Real-Time Cooperative Multi-Target Tracking by Communicating Active Vision Agents

Takashi Matsuyama
Department of Intelligent Science and Technology
Graduate School of Informatics, Kyoto University
Sakyo, Kyoto, JAPAN 606-8501
e-mail: tm@i.kyoto-u.ac.jp

Abstract Target detection and tracking is one of the most important and fundamental technologies to develop real world computer vision systems such as security and traffic monitoring systems. 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 exchanging object information with each other. With this cooperative tracking capability, the system as a whole can track multiple moving objects persistently even under complicated dynamic environments in the real world.

1 Introduction

Target detection and tracking is one of the most important and fundamental technologies to develop real world computer vision systems: e.g. visual surveillance systems, ITS (Intelligent Transport Systems) and so on.

To realize real-time flexible tracking in a wide-spread area, we proposed the idea of *Cooperative Distributed Vision* (CDV, in short)[1]. The goal of CDV is summarized as follows (Fig. 1):
Embed in the real world a group of *Active Vision Agents* (AVA, in short: a network-connected computer with an active camera), and realize
1. wide area dynamic scene understanding and
2. versatile scene visualization.

Applications of CDV include real-time wide area surveillance and traffic monitoring, remote conference and lecturing, 3D video[2] and intelligent TV studio, and navigation of mobile robots and disabled people.

While the idea of CDV shares much with those of DVMT (Distributed Vehicle Monitoring Testbed)[3] and the VSAM (Video Surveillance And Monitoring) project by DARPA[4], our primary interest rests in how we can realize intelligent systems which work adaptively in the real world. And we put our focus upon *dynamic interactions among perception, action, and communication.* That is, we believe that intelligence does not dwell solely in brain but emerges from active interactions with environments through perception, action, and communication.

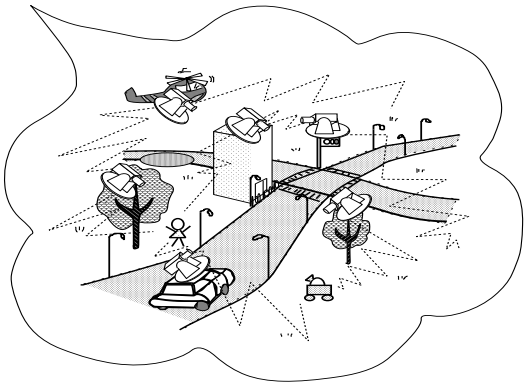


Figure 1: Cooperative distributed vision.

With this scientific motivation in mind, we designed a real-time cooperative multi-target tracking system, where we developed
Visual Sensor: a *Fixed-Viewpoint Pan-Tilt-Zoom Camera*[5] for wide area active imaging
Visual Perception: *Active Background Subtraction* for target detection and tracking[1]
Dynamic Integration of Visual Perception and Camera Action: *Dynamic Memory Architecture*[6] for real-time reactive tracking
Network Communication for Cooperation: a three-layered dynamic interaction ar-

chitecture for real-time communication among AVAs.

In this paper[1] , we address the key ideas of the above mentioned technologies and demonstrate their effectiveness in real-time multi-target tracking.

2 Fixed-Viewpoint Pan-Tilt-Zoom Camera for Wide-Area Active Imaging

To develop wide-area video surveillance systems, we first of all should study methods of expanding the visual field of a video camera:
1. Omnidirectional cameras using fish-eye lenses or curved mirrors[8][9][10], or
2. Active cameras mounted on computer controlled camera heads[5][7][11].

In the former optical methods, while omnidirectional images can be acquired at video rate, their resolution is limited. In the latter mechanical methods, on the other hand, high resolution image acquisition is attained at the cost of limited instantaneous visual field.

In our tracking system, we took the active camera method;
(a) High resolution images are of the first importance for object identification and scene visualization.
(b) Dynamic visual field and image resolution control can be realized by active zooming.
(c) The limited instantaneous visual field problem can be solved by incorporating a group of distributed cameras.

The next problem is how to design an active camera. Suppose we design a pan-tilt camera. This active camera system includes a pair of geometric singularities: 1) the projection center of the imaging system and 2) the pan and tilt rotation axes. In ordinary pan-tilt camera systems, no deliberate design about these singularities is incorporated, which introduces difficult problems in image analysis. That is, the discordance of the singularities causes photometric and geometric appearance variations during the camera rotation: varying highlights and motion parallax. To cope with these

[1] The original version of this paper will appear in IEEE Proceedings.

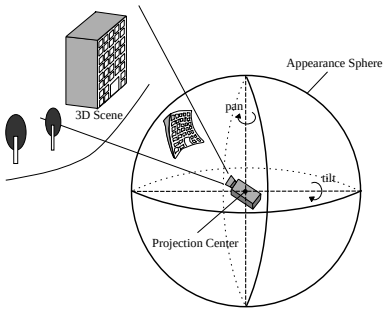
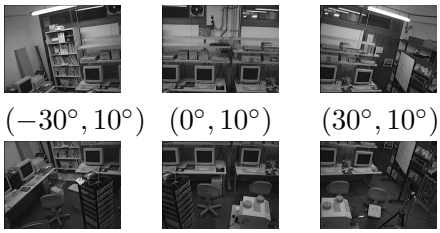


Figure 2: Fixed viewpoint pan-tilt camera.



