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

Norimichi Ukita

Abstract

The dynamic situation understanding is indispensable for realizing computer systems that autonomously work in the real world. This is because such computer systems have to understand dynamic situations in the real world, and adapt themselves to the situations reactively. The dynamic situation understanding has been studied in Computer Vision and many technologies have been developed. Some of them are applied to real-world computer systems to increase their flexibilities.

To realize the dynamic situation understanding, object tracking is one of the most important and fundamental technologies. This is because most of dynamic situations in the scene can be characterized by object motions. Therefore, we should focus on such moving objects and obtain the information of the focused objects to understand the dynamic situation. To apply object tracking to real-world systems, the object tracking method has to cope with complicated situations and conduct processing in real time.

In this thesis, we propose a real-time flexible object tracking system. Our objective is to realize the tracking system that can adaptively change its behavior depending on the situation and task, and persistently keep tracking focused target objects.

In order to realize real-time multi-target tracking in a wide-spread area, we employ the idea of *Cooperative Distributed Vision* (CDV, in short). The CDV system consists of communicating *Active Vision Agents* (AVAs, in short), where an AVA is a logical model of an *Observation Station* (real-time image processor with active camera(s)). For real-time object tracking by multiple AVAs, we have to solve (1) how to design an active camera for dynamic object detection and tracking, (2) how to realize real-time object tracking with an active camera and (3) how to realize cooperation among AVAs for real-time multi-target object tracking.

First of all, for wide-area active imaging, we developed a *Fixed-Viewpoint Pan-Tilt-Zoom* (FV-PTZ, in short) camera. This camera is designed so that the projection center is always placed at the rotational center irrespectively of pan, tilt and zoom controls. This property allows the system (1) to generate a wide panoramic image by mosaicing multiple images observed by changing pan-tilt-zoom parameters and (2) to synthesize an image taken with any pan-tilt-zoom parameters from the wide panoramic image. With the FV-PTZ camera, we can realize an active camera system that detects anomalous regions in the observed image by comparing it with the generated background image (background subtraction method).

Next, for real-time object detection and tracking, we designed an *Active Background Subtraction* method with the FV-PTZ camera. To successfully gaze at the target during tracking, the system incorporates a flexible control system named the *Dynamic Memory Architecture*, where multiple parallel processes share what we call the *Dynamic Memory*, to dynamically integrate visual perception and camera action modules. The dynamic memory enables parallel modules to asynchronously obtain the information of another process without disturbing their own intrinsic dynamics.

Finally, to implement the real-time cooperation among AVAs, we designed a three-layered interaction architecture:

1st layer (Intra-AVA layer): Visual perception, action and communication modules of an AVA work together by dynamically interacting with each other. Each module exchange its information through the dynamic memory.

2nd layer (Intra-Agency layer): AVAs that track the same target object form a group (*Agency*). AVAs in the same agency exchange the object information to cooperatively track the target. Each agency has its own dynamic memory, and all the member-AVAs exchange their information of the detected objects through the dynamic memory. The dynamic memory allows the agency to obtain the reliable result of object identification from multiple pieces of the object information asynchronously observed by the member-AVAs.

3rd layer (Inter-Agency layer): In order to adaptively restructure agencies taking into account targets' motions, agencies exchange the target and agency information with each other.

The dynamic interactions in each layer allow the entire system to track multiple moving objects under complicated dynamic situations in the real world.

Throughout the thesis, experimental results are presented to demonstrate the effectiveness of our real-time flexible tracking system.

Acknowledgements

First of all, I have to express my deepest thanks to professor Takashi Matsuyama at Kyoto university. Without him, this thesis could not never exist. He first showed me various possibilities of my research, and gave me valuable advice and enable me to complete my thesis.

I would also like to thank the members of my thesis committee, Prof. Michihiko Minoh and Prof. Hiroshi G. Okuno, for making many useful comments on my thesis.

Next, I express my gratitude to associate professor Toshikazu Wada. He introduced me to the world of Computer Vision and made me show a keen interest in the research. He always gave counsel to me, and his steady work in this field certainly influenced my research.

I would like to thank to lecturer Akihiro Sugimoto. He made smart comments on my research and supported me to accomplish this thesis. In addition, he read my manuscripts many times and gave me very useful comments both technically and grammatically.

I would also like to thank Shogo Tokai at Fukui university and Shinsaku Hiura at Osaka university. They used to debate with me on my idea and guide me in my study.

In addition, I am also grateful to the members of the Cooperative Distributed Vision Project (JSPS-RFTF 96P00501). They often discussed my work and gave me many useful comments.

I also wish to express my appreciation for cooperation and kindness of my fellows, members of Matsuyama laboratory at Kyoto university. They helped my experiments and debated on diverse issues of my work. Furthermore, they made my deserted student life enjoyable with their favors.

I would also like to thank Ms. Yuki Watanabe and Ms. Hiromi Monobe who assisted me in performing office jobs.

Although I did not mention all names, I am also grateful to my friends and many people for their warm support.

Finally, I would like to thank my families who were sympathetic to my work.

Contents

Chapter 1

Introduction

1.1 Background

The dynamic situation understanding is indispensable for realizing computer systems that autonomously work in the real world. This is because such computer systems have to understand dynamic situations in the real world and adapt themselves to the situations reactively. The dynamic situation understanding has been studied in Computer Vision and many technologies have been developed. Some of them are applied to real-world computer systems to increase their flexibilities.

To realize the dynamic situation understanding, object tracking is one of the most important and fundamental technologies. This is because most of dynamic situations in the scene can be characterized by object motions. Therefore, we should focus on such moving objects and obtain the information of the focused objects (e.g., the number, locations and behaviors of objects) to understand the dynamic situation.

From a practical point of view, on the other hand, object tracking technology allows us to develop various real-world vision systems such as

- Visual surveillance and monitoring systems [NKI98] [MWM98a] [KZ99],
- Remote conference and distance lecturing systems [KIM00] [KKD+98],
- ITS (Intelligent Transport System) [NO92] [HIZ00],
- Navigation of mobile robots and disabled people [Ish97] [AB99],
- 3D volume reconstruction and motion capturing systems [WWTM00] [YAT00] [BD00].

To apply object tracking to these real-world systems, the object tracking method has to cope with complicated situations and reactively conduct processing in real time. Most of the proposed object tracking methods, however, have some restrictions on functions and assumptions about the environment. These limitations reduce the effectiveness and generality of object tracking for real-world systems.

In this thesis, we propose a real-time flexible object tracking system. Our objective is to realize the tracking system that can (1) adaptively change its behavior depending on the situation and task and (2) persistently keep tracking focused target objects.

In what follows, we first categorize object tracking systems into several classes and review related previous works. We then discuss the advantages and disadvantages of each class (Section 1.2). Based on this discussion, we address our strategy for realizing real-time flexible object tracking (Section 1.3). We finally present an overview of the thesis in Section 1.5.

1.2 Categories of Object Tracking Systems

Real-time object tracking systems[1] can be classified into several different types. For their categorization, the following characteristics can be used:

Characteristic 1: How many objects in the scene?

Characteristic 2: How many target objects to be tracked?

Characteristic 3: Fixed camera or active camera?

Characteristic 4: How many cameras?

The first characteristic is an assumption on the scene. The second characteristic provides a task given to a tracking system. The latter two characteristics are concerned with a system architecture and its functions. These four characteristics define the task and complexity of the tracking system.

How many objects in the scene? Depending on the number of objects in the scene, a design of the tracking system changes.

- Single-object system: There is only one object in the scene.
- Multi-object system: There are one or more objects in the scene.

When the number of objects is assumed to be not more than one (i.e, single-object system), the detected object must be the target object. That is, the detected object is necessarily tracked by the system. The tracking system, then, need not discriminate between the target object and the non-target object. Thus, the design of the single-object system is simplified.

How many target objects to be tracked? Similarly, tracking systems can be classified into two classes depending the number of target objects.

- Single-target system: There is only one target object to be tracked.
- Multi-target system[2] : There are multiple target objects to be tracked.

[1] There are also many researches on object tracking by batch processing, namely non-real-time object tracking (see [LC83] [PDBSW92] [YM96], for example). This is one of the typical problems in distributed artificial intelligence. We do not, however, take this issue into account because we put our focus upon real-time object tracking.

[2] Obviously, the multi-target tracking system is included in the multi-object system.

In the single-target system, the system keeps tracking a single target object, and can ignore the information of all the other objects in the scene.

If the system has to track multiple target objects simultaneously, on the other hand, it need to discriminate between all target objects continuously. That is, temporal multi-object identification is required. The task of the multi-target tracking system, therefore, becomes complex in contrast to that of the single-target tracking system.

Fixed camera or active camera?: By employing an active camera as an image sensor, the system can control the camera parameters (e.g, pan, tilt and zoom parameters) depending on the given task and the situation in the real world. With the active camera, the system can continue the following procedures.

1. The system captures the image and analyzes the captured image.
2. Based on the result of the image analysis, the system control camera parameters to capture a required view.

This active scene observation allows the system to perform tasks adaptively.

In addition, controlling camera parameters gives the system the following advantages.

- The system can observe a wider area by changing the gazing direction and location of the camera.
- The system can dynamically adjust the visual field of the camera and the resolution of the captured image by changing the zoom parameter.

As the system can obtain many advantages, we have to design a real-time and reactive camera controlling method. In this method, we change camera parameters depending on (1) dynamic situations in the real world and (2) mechanical characteristics of the active camera.

How many cameras? The observable area by a single camera is limited. The single-camera system can, therefore, obtain little information simultaneously.

Compared with the single-camera system, the multi-camera system has the following advantages:

Simultaneous wide-area observation: By embedding many cameras in the scene, multiple different views can be observed simultaneously. The system can, therefore, gaze at target objects that are distant from each other.

Continuous wide-area observation: The arrangement of cameras in the wide area also enables the system to continuously track the focused target object even if the target object moves around the wide area.

Reconstruction of 3D information: If the external camera parameters (i.e., 3D positions of all cameras) are calibrated, 3D information of the object can be reconstructed from 2D information of the object observed by multiple cameras.

For example, a stereo vision method allows the system to reconstruct the 3D position of the object by employing the images observed from multiple directions.

To make full use of these advantages of the multi-camera system, we have to solve the following camera-cooperation problems:

1. To simultaneously track multiple objects with multiple cameras, the system has to identify each detected object among images observed by multiple cameras.
2. To keep tracking the moving target object without a break, a camera need to request another camera to take over tracking of the target object.
3. To robustly reconstruct 3D information of the object, 2D object information synchronized among multiple cameras is required. The system then need to integrate the object information observed by cameras.

Depending on how to realize the above cooperation among cameras, we establish two types of distributed camera systems. We will describe the properties of these systems in Section 1.3.

As mentioned above, there are four basic characteristics of object tracking, and each characteristic has two classes. In total, there are 16 ($= 2^4$) classes. The complexity of the system depends on combinations of the characteristics.

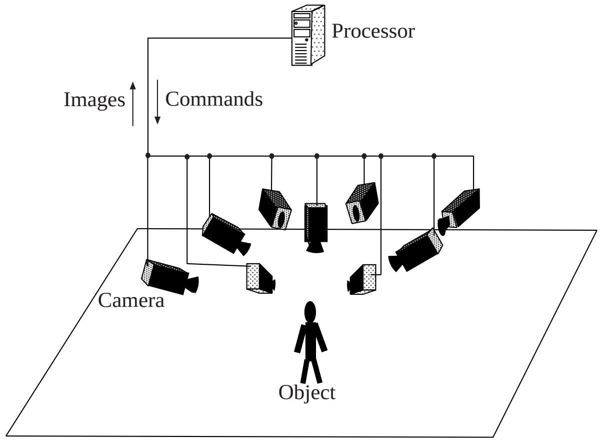
A large number of works about single-target tracking have been reported:

- Using a single fixed camera: [WADP97].
- Using a single active camera: [MWM99] [MB94].
- Using multiple fixed cameras: [CA99].
- Using multiple active cameras: [MWM98a].

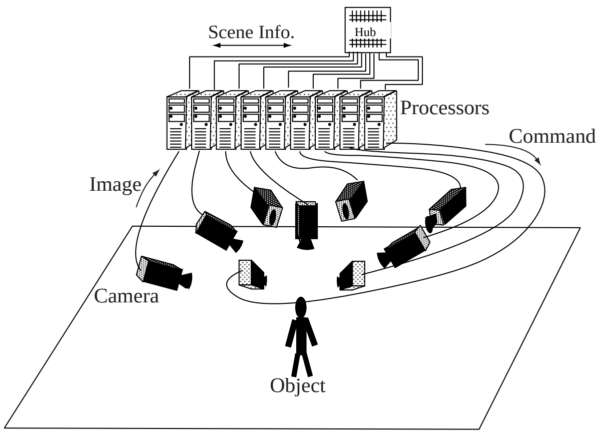
On the other hand, the number of reported multi-target tracking systems is not as many as that of the single-target tracking systems. Since the multi-target tracking system is required to apply the system to general purposes, researchers recently concentrate on multi-target tracking and the works about it has been increasing:

- Using a single fixed camera: [IB98] [HHD00a].
- Using multiple fixed cameras: [NO92] [NKI98].

