and tilt parameters are controlled so that the observed images reprojected onto a virtual plane can make the panoramic image without any aperture between the reprojected images. 2) Since the resolution of the panoramic background image is determined by the zooming factor, the zoom parameter is controlled such that the resolution is sufficient for detecting object regions by background subtraction.
(2) Extract a window image from the panoramic image according to the current pan-tilt-zoom parameters and regard it as the background image. A direct mapping exists between the position in the panoramic image and pan-tilt-zoom parameters of the camera.
(3) Compute differences between the generated background image and the observed image [3].
(4) If anomalous regions are detected in the difference image, select one and control the camera parameters to track the selected target [4]. Otherwise, move the camera along the predefined trajectory to search for objects.
(5) Points 2, 3, and 4 are repeated while the AVA is working.

In general, there are two kinds of agents:

[3] To utilize this object detection method with varying background scenes, a robust background subtraction (see [18] [19], for example) is required.
[4] How to control the camera parameters will be described in Section 3.3.2.

Software agent is a virtual agent without any physical body in the real world. Each agent corresponds to logical data (e.g., information of a detected object) in the system (see [16], for example).

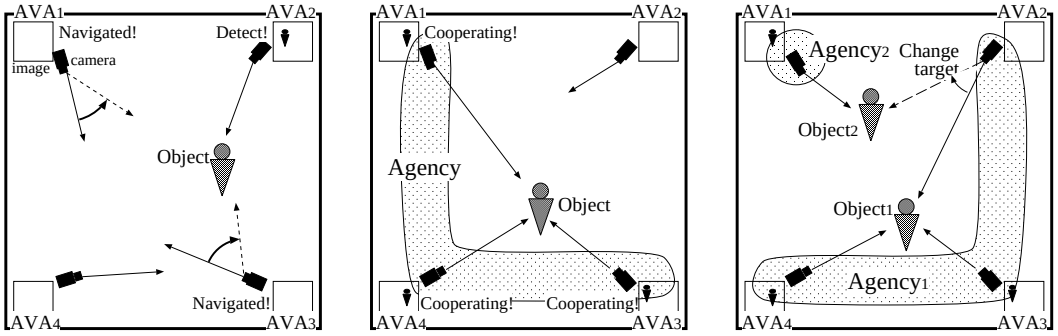
Real-world agent is an agent with its own physical body (e.g., an active camera and a mobile robot) that can be controlled by itself. There is a one-to-one correspondence between an agent in the system and the body in the real world (see [17], for example).

We believe that an intelligent system has to possess its own body to interact with the real world. Therefore, we define an *agent* to correspond to each physical body in the real world. That is, we call 1) a real-world agent simply an agent and 2) an agent without a body a software agent.

Although several object tracking systems with multiple cameras and multi-agent systems have been reported, most employ only software agents[10,11]. In these systems, 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. In addition to 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 employs active cameras. This makes it difficult for a camera to gaze at multiple objects simultaneously. This limitation prevents the realization of the intended system.

In our system, on the other hand, an agent (i.e., AVA) corresponds to a single active camera. That is, each agent monopolizes its own camera. Therefore, all AVAs can control their own cameras to gaze at their targets. Our definition of an agent has the advantage that it has a one-to-one correspondence between the agent and camera.

In addition, in our system, the information of each target should be managed intensively to 1) record the information of each object and 2) compare the information of different objects with each other. To realize these functions, a software agent, which has a 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 (described in detail later).



(a) Gaze navigation (b) Cooperative gazing (c) Adaptive tracking

Fig. 3. Basic scheme for cooperative tracking. A mesh region denotes an agency, and AVAs within the same mesh region belong to the same agency.

1.3 Basic Scheme for Cooperative Tracking

In our system, many AVAs are embedded in the real world and observe a wide area. With these AVAs, we realize a multi-AVA system that cooperatively detects and tracks multiple targets. The following are the tasks of the system:

(1) Initially, each AVA searches independently for objects that come into the observation scene.
(2) When an AVA detects an object, the AVA examines whether or not it should observe the detected object. If the object should be observed, the AVA regards it as a target and navigates the gazes of other AVAs toward the target (Fig. 3 (a)).
(3) An agency is a group of AVAs that cooperatively gaze at the same target form what we call an *Agency* (Fig. 3 (b)). In our system, there is a one-to-one correspondence between an agency and a target in the scene.
(4) Depending on target locations in the scene, each AVA changes its target dynamically (Fig. 3 (c)).
(5) When the target leaves the scene, an AVA decides whether it should search again for objects or track another target that is tracked by other AVAs depending on the situation.

To realize the above cooperative tracking, it is necessary to solve the following problems:

Multi-target identification: To gaze at each target, the system has to distinguish multiple objects in the scene.
Real-time and reactive processing: To adapt itself to dynamic changes in the scene (e.g., object motion), the system has to execute processing in real time and cope with variations in the scene reactively.
Adaptive resource allocation: it is necessary to implement a two-phase 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 objects even while tracking targets.
(2) To track each moving target persistently, the system has to adaptively determine which AVAs should track which targets.

In what follows, we address how these problems can be solved by real-time cooperative communications among AVAs and agencies.

2 Task Specification

The tracking system needs to search for objects in the scene. This role is called *Search*[4]. Once a target is detected, the system gazes at it to obtain its information. This role is called *Tracking*. Hereafter, slanted *search* and *tracking* denote the roles of the AVA. In addition, the system is required to gaze selectively at the object whose information is necessary for the given task.

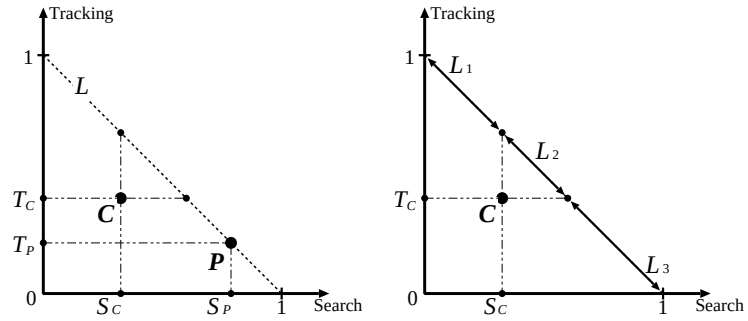
In our system, we specify the task for the system by the following three parameters:

Task-constraint represents the number of AVAs that execute *search* and *tracking*.
Object-priority specifies the priority of each object.
Utility-function specifies the fitness of an AVA for each task.

2.1 Task-constraint

The number of AVAs that execute *search* and *tracking* is adjusted in accordance with the given task-constraint. An AVA that searches for a new (undetected) object is called a *Freelancer*-AVA, and observes a wide area autonomously. AVAs that cooperatively track the same target form an agency. An AVA belonging to an agency is called a *Member*-AVA.

We can realize various capabilities of the system in terms of the combination of *search* and *tracking* as shown in Fig. 4. We call this graph a *System State Graph*. The horizontal and vertical axes indicate the rates of AVA that perform *search* and *tracking*, respectively. We call values of the horizontal and vertical axes *Search-level* and *Tracking-level*.



(a) Current state, Task-constraint (b) Three types of system state

Fig. 4. System state graph (system state representation with *search* and *tracking*).

Definition 1 (Search-level and Tracking-level)

$$\text{Search-level} = \frac{\text{The number of AVAs searching for an object}}{\text{The total number of AVAs}} \tag{1}$$

$$\text{Tracking-level} = \frac{\text{The number of AVAs tracking targets}}{\text{The total number of AVAs}} \tag{2}$$

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

Definition 2 (Current state $\boldsymbol{P}(S_P, T_P)$) *This parameter ($\boldsymbol{P}$ in Fig. 4 (a)) represents the search-level (S_P) and the tracking-level (T_P) at the present time. The range of the current state $\boldsymbol{P}$ is on the line L in Fig. 4 (a). That is, $(S_P + T_P)$ is always 1.*

Definition 3 (Task-constraint $\boldsymbol{C}(S_C, T_C)$) *This parameter ($\boldsymbol{C}$ in Fig. 4 (a)) represents the minimum search-level (S_C) and tracking-level (T_C), where $0 \leq S_C \leq 1$, $0 \leq T_C \leq 1$, and $0 \leq (S_C + T_C) \leq 1$. 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. 4. The system has to keep S_C and T_C while working. Therefore, the system 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 users as a pair of constants (i.e., S_C and T_C) depending on the task of the system.*

The following are the three states of the system determined by the relations between the task-constraint and the current state.

Deficiency of search-level: $T_C < T_P$ and $S_C > S_P$. That is, the current state $\boldsymbol{P}$ is on L_1 in Fig. 4 (b).

Task satisfaction: $T_C \leq T_P$ and $S_C \leq S_P$. That is, the current state $\boldsymbol{P}$ is on L_2 in Fig. 4 (b).

Deficiency of tracking-level: $T_C > T_P$ and $S_C < S_P$. That is, the current state $\boldsymbol{P}$ is on L_3 in Fig. 4 (b).

If the current state of the system does not satisfy the task-constraint, each AVA changes its own role dynamically between *search* and *tracking* to adjust the search-level and tracking level to the task-constraint.

Thus, the system can realize gradual variations in its behavior to adapt itself to versatile tasks by representing its behavior with numerical parameters.

2.2 Object-priority

In our system, object-priority is given to each object category that can be distinguished by the system.

Definition 4 (Object-priority I_P) *Let I_P denote the object-priority of the target of agency$_P$, where $0 \leq I_P \leq 1$. The number of member-AVAs in agency$_P$ (denoted by M_P) is determined by the object-priorities of the targets:*

$$M_P = (\text{The total number of member AVAs}) \times \frac{I_P}{S}, \tag{3}$$

$$S = \sum_{i=1}^{N_A} I_i, \tag{4}$$

where N_A is the total number of existing agencies. That is, the number of the member-AVAs is proportional to the object-priority of the target.

2.3 Utility-function

In our system, each AVA has to decide its role according to the given task-constraint and object-priority. These two parameters confer restrictions no the system regarding the numbers of AVAs that execute *search* and *tracking*. Under this restriction, however, each AVA can freely change its role taking into account the utility-function representing which AVA is suitable for which role.

The utility-function is represented by the total sum of the following search-value and tracking-value:

- **The search-value of a freelancer-AVA** is determined by the fitness of each freelancer-AVA for *search*.

- **The tracking-value of an agency** is the sum of all tracking-values of member-AVAs belonging to the agency.
  - **The tracking-value of a member-AVA** is determined by the fitness of each member-AVA for *tracking*.

The value of the utility-function is defined as the total sum of the search-values of all freelancer-AVAs and the tracking-values of all agencies. For example, when AVA_1 is a freelancer and AVA_2 and AVA_3 are member-AVAs, the sum of the search-value of AVA_1 and the tracking-values of AVA_2 and AVA_3 is the value of the utility-function. Each AVA changes its role dynamically to increase the utility-function under the restriction of the task-constraint and object-priority. The above values can be designed to adapt themselves to the task of the system by the users. We give an example in Section 5.2.

3 Dynamic Interaction for Cooperative Tracking

In our system, parallel processes work cooperatively by interacting dynamically 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. Therefore, the design of the whole system can be reduced to the design of each process. Furthermore, the states and these 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 in contrast to centralized processing systems (e.g., [15]).

3.1 Layers in the System

For the system to engage in multi-target tracking, *object identification* is significant. We, therefore, classify the system into three layers (namely, intra-AVA, intra-agency and inter-agency layers) depending on the types of object information employed for identification. Each layer corresponds to elements in the system as follows (Fig. 5):

Intra-AVA layer (the bottom layer): An AVA.
Intra-agency layer (the middle layer): An agency.
Inter-agency layer (the top layer): The total system.

