

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

Norimichi Ukita[†] and Takashi Matsuyama
Graduate School of Informatics, Kyoto University

This paper presents a real-time cooperative multi-target tracking system. The system consists of a group of Active Vision Agents (AVAs), where an AVA is a logical model of a network-connected computer with an active camera. All AVAs cooperatively track their target objects by dynamically interacting with each other. As a result, the system as a whole can track multiple moving objects simultaneously under complicated dynamic situations in the real world. To implement the real-time cooperation among AVAs, we designed a three-layered interaction architecture. In each layer, parallel processes mutually exchange various information for the effective cooperation. To realize the information exchange in real time, we employed the dynamic memory architecture[5]. Experimental results demonstrate that AVAs cooperatively track their targets in real-time while adaptively changing their roles.

能動視覚エージェント群による複数対象の実時間協調追跡

浮田宗伯[†] 松山隆司
京都大学大学院情報学研究科

能動視覚エージェント群の協調動作による複数対象の実時間追跡システムを提案する。能動視覚エージェント (Active Vision Agent, 略して AVA) とは、ネットワーク接続された計算機と能動カメラの組の論理モデルである。全ての AVA が相互に検出対象情報を交換しながら協調的にその役割の変更することによって、システム全体として複雑な動的環境下における複数対象の移動を同時に追跡することが可能となる。AVA 間の実時間協調を実現するために、我々は三層構成のインタラクションアーキテクチャを提案した。各層では、並列プロセス群が効果的な協調動作のために必要な情報を相互に交換している。この情報交換の実時間性を保証するために、ダイナミックメモリを利用した。また実験により、提案システムが適応的に役割分担を行いながら複数対象を実時間で追跡可能なことを確認した。

1 Introduction

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

To realize real-time flexible tracking in a wide-spread area, we employ the idea of *Cooperative Distributed Vision*[1]. Cooperative distributed vision system consists of a group of network-connected computers with active camera(s). A group of spatially distributed active cameras enable continuous wide-area observation as well as detailed measurement of 3D object information.

In this paper, we propose a real-time cooperative tracking system that gazes at multiple objects simultaneously. The system consists of communicating *Active Vision Agents* (AVAs, in short), where an AVA is a logical model of a network-connected com-

puter with an active camera. For real-time object tracking by multiple AVAs, we have to solve many problems (e.g., how to design an active camera for dynamic object detection[1] and how to realize real-time object tracking with an active camera[5]). In this paper, we put our focus upon how to realize a real-time cooperation among AVAs.

To implement the real-time cooperation among AVAs, we propose a three-layered interaction architecture. In each layer, parallel processes exchange different kinds of information for effective cooperation. To realize a real-time information exchange and processing, we employ the dynamic memory architecture proposed in [5]. The dynamic interaction in each layer allows the total system to track multiple moving objects under complicated dynamic situations in the real world.

2 Cooperative Tracking

2.1 Architecture of AVA and Its Functions

Each AVA possesses a single *Fixed-Viewpoint Pan-Tilt-Zoom* (FV-PTZ) camera[1]: its projection cen-

[†] 現所属：奈良先端科学技術大学院大学情報科学研究科 (Nara Institute of Science and Technology)

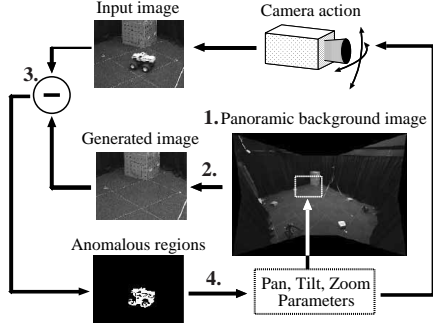


Figure 1: Object detection and tracking using an FV-PTZ camera.