$(-30°, 10°)$ $(0°, 10°)$ $(30°, 10°)$

$(-30°, -10°)$ $(0°, -10°)$ $(30°, -10°)$

(a) Observed images taken by changing (pan, tilt) angles.

(b) Generated panoramic image.

Figure 3: Panoramic image taken by the developed FV-PTZ camera.

appearance variations, consequently, sophisticated image processing should be employed[7].

Our idea to solve this appearance variation problem is very simple but effective[5][11]:
1. Make pan and tilt axes intersect with each other.
2. Place the projection center at the intersecting point.
We call the above designed active camera the *Fixed Viewpoint Pan-Tilt Camera*. With this camera, all images taken with different pan-tilt angles can be mapped seamlessly onto a common virtual screen (Appearance Sphere in Fig. 2) to generate a wide panoramic image. Note that once the panoramic image is obtained, images taken with arbitrary combina-

tions of pan-tilt parameters can be generated by back-projecting the panoramic image onto the corresponding image planes.

Usually, zooming can be modeled by the shift of the projection center along the optical axis[12]. Thus to realize the *Fixed Viewpoint Pan-Tilt-Zoom Camera* (FV-PTZ camera, in short), either of the following additional mechanisms should be employed:
(a) Design such a zoom lens system whose projection center is fixed irrespectively of zooming.
(b) Introduce a slide stage to align the projection center depending on zooming.

We found SONY EVI G20, an off-the-shelf active video camera, is a good approximation of an FV-PTZ camera ($-30^\circ \leq$ pan $\leq 30^\circ$, $-15^\circ \leq$ tilt $\leq 15^\circ$, and zoom: $15^\circ \leq$ horizontal view angle $\leq 44^\circ$); its projection center stays almost fixed irrespectively of zooming. Then, we developed a sophisticated internal-camera-parameter calibration method for this camera, with which we can use the camera as an FV-PTZ camera[1]. Fig. 3(a) illustrates a set of observed images taken by changing pan-tilt angles with the smallest zooming factor. Fig. 3(b) shows the panoramic image generated from the observed images.

3 Active Background Subtraction for Target Detection and Tracking

With an FV-PTZ camera, we can easily realize an active target tracking system. Fig. 4 illustrates the basic scheme of the active background subtraction for target detection and tracking we developed[1]:
STEP 1 Generate the panoramic image of the scene without any objects: Appearance Plane in the figure.
STEP 2 Extract a window image from the appearance plane according to the current pan-tilt-zoom parameters and regard it as the *current* background image.
STEP 3 Compute difference between the generated background image and the observed image.
STEP 4 If anomalous regions are detected in the difference image, select one and control the camera parameters to track the selected target.
STEP 5 Go back to STEP 2.

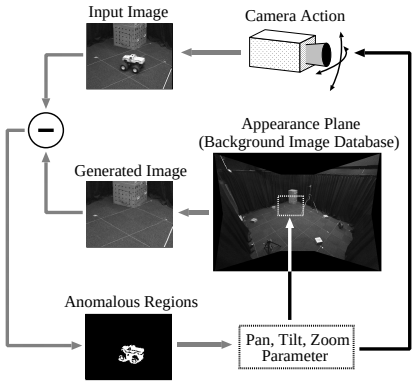


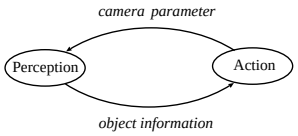
Figure 4: Active background subtraction with an FV-PTZ camera.

To cope with dynamically changing situations in the real world, we have to augment the above scheme in the following three points:
(a) Robust background subtraction which can work stably under non-stationary environments.
(b) Flexible system dynamics to control the camera reactively to unpredictable object behaviors.
(c) Multi-target tracking in cluttered environments.

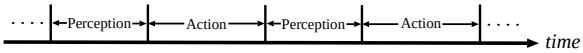
We do not address the first problem here, since various robust background subtraction methods have been developed [13][14]. As for the system dynamics, we will present a novel real-time system architecture in the next section and then, propose a cooperative multi-target tracking system in Section 6.

4 Dynamic Integration of Visual Perception and Camera Action for Real-Time Reactive Target Tracking

The active tracking system described in Fig. 4 can be decomposed into visual perception and camera action modules. The former includes image capturing, background image generation, image subtraction, and object region detection. The latter performs camera control and camera state (i.e. pan-tilt angles and



(a) Information flow between visual perception and camera action modules



(b) Dynamics in a sequential target tracking system.

Figure 5: Dynamic interaction between visual perception and camera action modules.

zooming factor) monitoring.

Here we discuss the dynamics of this system. Fig. 5(a) illustrates the information flow between the perception and action modules: the former obtains the *current* camera parameters from the latter to generate the background image and the latter the *current* target location from the former to control the camera. Fig. 5(b) shows the dynamics of the system, where the two modules are activated sequentially.