In each layer, object 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

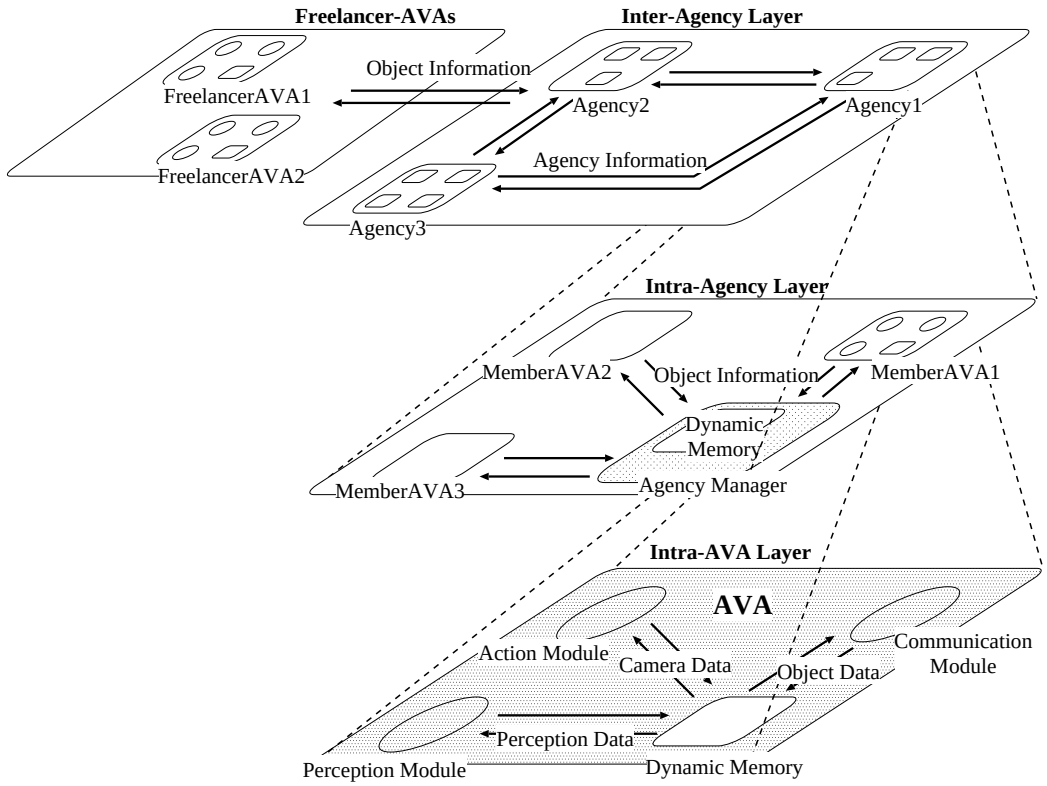


Fig. 5. Three layers in the system (bottom: intra-AVA, middle: intra-agency, top: inter-agency layers).

activated.

In what follows, we 1) first introduce the general concept and functions for real-time interaction among processes, and 2) address the interactions in each layer.

3.2 Dynamic Memory Architecture for Real-time Asynchronous Interaction

For real-time cooperation among parallel processes, they have to asynchronously exchange information maintained by themselves with each other. To support such asynchronous interactions, Matsuyama *et al.*[20] proposed a novel dynamic system architecture named the *Dynamic Memory Architecture*, where parallel processes share what they call the *Dynamic Memory*. 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. The dynamic memory supports asynchronous dynamic interactions (i.e., data exchange between the processes) without wasting time for synchronization. This no-wait asynchronous interaction capability greatly facilitates the implementation of real-time reactive systems such as a moving object tracking system.

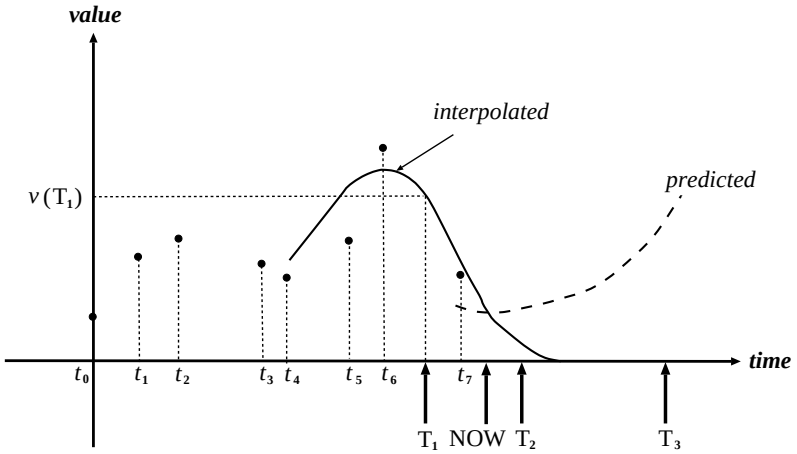


Fig. 6. Representation of a time varying variable in the dynamic memory.

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 (i.e., time-series data) and shared among the processes, the time information in all the processes must be consistent. In our system, therefore, all the processes that work in a single processor share a single internal clock in the processor and the internal clocks of all processors are synchronized.

The read/write operations from/to the dynamic memory are defined as follows (Fig. 6):

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 (the sequence of black dots in Fig. 6). Note that since the speed of the computation varies depending on input data, the temporal interval between a pair of consecutive values becomes irregular.

Read operation:

Temporal interpolation: A reader process runs in parallel to the writer process and tries to read the value of the variable from the dynamic memory at a certain moment according to its own dynamics: for example, the value at T_1 in Fig. 6. When no value is recorded at the specified moment, the dynamic memory interpolates it from its neighboring recorded discrete values. With this function, the reader process can read a value at any temporal moment along the continuous temporal axis.

Future prediction: A reader process may run fast and require data that have not yet been written by the writer process (for example, the value at

T_3 in Fig. 6). In such cases, the dynamic memory predicts an expected value in the future based on the data recorded so far and returns it to the reader process. Note that as illustrated in Fig. 6, multiple values may be defined by the interpolation and prediction functions, for example, at NOW and T_2 in Fig. 6. We have to define the functions to avoid such multiple value generation.

Since a variable in the dynamic memory represents a state of some dynamics of an object (e.g., pan-tilt-zoom parameters of an active camera), the interpolation and prediction functions associated with the variable should be designed to closely model the dynamics of the object. As stated in [20], therefore, off-line modeling and calibration of the object dynamics should be done *a priori* to define the functions.

In addition, to estimate the appropriate value from observed values with errors, the dynamic memory employs the Kalman Filter in the read operation, also as described in detail in [20].

3.3 Intra-AVA Layer: Interaction between Modules within an AVA

In the bottom layer in Fig. 5, perception, action and communication modules that comprise an AVA exchange the time-series information with each other via the dynamic memory[20] possessed by each AVA. With the dynamic memory, modules can exchange their information asynchronously at any time. The interactions among modules realize the functions of an AVA.

3.3.1 Perception Module

The tasks of the perception module are as follows:

(1) Capture an image.
(2) Detect anomalous regions in the captured image.
(3) Distinguish object regions in the observed image and obtain the information for each object.
(4) Determine its own target from the detected objects.

While tasks 1 and 2 above are similar to those of the single-target tracking system proposed previously in [20], the perception module must establish object identification (i.e., tasks 3 and 4 above) in the multi-target tracking system. Note that a freelancer-AVA performs only tasks 1 and 2 because it should observe a wide area without gazing at a specific object.

After the detection of anomalous regions, the perception module examines whether or not each detected pixel is adjacent to another: if adjoining pixels are detected as anomalous regions, they are regarded as the region of the same object. The centroid and the size of detected object$_p$ are denoted by (x_d^p, y_d^p) and S^p, respectively. The pan-tilt angles of object$_p$ at the image capturing time t (denoted by $(P_{obj}^p(t), T_{obj}^p(t))$) is represented by

$$\begin{pmatrix} P_{obj}^p(t) \\ T_{obj}^p(t) \end{pmatrix} = \begin{pmatrix} P_{cam}(t) \\ T_{cam}(t) \end{pmatrix} + \begin{pmatrix} \arctan(x_d^p / f(t)) \\ \arctan(y_d^p / f(t)) \end{pmatrix}, \tag{5}$$

where $(P_{cam}(t), T_{cam}(t))$ $f(t)$ denote the pan-tilt angle and focal length of the camera at t, respectively. That is, the 3D view direction from the projection center to object$_p$ at t is denoted by $(P_{obj}^p(t), T_{obj}^p(t))$. We call this 3D view direction a *3D view line* $L^p(t)$. The 3D view line $L^p(t)$ and the region size $S^p(t)$ are regarded as the information of object$_p$ at t.

When the module detects N objects at $t+1$, it computes and records $L^1(t+1), \cdots, L^N(t+1)$ and $S^1(t+1), \cdots, S^N(t+1)$ into the dynamic memory. Then, the module compares them with the 3D view line toward the currently tracked target at $t+1$, $\widehat{L}(t+1)$. Note that $\widehat{L}(t+1)$ can be read from the dynamic memory regardless of the temporal moment $t+1$. Comparison of the object information observed at the same time (i.e., $L^1(t+1), \cdots, L^N(t+1)$ and $\widehat{L}(t+1)$) makes object identification reliable. 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.

3.3.2 Action Module

The action module in a freelancer-AVA moves its camera along the predefined trajectory to search for objects, except when it receives the 3D position of a target object from an agency, and the camera is controlled toward that 3D position (described in detail in Section 3.6).

In a member-AVA, on the other hand, the action module controls the camera to gaze at a target. The action module can obtain two types of object information from its dynamic memory:

3D position-based control As described below, when an agency with multiple AVAs tracks a target, it measures the 3D position of the target (i.e., $\widehat{P}(t)$) and sends it to all member-AVAs, which is then written into the dynamic memory by the communication module.

2D appearance-based control The perception module estimates the object information from the observed image (i.e., $L^1(t+1), \cdots, L^N(t+1)$ and

$S^1(t+1), \cdots, S^N(t+1)$) and writes it into the dynamic memory.

When an active camera is ready to accept a control command, the action module first reads $\widehat{P}(now)$ from the dynamic memory.

3D position-based control Suppose the newest $\widehat{P}(t)$ in the dynamic memory is obtained at t_{new}. If the interval between now and t_{new} is shorter than the predefined threshold, the estimated $\widehat{P}(now)$ is considered to be valid. When $\widehat{P}(now)$ is valid, the module controls the camera to gaze at the target. Since $\widehat{P}(t)$ is sent to a member-AVA when it loses sight of the target (described in Section 3.4.4), the zooming factor is adjusted to the widest field of vision.

2D appearance-based control Otherwise, the module reads $\widehat{L}(now)$ and $\widehat{S}(now)$ from the dynamic memory to 1) control the view direction toward $\widehat{L}(now)$ and 2) adjust the zooming factor to capture the whole image of the target object.

Here, the camera is controlled by the prediction-based tracking method with the dynamic memory proposed in [20]: the module controls the camera toward the target position at the next camera control timing (t_{next}) in accordance with the target position at t_{next} and the camera pan-tilt angles at now, both of which are read from the dynamic memory.

The 2D appearance-based control is superior when a member-AVA observes its target object clearly and precisely because the camera can be controlled based on prediction and high-resolution images can be acquired. When a member-AVA fails in tracking its target object, for example, when a target object is (partially) occluded by other moving objects or obstacles, the camera should be controlled based on reliable information about the target object; an agency can provide the reliable 3D position of the target object, which is estimated by integrating 3D view lines observed by its member-AVAs. Therefore, we defined the algorithm for controlling the camera as described above.

3.3.3 Communication Module

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. Object information sent from an AVA is listed in Table 1. An AVA transmits the message in the different ways depending on its role:

- The message is broadcasted if the AVA functions as a freelancer-AVA. This broadcast message is accepted only by agencies.
- The message is sent only to its agency if the AVA functions as a member-AVA.

Table 1
Object information sent from an AVA to the agencies.

Entry			*Information*
AVA information	AVA-ID		ID of an AVA
	External parameters		External camera parameters (3D position and view direction)
Detected information $\{1, \cdots, N\}$	Number		The number of detected objects
	Time		The time at which the AVA observed the target
	Object_1	View line	3D view line from the camera to object_1 (L^1)
		Target flag	If an AVA is tracking object_1, the value is 1, otherwise 0.
	$\vdots$		$\vdots$
	Object_N	View line	3D view line from a camera to object_N (L^N)
		Target flag	If an AVA is tracking object_N, the value is 1, otherwise the value is 0.

3.3.4 Dynamic Memory: Interaction between the Modules

To function cooperatively as an AVA, perception, action and communication modules must dynamically exchange the time-series information maintained by each module. Each module provides the following information:

Perception: 3D view lines toward detected objects.
Action: Camera parameters (i.e., pan, tilt and zoom).
Communication: Received information of the target.

This information exchange is realized through the dynamic memory. Its contents are shown in Table 2. With the dynamic memory, all modules can exchange their information asynchronously at any time. Therefore, each module can function autonomously without damaging the reactivity required for a real-time system.

Table 2
Entries of the dynamic memory in the intra-AVA layer.