Nevertheless, there are few multi-target tracking systems using multiple cameras. In particular, hardly any multi-target tracking system that employs multiple active cameras has been reported. This is because we have to solve many problems to develop such systems. Since the multi-target tracking system with multiple active cameras (that is, (1) multi-object, (2) multi-target, (3) active camera and (4) multi-camera tracking system)



(a) Centralized processing system



(b) Distributed processing system

Figure 1.1: Types of multi-camera systems.

includes the properties of all other classes, it is the most powerful way to cope with various tasks and situations in the real world. We consider that this tracking system is a technology worth being developed. Accordingly, we aim at developing a real-time tracking system that can gaze at multiple target objects simultaneously by employing multiple active cameras.

1.3 Real-time Cooperative Multi-target Tracking

1.3.1 Cooperative Distributed System Architecture

As mentioned in the last section, we concern a multi-camera system for real-time object tracking. In general, we can design two types of multi-camera systems: centralized processing system and distributed processing system. We should examine the advantages and disadvantages of these systems, and choose the system which is appropriate for realization of our tracking system.

Centralized processing (Figure 1.1 (a)): To integrate the object information observed by multiple cameras, a single processor gathers all the captured images through the network. This processor then analyzes all the images and obtains the integrated object information.

Integrating the observed object information by a single processor causes the following disadvantages:

- Increasing network-load: Since all the captured images are transmitted to a single processor, a huge network-load is caused.

- Increasing computational processing: Since all the captured images are analyzed by a single processor, the computational load of the processor increases depending on the number of cameras.

These factors make real-time processing difficult. To solve these problems, Sogo–Ishiguro–Trivedi[SIT00] reduces the size of each image[3] and enables a single processor to conduct all required processes in real time. In this method, however, the lager the number of cameras becomes, the lower the image resolution is.

Besides these technological problems, the centralized processing system has an essential limitation: a large variety of the observed situations have to be managed by only the single processor. It has to, therefore, cope with all complicated situations in the real world by itself. This expands the computational complexity of the process that is conducted by one processor. Moreover, we have to design such a complex behavior taking into account all combinations of predictable situations.

Distributed processing (Figure 1.1 (b)): To solve the problem that arises in the centralized processing system, each camera is coupled to its own processor. That is, tasks of a single processor in the centralized processing system becomes decentralized. In this distributed processing system, each processor analyzes the image captured by its coupled camera, and exchanges the result of the image analysis with the other processors.

This distributed processing enables the system to solve the disadvantages in the centralized processing system as follows:

- Decreasing computational processing: Since the image captured by a camera is analyzed by each processor, the computational complexity for each processor can be simplified rather than that of the centralized processing system.
- Decreasing network-load: Since the image is analyzed by each processor, all processors exchange only the result on this image analysis. The amount of these results are much smaller than that of the image itself. This property can reduce network congestion.

By composing the system as a group of multiple processors, we can represent the complex behavior of the entire system through the interaction between processes. Designing the entire system can be, therefore, reduced to designing each process. Furthermore, the states and their transitions of the entire system increase enormously by combining those of all the processors. This property allows the system to cope with various and complicated situations in the real world. This is a great advantage of the distributed processing system in contrast to the centralized processing system. We believe that this property is indispensable to realize a flexible real-world system.

[3] All observed images are put together into a single image by a multi-to-one image converter. An image processor can, therefore, manage all observations as a single image.

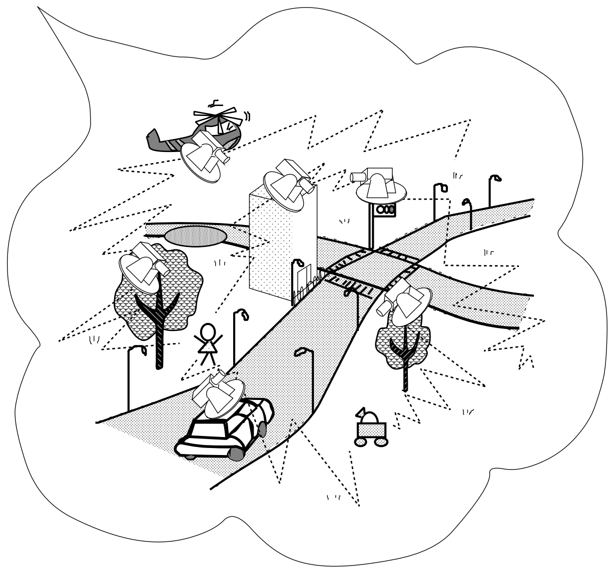


Figure 1.2: Cooperative distributed vision [Mat98].

Accordingly, the distributed processing system is suitable not only for tracking the target object in a wide-spread area but also for performing complex behaviors.

For all processors in the distributed processing system to effectively work together as an integrated tracking system, we consider that cooperation among the processors is significant. In order to realize real-time flexible tracking in a wide-spread area, we employ the idea of *Cooperative Distributed Vision* (CDV, in short) [Mat98]. The CDV system consists of a group of network-connected *Observation Stations* (real-time image processor with active camera(s)) as shown in Figure 1.2, and realizes

1. wide-area dynamic scene understanding and
2. versatile scene visualization.

In the CDV system, each observation station possesses *Visual Perception*, *Action* and *Network Communication* functions. By dynamically integrating these three functions, the observation station can behave as an intelligent autonomous system. With cooperation among observation stations, the system as a whole works persistently as a real world system.

In addition to these advantages, the CDV system has the following advantages that are the properties of a distributed system:

- Robustness achieved by integrating multilateral information.
- Flexibility of the system organization.

- Compensation for breakdowns.

Although we do not focus on these advantages in this thesis, they are also required functions to realize a real world system that works persistently. In recent years, therefore, a number of related researches are reported (see [Kan97], [SH97] and [CLK$^+$00], for example).

In this thesis, we propose a real-time cooperative tracking system that gazes at multiple objects simultaneously based on the concept of the CDV system. The system consists of communicating *Active Vision Agents* (AVAs, in short), where an AVA is a logical model of an observation station. For real-time object tracking by multiple AVAs, we have to solve the following problems:

1. How to design an active camera for dynamic object detection and tracking.
2. How to realize real-time object tracking with an active camera.
3. How to realize cooperation among AVAs for real-time multi-target object tracking.

In what follows, we propose our basic ideas to solve these problems.

1.4 Basic Ideas

1.4.1 Active Camera for Wide-area Imaging

The observation station for real-time object tracking should (1) keep tracking the moving object, (2) detect the object information in the observed image in real time, and (3) change its focusing visual field to effectively gaze at the target object. An active camera of the observation station is, therefore, required to

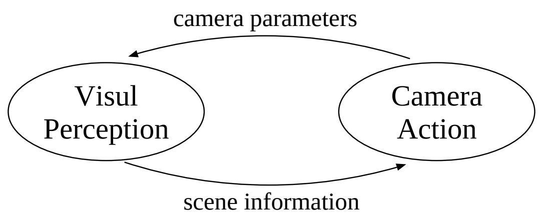
Function 1: observe a wide (omnidirectional) area,

Function 2: adjust its camera parameters to obtain the required information of a target object, and

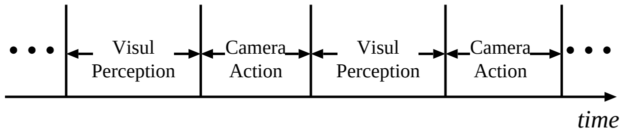
Function 3: capture an image that facilitates quick and robust detection of object regions while changing camera parameters.

With an active camera which can physically control its visual field, the above function 1 can be obtained. In previous active camera systems (e.g., Active Vision [AWB88] [Bal89], Visual Servo [Bro90] [WSN87] and Robot Vision [RH90] [BT93]), function 2 was mainly discussed. Without function 3, however, the system has to analyzes the complex observed images which involve various 3D information (geometric and photometric information). This makes real-time and robust processing difficult.

To solve this problem, we propose a well-calibrated rotational camera, where its optical projection center is placed at the rotational center (i.e., an intersection of pan and tilt axes). We call this camera a *Fixed-Viewpoint Pan-Tilt* camera. With this camera,



(a) Information flow



(b) System dynamics

Figure 1.3: Behavior of the system in active vision.

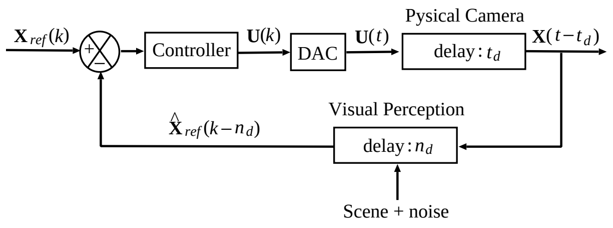


Figure 1.4: Position-based visual feedback system with delays.

appearance variations in the observed images are suppressed. This property greatly facilitates the image processing (e.g., detecting object region in the image while dynamically changing the view direction).

Usually, the projection center shifts according to zooming. It breaks down the fixed-viewpoint rotation. We can however realize a *Fixed-Viewpoint Pan-Tilt-Zoom* (FV-PTZ, in short) camera by adjusting the projection center to the rotational center while zooming.

1.4.2 Real-time Object Detection and Tracking

With an FV-PTZ camera, we can easily realize real-time object detection and tracking. The tasks of the system are as follows:

Task 1: Detect an object that comes into the scene,

Task 2: Track it by controlling pan and tilt parameters, and

Task 3: Capture object images in as high resolution as possible by controlling the zoom.

Task 1 is the function of visual perception, and tasks 2 and 3 are the functions of action. Since the action planning is determined based on image analysis by visual perception, the integration of visual perception and action is significant.

In many Active Vision systems, the visual perception and camera action modules are activated alternately as shown in Figure 1.3. With this procedure, one of the modules is

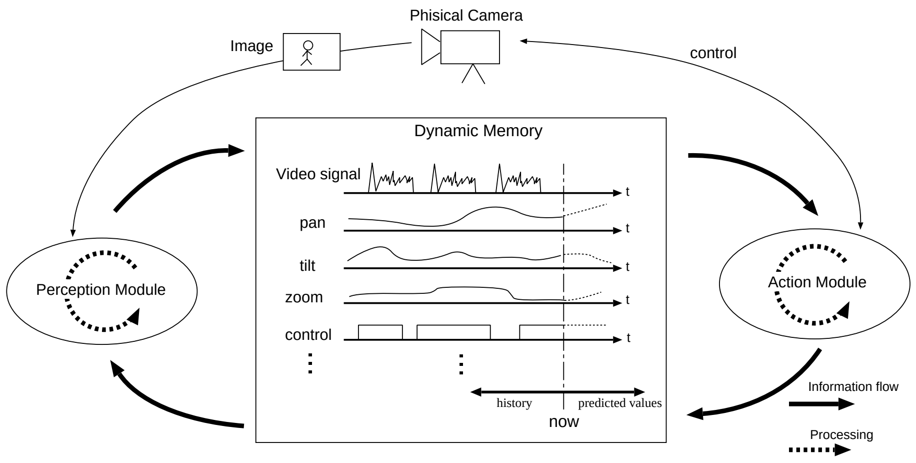


Figure 1.5: Real-time object tracking system using the dynamic memory (dynamic vision system).

suspended while the other module is working. It degrades the reactive response of the system. This disadvantage is fatal for real-time system that works in the real world.

In Visual Servo, various dynamic control methods have been studied based on the control theory. For example, visual feedback that takes into account both control and perception delays[Bro90] is employed as shown in Figure 1.4. In this system, the visual perception and camera action modules work in parallel and exchange the information with each other. Inter-module interactions, however, are simple and fixed. That is, this system cannot adapt itself to the situation with unpredictable dynamic variations.

To cope with unpredictable dynamic variations in the real world, more flexible dynamic interaction between the visual perception and action modules is required. We introduce a novel scheme named *Dynamic Vision*, where the event driven asynchronous interaction between the visual perception and action modules is realized. In dynamic vision, each module has its own dynamics and asynchronously exchange information with other modules as the occasion demands.

To implement a dynamic vision system, the *Dynamic Memory Architecture*[MHW$^+$00] is useful as illustrated in Figure 1.5. In this example, the visual perception and action modules share what we call the *dynamic memory*. In the dynamic memory, histories of control signals as well as state variables such as pan, tilt and zoom parameters are recorded as time-series data. The visual perception and action modules write into and read from the dynamic memory depending on their objectives and dynamics. It enables the modules not only to asynchronously exchange the information with other modules but also to obtain the information observed at any time, including prediction. Figure 1.5 illustrates the actual processing (i.e., image analysis and camera control) and information flow in object tracking with dynamic vision.