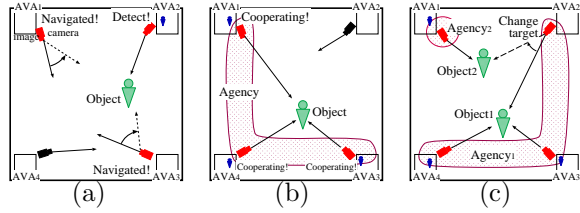


Figure 2: Basic scheme for cooperative tracking: (a) Gaze navigation, (b) Cooperative gazing, (c) Adaptive tracking.

ter stays fixed irrespectively of any camera rotations and zoomings. An AVA can track the moving object as illustrated in Fig.1[1][5]:

1. Generate a wide panoramic image of the scene by mosaicing multiple images observed by changing pan, tilt and zoom parameters.
2. Extract a background image from the panoramic image according to the current pan, tilt and zoom parameters.
3. Compute difference between the generated background image and an observed image.
4. If anomalous regions are detected in the difference image, select one and control the camera parameters to track the selected target.

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

In our system, an agent (i.e., AVA) corresponds to a single camera. All AVAs can, therefore, control their own cameras to gaze at the targets. Nishio-Ohta[2] and Nakazawa *et al*[3], on the other hand, defined an agent to correspond to the information of each object detected by the system. This definition forces each agent to examine the object information detected by all cameras for tracking its target. In addition, multiple agents may control a camera inconsistently in tracking the target, if the system employs active cameras. As we can see, our definition of the agent has the advantage in that it has the one-to-one correspondence between the agent and the camera.

2.2 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 tracks multiple targets. Following are the tasks of the system:

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

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

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

Real-time and reactive processing:

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

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

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

3 Task Specification

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

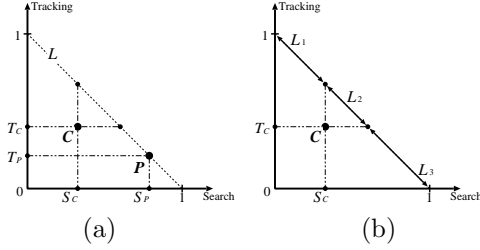


Figure 3: System state graph: (a) Current state and Task-constraint, (b) Three states of the system.

3.1 Task-Constraint

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

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

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

Def. 1 (Current state $P(S_P, T_P)$) This parameter (P in Fig.3) represents the search-level (S_P) and the tracking-level (T_P) at the present time. The range of P is on the line L in Fig.3 (a). That is, $(S_P + T_P)$ is always 1.

Def. 2 (Task-constraint $C(S_C, T_C)$) This parameter (C in Fig.3) represents the minimum search-level (S_C) and tracking-level (T_C), which the system has to keep while working. This parameter is determined by a user depending on the task given to the system.

Followings are the system states determined by the relations between the task-constraint and the current state (Fig.3 (b)):

Shortage of search-level: P is on L_1 .

Task satisfaction: P is on L_2 .

Shortage of tracking-level: P is on L_3 .

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

3.2 Object-Importance

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

Def. 3 (Object-importance I_P) Let I_P denote the object-importance of the target of agency P . The range of the object-importance is $0 \leq I_P \leq 1$. The number of the member-AVAs in agency P (denoted by M_P) is determined by the object-importance of the target: $M_P = (\text{The total number of AVAs}) \times (I_P/S)$. Provided that S is the total sum of the object-importance I_1, \dots, I_A , where A is the total number of the agencies.

3.3 Utility-Function

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

- **Search-value of a freelancer-AVA** is determined by a search ability of each freelancer-AVA.
- **Tracking-value of a member-AVA** is determined by a tracking ability of each member-AVA.

This utility-function can be designed to be adapt itself to the task given by a user¹.

4 Dynamic Interaction for Cooperative Tracking

In our system, parallel processes cooperatively work by dynamically interacting with each other. As a result, the system as a whole works as a tracking system. By composing the system as a group of multiple processes, we can represent various complex behaviors of the total system through the interaction between processes. Designing the total system can be, therefore, reduced to designing each process. Furthermore, the states and those transitions of the system increase enormously by combining with each other. We believe that this property allows the system to cope with complicated situations in the real world in contrast to centralized processing system[6].