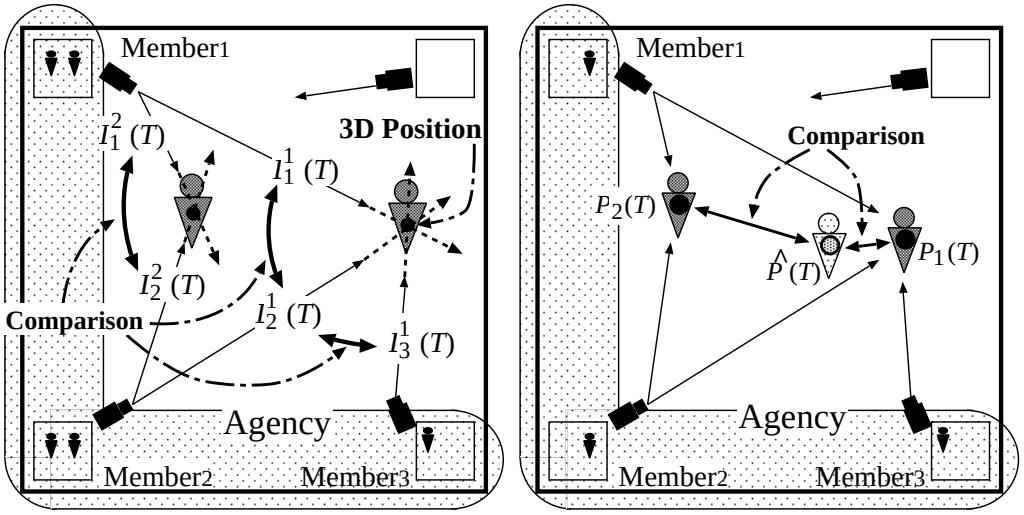
Entry			*Information*
Camera information	Pan-Tilt-Zoom		Pan-Tilt-Zoom parameters are recorded as time-series data ($P_{cam}(t)$, $T_{cam}(t)$, $Z_{cam}(t)$).
Target information	3D position		3D positions of the target are recorded as time-series data ($\widehat{P}(t)$).
Detected information $\{1, \cdots, N\}$	Number		The number of detected objects
	Object_1	View line	3D view line from a camera to object_1 is recorder as time-series data ($L^1(t)$).
		Target flag	If an AVA is tracking object_1, the value is 1, otherwise 0.
	⋮		⋮
	Object_N	View line	3D view line from a camera to object_N is recorded as time-series data ($L^N(t)$).
		Target flag	If an AVA is tracking object_N, the value is 1, otherwise the value is 0.

3.4 Intra-agency Layer: Interaction between AVAs

As defined above, an agency consists of a group of AVAs that track the same target. The intra-agency layer (the middle layer in Fig. 5) consists of member-AVAs belonging to the same agency simultaneously. In our system, an agency should show a one-to-one correspondence to a target. To make this correspondence dynamically established and persistently maintained, member-AVAs in the same agency are required to exchange information for both spatial and temporal object identification.

3.4.1 Object Identification

3.4.1.1 Spatial Object Identification The agency has to establish object identification between the groups of the 3D view lines detected and transmitted by its member-AVAs. When $\text{member-AVAs}_{1,\cdots,M}$ in the same agency capture the images, the agency must establish object identification between



(a) Spatial object identification (b) Temporal object identification

Fig. 7. Object identification established in the intra-agency layer.

the 3D view lines $\{L_1^i(t_1)|i = 1, \cdots, N_1\}$, $\cdots$, $\{L_M^i(t_M)|i = 1, \cdots, N_M\}$, where $\{L_m^i(t_m)|i = 1, \cdots, N_m\}$ denotes the 3D view lines detected by member-AVA$_m$ at t_m (Fig. 7 (a)). For spatial object identification, the system checks distances between the 3D view lines detected by different member-AVAs. Note that member-AVAs in an agency observe 3D view lines toward objects at different times because AVAs capture images autonomously. Furthermore, message transmission over the network introduces an unpredictable delay between the observation timing by a member-AVA and the object identification timing. The results of object identification are, therefore, unreliable if the asynchronous object data are compared with each other.

Other distributed systems that consist of autonomous cameras coped with this problem as follows:

- In [9], all object information regarding position, height, and width, observed by multiple cameras, is integrated by a Bayesian algorithm to increase the robustness of object identification. This method does not take into account the synchronization problem.
- In [11], the newest information gathered from each camera is considered to be observed at the same time. In [21,10], the object information includes a time stamp (Let t_i denote the time stamp of information$_i$). The system regards the information observed at t_i and t_j, where $|t_i - t_j|$ is sufficiently small, as simultaneous information. These approximate methods break down under complicated conditions and with network congestion.

To resolve this problem, our system employs the *Virtual Synchronization* to virtually adjust observation timings of the 3D view lines, which we proposed in [20,22] (see Section 3.4.2 for details). These virtually synchronized 3D view lines are compared between different AVAs, and the 3D distance between the

view lines is computed. If the 3D distance between the view lines is less than a given threshold, these view lines are considered to be information of the same object. In addition, the intersection of the identified 3D view lines is regarded as the 3D position of the object.

In the example shown in Fig. 7 (a), 3D view lines detected by member-AVAs$_1$, member-AVAs$_2$ and member-AVAs$_3$ are compared with each other. All the 3D view lines are virtually synchronized at T in advance. Then, '$L_1^1(T)$, $L_2^1(T)$ and $L_3^1(T)$' and '$L_1^2(T)$ and $L_2^2(T)$' are identified with each other. Based on these correspondences, the system computes the intersections of the 3D view lines that were identified with each other, and then regards these intersections as the 3D positions of the detected objects.

Note that an agency may find none or multiple sets of such nearly intersecting 3D view lines. To cope with these situations, the agency conducts temporal object identification as described below.

3.4.1.2 Temporal Object Identification To gaze at the target continuously, the agency compares the 3D trajectory of the target with the 3D positions of the objects newly computed by spatial object identification. When multiple 3D locations are obtained by spatial object identification, the agency selects the one closest to the target trajectory. On the other hand, when spatial object identification fails and no 3D object location is obtained, the agency selects the 3D view line that is closest to the latest recorded target 3D position. Then, the agency projects the target 3D position onto the selected view line to estimate the new 3D target position. Note that when an agency contains only a single AVA, neither spatial nor temporal object identifications succeed and hence the member-AVA conducts only 2D appearance-based tracking by itself.

In the example shown in Fig. 7 (b), the system estimates the 3D positions of two objects at T (denoted by $P_1(T)$ and $P_2(T)$) by spatial object identification. The system then compares these 3D positions with the 3D position of the target at T (denoted by $\widehat{P}(T)$). Note that $\widehat{P}(T)$ can be computed by employing the virtual synchronization regardless of the temporal moment T specifies (described later). As a result, $P_1(T)$ is identified with $\widehat{P}(T)$.

3.4.2 Virtual Synchronization

Here, we discuss the dynamic aspects of the above identification processes.

3.4.2.1 Virtual Synchronization for Spatial Object Identification The unpredictable delay between the observation timing by a member-AVA

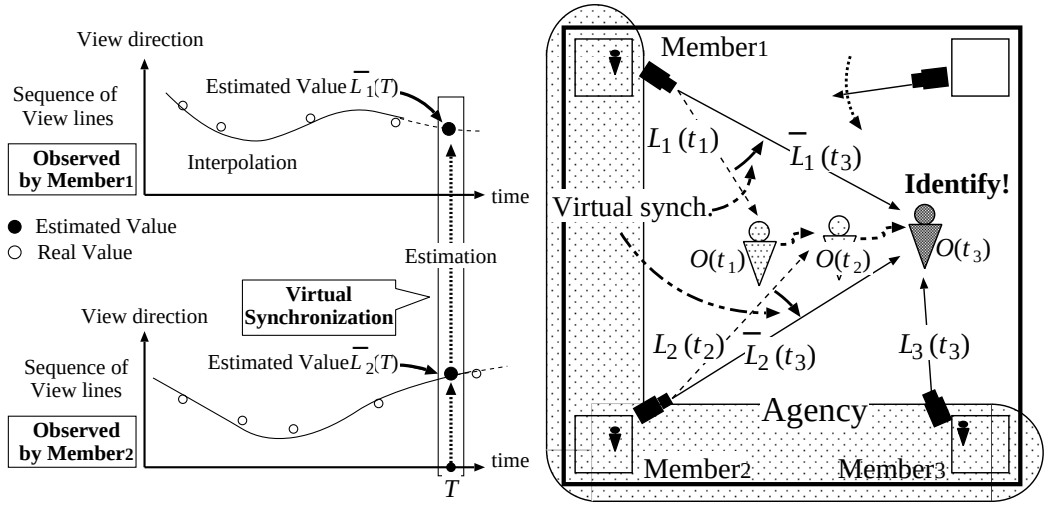
Table 3
Entries of the dynamic memory in the intra-agency layer.

Entry			*Information*
Target info.	3D position		3D positions of the target are recorded as time-series data.
	3D view line		3D view lines toward the target are recorded as time-series data.
	Object-priority		The object-priority of the target
Member info. $\{1, \cdots, M\}$	Number		The number of member-AVAs at the present time.
	member-AVA$_1$	AVA-ID	ID of member-AVA$_1$
		External parameters	External camera parameters of AVA$_1$'s camera
	$\vdots$		$\vdots$
	member-AVA$_M$	AVA-ID	ID of member-AVA$_M$
		External parameters	External camera parameters of AVA$_M$'s camera
Detected info. $\{1, \cdots, M\}$	Detected Information of member-AVA$_1$		Information sent from member-AVA$_1$ is recorded as time-series data.
	$\vdots$		$\vdots$
	Detected Information of member-AVA$_M$		Information sent from member-AVA$_M$ is recorded as time-series data.

and the object identification timing results in unreliable object identification. To resolve this problem, we introduce the dynamic memory into the intra-agency layer. By employing the dynamic memory, a 3D view line at an arbitrary time can be estimated from asynchronous discrete time-series data. The system can, therefore, estimate the 3D view line observed by all cameras at the same time. We call this function the *Virtual Synchronization*.

In our system, spatial object identification is realized as follows. When an agency is formed, an *Agency Manager* is generated at the same time. An agency manager is an autonomous software agent [5] independent of AVAs, and performs the following tasks as a delegate of an agency.

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



(a)Readout from the dynamic memory (b)Spatial object identification

Fig. 8. Virtual synchronization for spatial object identification among member-AVAs in the same agency.

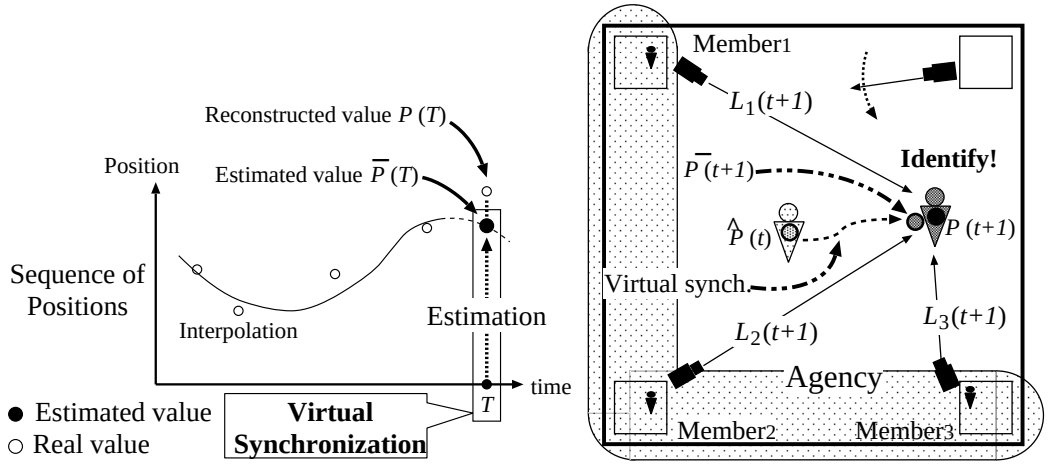
- Management of the dynamic memory in each agency. The contents of the dynamic memory in the intra-agency layer are shown in Table 3.
- Object identification.
- Communication with other agencies and AVAs.

That is, an agency is a conceptual group, and an agency manager is an entity of the agency.

All the target information is managed by each agency manager. For intensive management of the target information by each agency, the system handles the object information as follows:

- All member-AVAs send the information of the detected objects only to their agency manager.
- Even if the agency disappears because the target cannot be observed, the observed information will be managed by the agency that tracks the same target when it is detected again (described later).