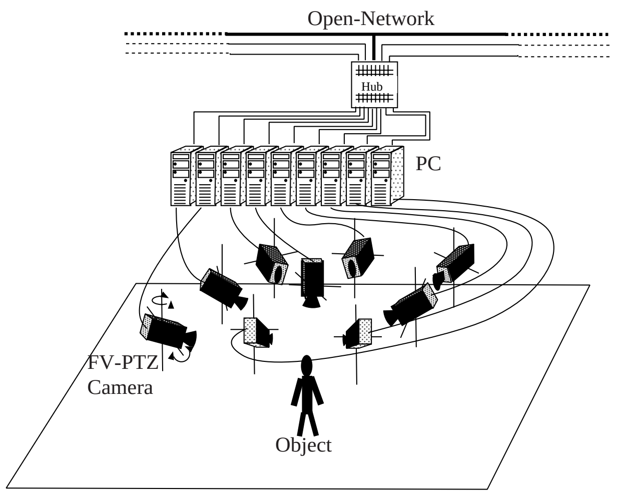


Figure 1.6: System architecture and organization.

In this thesis, we first show the active background subtraction with the FV-PTZ camera for object detection and tracking, and then present real-time object tracking system with the dynamic memory. By employing this real-time object tracking, each observation station can track its target object independently.

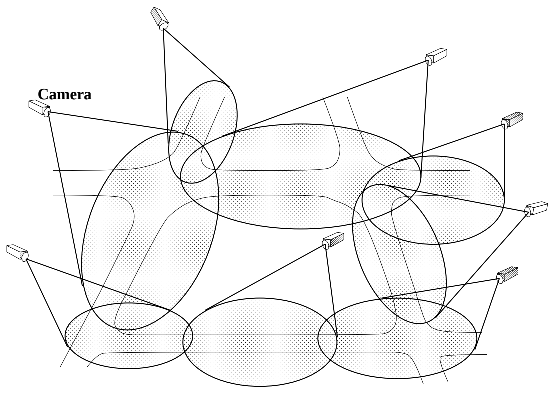
1.4.3 Real-time Cooperative Multi-target Tracking

Finally, we design real-time cooperative multi-target tracking by communicating AVAs. As mentioned before, for the system to effectively perform distributed processing, cooperation among AVAs is most important. We therefore put our focus upon how to realize real-time cooperation among AVAs.

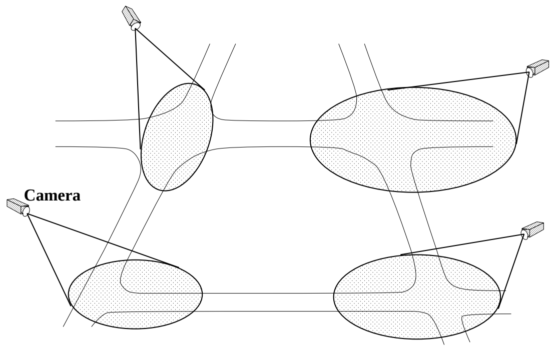
In our system, each AVA consists of a network-connected computer with a single FV-PTZ camera. By employing the properties of the FV-PTZ camera, each AVA can detect objects and track its target object independently in real time. Many cameras are embedded in the real world, and observe a wide area as shown in Figure 1.6. To effectively observe the scene, we have to arrange the cameras depending on the task:

Effective camera configuration: In Computer Vision and Robot Vision, many camera-configuration planning methods for various tasks have been studied [TAT95] [TTA95] [CK88]. In general, effective camera configuration is determined depending on the task given to the system. This is one of the most important problems for establishing multi-camera systems.

In this thesis, although we do not concentrate on realizing the effective camera configuration for object tracking, we impose the constraint about the camera configuration on the system: visual fields of cameras are overlapping with each other in

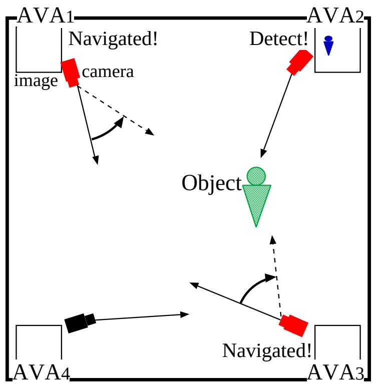


(a) Overlapped configuration

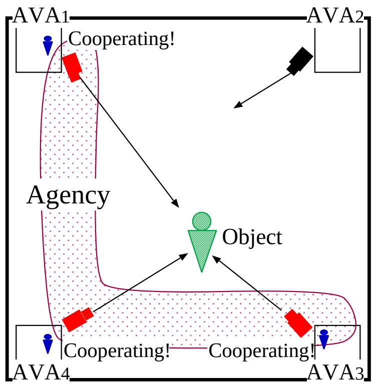


(b) Isolated configuration

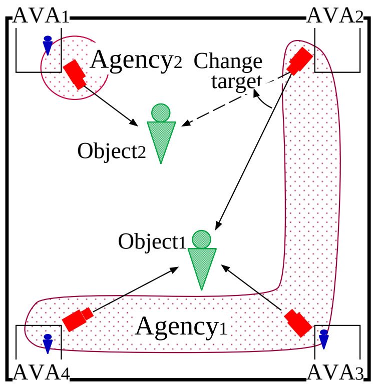
Figure 1.7: Camera configuration for wide-area observation.



(a) Gaze navigation



(b) Cooperative gazing



(b) Adaptive tracking

Figure 1.8: Basic scheme for cooperative tracking.

order to keep tracking a target object in the observation scene without a break (Figure 1.7 (a)). That is, in our system, the area of the observation scene is determined by the number of cameras and their visual fields.

With the above system organization, we realize a multi-AVA system that cooperatively detects and tracks a target object. Following are the tasks of the system:

1. Initially, each AVA independently searches for an object that comes into the observation scene.

2. When an AVA detects an object, the AVA examines whether or not the information of the detected object is required to the given task. If the information is required, the AVA regards the detected object as a target object.

3. If the AVA detects the target object, the AVA navigates the gaze of other AVAs towards the target object as illustrated in Figure 1.8 (a).

4. An AVA, which is required to gaze at the target object by another AVA, decides whether it accepts the navigation or continues its current role depending on the situation.

5. AVAs, all of which gaze at the same object, keep tracking the focused target object cooperatively without being disturbed by obstacles and other moving objects as illustrated in Figure 1.8 (b). A group of AVAs that track the same object is called an *Agency*.

6. Depending on the target motion, each AVA dynamically changes its target object as illustrated in Figure 1.8 (c).

7. When the target object gets out of the scene, the AVA decides whether it searches for an object again or tracks another target object that is tracked by other AVAs depending on the situation.

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 discriminate between multiple objects in the scene.

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 a new object even if the system is tracking the target object.
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.

In order to implement the real-time cooperation, 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. The dynamic interaction in each layer allows the whole system to track multiple objects under complicated dynamic situations in the real world.

1.5 Overview of Thesis

In this section, we present the organization of the subsequent chapters in the thesis. Figure 1.9 shows a flow of the thesis.

Chapter 2 presents a *Fixed-Viewpoint Pan-Tilt-Zoom* camera with the goal of establishing an active camera which is possessed by the observation station. The FV-PTZ camera can control its gazing direction (pan and tilt parameters) and the resolution of the image in its capturing (zoom parameter).

An active camera of the observation station is required to (1) be able to observe the wide area, (2) be able to adjust its camera parameters to obtain the required information of the target object, and (3) capture an image that facilitates quick and robust detection of object regions while changing camera parameters. We realize these functions with the FV-PTZ camera. Experimental results demonstrate the practical effectiveness of the FV-PTZ camera for the wide-area active sensing.

Chapter 3 presents an *Active Background Subtraction* method for object tracking by employing the FV-PTZ camera. In this section, we describe the basic idea for controlling pan, tilt and zoom parameters to continuously gaze at the target object.

To effectively gaze at the target object during tracking, we have to design an integration of visual perception and action functions of the camera. We realize this integration by the dynamic information exchange between the visual perception and action modules. We prove the effectiveness of the active background subtraction by experiments in tracking a moving object.

Chapter 4 first introduces a novel dynamic system architecture named *Dynamic Memory Architecture*[MHW$^+$00]. This architecture can be applied for the system consisting of multiple parallel processes. Through the dynamic memory, multiple parallel processes can (1) exchange the information without synchronization, and (2) obtain the value which are taken at arbitrary time by recording the information into the dynamic memory as the time-series data.

By employing the dynamic memory, we can dynamically integrate the visual perception and action modules of the tracking system to realize a flexible temporal coordination between them. As a result, the system can increase the tracking ability and obtain intimate information of the target object with smooth camera motion. Experimental results demonstrate the great improvement of the performance and stability of object tracking with the dynamic memory.

Chapter 5 proposes a *Real-time Cooperative Multi-target Tracking System* based on th concept of the CDV system, which is a main contribution of this thesis. This tracking system consists of communicating multiple AVAs. For real-time object tracking by multiple AVAs, we put our focus upon how to realize real-time cooperation among AVAs.

In order to implement the real-time cooperation among AVAs, we propose a three-layered interaction architecture. With the real-time cooperation through the dynamic interaction in each layer, the system as a whole can track multiple moving objects under complicated dynamic situations in the real world.

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

Chapter 6 presents an *Observable-area Model* of the scene for real-time cooperative object tracking by multiple AVAs. It is one of augmentations of our cooperative tracking system. We put our focus upon sharing knowledge of all the AVAs' abilities (i.e. observable area in the scene) for efficient object tracking. With this knowledge, the tracking system can plan an efficient role assignment to AVAs so that the system as a whole effectively tracks all the target objects.

To realize this efficient role assignment, the system gathers the observable-area information of all AVAs to incrementally generate the observable-area model (i.e., scene model) during tracking. We experiment to verify the effectiveness of the proposed scene model for cooperative tracking.

Chapter 7 summarizes the work of this thesis and points out next steps for the future real-world vision systems.

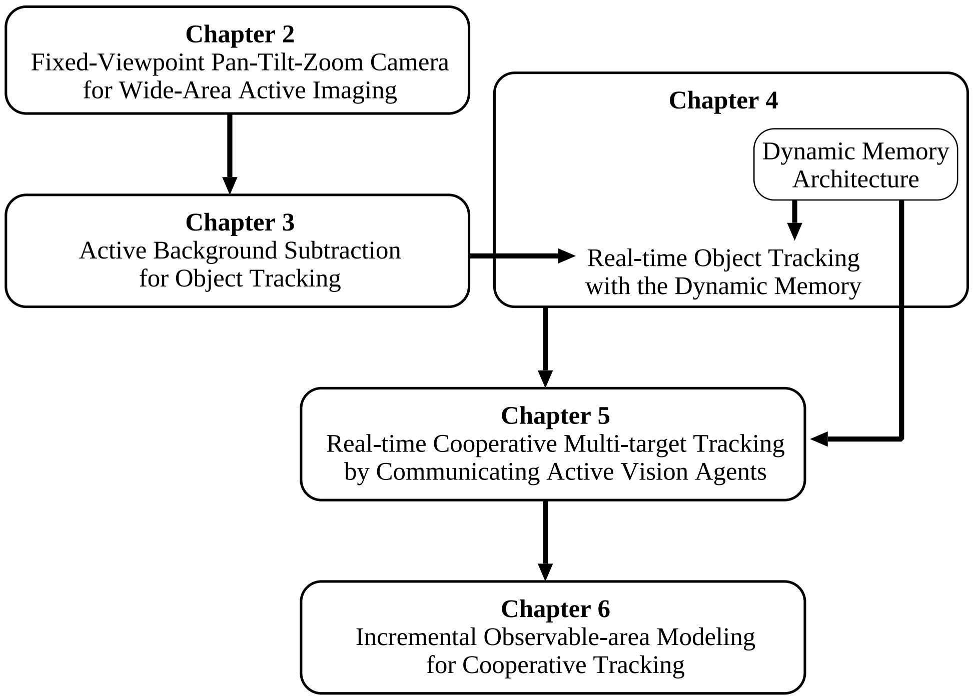

Chapter 2
Fixed-Viewpoint Pan-Tilt-Zoom Camera for Wide-Area Active Imaging
Chapter 3
Active Background Subtraction for Object Tracking
Chapter 4
Dynamic Memory Architecture
Real-time Object Tracking with the Dynamic Memory
Chapter 5
Real-time Cooperative Multi-target Tracking by Communicating Active Vision Agents
Chapter 6
Incremental Observable-area Modeling for Cooperative Tracking

Figure 1.9: Flow of the thesis.

Chapter 2

Fixed-Viewpoint Pan-Tilt-Zoom Camera for Wide-area Active Imaging

2.1 Wide-area Active Imaging for Object Detection and Tracking

In this chapter, we present a *Fixed-Viewpoint Pan-Tilt-Zoom* camera that has been designed to achieve the goal of active control by an observation station for real-time object tracking.