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

4.1 Intra-AVA layer

In the bottom layer in Fig.4, perception, action and communication modules that compose an AVA exchange the time-series information with each other via the dynamic memory[5]² possessed by each AVA. The interaction among three modules materializes the functions of the AVA.

Perception: This module continues to capture images and detect objects in the observed image. Let

¹ We give an example in Sec.5.1.

² With the dynamic memory, all modules can exchange their information asynchronously at any time.

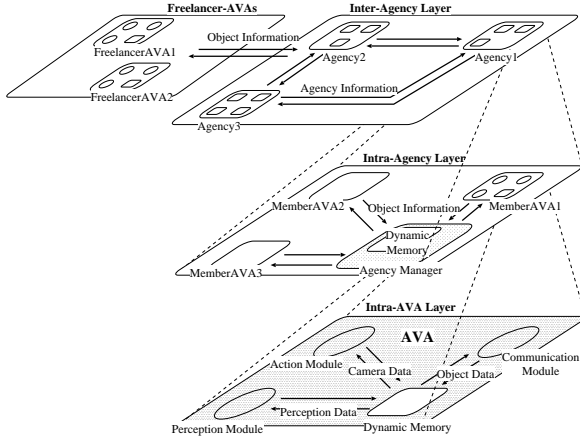


Figure 4: Three layers in the system.

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

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

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

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

4.2 Intra-Agency layer

As defined before, a group of AVAs which track the same target form an agency. The agency formation means the generation of an *agency manager*, which is an independent parallel process to coordinate interactions among its member AVAs. In our system, an agency should correspond one-to-one to a target. To make this correspondence dynamically established and persistently maintained, the following two kinds of object identification are required in the intra-agency layer (the middle layer in Fig.4):

Spatial object identification: The agency manager has to establish object identification between the groups of the 3D view lines detected and transmitted by its member AVAs. The manager checks distances between those 3D view lines detected by different member AVAs and computes the 3D target position from a set of nearly intersecting 3D view lines. The manager employs what we call the *Virtual Synchronization* to virtually adjust observation timings of the 3D view lines (see 4.2.1 for details). Note that the manager may find none or multiple sets of such nearly intersecting 3D view lines. To cope with these situations, the manager conducts the following temporal object identification.

Temporal object identification: The manager records the 3D trajectory of its target, with which the 3D object position(s) computed by spatial object identification is compared. That is, when multiple 3D locations are obtained by spatial object identification, the manager selects the one closest to the target trajectory. When spatial object identification failed and no 3D object location was obtained, on the other hand, the manager selects such 3D view line that is closest to the target trajectory. Note that when an agency contains only a single AVA, the member AVA just conducts appearance-based 2D tracking by itself.

4.2.1 Virtual Synchronization for Spatial Identification

Since AVAs capture images autonomously, member AVAs in an agency observe the target at different moments. Furthermore, the message transmission over the network introduces unpredictable delay between the observation timing by a member AVA and the object identification timing by the agency manager. In [3] and [4], the newest information gathered from each camera is considered to be observed at the same time. Such approximate methods, however, break down under complicated situations and network congestion. To solve this problem, we introduce the dynamic memory into an agency manager, which enables the manager to virtually synchronize

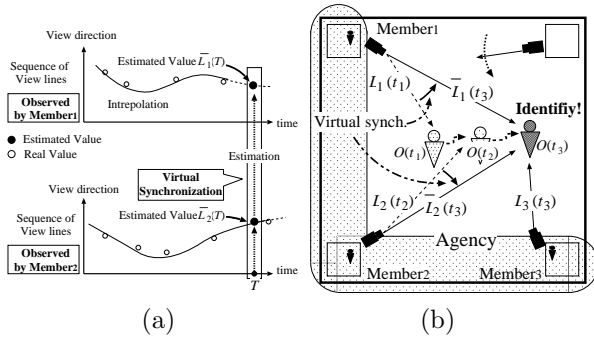


Figure 5: Virtual synchronization for spatial object identification: (a) Read values from the dynamic memory, (b) Spatial identification.

any asynchronously observed/transmitted data. We call this function *Virtual Synchronization* by the dynamic memory.