Figure 8 shows the mechanism of the virtual synchronization for spatial object identification. All 3D view lines computed by each member-AVA are transmitted to the agency manager, which then records them into its internal dynamic memory. For example, Fig. 8 (a) shows a pair of temporal sequences of 3D view line data (indicated by white points in the figure) transmitted from member-AVA_1 and member-AVA_2, respectively. When the manager wants to establish spatial object identification at T, it can read the pair of the synchronized 3D view line data at T from the dynamic memory (i.e. $\bar{L}_1(T)$ and $\bar{L}_2(T)$ in Fig. 8 (a)). Figure 8 (b) shows an example of spatial object identification with the virtual synchronization. In this example, AVA_1, AVA_2 and AVA_3 capture the



(a)Readout from the dynamic memory (b)Temporal object identification

Fig. 9. Virtual synchronization for temporal object identification.

images at t_1, t_2 and t_3, and detect the 3D view lines $L_1(t_1)$, $L_2(t_2)$ and $L_3(t_3)$, respectively. The agency manager synchronizes these 3D view lines at t_3 to read $\bar{L}_1(t_3)$, $\bar{L}_2(t_3)$ and $L_3(t_3)$ from the dynamic memory. By comparing these virtually synchronized values, the agency manager can estimate the reliable object position $P(t_3)$ by spatial object identification.

3.4.2.2 Virtual Synchronization for Temporal Object Identification

The virtual synchronization is also effective in temporal object identification. Let $\widehat{P}(t)$ denote the 3D target trajectory recorded in the dynamic memory and $\{P_i(T)|i = 1, \cdots, M\}$ the 3D positions of the objects reconstructed at T by spatial object identification. Then the agency manager 1) reads $\widehat{P}(T)$ (i.e. the estimated target position at T) from the dynamic memory, 2) selects the one among $\{P_i(T)|i = 1, \cdots, M\}$ closest to $\widehat{P}(T)$, and 3) records it into the dynamic memory as the target position.

Figure 9 shows an example of temporal object identification with the virtual synchronization. By interpolating the reconstructed 3D positions of the target, the agency manager can estimate the target position at T (Fig. 9 (a)). In Fig. 9 (b), 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., $\widehat{P}(t+1)$) and compares $\widehat{P}(t+1)$ with $P(t+1)$ for temporal object identification.

As mentioned above, information exchange in the intra-agency layer through the dynamic memory allows the system to stabilize both spatial and temporal object identification. Depending on whether or not temporal object identification is successful, the dynamic interactions are activated. These dynamic interactions are defined by the following three cooperative-tracking protocols:

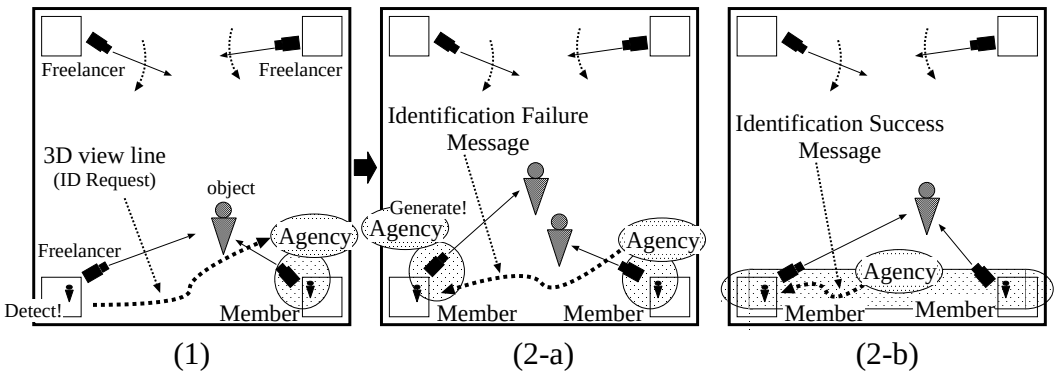


Fig. 10. Agency formation protocol.

Agency formation protocol defines a new agency generation procedure by a freelancer-AVA and a participation procedure of a freelancer-AVA into an existing agency.

Agency maintenance protocol defines procedures for cooperative tracking, continuous maintenance of an agency and the elimination of an agency.

Agency spawning protocol defines a new agency generation procedure from an existing agency.

3.4.3 Agency Formation Protocol

Initially, each AVA searches for objects independently. When a freelancer-AVA finds a new object, it requests object identification from existing agencies between the newly detected object and the target of each agency (see Section 3.6 for details) (Fig. 10 (1)). Depending on whether or not the result of this object identification is successful, the freelancer-AVA works as follows:

Case A: If no agency establishes successful identification, the freelancer-AVA generates a new agency manager and joins this agency. The agency that tracks the newly detected object is then formed (Fig. 10 (2-a)).

If the freelancer-AVA detects multiple objects and two or more of them are regarded as newly detected objects, it selects the object with the highest object-priority as its target.

Case B: If an agency establishes successful identification, the freelancer-AVA joins the agency that has made the successful identification (Fig. 10 (2-b)).

If the freelancer-AVA detects multiple objects and two or more of them are identified with the target objects of existing agencies, it determines its agency depending on their object-priorities.

Depending on the relationship between the current state of the system and the task-constraint, the agency formation is rejected even if the freelancer-AVA finds an object. That is, if there are few freelancer-AVAs with regard to *search*, the number of freelancer-AVAs must not decrease. Then, in the above cases A

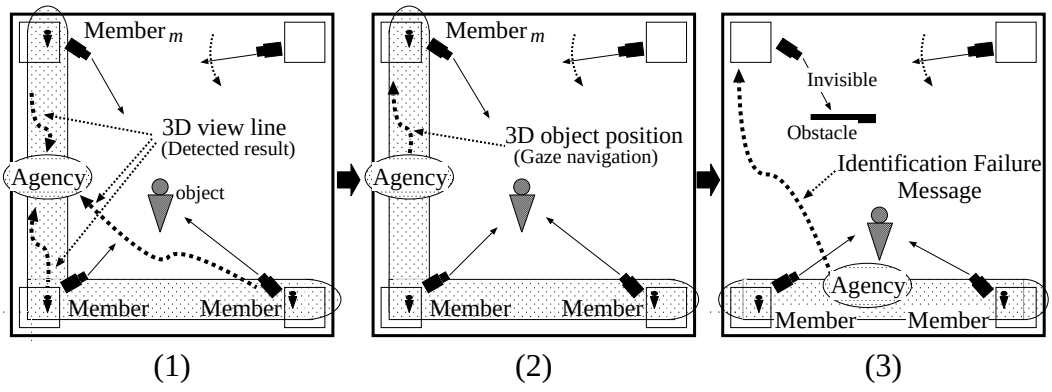


Fig. 11. Agency maintenance protocol.

and B, the freelancer-AVA and the agency work as follows:

Case A: The freelancer-AVA cannot become a member-AVA, and a new agency is not generated.

Case B: The agency manager that established successful identification examines the values of the utility-function in the case that the freelancer-AVA replaces each of the current member-AVAs in order to determine 1) whether to absorb the freelancer-AVA instead of one of the current member-AVAs, and then 2) which member-AVA should be released if the freelancer-AVA joins the agency.

3.4.4 Agency Maintenance Protocol

After an agency is generated, the agency manager continues spatial and temporal object identification for cooperative tracking (Fig. 11 (1)). If temporal object identification between the targets of the agency and member-AVA$_m$ fails (i.e., if member-AVA$_m$ does not gaze at the target of the agency), the agency manager reports the 3D position of the target to member-AVA$_m$. This information navigates the gaze of member-AVA$_m$ toward the target (Fig. 11 (2)). Nevertheless, if this identification fails for a long time, the agency manager forces member-AVA$_m$ out of the agency making it a freelancer-AVA (Fig. 11 (3)). If all member-AVAs are unable to observe the target, the agency manager eliminates the agency and all its member-AVAs become freelancer-AVAs.

In addition,the agency manager adjusts the number of member-AVAs depending on the relationship between the current state of the system and the task-constraint: if there are insufficient freelancer-AVAs, the agency manager must release its member-AVAs to increase the number of freelancer-AVAs.

In our system, all information about the same object should be managed together. For this purpose, the agency manager records its object information to an object database when it is eliminated. When an agency manager is newly

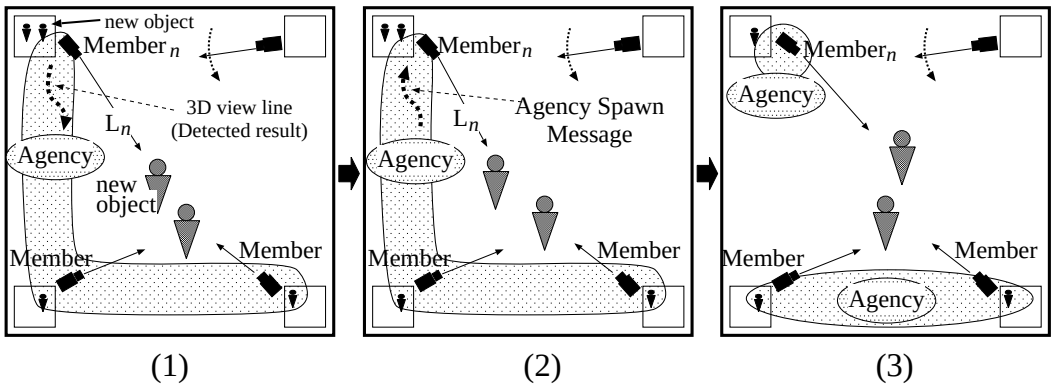


Fig. 12. Agency spawning protocol.

generated, it 1) reads object information from the database and 2) compares its target information with the read information to check if its target corresponds to a target that has been tracked previously. If so, the corresponding target information is moved from the database into the dynamic memory of the newly generated agency.

In the system proposed in this paper, 1) the object information recorded in the database consists only of its trajectory and is represented as time-series information in the same way as the information in the dynamic memory, and 2) object identification is realized only by comparing the trajectories of observed objects as described in Sections 3.4.1 and 3.4.2. Accordingly, 1) this system can identify an object that has lost track of for a short interval but 2) identification fails if an object is not observed for a long time [6].

3.4.5 Agency Spawning Protocol

After spatial and temporal object identification, the agency manager may find such a 3D view line(s) that does not correspond to the target; this represents the detection of a new object by its member-AVA. Let L_n denote such a 3D view line detected by AVA$_n$ (Fig. 12 (1)). The agency manager then requires other agencies to compare L_n with their own targets for object identification. The results of object identification are returned from other agency managers. If none of identification is successful (namely, if none of the agencies track L_n), the agency manager orders member-AVA$_n$ to generate a new agency (Fig. 12 (2)) and join it (Fig. 12 (3)).

Note that agency generation by the agency spawning is also restrained if the search-level of the current state is less than that of the task-constraint.

[6] To improve object identification, it is necessary to employ face, gait, shape, and other recognition methods.

Table 4
Agency information.

Entry			*Information*
Target info.	3D information		3D position of the target $\widehat{P}$ or 3D view line of the target $\widehat{L}$
	Object-priority		The object-priority of the target
	Time		The time when the target is observed
Member info. $\{1, \cdots, M\}$	Number		The number of member-AVAs
	member-AVA$_1$	AVA-ID	ID of member-AVA$_1$
		External parameters	External camera parameters of AVA$_1$'s camera
	$\vdots$		$\vdots$
	member-AVA$_M$	AVA-ID	ID of member-AVA$_M$
		External parameters	External camera parameters of AVA$_M$'s camera

3.5 Inter-agency Layer: Interaction between Agencies

The inter-agency layer consists of all existing agencies (top layer in Fig. 5). In multi-target tracking, the system should allocate resources adaptively: the system must adaptively determine which AVAs should track which targets. To realize this adaptive resource allocation, information about targets and member-AVAs is exchanged between agency managers. We call this information *Agency Information*. The contents of the agency information are listed in Table 4.

The dynamic interactions between agency managers are triggered based on object identification of target 3D positions across agencies. That is, when a new target 3D position is obtained, agency manager$_i$ broadcasts it to the others. Agency manager$_j$, which receives this information, compares it with the 3D position of its own target to check object identification. This object identification is not reliable if these 3D positions are observed at different times. This problem can be resolved by the virtual synchronization in the same way as temporal object identification in the intra-agency layer. With the 3D positions of the target recorded as time-series data in the dynamic memory, agency manager$_j$ can synchronize the 3D position of its target with the received 3D position.