The observation station for real-time object tracking should

- monitor a wide-spread area,
- detect object(s) from the observed image in real time,
- keep tracking a target object as it moves around, and
- change its visual field for effectively gazing at the target object.

Accordingly, a camera controlled by the observation station must perform the following functions:

Function 1: Observe a wide area.

Function 2: Focus on a target object to obtain a high-resolution object image.

Function 3: Capture an image that facilitates quick and robust detection of object regions while camera parameters are changed.

That is, adaptive wide-area observation is required for real-time object detection and tracking. We have realized these functions with the FV-PTZ camera.

2.1.1 Adaptive Wide-area Observation

We categorize wide-area observation methods into two classes.

Optical methods: Omnidirectional cameras using fish-eye lenses and curved mirrors [YY91] [YYY95] [PN97].

Mechanical methods: Computer-controlled active cameras that can change their visual field [IYT90] [MB94].

In the optical methods, a fixed camera captures the omnidirectional image on the curved mirror. Yagi *et al.*[YY91], Yamazawa *et al.*[YYY95], and Peri *et al.*[PN97] presented omnidirectional image acquisition methods using cone-shaped, hyperbolic-curved, and parabolic mirrors, respectively. Since these methods capture the omnidirectional image onto a single sensing element (CCD), they can provide video-rate wide-area image capturing. Their image resolutions are, however, fixed and limited:

- Since all of the omnidirectional view is captured onto a single CCD, the acquired appearance information is coarse.
- Optical methods cannot control the zooming factor of the camera to acquire the high-resolution image of the focused object.

Moreover, since the camera observes through a mirror in these methods, objects beyond the mirror cannot be observed.

In the mechanical methods, on the other hand, the instantaneous visual field of an active camera is limited. However, an omnidirectional scene can be observed by controlling the camera parameters, and high-resolution images can be acquired by controlling the zoom parameter.

Visual surveillance tasks, including object tracking, require high-resolution image acquisition because accurate information on the object is useful for reliable object identification. Furthermore:

- An active camera can adaptively control the resolution to accommodate the dynamic situation in the real world.
- We can avoid the problem of a narrow instantaneous visual field by employing many cameras. That is, multiple cameras can supplement each other to cover their unobservable fields.

For these reasons, we have adopted an active camera as the image sensor to realize functions 1 and 2 described in Section 2.1 (i.e., wide area observation and close observation for gazing at the target object). Next, we discuss how to realize function 3, namely how to capture images that facilitate image processing for object detection.

2.1.2 Image Appearance Variations caused by Changing Camera Parameters

In active camera control, an observed image is first analyzed, and then camera parameters are determined based on the result of the image analysis. For example, in object tracking:

1. Detect the object region in the observed image.
2. Determine the next camera parameters so that the object will be projected in the center of the observed image.

Such an observation method is called *Active Vision*[AWB88][Bal89].

By controlling the camera parameters, various appearance variations are caused in the observed image:

- **Geometric variations:** These are caused by changing the camera location, view direction and zoom.
- **Photometric variations:** These are caused by changing focus[1] , iris, gain control and shutter speed.

When we employ an active camera to detect and track objects over a wide area, two camera parameters, which produce the above types of variations, should be controlled:

- **Geometric camera parameters:** These must be controlled to change the visual field of the camera for wide-area observation.
- **Photometric camera parameters:** In general, when the visual field of the camera is changed, illumination conditions in the visual field are variable. In addition, since target objects move around the scene at various speeds, constant shutter speed is not enough to prevent motion blurs from being included in the observed image. Depending on the illumination conditions and the target motion, therefore, the photometric camera parameters should be controlled to acquire the meaningful object information. The following examples show how this is done:
  - Iris and gain control are used to adjust the dynamic range of the camera to the illumination.
  - If an object moves at high speed, the shutter speed is increased. Since a fast shutter speed darkens the observed image, iris and gain control should be controlled to acquire a meaningful image.

In this thesis, we assume that the system does not control the photometric parameters and concentrate on controlling the geometric parameters.

To analyze images taken with different variations of geometric camera parameters, we have to discuss the factors that cause appearance variations in the observed image (Figure 2.2 and Figure 2.3).

[1] Strictly speaking, geometric variation is also caused by changing the focus parameter. However, this geometric variation can be easily calibrated and is negligible with a telecentric lens. Hence, here we simplify the camera model so that focus control only causes photometric variation.

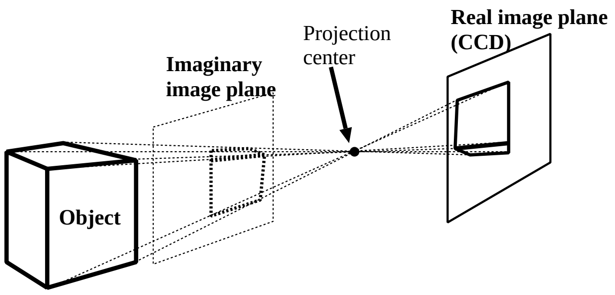


Figure 2.1: Perspective projection.

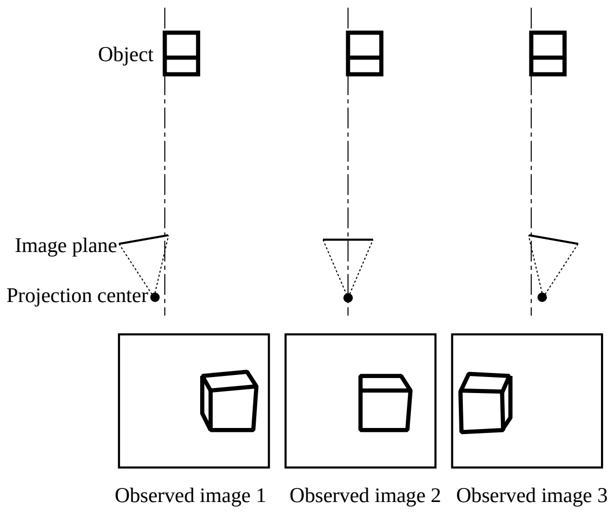


Figure 2.2: Images with parallax.

We suppose a projection from a 3D scene to a 2D image to be a *Perspective Projection*[Fau93]. With this projection, all rays are projected onto the image plane through the projection center (Figure 2.1). In a practical projection system, the projected image is reversed on a CCD. To simplify this concept, we assume an imaginary image plane at the opposite side of the projection center and let the image be projected onto the imaginary image plane without reversal. Hereafter, we call such an imaginary image plane simply the image plane.

If the camera observes the 3D scene while changing its camera parameters, the object appearance in the observed image changes. This variation is caused by the following two factors, namely parallax and image deformation.

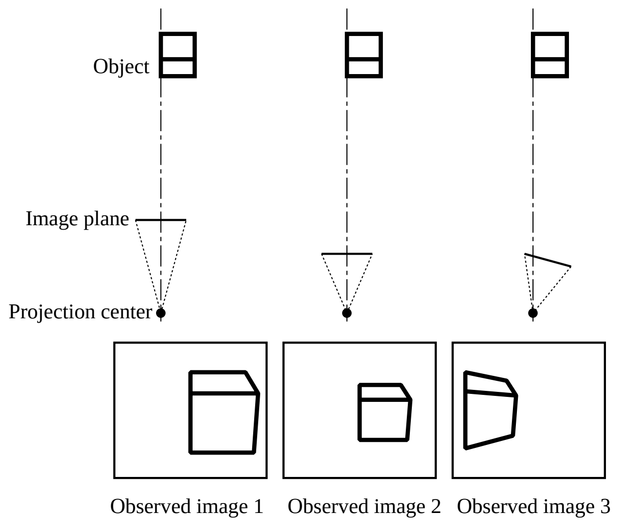


Figure 2.3: Images without parallax.

2.1.2.1 Parallax

If the camera observes the 3D scene while moving its projection center, the camera observes objects in the scene from different view positions and angles. Consequently, object appearances in the observed images vary. These variations between the observed images are called *Parallax*. Figure 2.2 shows examples of parallax. Observed images 1, 2, and 3 are captured while the projection center of the camera is moved. In these images, the side of the object is visible or invisible depending on the geometric configuration between the projection center and the object.

By employing parallax information, several real-time stereo vision system reconstruct 3D information of the scene based on the concept of the triangulation [TWM01] [SO00]. These systems posses multiple cameras, and these cameras observe the scene simultaneously. In [TWM01], 3D depth map is obtained from trinocular images based on the multiple-baseline stereo method[YK86] [OK93]. These images are taken by the trinocular lenses mounted on the pan-tilt camera head. The system can (1) detect a moving object by comparing the input 3D map with the 3D map of the stationary scene and (2) track it by controlling pan-tilt directions, without being interfered by variations of illumination conditions. In [SO00], the system with five cameras integrates the results of (1) active region extraction, (2) multi-view stereo with occlusion handling, and (3) multi-view stereo without occlusion handling, each of which is analyzed by different PCs. By effectively integrating all the results, the system can acquire the precise 3D depth map at video-rate.

If the projection center is fixed during observation, on the other hand, the geometric configurations between the projection center and objects in the scene are unchanged. This property is kept even if the location and the posture of the image plane change. We call a camera whose projection center is fixed a *Fixed-Viewpoint* camera (FV camera, in short).

An FV camera enables us to observe and take images as shown by the examples in Figure 2.3. These images are observed while the location and posture of the image plane are changed. Since the geometric configuration between the projection center and the object is fixed, parallax does not occur in the observed images. These images facilitate image processing with appearance based analysis.

In the above camera configuration, variable camera parameters are restricted to the view direction (pan and tilt angles) and the zoom. Such an FV camera is defined as a *Fixed-Viewpoint Pan-Tilt-Zoom* camera (FV-PTZ camera, in short).

Restricting variable camera parameters reduces the visual field of the camera because the 3D location of the camera is fixed. This disadvantage can be overcome by employing many cameras. That is, they can cover each others unobservable visual fields.

2.1.2.2 Image Deformation

Even if the projection center of the camera is fixed, the geometric configuration between the projection center and the image plane varies when the posture of the image plane is changed. Therefore, while images observed by an FV-PTZ camera do not include any geometric variations due to the 3D scene geometry, object shapes in the images vary with the camera motion as shown in Figure 2.3. These variations are caused by the movement of the image plane (i.e., variations in the location and posture of the image plane). We call this difference in the image appearance an *Image Deformation.* We cope with this problem by employing a geometric transformation between images taken by the FV-PTZ camera.

In what follows, Section 2.2 first presents a geometric configuration of the projection center for realizing the FV-PTZ camera and then describes a geometric property of images taken by the FV-PTZ camera (Section 2.2.1) and high-speed image transformation between these images (Section 2.2.3). Section 2.3 shows (1) a scene model representation with a group of images taken by the FV-PTZ camera and (2) an image generation method from this scene model. Section 2.4 proposes a practical camera calibration method to realize the FV-PTZ camera system. In Section 2.5, experimental results demonstrate the practical effectiveness of our idea for wide-area active sensing.

2.2 Fixed-Viewpoint Pan-Tilt Zoom Camera

This section presents an active image sensing and processing method to cope with appearance variations in observed images.

2.2.1 Designing the FV-PTZ camera

We realize the FV-PTZ camera configuration with the following practical active camera design:

1. Make pan and tilt axes intersect with each other. The intersection should be at a right angle to facilitate later geometric computations.

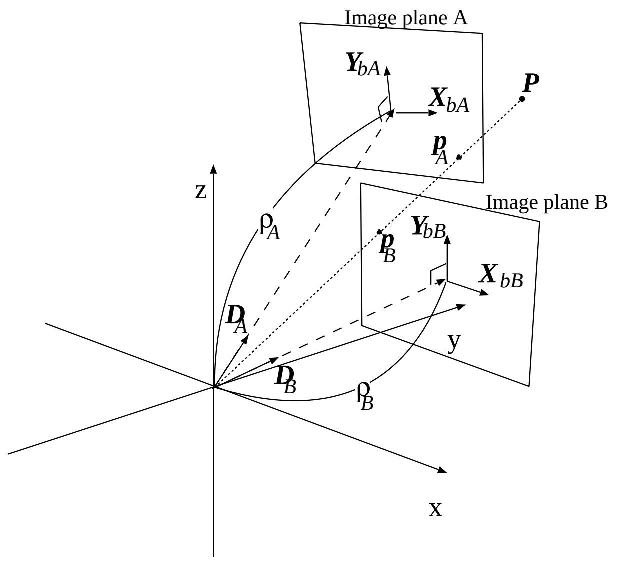


Figure 2.4: Projection onto two different image planes from a 3D point.

2. Place the projection center of the camera at the intersection point. The optical axis of a camera should be perpendicular to the plane defined by the pan and tilt axes.

We call the above designed active camera the *Fixed-Viewpoint Pan-Tilt* (FV-PT, in short) camera.