While this system worked stably[1], the camera motion was not so smooth nor could follow abrupt changes of target motion;
(a) The frequency of image observations is limited due to the sequential system dynamics. That is, the perception module should wait for the termination of the *slow* mechanical camera motion.
(b) Due to delays involved in image processing, camera state monitoring, and mechanical camera motion, the perception and action modules cannot obtain accurate *current* camera state or target location respectively.

To solve these problems and realize real-time reactive target tracking, we proposed a novel dynamic system architecture named *Dynamic Memory Architecture*[6], where the visual perception and camera action modules run in parallel and dynamically exchange information via a specialized shared memory named the *Dynamic Memory* (Fig. 6).

4.1 Access Methods for the Dynamic Memory

While the system architecture consisting of multiple parallel processes with a common shared memory looks similar to the

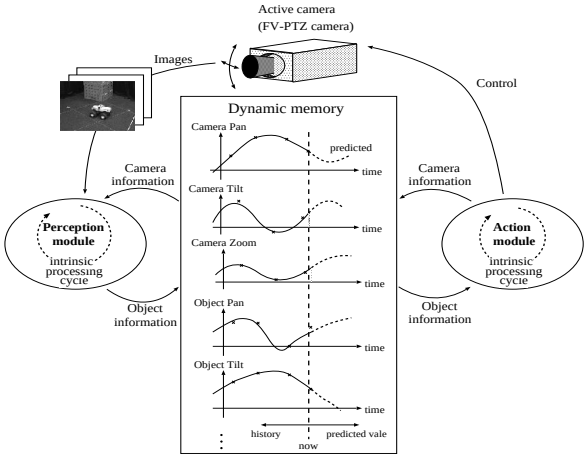


Figure 6: Real-time reactive target tracking system with the dynamic memory.

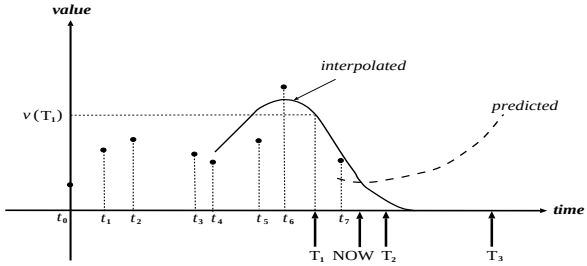


Figure 7: Representation of a time varying variable in the dynamic memory.

"whiteboard architecture"[15] and the "smart buffer"[16], the critical difference rests in that each variable in the dynamic memory stores a discrete temporal sequence of values and is associated with the following temporal interpolation and prediction functions (Fig. 7).

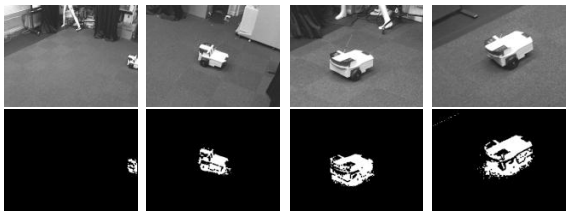
The write and read operations to/from the dynamic memory are defined as follows:

(a) Write Operation

When a process computes a value *val* of a variable **v** at a certain moment t, it writes (val, 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 (a sequence of black dots in Fig. 7).

(b) Read Operation

Temporal Interpolation: A reader process runs in parallel to the writer process and tries to read from the dynamic memory the value of the variable **v** at a certain moment: e.g.



frame0 frame50 frame100 frame150

Figure 8: Observed image sequence taken by the system. Upper: input images, Lower: detected object regions.

	rate of image observations	deviation of the target location from the image center	target region size
System A	1.83 [ftp]	44.0 [pixel]	5083 [pixel]
System B	11.04 [ftp]	16.7 [pixel]	5825 [pixel]

Table 1: Performance evaluation

the value at T_1 in Fig. 7. When no value is recorded at the specified moment, the dynamic memory interpolates it from recorded data. With this function, the reader process can read a value at any temporal moment along the continuous temporal axis without making any synchronization with the writer process.

Future Prediction: A reader process may run fast and require data which are not written yet by the writer process (for example, the value at T_3 in Fig. 7). In such case, the dynamic memory predicts an expected value in the future based on those data so far recorded and returns it to the reader process.

With the above described functions, each process can get any data along the temporal axis freely without waiting (i.e. wasting time) for synchronization with others. This no-wait asynchronous module interaction capability greatly facilitates the implementation of real-time reactive systems. As will be shown later in Section 5.2.3, moreover, the dynamic memory supports the *virtual synchronization* between multiple network-connected systems (i.e. AVAs), which facilitates the real-time dynamic cooperation among the systems.