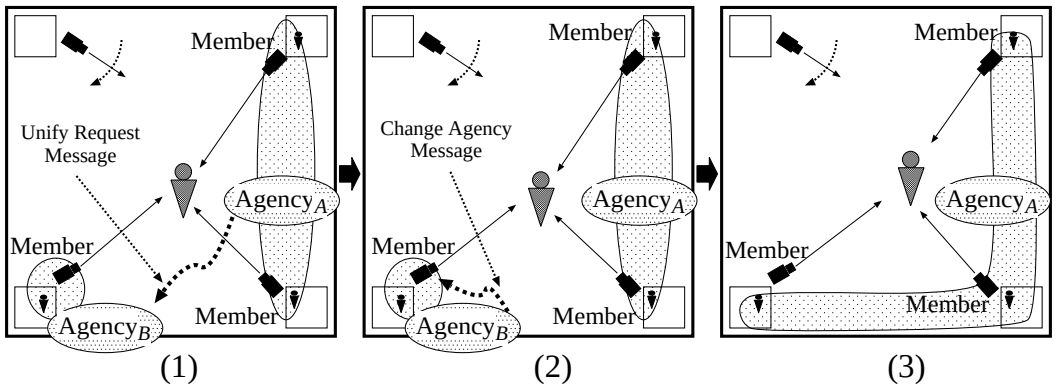


Fig. 13. Agency unification protocol.

Depending on whether or not the result of the above object identification is successful, either of the following cooperative-tracking protocol is activated:

Agency unification protocol defines a merging procedure of the agencies that happen to track the same object.

Agency restructuring protocol defines a reformation procedure of the member-AVAs between agencies.

3.5.1 Agency Unification Protocol

This protocol is activated when the result of the inter-agency object identification is successful.

In principle, the system should maintain the one-to-one correspondence between agencies and detected targets. However, this correspondence is sometimes violated due to failure of object identification and discrimination. Actual examples of such situations are as follows:

- **Asynchronous observations and/or errors in object detection by individual AVAs:** If a single object is first regarded as multiple objects due to failure of object identification, multiple agencies are formed for the same object by mistake. This error is recovered by subsequent observation. That is, when object identification between the agencies is successful, these agencies merge by the agency unification.
- **Multiple targets that cannot be separated:** The agency manager may consider multiple objects in the scene as a single object due to failure of object identification. This may occur when different objects become too close to separate. In this case, the agencies that make successful identification merge temporally to maintain the one-to-one correspondence between the agency and the detected target.

This protocol is required to cope with failures of object identification and

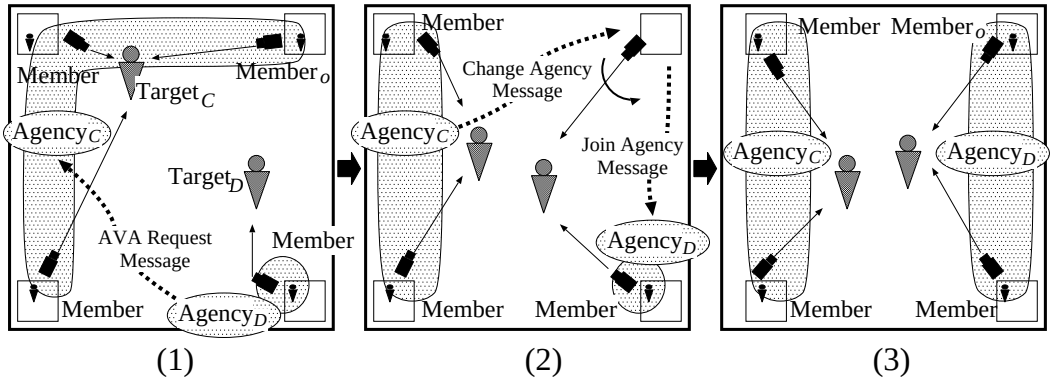


Fig. 14. Agency restructuring protocol.

discrimination.

Figure 13 shows an example of the agency unification. Agency manager$_A$, which has performed successful object identification with the target of agency$_B$, sends a message to agency manager$_B$. This message asks agency manager$_B$ to join agency$_A$ (Fig. 13 (1)). Then, agency manager$_B$ sends messages to its member-AVAs (Fig. 13 (2)) and eliminates itself. Thus, the two agencies merge (Fig. 13 (3)).

As noted above, agencies corresponding to multiple different targets may be unified if they are very close. However, this heterogeneously unified agency can be separated again into two agencies by the agency spawning protocol when the distance between the targets becomes larger.

3.5.2 Agency Restructuring Protocol

This protocol is activated when the result of inter-agency object identification is unsuccessful. The agency manager activates this protocol taking into account the following factors:

- The number of member-AVAs in each agency is determined by the object-priority of its target.
- Under the condition of restriction of number, each agency is attended by AVAs that are suitable for gazing at its target. This criterion is given by the utility-function.

Actual examples of situations that cause the agency restructuring are as follows:

- **Due to new agency generation:** When a new agency is generated, this agency requests member-AVAs from other agencies based on the object-priority of the target.

- **Due to object motion:** Depending on the 3D position of the target, agencies exchange their member-AVAs to employ those AVAs that are suitable for gazing at each target based on the object-priority of the utility-function.

Figure 14 shows an example of the agency restructuring.

(1) If agency manager$_D$ makes unsuccessful object identification with agency$_C$, it decides whether or not to request a member-AVA from agency$_C$. Such an AVA-request message is sent to agency manager$_C$ (Fig. 14 (1)).
(2) When agency manager$_C$ is requested to transfer its member-AVA, it determines the member-AVA that is more suitable for tracking target$_D$ rather than target$_C$, where target$_C$ and target$_D$ indicate the targets of agency$_C$ and agency$_D$, respectively. Agency manager$_C$ orders the selected AVA (member-AVA$_o$) to transfer to agency$_D$ ("Change agency message" in Fig. 14 (2)). Then, member-AVA$_o$ informs agency$_D$ that it has joined agency$_C$ ("Join agency message" in Fig. 14, (2)).
(3) Member-AVA$_o$ begins working as a member of agency$_D$ (Fig. 14 (3)).

3.5.3 Exclusive Interaction in the Inter-agency Layer

A member-AVA transfers between agencies as a result of dynamic interactions in the inter-agency layer. Although this is necessary for continuous tracking, the state of the system is unstable during agency reformation.

The following examples show the actual problems that occur during the dynamic interaction between agencies:

Agency unification: If two agency managers decide to join one side at the same time, both agencies may fail in the agency unification because the destination agency will have disappeared.
Agency restructuring: If multiple agencies transfer their member-AVAs to another agency at the same time, these AVAs may swing between agencies due to radical reformation of agencies.

To resolve these problems, each agency activates an inter-agency cooperative-tracking protocol with only a single agency simultaneously. In addition, each protocol is activated depending on the following conditions:

Agency unification: When agency $_Q$ is requested to join agency$_P$, it decides whether or not to accept the request according to its state as follows:
- If agency$_Q$ is not concerned with any inter-agency interaction, agency$_Q$ accepts the request.
- If agency$_Q$ has requested the agency unification from agency$_P$, agency$_Q$ compares the times when each agency sent the message to one side. Agency$_Q$ accepts the request only if agency$_P$ sent the message earlier than

agency$_Q$.
- If agency$_Q$ is interacting with another agency, agency$_Q$ rejects the request.

Agency restructuring: When an agency receives a request for transfer of a member-AVA, it decides whether or not to accept the request according to its condition as follows:
- If the agency is not concerned with any inter-agency interaction, it accepts the request.
- If the agency is interacting with another agency, it rejects the request.

3.6 Communication with Freelancer-AVAs

In our system, an agency manager communicates not only with other agency managers but also with all freelancer-AVAs through broadcast messages (top row in Fig. 5). A freelancer-AVA must exchange object information with agency managers to achieve the following functions:

- Maintain a consistency of the one-to-one correspondence between an agency and its target. If there is no agency tracking the object detected by a freelancer-AVA, the system can generate a new agency. If an agency already exists that is tracking this object, the system should not generate a new agency.
- Investigate the current state of the system. If a message from a freelancer-AVA is newly received, the search-level of the system increases. Similarly, if a message from an agency manager is newly received, the tracking-level of the system increases based on the number of member-AVAs belonging to the agency.

The contents of this message are listed in Table 1.

As described in Section 3.4.3 describing the agency formation protocol, a freelancer reports object information when it detects an object. An agency manager that has received object information from a freelancer-AVA establishes spatial object identification between its target information and the received object information. Then, the agency manager sends the result of identification to the freelancer-AVA in reply. The freelancer-AVA that has received this reply activates the agency formation depending on the received issue.

On the other hand, an agency manager broadcasts its agency information. A freelancer-AVA that has received the agency information refers to the target position included in the received message. The freelancer-AVA determines the next role depending on the current state of the system and the task-constraint as follows:

(Search-level of the current state) > (Search-level of the task-constraint)

The freelancer-AVA can start tracking the target. If it starts *tracking*, it points its gaze toward the 3D position of the target.

(Search-level of the current state) $\leq$ (Search-level of the task-constraint)
The freelancer-AVA should continue to search for a new object.

4 Performance Characteristics of the System

4.1 Feasibility of Persistent Tracking

In general, it is difficult to guarantee that the system will always be able to track all targets in every situation in the real world. Tracking ability is dependent on various factors (e.g., the numbers of cameras and targets, mechanical limitations of the cameras, etc.). Here, we discuss the relationships between the numbers of cameras and targets, and show the upper limitation of the targets to be tracked simultaneously in the proposed system.

To track a target, an agency is generated. We define an agency as a representation of a target in the system. Based on this definition, the maximum number of targets is equal to the maximum number of agencies. An agency is generated by an AVA that detects a target, and the agency must be attended by at least one member-AVA to track its target. Therefore, the maximum number of agencies 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 assists the agency to keep tracking the target as follows:

- The reconstructed 3D position of the target is useful for object identification not only among AVAs but also among agencies.
- Even if a member-AVA cannot observe its target because of being disturbed by obstacles or other moving objects, it can gaze at the target by receiving the 3D position of the target.
- Comparing the 3D positions of the targets with those of the cameras allows the system to determine which AVA is appropriate for gazing at each target.

Thus, each agency should have at least two member-AVAs for reliable object tracking.

Based on the above discussion, the relationships between the tracking ability of the system and the numbers of targets and AVAs (denoted by n_t and n_a, respectively) can be summarized 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 only one 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 that is not more than n.

The limitation of the number of targets results because 1) an agency receives the information of objects only from its member-AVAs and 2) an agency must 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 AVAs that are not its member-AVAs, it can reconstruct the 3D information of the target even when it has only a single member-AVA. This information exchange allows all agencies to stably track their targets even in case 2 above. In this case, however, member-AVAs must send the information regarding the detected objects not only to their own agency manager but also to other agency managers, thus increasing network load. In particular, the broadcast of object information to all agencies by each member-AVA will produce a huge network load.

Active camera control for vacant agencies: If an agency can exist without any member-AVA, the system can track all targets even in case 3 above. To obtain the information of the target, the agency without any member-AVA must gather the object information from non-member-AVAs. Therefore, it is necessary to resolve the problem of increasing network load described above. In addition, a vacant agency has an essential problem: since the vacant agency cannot control any camera, it is not guaranteed that this agency will 1) keep tracking the target by controlling pan-tilt parameters of a camera(s) or 2) acquire high-resolution images by adjusting the zoom parameter of a camera(s).

To avoid the problems mentioned above, we designed the system such that 1) the agency reconstructs the 3D information of its target only from the information received from its member-AVAs and 2) each agency has at least one member-AVA.

4.2 Completeness of Cooperative-tracking Protocols

In the proposed system, 1) all events occurring in the real world are characterized by the results of object identification and 2) object identification is established when an agency receives a message including the information

Table 5
Cooperative-tracking protocols activated depending on the result of object identification.

Received object information	*Result of object identification*	
	Success	Failure
3D view lines toward detected objects from freelancer-AVA	Agency Formation	Agency Formation
3D view line of the target from member-AVA	Agency Maintenance	Agency Maintenance, Agency Spawning
3D view lines toward non-targets from member-AVA	Agency Maintenance	Agency Spawning
3D point of the target from agency	Agency Unification	Agency Restructuring

of detected objects. Therefore, by verifying the types of cooperative-tracking protocols activated depending on the relation between the type of received object information and the result of object identification, we can confirm the necessity and sufficiency of protocols for multi-target tracking.

Table 5 shows the types the cooperative-tracking protocols designed in accordance with the types and result of object identification.