Fig.5 shows the mechanism of the virtual synchronization. All 3D view lines computed by each member AVA are transmitted to the agency manager, which then records them into its internal dynamic memory. In Fig.5, for example, 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.5).

4.2.2 Virtual Synchronization for Temporal Identification

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

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

4.2.3 Agency Formation Protocol

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

When a freelancer-AVA finds a new object, it requests from the existing agencies object identification between the newly detected object and the target of each agency (Fig.6, (1)). Depending on

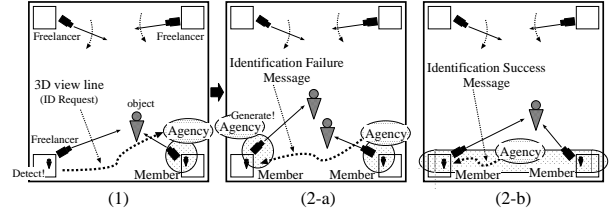


Figure 6: Agency formation protocol.

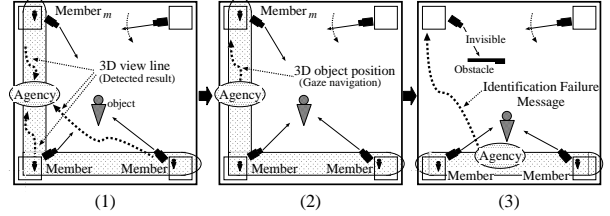


Figure 7: Agency maintenance protocol.

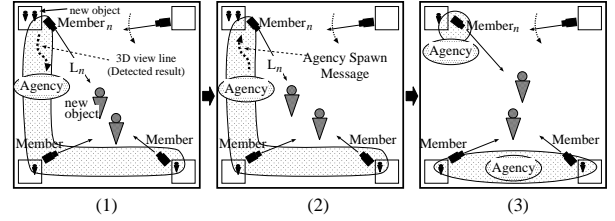


Figure 8: Agency spawning protocol.

whether or not the result of object identification is successful, the freelancer-AVA works as follows:

- **When no agency established a successful identification**, the freelancer-AVA that finds the new object starts a new agency manager and joins this agency (Fig.6, (2-a)).
- **When an agency established a successful identification**, the freelancer-AVA joins the agency that has made successful identification (Fig.6, (2-b)).

4.2.4 Agency Maintenance Protocol

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

After an agency is generated, the agency manager continues spatial and temporal object identifications for cooperative tracking (Fig.7, (1)). If temporal object identification between the target of the agency and the object detected by member-AVA_m fails, the agency manager reports the 3D position of the target to member-AVA_m. This information navigates the gaze of member-AVA_m towards the target (Fig.7, (2)). Nevertheless, if the failure of identification continues for a long time, the agency manager puts member-AVA_m out of the agency (Fig.7, (3)).

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

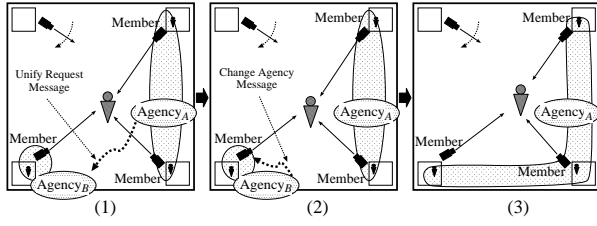


Figure 9: Agency unification protocol.

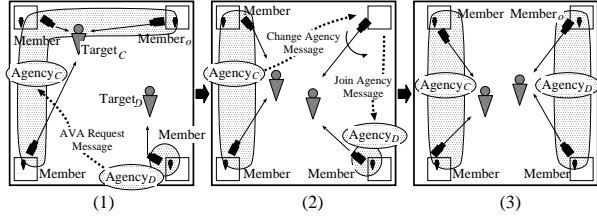


Figure 10: Agency restructuring protocol.