4.2 Effectiveness of the Dynamic Memory

To verify the effectiveness of the dynamic memory, we developed a real-time single-target tracking system and conducted experiments of tracking a radio-controlled car in a computer room. The system employed the parallel active background subtraction method with the FV-PTZ camera, where the perception and action modules were implemented as UNIX processes sharing the dynamic memory. Fig. 8 illustrates a partial sequence of observed images and detected object regions. Note that the accurate calibration of the FV-PTZ camera enabled the stable background subtraction even while changing pan, tilt, and zooming.

Table 1 compares the performance between System A: sequential dynamics and System B: parallel dynamics with the dynamic memory. Both systems tracked a computer-controlled toy car under the same experimental settings and performance factors were averaged over about 30 sec. The left column of the table shows that the dynamic memory greatly improved the rate of image observations owing to the no-wait asynchronous execution of the perception module. The other two columns verify the improvements in the camera control. That is, with the dynamic memory, the camera was directed toward the target more accurately (the middle column) and hence could observe the target in higher resolution (the right column). Note that our system controls pan-tilt angles to observe the target at the image center and adjusts the zooming factor depending on deviations of the former from the latter: smaller deviations lead to zooming in to capture higher resolution target images, while larger deviations to zooming out not to miss the target[1].

5 Cooperative Multi-Target Tracking

Now we address cooperative multi-target tracking by communicating active vision agents (AVAs), where an AVA denotes an augmented target tracking system described in the previous section. The augmentation means that an AVA consists of visual percep-

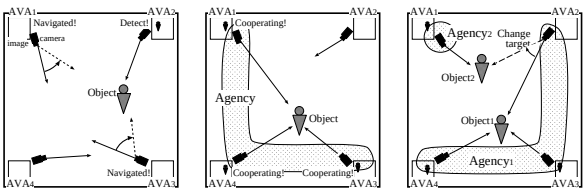


Figure 9: Basic scheme for cooperative tracking: (a) Gaze navigation, (b) Cooperative gazing, (c) Adaptive target switching.

tion, camera action, and network communication modules, which run in parallel exchanging information via the dynamic memory.

5.1 Basic Scheme for Cooperative Tracking

Our multi-target tracking system consists of a group of AVAs embedded in the real world (Fig. 1). The system assumes that the cameras are calibrated and densely distributed over the scene so that their visual fields are well overlapping with each other.

Followings are the basic tasks of the system:
1. Initially, each AVA independently searches for a target that comes into its observable area. Such AVA that is searching for a target is called a *freelancer*.
2. If an AVA detects a target, it navigates the gazes of the other AVAs towards that target (Fig.9 (a)).
3. A group of AVAs which gaze at the same target form what we call an *Agency* and keep measuring the 3D information of the target from multi-view images (Fig.9 (b)).
4. Depending on target locations in the scene, each AVA dynamically changes its target (Fig.9 (c)).

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 targets.
Real-time and reactive processing: To adapt itself to dynamic changes in the scene, the system has to execute processing in real-time and quickly react to the changes.
Adaptive resource allocation: We have to implement two types of dynamic resource allocation (i.e. grouping AVAs into agencies):

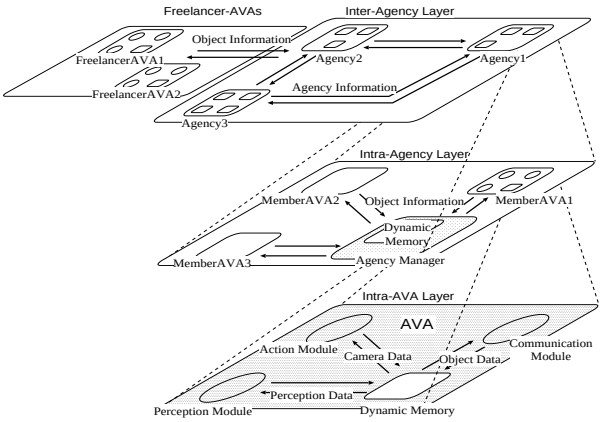


Figure 10: Three layered dynamic interaction architecture.

(1) To perform both target search and tracking simultaneously, the system has to preserve AVAs that search for new targets 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.

5.2 Three Layered Dynamic Interactions for Cooperative Tracking

We designed and implemented the three layered dynamic interaction architecture illustrated in Fig. 10 to realize real-time cooperative multi-target tracking.

5.2.1. Intra-AVA layer

In the lowest layer in Fig.10, perception, action and communication modules that compose an AVA interact with each other via the dynamic memory.

An AVA is an augmented target tracking system described in Section 5, where the augmentation is threefold:

(1) Multi-target detection while single-target tracking

When the perception module detects N objects at $t+1$, it computes and records into the dynamic memory the 3D view lines toward the objects[2] (i.e. $L^1(t+1), \cdots, L^N(t+1)$). Then, the module compares them with the 3D

[2] The 3D line determined by the projection center of the camera and an object region centroid.

view line toward its currently tracking target at $t+1$, $\widehat{L}(t+1)$. Note that $\widehat{L}(t+1)$ can be read from the dynamic memory whatever temporal moment $t+1$ specifies. Suppose $L^x(t+1)$ is closest to $\widehat{L}(t+1)$, where $x \in \{1, \cdots, N\}$. Then, the module regards $L^x(t+1)$ as denoting the newest target view line and records it into the dynamic memory.