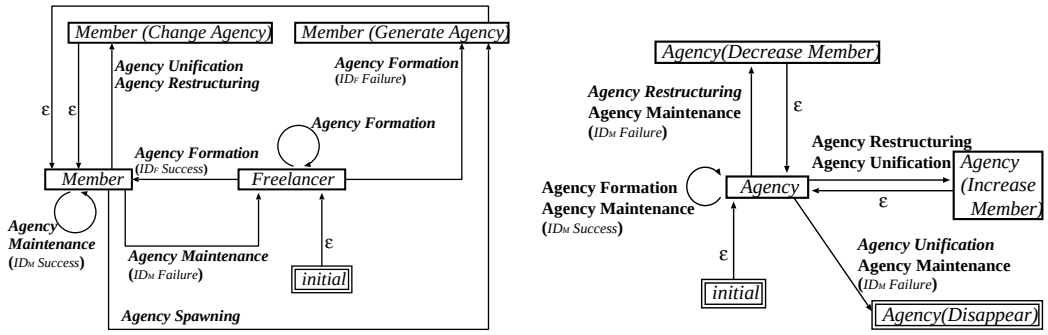
4.3 Soundness of Communication and State Transition

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

Intra-AVA layer: The perception, action, and communication modules exchange 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 an arbitrary time.

Intra-agency layer: An agency manager receives observed information from its member-AVAs and continues spatiotemporal object identification while maintaining its own intrinsic dynamics. Since each agency manager has its own dynamic memory, 1) message transmission from a member-AVA to its agency manager is guaranteed and 2) reliable object identification in the intra-agency layer is achieved.

In addition, several types of information (e.g., 3D position of the target and messages based on the cooperative-tracking protocols) are reported from the agency manager to its member-AVAs by message transmission. A



(a) State transition of the AVA (b) State transition of the agency

Fig. 15. State transition networks of the AVA and agency. The boxes and arrows indicate states and state transitions, respectively. A cooperative-tracking protocol with each arrow causes a state transition. Protocols shown in bold and italics 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.

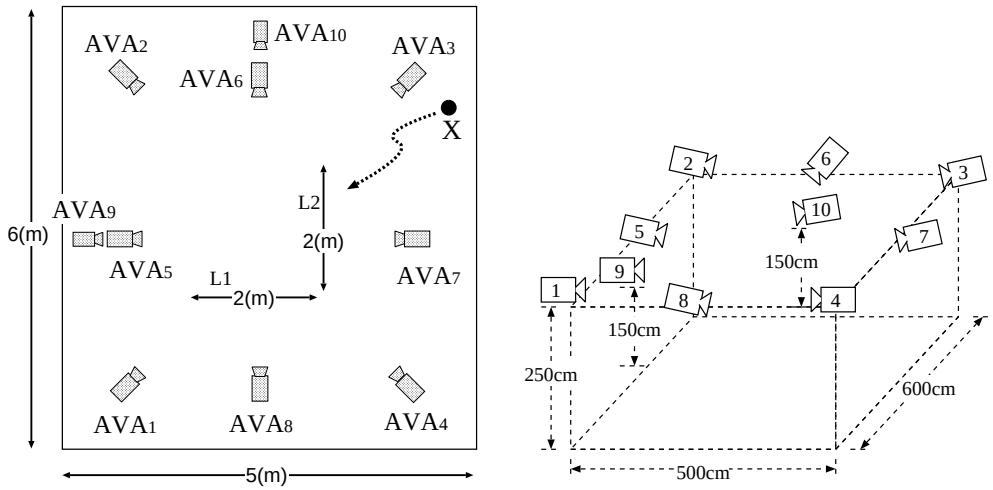
member-AVA accepts only the message from its agency manager to prevent it being affected inconsistently by multiple agencies: for example, a message delay incurs invalid communication between an agency manager and a member-AVA belonging to another agency.

Fig. 15 (a) shows the state transition network of the AVA. All the state transitions of the AVA are caused by the cooperative-tracking protocols described in Section 3, provided that the state transitions indicated by arrows with ϵ occur automatically and immediately.

Inter-agency layer: Each agency broadcasts its agency information to other agencies. 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. 15 (b) shows the state transition network of the agency. All the state transitions of the agency are caused by the cooperative-tracking protocols, provided that the automatic state transition ϵ in the same as that of the AVA.

Thus, the dynamic interactions and state transitions in each layer can be realized without inconsistencies or deadlock.



(a) Top view (b) Camera settings

Fig. 16. Experimental environment.

5 Experiments

We conducted several experiments to verify the effectiveness of cooperative tracking with the proposed system.

5.1 System Organization

In our experiments, we employed ten AVAs, each of which consisted of a network-connected PC with an active camera.

PC: PentiumIII 600MHz × 2 with Linux operating system.
Active camera: FV-PTZ camera (SONY EVI-G20).
Network: 100M-base Ethernet.

The perception, action and communication modules and the dynamic memory are implemented by threads on a PC. The communication module sends messages using UDP. In addition, the internal clocks of all the PCs are synchronized by the Network Time Protocol (NTP)[23]. With these resources, the perception module can capture images and detect objects in the observed image at intervals of about 0.1 [sec] on average. The size of each image is 320 × 240 [pixels].

Figure 16 illustrates the camera layout in the experimental environment. The external camera parameters (i.e., the 3D position and view direction of each camera) were calibrated in advance.

5.2 Designing Utility-function

The utility-function is examined by each agency manager 1) when an agency obtains/releases its member-AVA based on the agency formation/maintenance protocol and 2) when agencies exchange their member-AVAs based on the agency restructuring protocol.

In our experiments, we designed the utility-function as follows:

Search-value of freelancer-AVA$_f$: For a freelancer-AVA to search for new objects, it should monitor as wide an observation area as possible.

Let W_f denote the area size of the floor that is visible from AVA$_f$. The search-value of AVA$_f$ (denoted by V_{S_f}) is determined as follows:

$$V_{S_f} = \alpha_S \times W_f, \tag{6}$$

where α_S is a constant that is determined such that V_{S_f} is well-balanced with the tracking-value. V_{S_f} becomes larger as freelancer-AVAs observe wider areas.

Tracking-value of member-AVA$_m$: For a member-AVA to gaze at its target object without failure, it should 1) hold the region of the target in the center of the observed image and 2) gaze at an object that is close to its camera to allow capture of high-resolution images of the observed object.

Let D_m^n denote the 3D distance between the camera of AVA$_m$ and the target of agency$_n$, and A_m^n denote the angle between the central direction of AVA$_m$'s view angle and the direction from the camera to the target. The tracking-value of AVA$_m$ (denoted by $V_{T_m^n}$) is determined as follows:

$$V_{T_m^n} = \frac{1}{D_m^n} \times \frac{1}{A_m^n}. \tag{7}$$

5.3 Active Tracking changing Pan-Tilt-Zoom Parameters

First, we demonstrated the performance of the pan-tilt-zoom control algorithm with the prediction-based tracking method. The system tracked a computer-controlled mobile robot that moved in the room.

Here, we point out the effectiveness of the proposed tracking method by the following comparative study. Figures 17 and 18 show observed images taken by AVA$_4$ in the proposed system (called **system A**) and the simple active background subtraction system without modular (perception and action) functions or prediction-based camera control (**system B**), respectively. When each system worked, the robot moved along the same trajectory at the same speed.

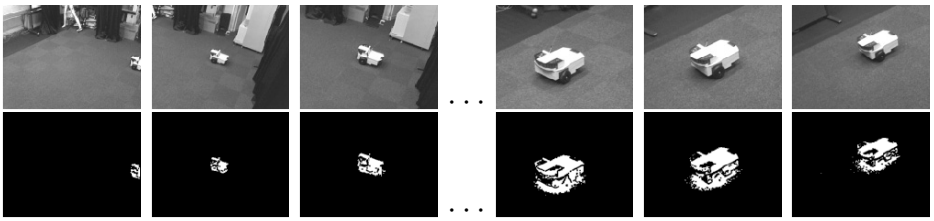


Fig. 17. Example of observed image sequence taken by the proposed system. Top: input images, Bottom: detection images.

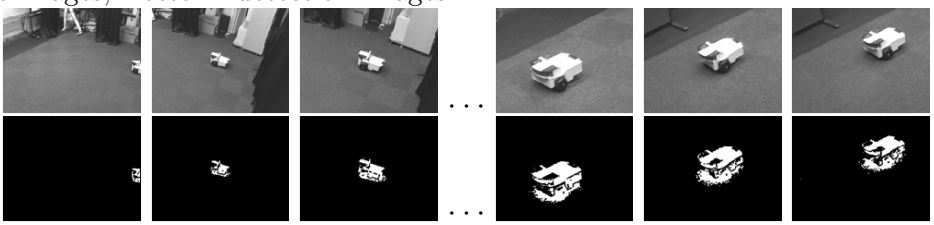


Fig. 18. Example of observed image sequence taken by the simple active background subtraction system. Top: input images, Bottom: detection images.

Table 6
(1) Distance between the image center and the centroid of the detected object region, and (2) Area size of the detected object.

	(1) average	(1) variance	(2) average	(2) variance
System A	44.0[pixels]	6.7[pixels]	5083[pixels]	145[pixels]
System B	16.7[pixels]	1.5[pixels]	5825[pixels]	108[pixels]

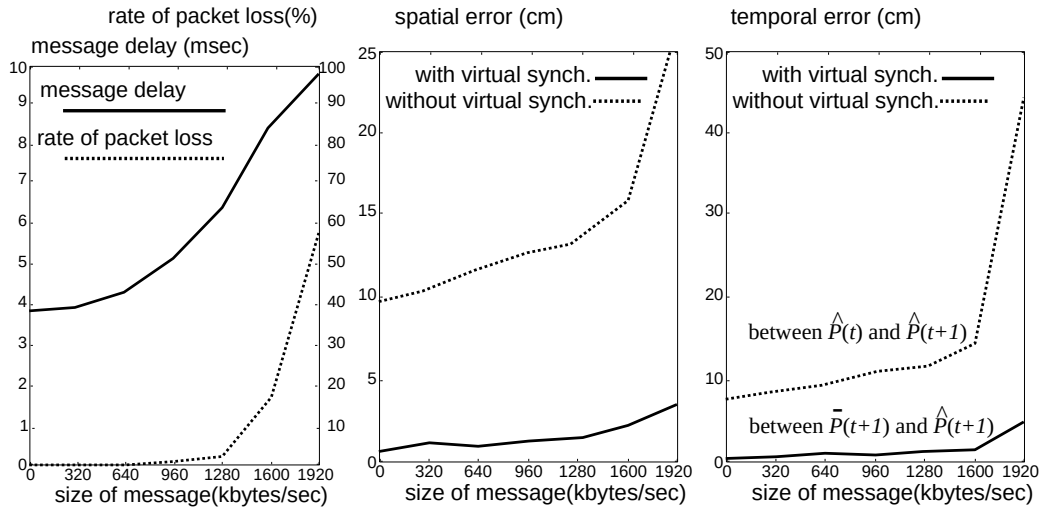
The average and variance of (1) the distance between the image center and the centroid of the detected object region and (2) the area size of the detected object are shown in Table 6. These results indicated that the proposed system improved the stability of tracking and the feasibility of acquiring high-resolution images of a target.

5.4 Performance Evaluation of the Virtual Synchronization

We conducted experiments with systems with/without the virtual synchronization. To verify the effectiveness of the virtual synchronization against not only the asynchronous observations but also 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 area. L1 and L2 in Fig. 16 (a) show the trajectories of the robots.

Figure 19 (a) shows variations in network conditions when the packet size of the vain broadcast messages is changed. The errors of spatial identification in



(a) (b) (c)

Fig. 19. Performance evaluation of the virtual synchronization: (a) Delay of messages (solid line) and Rate of packet loss (dotted line), (b) Error in spatial object identification, (c) Error in temporal object identification. The horizontal axis indicates the total size of the vain broadcast messages per second.

Fig. 19 (b) denote the average distances between the reconstructed 3D position and the 3D view line detected by each member-AVA. The errors of temporal identification in Fig. 19 (c) denote the average distances between the 3D positions of the target, which were estimated/reconstructed at different times (i.e., $\bar{P}(t+1)/\widehat{P}(t)$ and $\widehat{P}(t+1)$ in the system with/without the virtual synchronization). Note that since temporal identification was established at intervals 0.1 [sec] in this experiment, the robot moved 5 [cm] between subsequent temporal identifications.

The result indicated that the virtual synchronization helps both spatial and temporal object identification, especially in the case of poor network conditions.

5.5 Verifying Cooperative-tracking Protocols

Finally, we verified the effectiveness of the cooperative-tracking protocols. Our experimental results demonstrated flexible and reliable multi-target tracking by cooperation among AVAs.

To verify the effectiveness of the task representation with the task-constraint and the object-priority, we performed two experiments in the same environment. In each experiment, we provided the following parameters as the task representation.

Experiment 1
Task-constraint: Search-level was 0.1. Tracking-level was 0.9.
Object-priority: Values for all objects were 1.0.
Experiment 2
Task-constraint: Search-level was 0.3. Tracking-level was 0.7.
Object-priority: Values for $object_1$ and $object_2$ were 1.0 and 0.5, respectively [7].