Zooming can be usually modeled by shifting the projection center along the optical axis[LDPD97]. Therefore, to realize the FV-PTZ camera, either of the following additional mechanisms should be employed:

- Design a zoom lens system whose projection center is fixed irrespectively of zooming.
- Introduce a slide stage that keeps the projection center fixed irrespectively of zooming.

With the fixed-viewpoint camera, the reflection on the object and other photometric appearances are also invariable between observed images. This is because the geometric configuration between the projection center and the light source is fixed. This guarantees that variations in the view point do not produce parallax, as do photometric variations due to the 3D scene structure.

2.2.2 Imaging geometry of the FV-PTZ camera

If a 3D point is projected onto different multiple images by the FV-PTZ camera, the projected 2D points can be transformed alternately irrespective of the location of the projected 3D point. This property can be proven as follows.

Figure 2.4 illustrates a projection onto two different image planes (denoted by plane$_A$ and plane$_B$) from a 3D point (denoted by $\boldsymbol{P}$) in the scene. In this figure, the origin of the coordinate system denotes the projection center of the camera. Plane$_A$ is an image plane that is determined by the following parameters:

- Distance from the origin to plane$_A$ is ρ_A (> 0). ρ_A is identical to the focal length of the camera (i.e., the zoom parameter).
- 3D vector $\boldsymbol{D}_A$ ($= (k_A, l_A, m_A)^T$) is a unit normal vector of plane$_A$. $\boldsymbol{D}_A$ corresponds to the view direction of the camera (i.e., the pan and tilt parameters).
- Plane$_A$ is expressed as $k_A x + l_A y + m_A z = \rho_A$.

$\boldsymbol{P}_A$, which is a 2D point projected from a 3D point $\boldsymbol{P}$ ($\boldsymbol{D}_A^T \boldsymbol{P} > 0$) onto plane$_A$, is then expressed by

$$\boldsymbol{P}_A = \frac{\rho_A}{\boldsymbol{D}_A^T \boldsymbol{P}} \boldsymbol{P}. \tag{2.1}$$

A 3D point $\boldsymbol{P}_A$ can be expressed as the 2D point $\boldsymbol{p}_A$ in a 2D image coordinate system whose origin is $\rho_A \boldsymbol{D}_A$.

$$\boldsymbol{p}_A = \begin{pmatrix} \boldsymbol{X}_{bA}^T \\ \boldsymbol{Y}_{bA}^T \end{pmatrix} \boldsymbol{P}_A, \tag{2.2}$$

where $\boldsymbol{X}_{bA}$ and $\boldsymbol{Y}_{bA}$ are the 3D orthogonal basis vectors of the image coordinate system.

$$\boldsymbol{X}_{bA} = \frac{1}{\sqrt{1 - m_A^2}} \begin{pmatrix} l_A \\ -k_A \\ 0 \end{pmatrix} \tag{2.3}$$

$$\boldsymbol{Y}_{bA} = \frac{m_A}{\sqrt{1 - m_A^2}} \begin{pmatrix} -k_A \\ -l_A \\ \frac{1-m_A^2}{m_A} \end{pmatrix} \tag{2.4}$$

That is, $\boldsymbol{X}_{bA}^T \boldsymbol{Y}_{bA} = 0$, $\boldsymbol{X}_{bA}^T \boldsymbol{D}_A = 0$, $\boldsymbol{Y}_{bA}^T \boldsymbol{D}_A = 0$ and $||\boldsymbol{X}_{bA}|| = ||\boldsymbol{Y}_{bA}|| = 1$.

Conversely, the 2D point $\boldsymbol{p}_A$ can be expressed as the 3D point $\boldsymbol{P}_A$:

$$\boldsymbol{P}_A = (\boldsymbol{X}_{bA} \boldsymbol{Y}_{bA}) \boldsymbol{p}_A + \rho_A \boldsymbol{D}_A \tag{2.5}$$

If a 3D point $\boldsymbol{P}$ is projected onto plane$_B$, a 2D point $\boldsymbol{p}_B$ is represented by the following equation from equations (2.1), (2.2) and (2.5).

$$\boldsymbol{p}_B = \frac{\rho_B \begin{pmatrix} \boldsymbol{X}_{bB}^T \\ \boldsymbol{Y}_{bB}^T \end{pmatrix} \{(\ \boldsymbol{X}_{bA}\ \boldsymbol{Y}_{bA}\ ) \boldsymbol{p}_A + \rho_A \boldsymbol{D}_A\}}{\boldsymbol{D}_B^T \{(\ \boldsymbol{X}_{bA}\ \boldsymbol{Y}_{bA}\ ) \boldsymbol{p}_A + \rho_A \boldsymbol{D}_A\}}, \tag{2.6}$$

where plane$_B$ has a unit normal vector $\boldsymbol{D}_B$, distance from the origin ρ_B, and 3D orthogonal basis vectors $(\boldsymbol{X}_{bB}, \boldsymbol{Y}_{bB})$.

From equation (2.6), it is obvious that $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$, each of which are projected from the same 3D point $\boldsymbol{P}$ onto different image planes, can be mutually transformed independent of $\boldsymbol{P}$. By employing this property, we can rectify the image deformation between images taken by the FV-PTZ camera. Thus, we can easily compare multiple observed images for image processing and analysis.

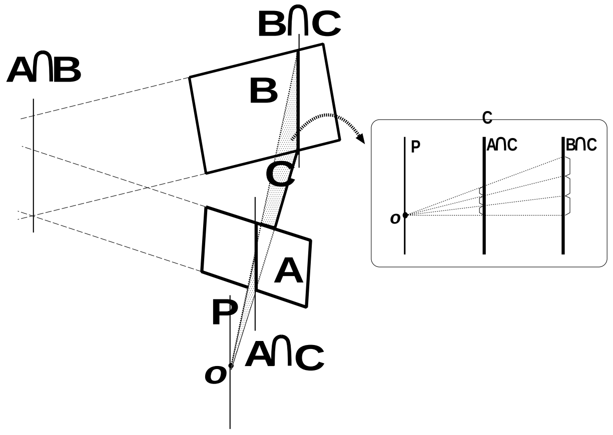


Figure 2.5: Basic idea of linear-wise transformation between images.

2.2.3 High-Speed Image Generation between Multiple Screens

Since 2D coordinate transformation between different planar screens is defined by equation (2.6), we can generate (1) a seamless wide panoramic image (described below) from observed images and (2) an image taken with any pan-tilt-zoom parameters from the panoramic image, by a transformation between 2D coordinates. This transformation, however, requires a complex non-linear computation and degrades real-time processing.

To reduce the number of arithmetic operations in a transformation between two planar images, we can exploit the following property (Figure 2.5):

For any combination of two planes A and B, there exists at least one set of planes $\{C\}$ whose member C satisfies the condition that two intersection lines $A \bigcap C$ and $B \bigcap C$ are parallel and C involves a fixed line P passing a point o.

Proof

- **Planes A and B are not parallel**, there exists an intersection line $A \bigcap B$. The planes A and B can be decomposed into two disjoint sets of parallel lines $\{L_A\}$ and $\{L_B\}$, both of which are parallel to $A \bigcap B$. Let a line P passing the point o be parallel to $A \bigcap B$. A plane C involving line P makes two intersection lines $A \bigcap C$ and $B \bigcap C$.

  Since plane C involves P and P is parallel to $A \bigcap B$, plane C is also parallel to $A \bigcap B$. Therefore, plane C can also be decomposed into disjoint set of parallel lines $\{L_C\}$, whose members are parallel to $A \bigcap B$. Hence, the parallel line decompositions $\{L_A\}$, $\{L_B\}$, and $\{L_C\}$ all consist of lines parallel to $A \bigcap B$. Any two different lines chosen from $\{L_A\} \bigcup \{L_B\} \bigcup \{L_C\}$ never intersect because all of these lines are parallel. That is, $A \bigcap C = \{L_A\} \bigcap \{L_C\} = L_{AC}$, and $B \bigcap C = \{L_B\} \bigcap \{L_C\} = L_{BC}$. Since $L_{AC}, L_{BC} \in \{L_C\}$, $A \bigcap C$ and $B \bigcap C$ are parallel.

- **Planes A and B are parallel**, any plane C that is not parallel to these planes makes two intersection lines $A \bigcap C$ and $B \bigcap C$, both of which are obviously parallel. Hence, $\{C\}$ can be defined for any line P passing point o.

Q.E.D.

According to the above property, the projection between two planes A and B can be decomposed into projections between two parallel lines $A \bigcap C$ and $B \bigcap C$ ($C \in \{C\}$) by regarding point o as the center of projection. Since these two lines are parallel, the projection between two lines is simplified to 1D scaling. This can be implemented as linear scanning: starting from two corresponding points $\overrightarrow{x_0}$ and $\overrightarrow{X_0}$, subsequent corresponding points on these lines are computed by adding constant 2D vectors $\overrightarrow{\delta}$ and $\overrightarrow{\Delta}$ to these points:

$$\overrightarrow{x_{i+1}} = \overrightarrow{x_i} + \overrightarrow{\delta}, \quad \overrightarrow{X_{i+1}} = \overrightarrow{X_i} + \overrightarrow{\Delta}. \tag{2.7}$$

This means **4** additions are enough to compute one-to-one correspondence between points in two different planes. However, there are initialization overheads to compute (1) starting point pair $\overrightarrow{x_0}$ and $\overrightarrow{X_0}$, and (2) scaling coefficients. These overheads are equivalent to computing two one-to-one correspondences between points in two planes, two vector subtractions, and two scalar divisions per line pair. In the case of transformation to a plane consisting of $n_x \times n_y$ elements, the overheads per point are estimated as: $16/\sqrt{n_x n_y}$ additions, $12/\sqrt{n_x n_y}$ multiplications, and $6/\sqrt{n_x n_y}$ divisions, where $\sqrt{n_x n_y}$ represents the estimated number of lines on the plane.

2.3 Scene Model Representation for the FV-PTZ Camera

Rectifying the image deformation between images taken by the FV-PTZ camera allows us to integrate all observed images by projecting them onto a common virtual screen. On the virtual screen, the projected images form a seamless wide panoramic image.

For the integration, we can use arbitrarily shaped virtual screens. Sections 2.3.1 and 2.3.2 describe typical examples.

2.3.1 Appearance Sphere

When we observe the 360° panoramic view by rotating the camera, a spherical screen can be used (Figure 2.6). We call this spherical screen *APpearance Sphere* (APS, in short). The omnidirectional image re-projected to the APS is called an APS image. In general, omnidirectional images can be re-projected to any star-shaped closed screen. In the APS, since the distances from the projection center to all positions on the virtual screen are equal to each other, the image resolutions are uniform for all directions.

We will first describe the APS image generation method. All pixel values in the APS image are projected from observed images taken by the FV-PTZ camera. Figure 2.7

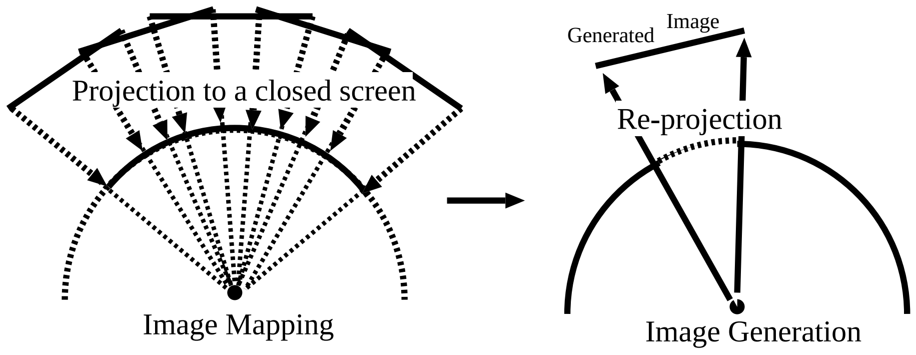


Figure 2.6: Appearance sphere.

illustrates a projection onto an image plane and a spherical screen (APS) from a 3D point. The 3D points $\boldsymbol{P}_A$ and $\boldsymbol{S}(\varphi^*, \theta^*)$ are projected points from a 3D point $\boldsymbol{P}$ onto image plane$_A$ and a spherical screen, respectively. Let a 3D point $\boldsymbol{P}_{AS}$ be a projected point from $\boldsymbol{P}_A$ onto a spherical screen.

A spherical screen is expressed as follows:

$$\boldsymbol{S}\left(\varphi, \theta\right) = r\left(\cos\varphi\cos\theta, \cos\varphi\sin\theta, \sin\varphi\right), \tag{2.8}$$

where r is a radius of the sphere and $-\frac{\pi}{2} \leq \varphi \leq \frac{\pi}{2}$, $0 \leq \theta < 2\pi$. $\boldsymbol{S}(\varphi^*\theta^*)$ is then expressed as follows:

$$\boldsymbol{S}\left(\varphi^*, \theta^*\right) = \frac{r}{||\boldsymbol{P}||}\boldsymbol{P}, \tag{2.9}$$

where $\varphi^* = \cos^{-1}\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right)$, $\theta^* = \cos^{-1}\left(\frac{x}{\sqrt{x^2+y^2}}\right) = \sin^{-1}\left(\frac{y}{\sqrt{x^2+y^2}}\right)$.