4.2.5 Agency Spawning Protocol

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

After spatial and temporal object identifications, the agency manager may find such a 3D view line(s) that does not correspond to the target. Let L_n denote such 3D view line detected by member-AVA $_n$ (Fig.8, (1)). The manager requires other agencies to compare L_n with their own targets for object identification. If none of the identification is successful, the manager orders member-AVA $_n$ to generate a new agency (Fig.8, (2)). Member-AVA $_n$ then joins a new agency (Fig.8, (3)).

4.3 Inter-Agency layer

In multi-target tracking, the system should adaptively allocate resources: the system has to adaptively determine which AVAs should track which targets. To realize this adaptive resource allocation, the information about targets and member AVAs is exchanged between agency managers (the top layer in Fig. 4).

The dynamic interactions between agency managers are triggered based on object identification of target 3D locations across agencies. Note that here also the virtual synchronization between a pair of 3D target locations is employed to increase the reliability of object identification. Depending on the result of this inter-agency object identification, either of the following two protocols are activated.

4.3.1 Agency Unification Protocol

An *Agency Unification* protocol defines the merging procedure of the agencies, both of which happen to track the same object. This protocol is activated when the result of object identification between the agencies is successful. Figure 9 shows an example.

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

- When the agency considers multiple objects in the scene as a single object because of the identification failure.
- When a single object is regarded as multiple objects because of the identification failure, and then multiple agencies are formed for the same object.

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

4.3.2 Agency Restructuring Protocol

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

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

In an example illustrated in Fig.10, agency $_D$ requests a member-AVA from agency $_C$, and then agency $_C$ transfers member-AVA $_o$ to agency $_D$.

4.3.3 Communication with Freelancers

An agency manager communicates with freelancer-AVAs as well as with other managers (the top layer in Fig.4). As described in the agency formation protocol in Sec.4.2.3, a freelancer-AVA activates the communication with agency managers when it detects an object. To determine whether or not generate a new agency based on the agency formation protocol, a freelancer-AVA communicates with agency managers when it detects an object. An agency manager, on the other hand, sends to freelancer-AVAs its target position when the new data are obtained. Then, each freelancer-AVA decides whether it continues to be a freelancer-AVA or joins into the agency depending on the task specification.

5 Experiments

We experimented to verify the effectiveness of the proposed system. We employed ten AVAs. Each AVA is implemented on a network-connected PC (PentiumIII 600MHz \times 2) with an active camera (SONY EVI-G20), where the perception, action, and communication modules as well as agency managers are realized as UNIX processes. In addition, the internal clocks of all the PCs are synchronized by Network Time Protocol to realize the virtual synchronization. With this architecture, the perception

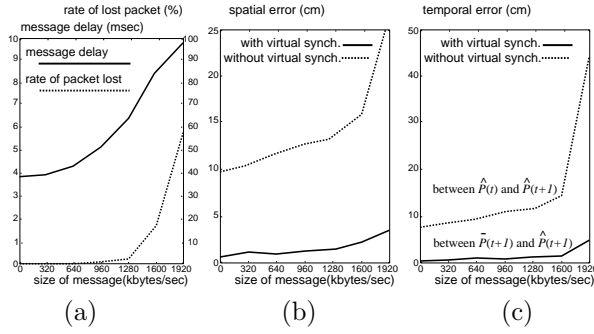


Figure 11: (a) Delay of the message (solid line) and Rate of lost packet (dotted line), (b) Error in spatial object identification, (c) Error in temporal object identification.

module can capture images and detect objects in the observed image at 11[frame/sec].

Figure 13 (a) illustrates the camera layout in the room. Camera₉ and camera₁₀ are placed about 1.5m above the floor. All other cameras are placed about 2.5m above the floor. The external camera parameters are calibrated.