(2) Gaze control based on the 3D target position

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

(3) Incorporation of the communication module

Data exchanged by the communication module over the network can be classified into two types: detected object data and messages for cooperations among AVAs. The former include 3D view lines toward detected objects: AVA $\rightarrow$ other AVAs and agencies, and 3D target position: agency $\rightarrow$ member AVAs. The latter realize various communication protocols, which will be described later.

5.2.2. 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. The middle layer in Fig.10 specifies dynamic interactions between an agency manager and 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.

(a) Spatial object identification

The agency manager has to establish the object identification between the groups of the 3D view lines detected and transmitted by its member AVAs. The agency 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 5.2.3 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.

(b) Temporal object identification

The manager records the 3D trajectory of its target, with which the 3D object position(s) computed by the spatial object identification is compared. That is, when multiple 3D locations are obtained by the spatial object identification, the manager selects the one closest to the target trajectory. When the 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 latest recorded target 3D position. Then the manager projects the target 3D position onto the selected view line to estimate the new 3D target position.

5.2.3. Virtual Synchronization

Here we discuss dynamic aspects of the above identification processes.

(a) Spatial object 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. These asynchronous activities can significantly damage the reliability of the spatial object identification.

To solve this problem, we introduce the dynamic memory into an agency man-

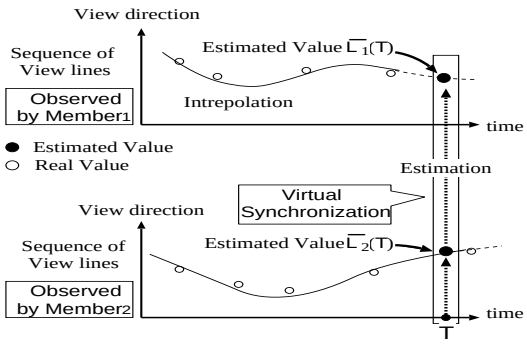


Figure 11: Virtual synchronization for spatial object identification

ager, which enables the manager to virtually synchronize any asynchronously observed/transmitted data. We call this function *Virtual Synchronization* by the dynamic memory.

Fig.11 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. Fig.11, for example, shows a pair of temporal sequences of 3D view line data transmitted from member AVA_1 and member AVA_2, respectively. When the manager wants to establish the 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.11). That is, the values of the 3D view lines used for the identification are completely synchronized with that identification timing even if their measurements are conducted asynchronously.

(b) Temporal object identification

The virtual synchronization is also effective in the 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 identified at T. Then the 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 new target position.

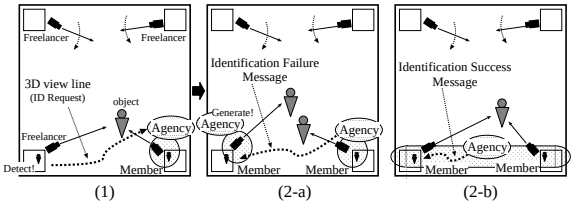


Figure 12: Agency Formation

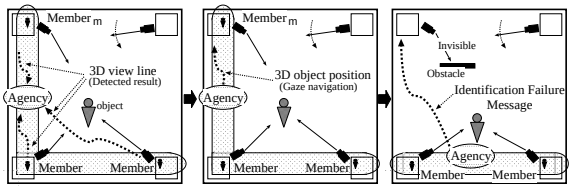


Figure 13: Agency Maintenance

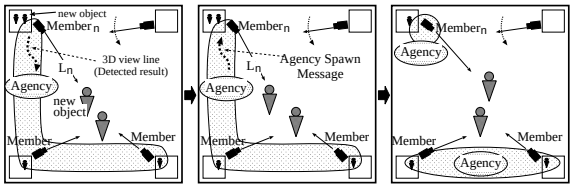


Figure 14: Agency Spawning

5.2.4. Communications at Intra-Agency Layer

The above mentioned temporal object identification fails if the closest distance between the estimated and observed 3D target locations exceeds a threshold. The following three communication protocols are activated depending on the success or failure of the object identification. They materialize dynamic interactions at the intra-agency layer.

(a) Agency formation protocol

This protocol defines (1) the new agency generation procedure by a freelancer AVA and (2) the participation procedure of a freelancer AVA into an existing agency.

When a freelancer AVA detects an object, it requests the existing agency managers to examine the identification between the detected object and the target object of each agency (Fig.12, (1)). Depending on the result of this object identification, the freelancer AVA works as follows:

No agency established the object identification: The freelancer AVA generates a new agency manager to track the newly detected object and joins into that agency as its member AVA (Fig.12, (2-a)).

An agency established the object iden-

tification: The freelancer-AVA joins into the agency that has made successful object identification, if requested (Fig.12, (2-b)).

(b) Agency maintenance protocol

This protocol defines procedures for the continuous maintenance of an agency and the elimination of an agency.