To generate the APS image from observed images, every pixel value at 2D coordinates $\boldsymbol{p}_A$ in the observed image is re-projected onto the APS. $\boldsymbol{P}_A$ is expressed with $\boldsymbol{p}_A$ by equation (2.5). Therefore, $\boldsymbol{P}_{AS}$ can be obtained by substituting the right side of equation (2.1) for $\boldsymbol{P}$ in equation (2.9).

$$\boldsymbol{P}_{AS} = \frac{r}{||\left(\boldsymbol{X}_{bA}\boldsymbol{Y}_{bA}\right)\boldsymbol{p}_A + \rho_A\boldsymbol{D}_A||}\left\{\left(\boldsymbol{X}_{bA}\boldsymbol{Y}_{bA}\right)\boldsymbol{p}_A + \rho_A\boldsymbol{D}_A\right\} \tag{2.10}$$

$\boldsymbol{X}_{bA}$, $\boldsymbol{Y}_{bA}$, ρ_A and $\boldsymbol{D}_A$ are given depending on the posture of the observed image plane. For every $\boldsymbol{p}_A$, $\boldsymbol{P}_{AS}$ is determined by employing the above equation. A pixel value at $\boldsymbol{p}_A$ is re-projected onto the corresponding $\boldsymbol{P}_{AS}$ in the APS.

Next, we will describe the image generation method using the generated APS image. Pixel values in the APS image are projected onto the image plane. To generate an image from the APS, a transformation between $\boldsymbol{p}_A$ and $\boldsymbol{S}(\varphi^*\theta^*)$ is required. Here again, equation (2.10) can be used. Since $\boldsymbol{P}_{AS}$ is the 3D point on the APS (i.e., $\boldsymbol{S}(\varphi^*\theta^*)$), $\boldsymbol{S}(\varphi^*\theta^*)$ is determined for every $\boldsymbol{p}_A$ by employing equation (2.10). A pixel value at $\boldsymbol{S}(\varphi^*\theta^*)$ is projected onto the corresponding $\boldsymbol{p}_A$ in the image plane.

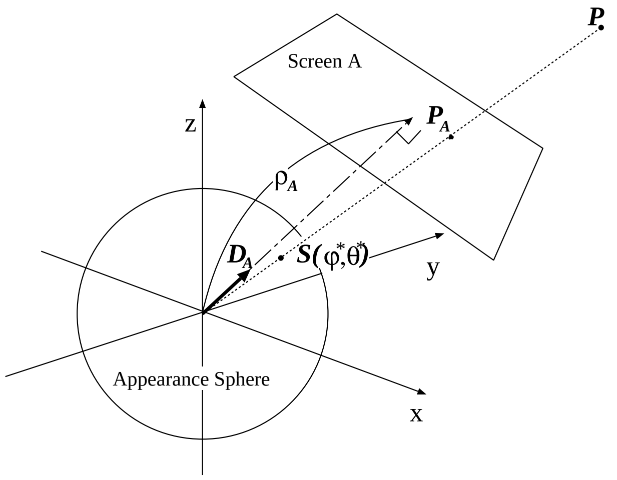


Figure 2.7: Projection onto an image plane and a spherical screen from a 3D point.

Depending on r (i.e., radius) of the APS, the surface area of the APS changes. As the surface area is increased, the accuracy of the APS image increases. We can adjust the radius of the APS depending on the task as follows:

Large radius: If high resolution is required: surveillance in a large space, inspection of precise parts.

Small radius: If high resolution is not required: Surveillance in a narrow area.

2.3.2 Appearance Plane

When the rotation angle of the camera is limited and all observed images can be re-projected to one side of a bounded plane, we can use a planar screen (Figure 2.8). We call this planar screen *APpearance Plane* (APP, in short). The panoramic image on the APP is called an APP image.

Since both the APS screen and the observed image plane are planar, the transformation between them is expressed as in equation (2.6). As we proved in Section 2.2.3, this transformation can be quickened. This property allows the system to establish real-time processing in contrast to the image generation method using the APS.

By combining multiple APPs, we can utilize the following virtual screens:

- With multi-view APPs, an omnidirectional virtual screen identical to the APS can be generated.
- By adjusting the distance from the projection center to each APP, variable image resolution can be implemented. This enables us to change the quality of the image resolution depending on the direction of the camera.

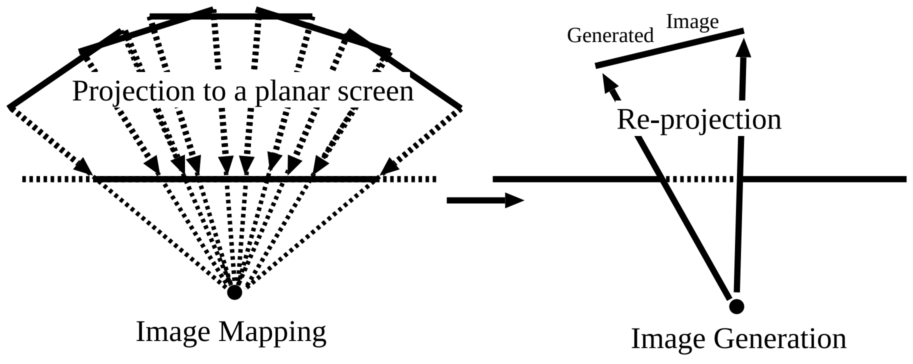


Figure 2.8: Appearance plane.

Since the image generation from the APP is a 2D coordinate transformation between two image planes, we can utilize the high-speed image transformation technique described in Section 2.2.3. This technique greatly improves the real-time image processing.

2.4 Calibration for Realizing the FV-PTZ Camera

The omnidirectional image representations mentioned in Section 2.3.1 and Section 2.3.2 are equivalent to those in [Gre86] and [Che95] in Computer Graphics and Virtual Reality. Our objective, however, is not to synthesize panoramic images natural to human viewers but to develop an active camera system that facilitates image analysis for wide area surveillance. That is, in our case both the image acquisition and the projections on/from virtual screens should be accurate enough to closely match physical camera motions. To attain such an accuracy, we have to develop sophisticated camera calibration methods.

2.4.1 Fixing the Viewpoint

In this section, we explain how to place the projection center and the intersection of pan and tilt axes for proper setup of an FV-PTZ camera.

To confirm the geometric configuration of the projection center and the rotation center of the camera, the equipment illustrated in Figure 2.9 is used. Since a laser beam passes through two translucent screens, bright spots appear on both screens. The camera observes them while rotating its pan and tilt angles[2] .

We employ the following two properties to adjust the position of the projection center to the rotational axes:

Property 1: If a ray passes through the projection center, all 3D points along the ray are projected to the same point on the image plane.

[2] For simplification, only the pan angle is rotated in Figure 2.9

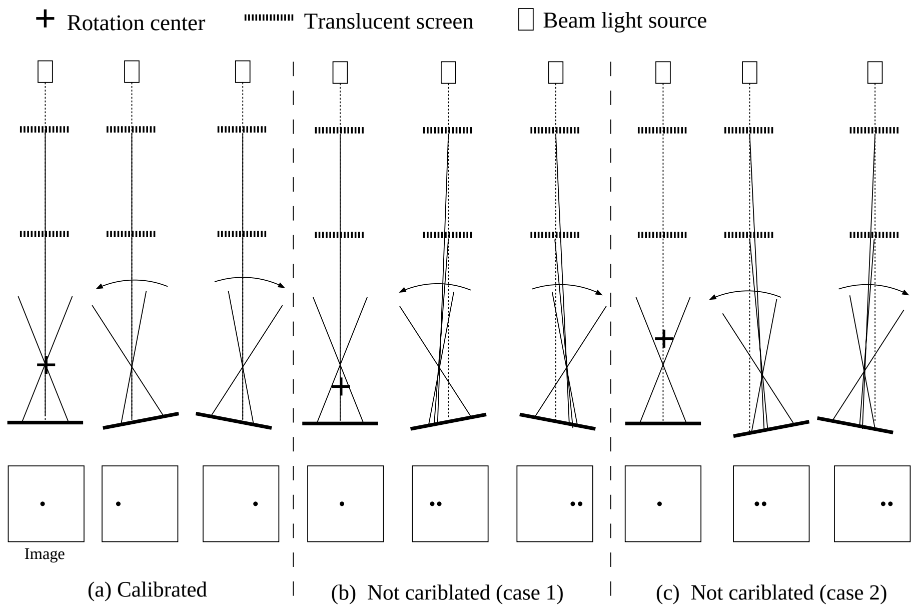


Figure 2.9: Calibration using laser beam.

Property 2: If the projection center is placed precisely on the rotational axes, a ray passing through the projection center always passes the point while the camera is rotated. All 3D points along the ray are necessarily projected to the same point on the image plane.

Because of the above properties 1 and 2, if the projection center is placed exactly at the intersection of the rotational axes, both bright spots are always projected onto the same position, even while the camera angle is rotated. That is, only one bright spot appears in the image (Figure 2.9 (a)). With other arrangements of the projection center and the rotation center, on the other hand, each bright spot is projected onto different positions in the image plane while the camera angle is rotated. In Figure 2.9 (b), the projection center is between the scene and the rotational axis. In Figure 2.9 (c), the rotational axis is between the scene and the projection center.

We actually establish the above calibration with a linear slide stage mounted on a rotational stage and a laser beam oscillator. The following procedures are applied for calibration:

Step 1: Set a laser beam so that it passes through the projection center. A bright spot appears on a translucent screen by the laser beam as shown in Figure 2.10, Step 1. If the beam passes through the projection center, two spots on translucent screens are projected to the same position on the image plane.

Step 2: Rotate the camera stage to the left side and measure the distance between beam spots in the observed image while sliding the linear stage (Figure 2.10, Step 2). The

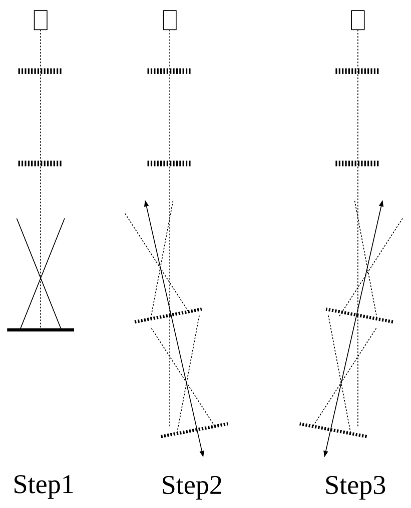


Figure 2.10: Calibrating the geometric configuration of the projection center and the rotation center.

same measurement is performed for the same angle to the right side (Figure 2.10, Step 3).

Step 3: The stage location, which minimize the sum of the distances between beam spots, is considered to be optimal.

These procedures place the projection center of the camera at the pan axis. The same procedures enable adjustment of the tilt axis.

2.4.2 Calibrating Internal Camera Parameters

In general, both the projection center and the image plane shift along the optical axis while the focal length (i.e., the zoom parameter) is changed. In the FV-PTZ camera system, however, the projection center is fixed. This enables us to consider the variation in the focal length as only a shifting of the image plane.

If the camera captures images while the focal length is changed, the captured images are enlarged or reduced around a specific position in the image. This position is called a *Focus of Expansion* (FOE, in short). The FOE is always placed on the optical axis of the camera, even while the focal length is changed.

In accordance with the above discussion, we define a camera model of the FV-PTZ camera as follows:

Camera model of the FV-PTZ camera: An optical axis is equivalent to a straight line between the projection center and the FOE in the image plane. Depending on the zoom parameter, the position of the FOE shifts along the optical axis, and the position and posture of the image plane changes. The x and y axes of the image

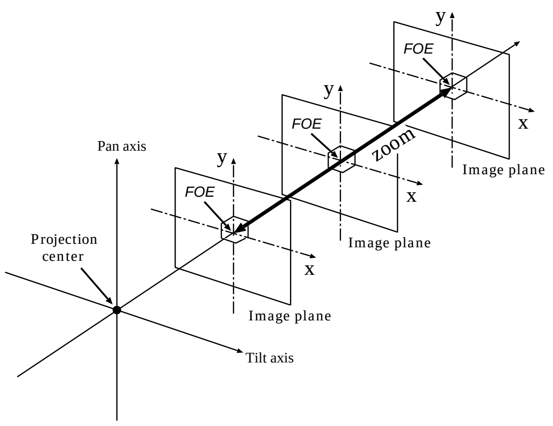


Figure 2.11: Camera model of the FV-PTZ camera.