5.1 Designing Utility-Function

We designed the utility-function as follows:

Search-value of freelancer-AVA_f: Let W_f denote the floor size that is visible from AVA_f . The search-value of AVA_f (denoted by V_{S_f}) is determined as follows: $V_{S_f} = \alpha_s \times W_f$, where α_s is a constant that is determined so that V_{S_f} is well-balanced with the tracking-value.

Tracking-value of member-AVA_m: 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 object. The tracking-value of AVA_m (denoted by $V_{T_m}^n$) is determined as follows: $V_{T_m}^n = (1)/(D_m^n) \times (1)/(A_m^n)$.

5.2 Performance Evaluation

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

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

Figure 11 (a) shows variations of network conditions when the size of the vain messages is changed. The error of spatial identification in Fig.11 (b) denotes the average distance between the reconstructed 3D position and the 3D view lines detected

by member-AVAs. The error of temporal identification in Fig.11 (c) denotes the average distance between the 3D positions of the same target, each of which are reconstructed/estimated at different times (i.e., $\hat{P}(t)/\hat{P}(t+1)$ and $\hat{P}(t+1)$).

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

5.3 Verification of Protocols

In the next experiment, the system tracked two people. Object₁ first came into the scene. Next, object₂ came into the scene. Both objects then moved freely.

In this reexperiment, the task specification was designed as follows: search-level=0.1, tracking-level=0.9 and object-importances for all objects were 1.0.

The upper part of Fig.12 shows the partial image sequences observed by AVA_2 , AVA_5 and AVA_9 . The images on the same column were taken at almost the same time. The regions enclosed by black and gray lines in the images show the detected regions of object₁ and object₂, respectively. Each figure in the bottom of Fig.12 shows the role of each AVA and the agency organization at such a moment when the same column of images in the upper part were observed. White circles denote freelancer AVAs, while black and gray circles indicate member AVAs belonging to agency₁ and agency₂, respectively. Black and gray squares indicate computed locations of target₁ and target₂ respectively.

The system worked as follows:

- a:** Initially, each AVA searched for an object independently.
- b:** AVA_5 first detected object₁, and agency₁ was formed.
- c:** All AVAs except for AVA_5 were tracking object₁ as the member-AVAs of agency₁, while AVA_5 was searching for a new object as a freelancer-AVA.
- d:** AVA_5 detected object₂ and generated agency₂.
- e:** The agency restructuring balanced the numbers of member-AVAs in agency₁ and agency₂.
- f:** Since no AVA could distinguish two objects, the agency unification protocol merged agency₂ into agency₁.
- g:** When the targets got apart, agency₁ detected a 'new' target. Then, it activated the agency spawning protocol to generate agency₂ again for target₂.
- h:** Object₁ was going out of the scene. **i:** After agency₁ dissolved, all the AVAs except for AVA_4 tracked object₂ as the member-AVAs of agency₂.

Figure 13 (a) shows the trajectories of the targets reconstructed by the agencies. Figure (b) shows the dynamic population changes of freelancer AVAs,