After an agency is generated, the agency manager repeats the spatial and temporal object identifications for cooperative tracking (Fig.13 (1)). Following the spatial object identification, the manager transmits the newest 3D target location to each member AVA (Fig.13 (2)), which then is recorded into the dynamic memory of the member AVA.

Suppose a member AVA_m cannot detect the target object due to an obstacle or processing errors (Fig.13 (3)). Even in this case, the manager informs AVA_m the 3D position of the target observed by the other member AVAs. This information navigates the gaze of AVA_m towards the (invisible) target. However, if such mis-detection continues for a long time, the agency manager forces AVA_m out of the agency to be a freelancer.

If all member AVAs cannot observe the target being tracked so far, the agency manager destroys the agency and makes all its member AVAs become freelancers.

(c) Agency spawning protocol

This protocol defines a new agency generation procedure from an existing agency.

After the spatial and temporal object identifications, the agency manager may find such a 3D view line(s) that does not correspond to the target. This means the detection of a new object by its member AVA. Let L_n denote such 3D view line detected by AVA_n (Fig.14 (1)). Then, the manager broadcasts L_n to other agency managers to examine the identification between L_n and their tracking targets.

If none of the identification is successful, the agency manager makes AVA_n quit from the current agency and generate a new agency (Fig.14 (2)). AVA_n then joins into the new agency (Fig.14 (3)).

5.2.5. Inter-Agency layer

In multi-target tracking, the system should adaptively allocate resources: the system has

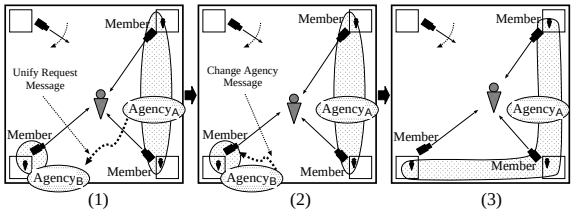


Figure 15: Agency Unification

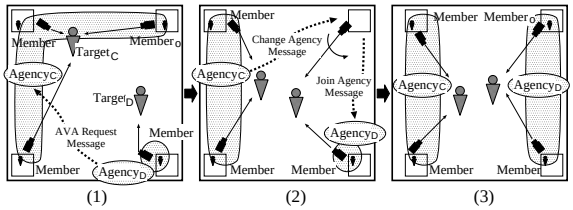


Figure 16: Agency Restructuring

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. 10).

The dynamic interactions between agency managers are triggered based on the object identification across agencies. That is, when a new target 3D location is obtained, agency manager AM_i broadcasts it to the others. Agency manager AM_j, which receives this information, compares it with the 3D position of its own target to check the object identification. Note that here also the virtual synchronization between a pair of 3D target locations is employed to increase the reliability of the object identification.

Depending on the result of this inter-agency object identification, either of the following two protocols are activated.

(a) Agency unification protocol

This protocol is activated when the inter-agency object identification is successful and defines a merging procedure of the agencies which happen to track the same object.

In principle, the system should keep the one-to-one correspondence between agencies and target objects. However, this correspondence sometimes is violated due to failures of object identification and discrimination:
(a) asynchronous observations and/or errors in object detection by individual AVAs or
(b) multiple targets which come too close to separate.

Fig.15 shows an example. When agency

manager AM_A of $agency_A$ establishes the identification between its own target and the one tracked by AM_B, AM_A asks AM_B to be merged into AM_A (Fig.15(1)). Then, AM_B asks its member AVAs to join into $agency_A$ (Fig.15(2)). After copying the target information recorded in the dynamic memory into the object trajectory database, AM_B eliminates itself (Fig.15(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 back by the agency spawning protocol when the distance between the targets get larger. In such case, characteristics of the newly detected target are compared with those recorded in the object trajectory database to check if the new target corresponds to a target that had been tracked before. If so, the corresponding target trajectory data is moved from the database into the dynamic memory of the newly generated agency.

(b) Agency restructuring protocol

When the inter-agency object identification fails, agency manager AM_j checks if it can activate the agency restructuring protocol taking into account the numbers of member AVAs in $agency_j$ and $agency_i$ and their target locations.

Fig.16 illustrates an example. agency manager AM_C of $agency_C$ sends its target information to AM_D, which fails in the object identification. Then, AM_D asks AM_C to trade its member AVA into AM_D (Fig.16(a)). When requested, AM_C selects its member AVA and asks it to move to $agency_D$ (Fig.16(b) (c)).

5.2.6. Communication with Freelancer AVAs

An agency manager communicates with freelancer AVAs as well as with other managers (the top row of Fig. 10). As described in the agency formation protocol in Section 5.2.4, a freelancer activates the communication with agency managers when it detects an object. An agency manager, on the other hand, sends to freelancers its target position when the new data are obtained. Then, each freelancer decides whether it continues to be a freelancer or joins into the agency depending on the target position and the current number of freelancers in the system. Note that in our system a user can specify the number of freelancers to be preserved while tracking targets.