In both experiments, the system tracked two people. $Object_1$ first came into the observation space from the location 'X' (shown in Fig. 16 (a)). Next, $object_2$ came into the observation space, and both objects then moved freely.

First, we present the results of the first experiment.

Figure 20 shows the partial image sequences observed by each AVA. The images on the same row were taken by the same AVA. Figure 20 shows images taken by AVA_1, AVA_2, AVA_4, AVA_5, AVA_7, AVA_8, and AVA_9 as examples. The images on the same column were taken at almost the same time. The regions enclosed by red and blue lines in the images show the detected regions of $object_1$ and $object_2$, respectively.

Figure 21 shows the role of each AVA and the agency organization at such a moment when the same column of images in Fig. 20 were observed. Green circles indicate freelancer-AVAs. Red and blue circles indicate member-AVAs belonging to $agency_1$ and $agency_2$, respectively. Red and blue squares indicate computed locations of $object_1$ and $object_2$ respectively. 'X' shown in Fig. 21 indicates location X in Fig. 16 (a).

In this experiment, the system worked as follows:

a: Initially, each AVA searched for objects independently.
b: AVA_5 first detected $object_1$ (Fig. 20, 5-b), and then $agency_1$ was formed.
c: All AVAs, with the exception of 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 (Fig. 20, 5-c).
d: AVA_5 detected a new object (Fig. 20, 5-d). AVA_5 then regarded this object as the target ($object_2$) and generated $agency_2$.
e: In this experiment, the object-priorities of both $object_1$ and $object_2$ were equivalent. The agency restructuring, therefore, balanced the numbers of the member-AVAs in $agency_1$ and $agency_2$.
f: Since $object_1$ came close to $object_2$, no AVA could distinguish these objects, and object identification between two agencies was successful. Then, the agency unification protocol merged $agency_2$ into $agency_1$.

[7] In this experiment, the system gave object-priorities of 1.0 and 0.5 to the objects detected first and second, respectively

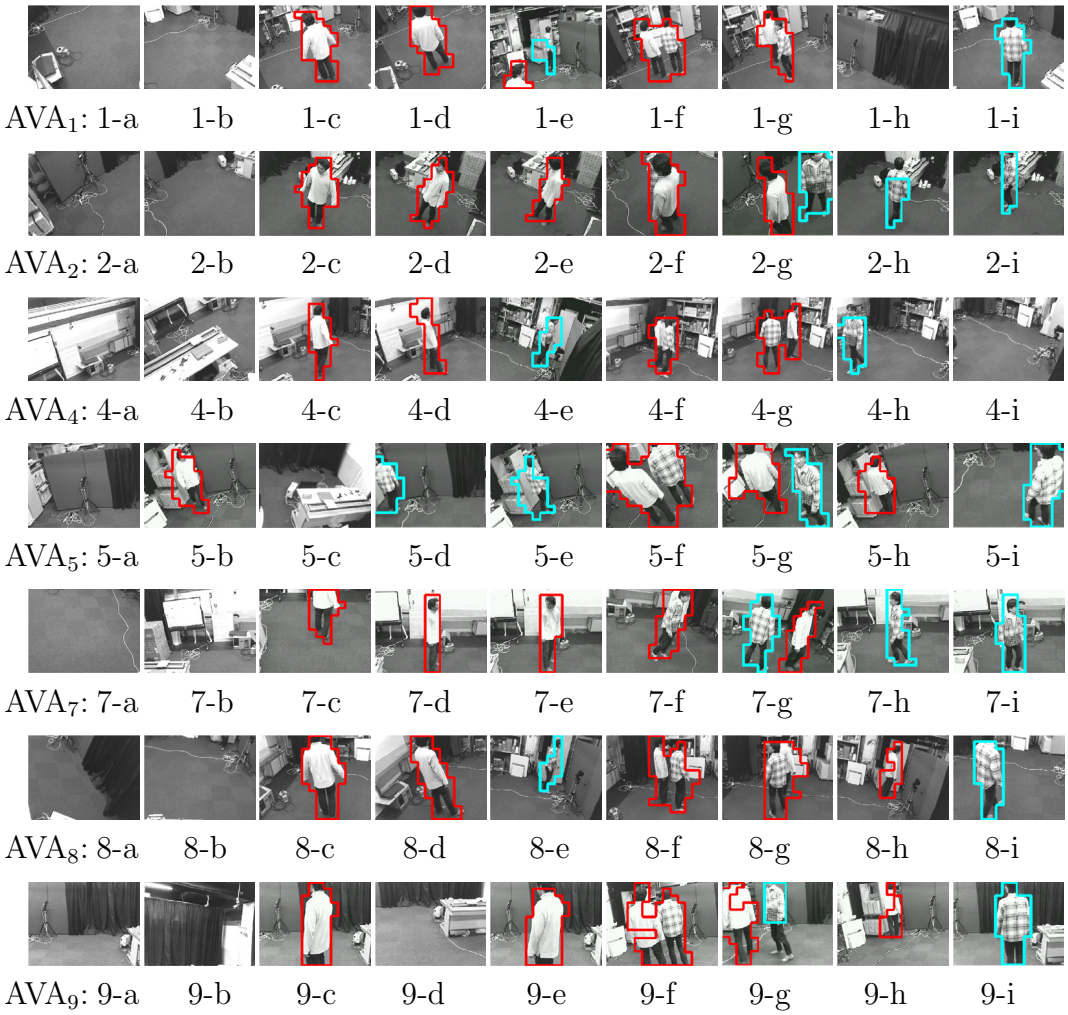


Fig. 20. Experiment 1: Partial image sequences observed by AVAs.

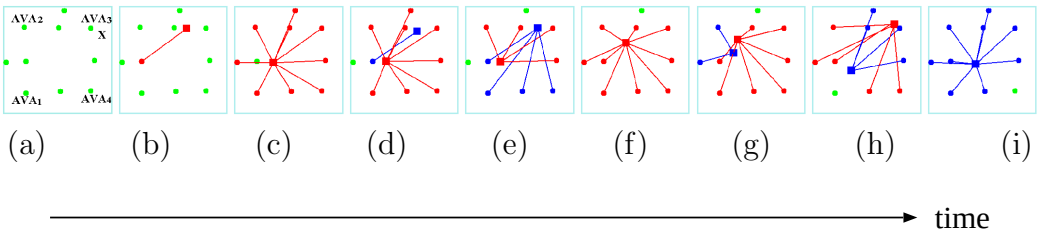
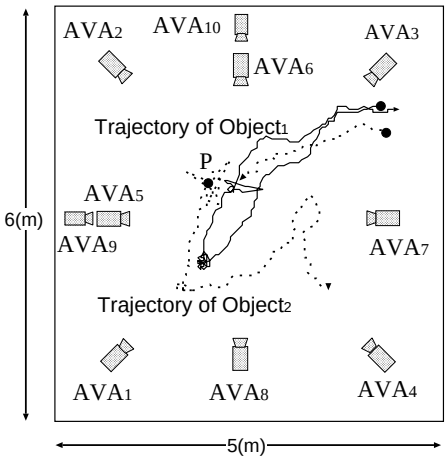
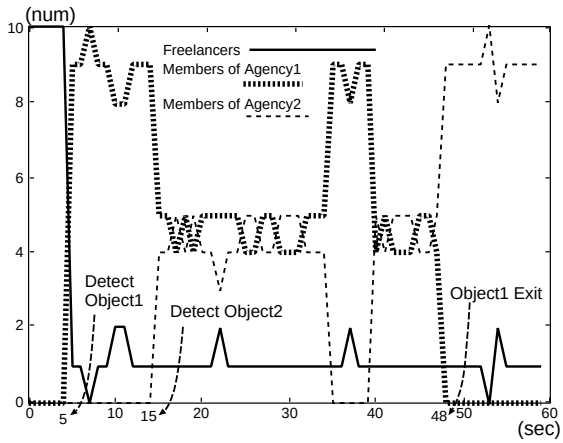


Fig. 21. Experiment 1: Transitions of AVA roles and agency organization.

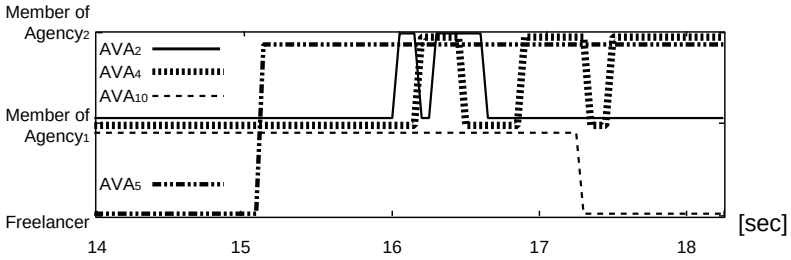
g: When the objects were separated, agency$_1$ detected a 'new' object. Then, it activated the agency spawning protocol to generate agency$_2$ again for object$_2$. At this time, agency$_2$ read the information of object$_2$ obtained in the past and compared its target information with the read object information (i.e., the past trajectory of object$_2$) for temporal object identification. Since this object identification was successful, the newly detected object was regarded as object$_2$.



(a) Trajectories of the targets



(b) The number of AVAs performing each role ('*search*' and '*tracking* of each object')



(c) Histories of AVA roles

Fig. 22. Experiment 1: Experimental results.

h: Object$_1$ was leaving the scene.

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

Figure 22 (a) shows the trajectories of the targets, which were reconstructed by the agencies. When the agency spawning was initiated for object$_2$, tracking of object$_2$ was started at location P.

Figure 22 (b) shows the dynamic population changes of freelancer-AVAs, AVAs tracking target$_1$, and those tracking target$_2$. The horizontal and vertical axes indicate the time and the number of AVAs undertaking each role, respectively. This graph shows the system states at intervals of 1 [sec]. Figure 22 (c) shows the histories of the AVAs' roles. The horizontal and vertical axes indicate the time and the roles of the AVAs. Bumps in Fig. 22 (b) and (c) indicate the temporal states of the system while each AVA was changing its role depending on the target motions. These results indicated that the system as a whole worked to cope with the dynamic situations in the scene by dynamically changing the role of each AVA.

In the second experiment, we verified the efficacy of the task representation.

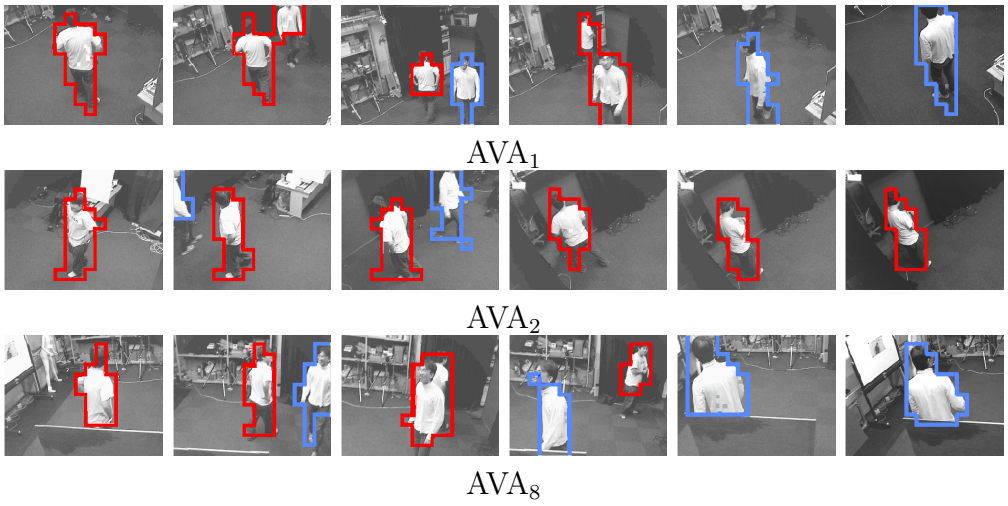


Fig. 23. Experiment 2: Partial image sequences observed by AVAs. These images were taken at intervals of about 1 [sec].