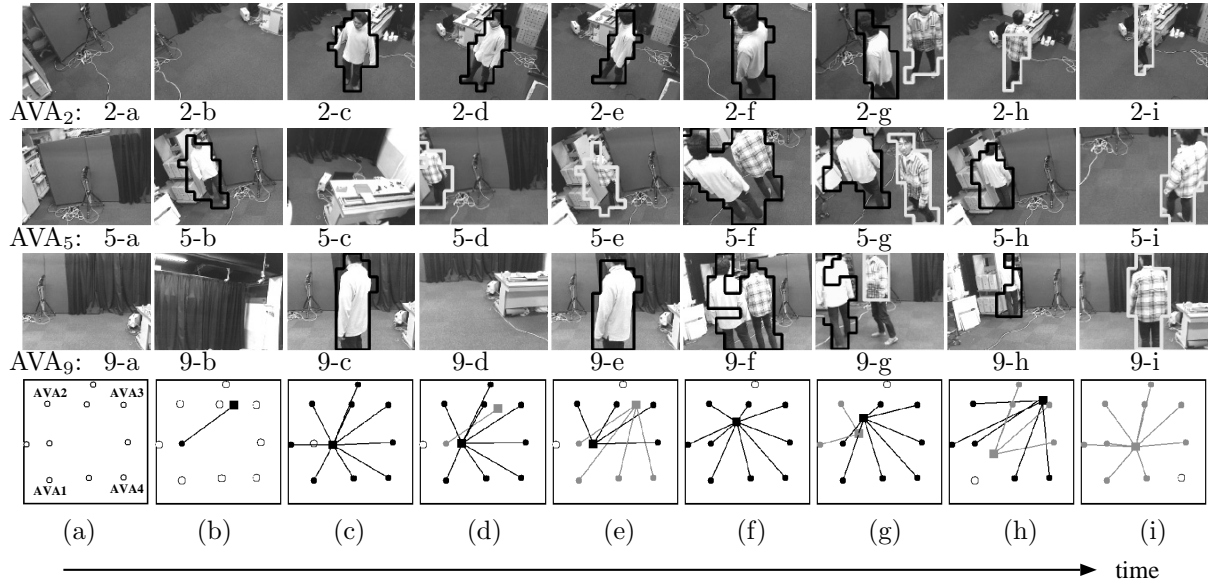


Figure 12: Upper: Partial image sequences, Lower: The role of each AVA and the agency organization

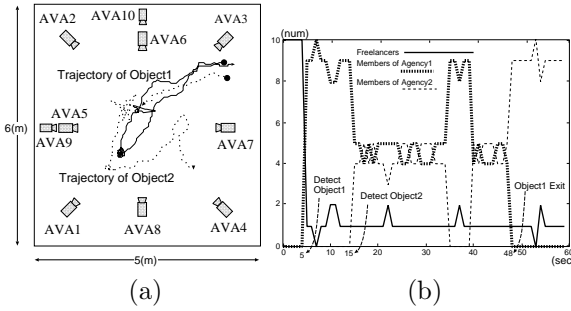


Figure 13: (a) Trajectories of the targets, (b) The number of AVAs that performed each role.

AVAs tracking target₁ and those tracking target₂.

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

6 Concluding Remarks

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

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

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

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

Inter-Agency: To adaptively restructure agencies taking into account targets' motions, agencies mutually exchange their information.

- Employing the dynamic memory realized the dynamic interactions in each layer without synchronization. The system is endowed with a high reactivity.

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

References

- [1] T. Matsuyama, "Cooperative Distributed Vision - Dynamic Integration of Visual Perception, Action and Communication -", *Proc. of Image Understanding Workshop*, pp.365-384, 1998.
- [2] S. Nishio and Y. Ohta, "Tracking of Vehicles at an Intersection by Integration of Multiple Image Sensors", *Proc. of MVA*, pp.321-324, 1992.
- [3] A. Nakazawa, H. Kato S. Hiura and S. Inokuchi, "Tracking Multiple Persons Using Distributed Vision Systems", *IPSJ Journal*, Vol. 42, NO. 11, pp.2699-2710, 2001 (Written in Japanese).
- [4] T. Kato, Y. Mukaigawa and T. Shakunaga, "Cooperative Distributed Tracking for Effective Face Recognition", *Proc. of MVA2000*, pp.353-358, 2000.
- [5] T. Matsuyama, *et al*, "Dynamic Memory: Architecture for Real Time Integration of Visual Perception, Camera Action, and Network Communication", *Proc. of CVPR*, pp.728-735, 2000.
- [6] T. Sogo, H. Ishiguro and M. M. Trivedi, "Real-time target localization and tracking by N-ocular stereo", *Proc. of IEEE Workshop on Omnidirectional Vision*, pp.153-160, 2000.