6 Experiments

To verify the effectiveness of the proposed system, we conducted experiments of multiple human tracking in a room (about $5m \times 5m$). The system consists of ten AVAs. Each AVA is implemented on a network-connected PC (PentiumIII 600MHz $\times$ 2) with an FV-PTZ camera (SONY EVI-G20), where the perception, action, and communication modules as well as agency managers are realized as UNIX processes. Fig.18 (a) illustrates the camera layout: $camera_9$ and $camera_{10}$ are on the walls, while the others on the ceiling. The external camera parameters are calibrated. Note that the internal clocks of all the PCs are synchronized by the Network Time Protocol to realize the virtual synchronization. With this architecture, the perception module of each AVA can capture images and detect objects at about 10 frames per second on average.

In the experiment, the system tracked two people. $Target_1$ first came into the scene and after a while, $target_2$ came into the scene. Both targets then moved freely. The upper part of Fig. 17 shows the partial image sequences observed by AVA_2, AVA_5 and AVA_9. The images on the same row were taken by the same AVA. 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 corresponding to $target_1$ and $target_2$ respectively.

Each figure in the bottom of Fig.17 shows the role of each AVA and the agency organization at such a moment when the same column of images in the upper part were observed. White circles denote freelancer AVAs, while black and gray circles indicate member AVAs belonging to $agency_1$ and $agency_2$, respectively. Black and gray squares indicate computed locations of $target_1$ and $target_2$ respectively.

The system worked as follows.

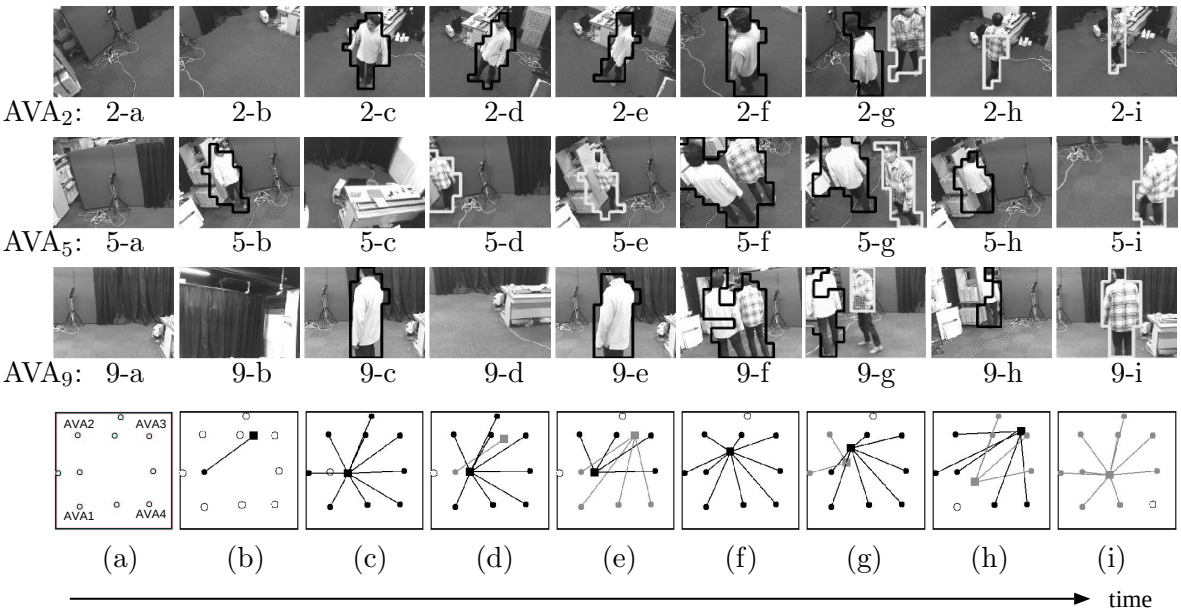


Figure 17: Experimental results.

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

b: AVA_5 first detected $target_1$, and $agency_1$ was formed.

c: All AVAs except for AVA_5 were tracking $target_1$, while AVA_5 was searching for a new object as a freelancer.

d: Then, AVA_5 detected $target_2$ and generated $agency_2$.

e: The agency restructuring protocol balanced the numbers of member AVAs in $agency_1$ and $agency_2$. Note that AVA_9 and AVA_{10} were working as freelancers.

f: Since two targets came very close to each other and no AVA could distinguish them, the agency unification protocol merged $agency_2$ into $agency_1$.

g: When the targets got apart, $agency_1$ detected a 'new' target. Then, it activated the agency spawning protocol to generate $agency_2$ again for $target_2$.

h: $Target_1$ was going out of the scene.

i: After $agency_1$ was eliminated, all the AVAs except AVA_4 tracked $target_2$.

Fig.18 (a) shows the trajectories of the targets computed by the agency managers. Fig.18 (b) shows the dynamic population changes of freelancer AVAs, AVAs tracking $target_1$ and those tracking $target_2$.

As we can see, the dynamic cooperations among AVAs and agency managers worked very well and enabled the system to persistently track multiple targets.