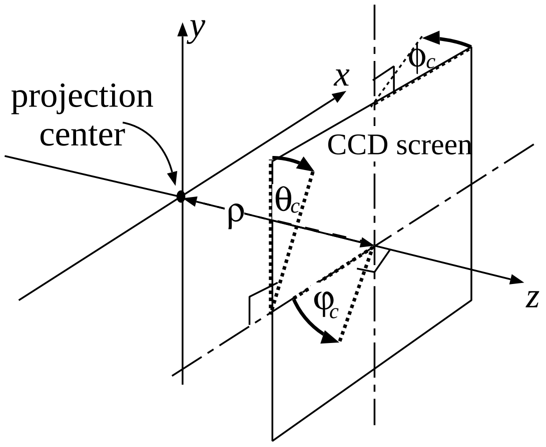


Figure 2.12: Slant angles of the image plane (CCD).

plane are parallel to the tilt and pan axes, respectively. Figure 2.11 illustrates variations in the image planes while the zoom parameter is changed.

Based on this camera model, we define the internal camera parameters required for analyzing images taken by the FV-PTZ camera. We establish a two-phased calibration procedure for estimating the internal camera parameters:

1. Initially, the FOE is estimated.
2. Next, all other parameters (i.e., radial distortion, aspect ratio, and CCD slant angle) are estimated simultaneously.

The FOE is estimated by analyzing images taken with varying zoom parameters. The 2D points, which are projected from the same 3D point onto the image plane while the zoom is changed, are on a straight line in the image plane. This straight line necessarily passes through the FOE. Therefore, we can estimate the FOE as follows:

1. Draw multiple points randomly on a 3D plane that is at a right angle with the optical axis of the camera (Figure 2.13 (a)).
2. Observe these points and capture images while the zoom is changed (Figure 2.13 (b)).
3. Estimate straight lines that are each determined by 2D points projected from the same 3D point (Figure 2.13 (c)).
4. Obtain the intersection point of all straight lines. This point is considered to be the FOE (Figure 2.13 (c)).

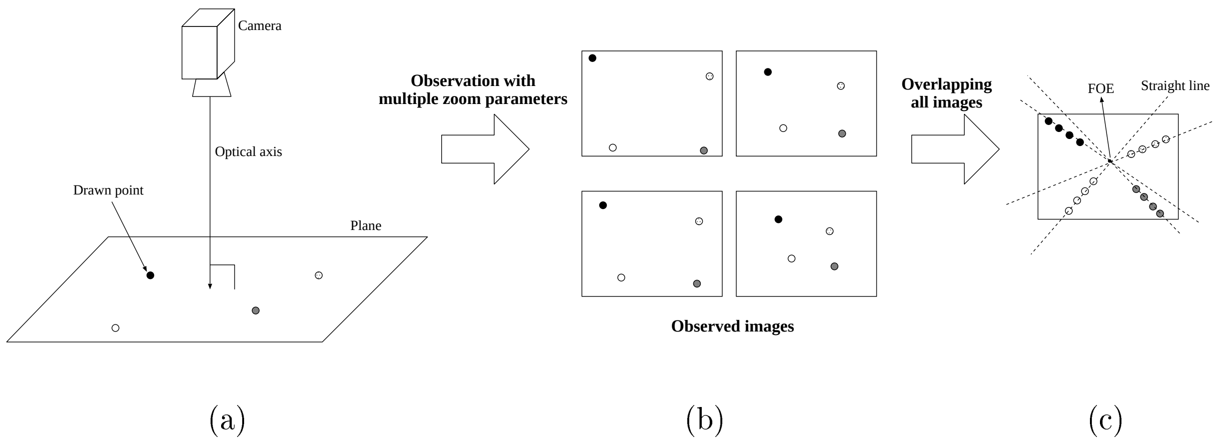


Figure 2.13: Calibration of the FOE.

To realize the FV-PTZ camera, we suppose a projection from a 3D scene to a 2D image to be a perspective projection. In a practical lens system, however, the actual projected image is different from the image generated by the perspective projection due to various distortion factors. Because of these factors, the assumption about the projection is not accurate enough to facilitate image analysis.

In [Tsa86], the following factors were employed to correct the observed image with distortion.

Radial distortion: A projection point (x_u, y_u) on a perpendicular plane to the optical axis will be shifted to (x_{rd}, y_{rd}) by the radial distortion:

$$\begin{aligned} x_{rd} &= (x_u - x_0)(1 + \kappa\Delta^2) + x_0, \\ y_{rd} &= (y_u - y_0)(1 + \kappa\Delta^2) + y_0, \end{aligned} \tag{2.11}$$

where κ represents radial distortion coefficient, and

$$\Delta^2 = (x_u - x_0)^2 + (y_u - y_0)^2 \quad . \tag{2.12}$$

Aspect ratio: The aspect ratio only affects x coordinate values, and the point (x_{sd}, y_{sd}) will be shifted to (x_{ad}, y_{sd}), where

$$x_{ad} = \alpha(x_{sd} - x_c) + x_c \quad . \tag{2.13}$$

To increase the calibration accuracy, we additionally introduce slant angles of the image plane (CCD) as illustrated in Figure 2.12.

CCD slant angle: The point (x_{rd}, y_{rd}) will be shifted to (x_{sd}, y_{sd}) by the CCD rotations $(\theta_c, \phi_c, \varphi_c)$ around the x-axis, y-axis and z-axis.

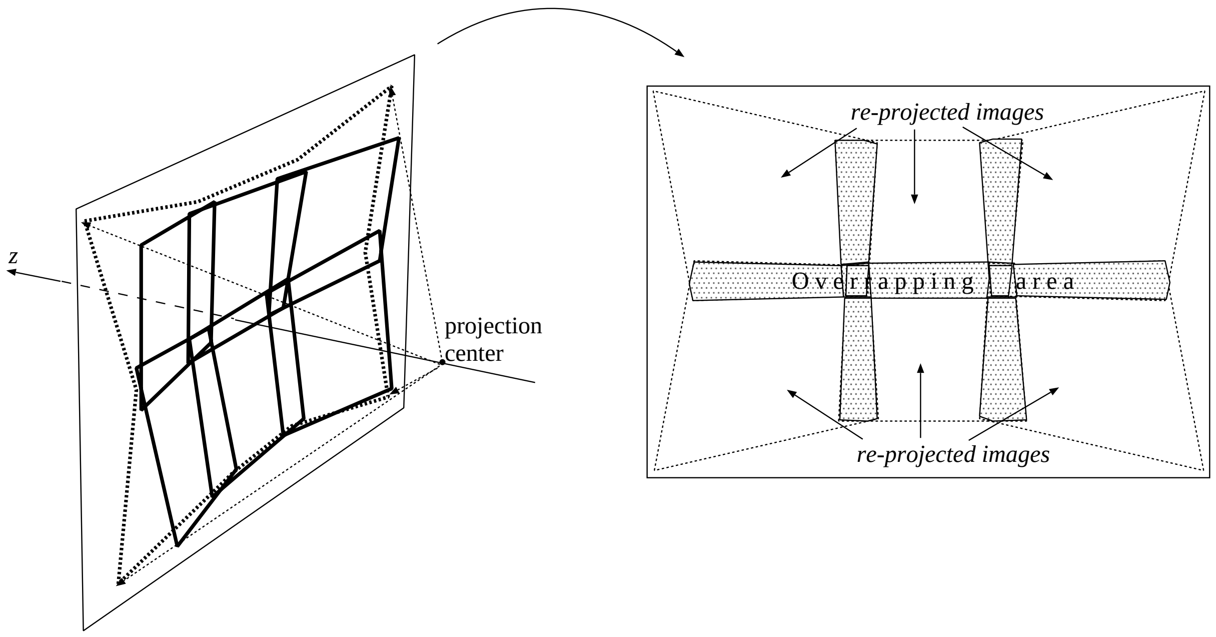


Figure 2.14: Image stitching based calibration with APP.

$$\begin{pmatrix} x_{sd} \\ y_{sd} \end{pmatrix} = \rho \cos\theta_c \cos\phi_c \frac{A \begin{pmatrix} x_{rd} - x_c \\ y_{rd} - y_c \end{pmatrix}}{\boldsymbol{D}_r^T \begin{pmatrix} x_{rd} - x_c \\ y_{rd} - y_c \\ \rho \end{pmatrix}} + \begin{pmatrix} x_c \\ y_c \end{pmatrix} , \tag{2.14}$$

where (x_c, y_c) represents the rotation center, ρ the focal length, and

$$\begin{aligned} A &= \begin{pmatrix} \cos\varphi_c \cos\phi_c - \sin\theta_c \sin\varphi_c \sin\phi_c & -\cos\theta_c \sin\varphi_c \\ \sin\varphi_c \cos\phi_c + \sin\theta_c \cos\varphi_c \sin\phi_c & \cos\theta_c \cos\varphi_c \end{pmatrix}, \\ \boldsymbol{D}_r &= \begin{pmatrix} -\cos\theta_c \sin\phi_c \\ \sin\theta_c \\ \cos\theta_c \cos\phi_c \end{pmatrix}. \end{aligned} \tag{2.15}$$

In order to estimate x_0, y_0, κ, α, θ_c, ϕ_c, and φ_c, we employ the properties of the FV-PTZ camera. If images taken by the perspective projection FV-PTZ camera system are projected onto the same virtual screen (e.g., APS and APP), the projected images are seamless on the virtual screen. Accordingly, if we have correct camera parameters as denoted above (i.e., pan, tilt, focal length, aspect ratio, radial distortion coefficient, radial distortion center, and slant angles of CCD plane), a seamless image stitching can be realized. In other words, these camera parameters can be calibrated in such a way as to achieve seamless image stitching. According to this scheme, camera parameters can be calibrated by minimizing the image difference in the overlapping area on the virtual screen shown in Figure 2.14.

We set the internal camera calibration as follows:

1. We capture a set of partially overlapping images of a stationary scene by changing pan and tilt angles with a fixed zoom parameter.

2. We give appropriate initial parameters.

3. Based on equations (2.11) $\sim$ (2.15), we obtain the corrected images (x_u, y_u) from the observed images (x_{ad}, y_{sd}). We then project the corrected images onto the virtual screen.

4. The image difference in the overlapping area should be minimized. We evaluate the similarity between images by the normalized correlation and maximize it by employing non-linear optimization. Let f and g be the overlapping areas of the different observed images projected onto the virtual screen. If pixels (x_1, y_1) and (x_2, y_2) in f and g, respectively, are projected to the same position in the virtual screen, the normalized correlation of these pixels (denoted by S) is computed as

$$S = \frac{\sum f(x_1, y_1) \cdot g(x_2, y_2)}{\sqrt{\sum f^2(x_1, y_1)}\sqrt{\sum g^2(x_2, y_2)}}. \tag{2.16}$$

   The range of S is $0 \leq S \leq 1$. Let P be the combination number of image pairs that are overlapped in the virtual screen. We calculate all $S_i (i = 1, \cdots, P)$ and obtain the evaluation value for non-linear optimization.

$$D = \left(1 - \frac{\sum_{k=1}^{p} S_k \cdot N_k}{\sum_{k=1}^{p} N_k} \right)^2 , \tag{2.17}$$

   where $N_i (i = 1, \cdots, P)$ denotes the total sum of pixels that overlap with another image in the virtual screen.

5. If the value of equation (2.17) is small enough, optimization is finished. Otherwise, values of all parameters are adjusted, and go back to 2.

The advantages of this calibration method are:

No Calibration Object: This method does not require any specially designed calibration object.

No Human Inspection: Since this method only uses observed images taken by changing view directions, it will not be affected by human mistakes.

No External Camera Parameters: Most camera calibration methods (e.g., [Tsa86]) estimate internal and external camera parameters simultaneously. In these methods, internal and external parameters may interfere with each other, and inconsistent internal parameters may be obtained depending on the calibration object. In our method, however, only the internal camera parameters are obtained.

2.4.3 Pixel-wise Image Calibration

Even if the above two calibrations work out satisfactorily, it is difficult to attain the exact pixel-wise alignment between the observed image and the scene image generated from the APS/APP. Various factors cause this alignment error:

Imprecise calibration: Depending on the precision of the calibration result, the accuracy of the generated image changes. For example, since we estimate the internal camera parameters by the optimization method so that the total error is minimized, the precision the generated image varies at each position in the APS/APP. Furthermore, a divergence of the camera's projection center may occur during extended utilization.

Quantization error: In practical camera systems, the quantization error in the observed image is caused by the A-D conversion done for image capturing. This produces the difference between each observed image and the generated image.

Mechanical limitation: In general, the camera angle and the zoom are also included in the internal camera parameters that should be estimated for active imaging with the FV-PTZ camera.