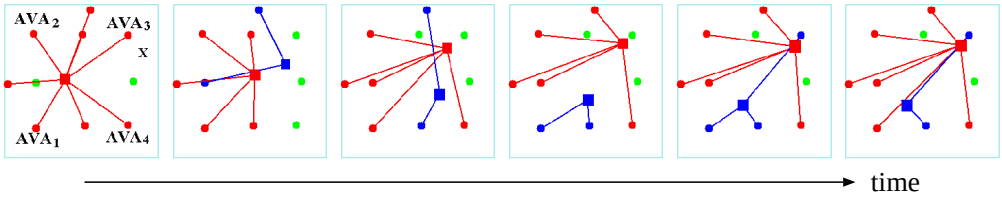


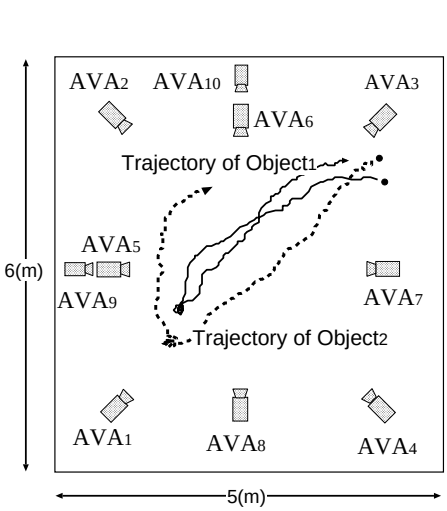
Fig. 24. Experiment 2: Transitions of AVA roles and agency formations.

Figures 23 and 24 show partial image sequences observed by AVA_1, AVA_2, and AVA_8 as well as the dynamic population changes of freelancer-AVAs and member-AVAs. Figure 24 shows the role of each AVA and the formation of each agency. Figures 25 (a), (b) and (c) show the results of the same analyses as shown in Fig. 22 (a), (b) and (c), respectively. The system detected object_1 and object_2 at 7 seconds and 28 seconds, respectively, and kept tracking both objects. Since two objects had different object-priorities, the numbers of member-AVAs in agency_1 and agency_2 were different from each other.

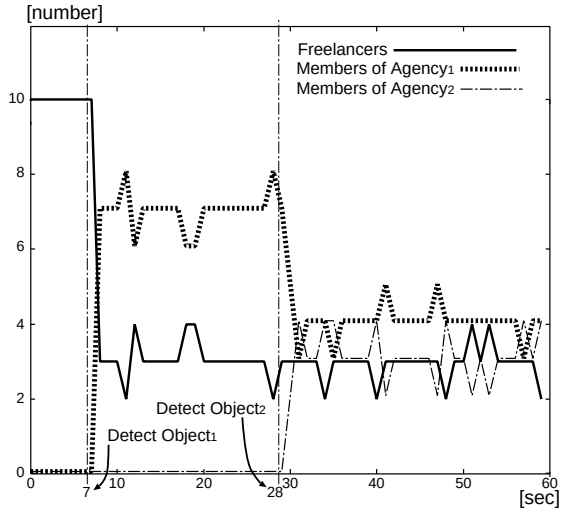
As can be seen from these results, the dynamic interactions among AVAs and agencies enable the system to persistently track multiple objects taking into account the given task, i.e., the given task-constraint and object-priority.

Although the system always functioned as designed in all experiments, its behavior was not entirely satisfactory depending on the given utility-function:

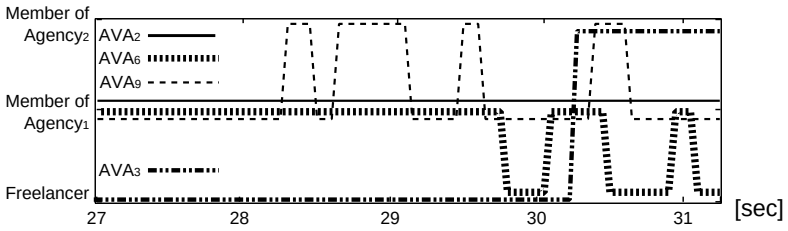
- If the weight constant α_S is not determined appropriately, dynamic role assignments among AVAs are disrupted; for example, an AVA that should gaze at a target object is selected as a freelancer-AVA. That is, the system changes its behavior sensitively depending on the relationship between the search-value and the tracking value.



(a) Target motion trajectories reconstructed by agencies



(b) The number of AVAs performing each role ('*search*' and '*tracking* of each object')



(c) Histories of AVA roles

Fig. 25. Experimental results 2: Experimental results.

- The tracking-value defined in Section 5.2 could select a pertinent agency organization in most cases. The system, however, sometimes assigned an unsuitable target to an AVA; for example, an AVA that should gaze at object_1 may be joined to an agency tracking object_2. Such unsuitable target assignment results not only in difficulty in acquiring high-resolution images of a target but also in unstable system states, including the case in which an AVA returns to an agency at quite frequent intervals. The target assignment is strongly dependent on the definition of the tracking-value.

Based on these observations, it will be important in future studies to develop a method for automatically designing the utility-function.

6 Concluding Remarks

In this paper, we proposed a system for real-time multi-target tracking employing a distributed active-camera system. The proposed system has the following

properties:

- Multiple parallel processes interact dynamically with each other, resulting in a system that functions as a whole for cooperative tracking.
- The system is classified into three layers to efficiently establish various types of object identification.
  Intra-AVA layer: Perception, action and communication modules work together as a single AVA by interacting dynamically with each other.
  Intra-agency layer: A group of AVAs in the same agency share object information to track a target.
  Inter-agency layer: To adaptively restructure agencies taking into account target motions, the agencies exchange agency information with each other.
- Employing the dynamic memory architecture achieves dynamic interactions in each layer without synchronization. The system is endowed with high reactivity.

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

While we proposed the three-layered interaction architecture for real-time cooperative tracking, we consider the three-layered architecture to be adaptable to other cooperative systems with autonomous agents:

Intra-AVA layer: To perform versatile and complex behaviors, an intelligent autonomous agent (e.g., AVA) should consist of several functional modules required for the task.
Intra-agency layer: Agents, all of which have the same purpose, should form an agency to work together in a cooperative manner.
Inter-agency layer: For agencies to work cooperatively by negotiation, they should interact with each other.

As mentioned above, our real-time tracking system can work under complicated dynamic situations in the real world. We believe that this system represents fundamental technology to realize various real-world vision systems. For the practical application of our system to real-world vision systems, other issues in Computer Vision should be discussed. Here, we summarize several aspects examined in the present study and discuss directions for future work.

1. The number of trackable targets:

To enable the system to track more targets than the number of AVAs, the system can be modified such that an agency without any member-AVAs can be generated. In this definition, it is necessary to consider how a vacant agency can continuously obtain the target information? A vacant agency must receive the object information detected by non-member-AVAs (i.e., freelancer-AVAs

and member-AVAs in other agencies). While a freelancer-AVA broadcasts the information of the detected objects, a member-AVA sends the information only to its agency manager. For the member-AVA to report the object information to other agencies without increasing the network load, the message should be sent only to agencies that require the information. Such a member-AVA that sends information to other agencies is called a *Supporter-AVA*. While an AVA can be a supporter-AVA for multiple agencies, it must belong to only one agency as a member-AVA to avoid inconsistent camera-control from different agencies.

2. Camera configuration planning:

There have been many studies to determine effective camera configuration for a given task [25]. Similarly, the effective camera configuration for object tracking should be planned depending on the given task.

3. Tracking with isolated camera configuration:

In all the experiments conducted in the present study, visual fields of all AVAs overlapped with each other. In the real world, however, this situation does not always hold true. To keep tracking a target even if cameras are embedded sparsely in the scene, the system must employ not only the 3D trajectory of the target but also other information for object identification:

- Appearance-based object identification is useful (e.g., [24]).
- In general, for object identification among widely distributed cameras, the system searches through an enormous number of candidates for an optimal solution. In [26], several constraints on the route and lapse assist object identification in addition to the appearance information.

4. Capturing selective object image depending on the task:

The required information of a target varies depending on the task; e.g., whole body, face, hands, etc. For example:

- To acquire the precise volumetric and appearance information of an object (e.g., [27]), the system should control cameras to capture high-resolution and meaningful object images.
- For individual identification, information on human face is significant. In [28], the human head was detected based on the appearance and feature models.

`acknowledge` This study is supported by PRESTO program of Japan Science and Technology Agency (JST), National Project on Development of High Fidelity Digitization Software for Large-Scale and Intangible Cultural Assets, and Grant-in-Aid for Scientific Research on Priority Areas, No.13224051.

References

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

[2] T. Kanade, "Cooperative multisensor video surveillance," in *Proc. of Image Understanding Workshop*, pp.3–10, 1997.

[3] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Occlusion Robust Tracking Utilizing Spatio-Temporal Markov Random Field Model," in *Proc. of International Conference on Pattern Recognition 2000*, Vol.1, pp142–147, 2000.

[4] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(7), pp.780–785, 1997.

[5] D. Murray and A. Basu, "Motion tracking with an active camera," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(5), pp.449–459, 1994.

[6] Q. Cai and J. K. Aggarwal, "Tracking human motion in structured environments using a distributed camera system," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 21(11), pp.1241–1247, 1999.

[7] J. MacCormick and A. Blake, "A probabilistic exclusion for tracking multiple objects," In *Proc. of International Conference on Computer Vision*, pp.572–578, 1999.

[8] I. Haritaoglu, D. Harwood, and L. S. Davis, "An appearance-based body model for multiple people tracking," in *Proc. of 15th International Conference on Pattern Recognition*, Vol.4, pp.184–187, 2000.

[9] B. S. Rao and H. Durrant-Whyte, "A Decentralized Bayesian Algorithm for Identification of Tracked Targets," in *IEEE Transaction on Systems, Man, and Cybernetics*, Vol.23, No.6, p.1683–1698, 1993.

[10] B. Horling, R. Vincent, R. Mailler, J. Shen, R. Becker and K. Rawlins and Victor Lesser, "Distributed Sensor Network for Real Time Tracking," in *Proc. of the 5th International Conference on Autonomous Agents*, pp.417–424, 2001.

[11] A. Nakazawa, H. Kato, S. Hiura and S. Inokuchi, "Tracking multiple people using Distributed Vision Systems," in *Proc. of IEEE International Conference on Robotics and Automation 2002*, pp.2974–2981, 2002.

[12] E. Borovikov and L. S. Davis, "A Distributed System for Real-time Volume Reconstruction," in *Proc. of IEEE Workshop on Computer Architecture for Machine Perception*, pp.183–189, 2000.

[13] P. Rander, P.J. Narayanan, and T. Kanade, "Virtualized Reality: Constructing Time-Varying Virtual Worlds from Real Events," in *Proc. of IEEE Visualization '97*, pp.277–283, 1997.

[14] D. R. Karuppiah, Z. Zhu, P. Shenoy, and E. M. Riseman, "A Fault-Tolerant Distributed Vision System Architecture for Object Tracking in a Smart Room," in *Proc. of International Conference on Computer Vision Systems 2001*, pp.201–219, 2001.

[15] T. Sogo, H. Ishiguro and M. M. Trivedi, "Real-time target localization and tracking by N-ocular stereo," in *Proc. of IEEE Workshop on Omnidirectional Vision*, pp.153-160, 2000.

[16] N. Yoshida and T. Fuki, "Target Tracking Using Tuple-Space-Based Mobile Agents," in *Proc. of 19th IASTED International Conference on Applied Informatics*, pp.389–393, 2001.

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

[18] K. Toyama, J. Krumm, B. Brumitt and B. Meyers, "WallFlower: Principle and Practice of Background Maintenance," in *Proc. of International Conference on Computer Vision*, pp.255–261, 1999.

[19] C. Stauffer and E. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. of Computer Vision and Pattern Recognition*, Vol.II, pp.246–252, 1999.

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

[21] G. P. Stein, "Tracking from Multiple View Points: Self-calibration of Space and Time," in *Proc. of Computer Vision and Pattern Recognition*, Vol. I, pp.521–527, 1999.

[22] N. Ukita and T. Matsuyama, "Real-time Multi-target Tracking by Cooperative Distributed Active Vision Agents," in *Proc. of 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp.829–838, 2002.

[23] D. L. Mills. "Internet time synchronization: the network time protocol," *IEEE Transaction of Communications*, 39(10), pp.1482–1493, 1991.

[24] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, 29(1), pp.5–28, 1998.

[25] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai. "A survey of sensor planning in computer vision," *IEEE Transaction on Robotics and Automation*, 11(1), pp.86–104, 1995.

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

[27] T. Wada, X. Wu, S. Tokai, and T. Matsuyama. "Homography based parallel volume intersection: Toward real-time volume reconstruction using active cameras," in *Proc. of IEEE Workshop on Computer Architecture for Machine Perception*, pp.331–339, 2000.

[28] K. Yachi, T. Wada, and T. Matsuyama. "Human head tracking using adaptive appearance models with a fixed-viewpoint pan-tilt-zoom camera," in *Proc. of Fourth International Conference on Automatic Face and Gesture Recognition*, pp.150–155, 2000.