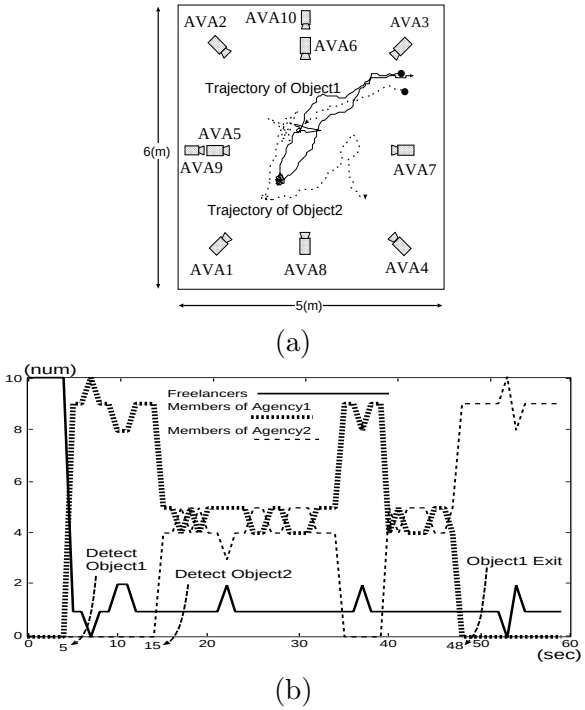


Figure 18: Experimental results: (a) Trajectories of the targets, (b) The number of AVAs that performed each role

7 Concluding Remarks

This paper presented a real-time active multi-target tracking system, which is the most powerful and flexible but difficult to realize among

various types of target tracking systems.

To implement the system, we developed 1) a Fixed-Viewpoint Pan-Tilt-Zoom Camera for wide area active imaging, 2) Active Background Subtraction for target detection and tracking, 3) Dynamic Memory Architecture for real-time reactive tracking, and 4) a three-layered dynamic interaction architecture for real-time communication among AVAs.

In our system, parallel processes (i.e. AVAs and its constituent perception, action, and communication modules) cooperatively work interacting with each other. As a result, the system as a whole works as a very flexible real-time reactive multi-target tracking system. We believe that this cooperative distributed processing greatly increases the flexibility and adaptability of the system, which has been verified by experiments of multiple human tracking.

This work was supported by the Research for the Future Program of the Japan Society for the Promotion of Science (JSPS-RFTF96P00501). Research efforts by all former and current members of our laboratory are gratefully acknowledged.

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.Moezzi, L.Tai, and P.Gerard: Virtual View Generation for 3D Digital Video, IEEE Multimedia, pp.18-26, 1997.

[3] V. R. Lesser and D. D. Corkill: The Distributed Vehicle Monitoring Testbed: a Tool for Investigating Distributed Problem Solving Networks, AI Magazine, Vol. 4, No. 3, pp.15–33, 1983.

[4] Video Surveillance and Monitoring, Proc. of Image Understanding Workshop, Vol.1, pp.3-400, 1998

[5] T. Wada and T. Matsuyama: Appearance Sphere: Background Model for Pan-Tilt-Zoom Camera, Proc. of ICPR, Vol. A, pp. 718-722, 1996.

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

[7] D. Murray and A. Basu: Motion Tracking with an Active Camera, IEEE Trans. of PAMI, Vol. 16, No. 5, pp. 449-459, 1994.

[8] Y. Yagi and M. Yachida : Real-Time Generation of Environmental Map and Obstacle Avoidance Using Omnidirectional Image Sensor with Conic Mirror, Prof. of CVPR, pp. 160-165, 1991.

[9] K. Yamazawa, Y. Yagi, and M. Yachida: Obstacle Detection with Omnidirectional Image Sensor HyperOmni Vision, Proc. of ICRA, pp.1062 - 1067, 1995.

[10] V.N. Peri and S.K. Nayar: Generation of Perspective and Panoramic Video from Omnidirectional Video, Proc. of IUW, pp.243 - 245, 1997.

[11] S. Coorg and S. Teller: Spherical Mosaics with Quaternions and Dense Correlation, Int'l J. of Computer Vision, Vol.37, No.3, pp.259-273, 2000.

[12] J.M. Lavest, C. Delherm, B. Peuchot, and N. Daucher: Implicit Reconstruction by Zooming, Computer Vision and Image Understanding, Vol.66, No.3, pp.301-315, 1997.

[13] K. Toyama, et al: Wallflower: Principles and Practice of Background Maintenance, Proc. of ICCV, pp. 255-261, 1999

[14] T. Matsuyama, T. Ohya, and H. Habe: Background Subtraction for Non-Stationary Scenes, Proc. of 4th Asian Conference on Computer Vision, pp.662-667, 2000

[15] C. Thorpe, M.H. Herbert, T. Kanade, and S.A. Shafer: Vision and Navigation for the Carnegie-Mellon Navlab, IEEE Trans., Vol.PAMI-10, No.3, pp.362-373, 1988

[16] J.J. Little and J. Kam: A Smart Buffer for Tracking Using Motion Data, Proc. of Computer Architecture for Machine Perception, pp.257-266, 1993