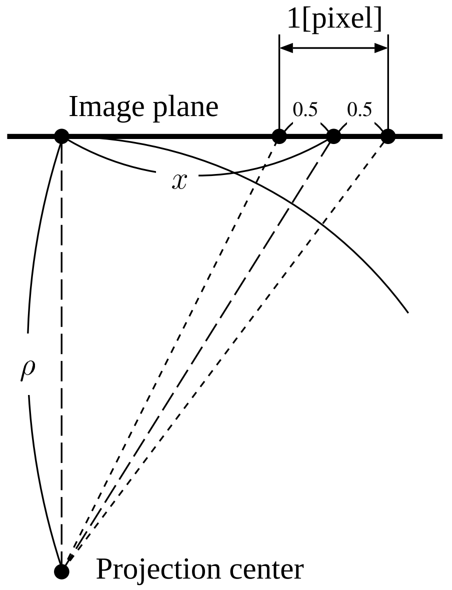
Camera angle: The camera angle is represented by the pan (θ) and tilt (φ) angles. In equation (2.1), the view direction is expressed by a unit vector $\boldsymbol{D}$.

Zoom: The zoom of the camera is represented by the focal length ρ.

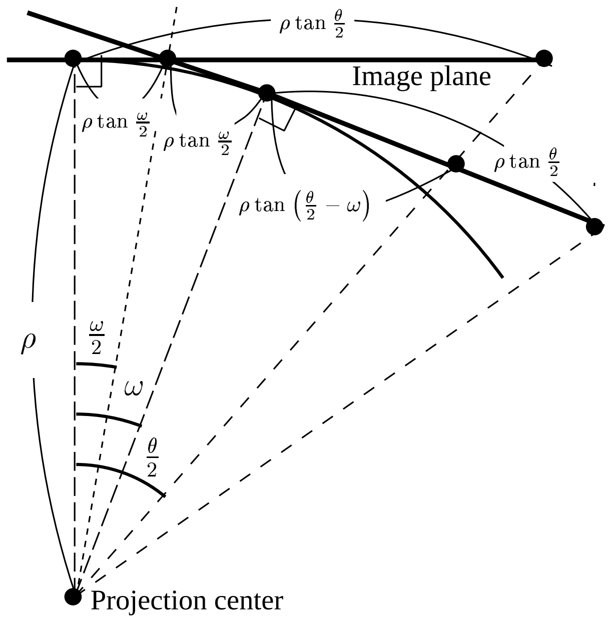
We control pan, tilt and zoom parameters to observe the wide area and acquire the required object information. Since we employ a computer-controlled active camera, we can obtain current values of these parameters during observation. The accuracies of the obtained parameters depend on the mechanical characteristics of the camera (i.e., the resolution of the rotational angle and the zooming factors). If its performance is not high enough to generate an image that is identical to an observed image, the result of image analysis becomes unreliable.

Active control: In particular, if the camera captures the image while changing pan-tilt-zoom parameters with smooth (non-stop) camera motion, it is difficult to align the observed image exactly with the image generated from the APS/APP.

To obtain a reliable result of image analysis, an accurate rotational angle is especially required. This is because the distance from the projection center to the image plane is much longer than the pixel size in most cases, and hence a small angular error produces



(a) Angular resolution


$\rho\tan\frac{\theta}{2}$
Image plane
$\rho\tan\frac{\omega}{2}$
$\rho\tan\frac{\omega}{2}$
$\rho\tan\frac{\theta}{2}$
$\rho\tan\left(\frac{\theta}{2}-\omega\right)$
ρ
$\frac{\omega}{2}$
ω
$\frac{\theta}{2}$
Projection center

(b) Discrepancy of the pixel and the rotational angle

Figure 2.15: Consideration of the angular accuracy.

a crucial geometric error in generating images from the APS/APP. To obtain the actual (accurate) rotational angle, therefore, we examine the difference between the observed image and images generated with different combinations of pan-tilt parameters. Let image$_o$ whose rotational angle is (θ_o, φ_o) have the minimum error with the observed image. (θ_o, φ_o) is then considered to be the current actual rotational angle.

Although an accurate angle can be found by minimizing the error, this procedure is computationally expensive. This is because all synthesized images compared with the observed image are generated from the APS/APP. Fortunately, since a small angle of the rotation can be approximated by the translation, it is sufficient to find the optimal translation minimizing the error.

The angle resolution $\delta(x)$ corresponding to the pixel resolution at x on the plane is represented as:

$$\delta(x) = \left|\tan^{-1}\left(\frac{x+0.5}{\rho}\right) - \tan^{-1}\left(\frac{x-0.5}{\rho}\right)\right|, \tag{2.18}$$

where ρ represents the distance from the projection center to the plane (Figure 2.15 (a)). The angular error ω that produces 1-pixel image distortion satisfies the equation

$$\rho \times \left| tan\left(\frac{\theta}{2}\right) - \tan\left(\frac{\theta}{2} - \omega\right) - 2\tan\left(\frac{\omega}{2}\right)\right| = 1, \tag{2.19}$$

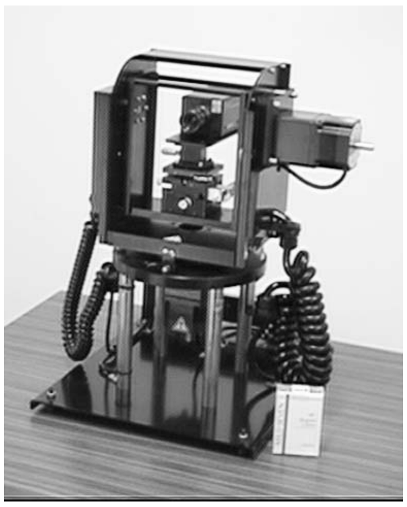


Figure 2.16: Developed FV-PT camera head.

where θ represents the view angle of the camera (Figure 2.15, (b)). Accordingly, if the error of the rotational angle is less than ω, we can approximate the angular error by the translation of the image without any errors.

2.5 Experiments

We actually developed two types of fixed-viewpoint cameras (FV-PT camera and FV-PTZ camera). In this section, we demonstrate (1) the accuracy of our calibration method for implementing the proper setup of a fixed-viewpoint camera and (2) the effectiveness of the image generation method from the scene model (i.e., APS and APP).

2.5.1 Developing the FV-PT Camera

2.5.1.1 Hardware Specification

We experimented to verify the effectiveness of our idea by using a pan-tilt rotation camera head. Figure 2.16 shows the FV-PT camera head we developed, where the pan and tilt axes intersect at a right angle and a video camera is mounted on a group of adjustable slide and slant stages. Table 2.1 shows the mechanical specifications of the camera head. Rotational angles of the camera head can be controlled via RS-232C by a computer.

We mounted a SONY camera XC-003 with a C-mount lens VCL-08WM on this camera head. By using this camera head, any (compact) video camera with any lens system can be calibrated to realize a fixed-viewpoint camera. In this experiment, the focal length was fixed. With these resources, we developed an FV-PT camera.

Table 2.1: Specification of the FV-PT camera head.

Pan rotation	$-180^\circ \sim 180^\circ$
Tilt rotation	$0^\circ \sim 45^\circ$
Maximum rotational velocity	100°/sec
Maximum rotational acceleration	$100^\circ/\text{sec}^2$
Angular resolution	0.012°
Backlash	less than 0.167°

2.5.1.2 Fixing the Viewpoint

For the calibration to realize the FV-PT camera system, we employ the calibration method described in Section 2.4.1. All of the observed images were projected onto a single screen (i.e., APP) to evaluate the image differences in the overlapping areas of different images.

Figure 2.18 shows the calibration result. We turned the pan and tilt angles by 6° and 10° in either direction (i.e., pan: left and right directions, tilt: up and down directions), respectively. The horizontal and vertical axes represent the linear stage location and the distance between the projected bright spots, respectively. From the calibration result, the optimal location of the slide stage for realizing the FV-PT camera was determined to be 5.45 [cm] for pan axis and 1.55 [cm] for tilt axis.

2.5.1.3 Calibrating Internal Camera Parameters

We captured 18 images by panning and tilting the camera. Figure 2.19 shows the observed images. The size of each image is 320×240 [pixel]. Pixel values in the observed images are darkened due to the vignetting distortion[AAB97]. We can cancel the vignetting distortion in the observed image by the following procedures:

1. Capture a white plane illuminated uniformly. The vignetting distortion appears in the captured image.

2. Compute rates of pixel values between the brightest pixel and all other pixels. Each rate shows the relative extent of the vignetting distortion for each pixel.

3. Correct the vignetting distortion based on the analysis achieved in 2.

In this experiment, although the focal length was known as a physical length, we need the focal length as a pixel length to integrate all of the observed images and generate a scene image model. This is because 2D coordinates in the observed image are represented by pixels, and so the unit length is also represented by pixels in equations (2.1) $\sim$ (2.15).

We estimated the internal camera parameters (i.e., focal length ρ, radial distortion κ, x_0, y_0, aspect ratio α, and slant angle of CCD θ_c, ϕ_c, φ_c) by the calibration method described in Section 2.4.2. In this calibration, all of the observed images were re-projected

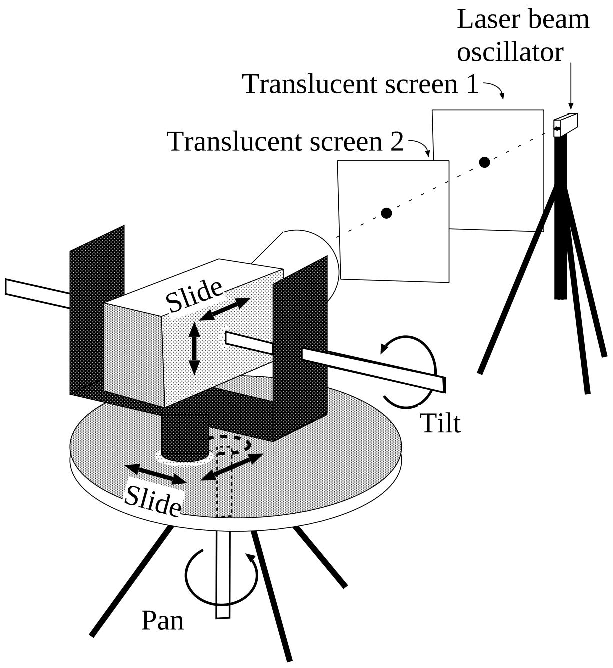
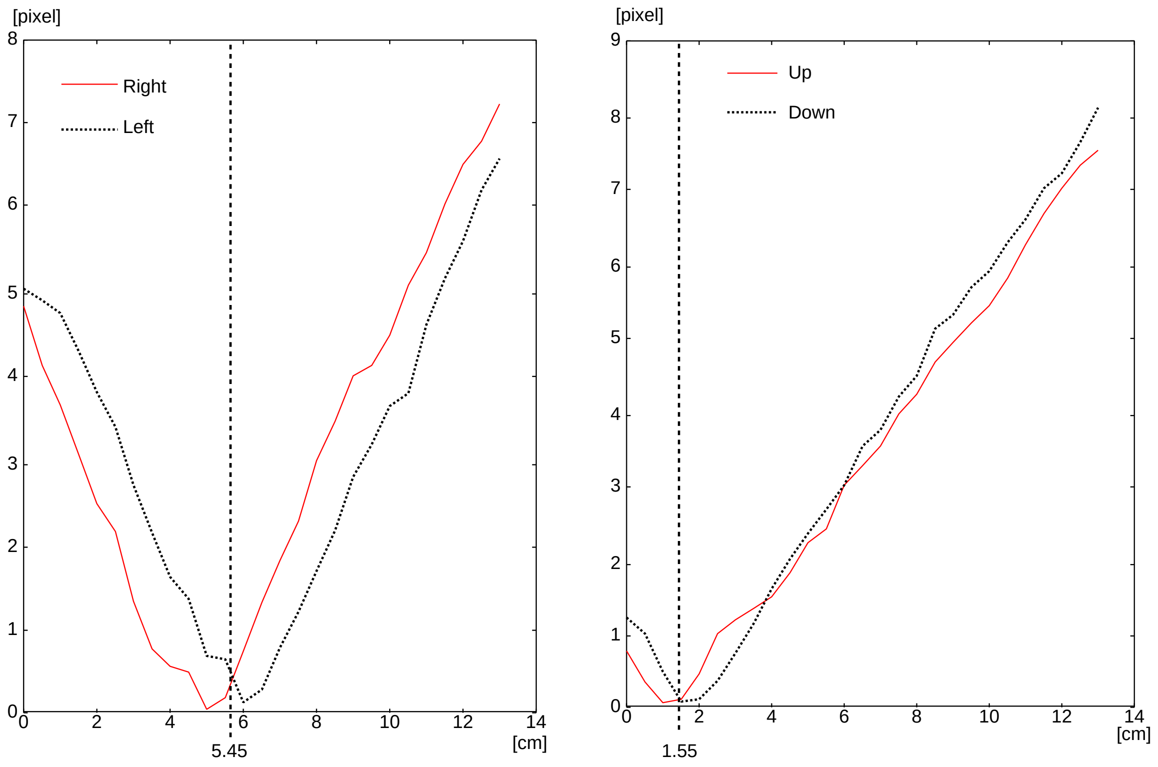


Figure 2.17: Experimental environment.



(a) Pan rotation (b) Tilt rotation

Figure 2.18: Distance between two bright spots on translucent screens.

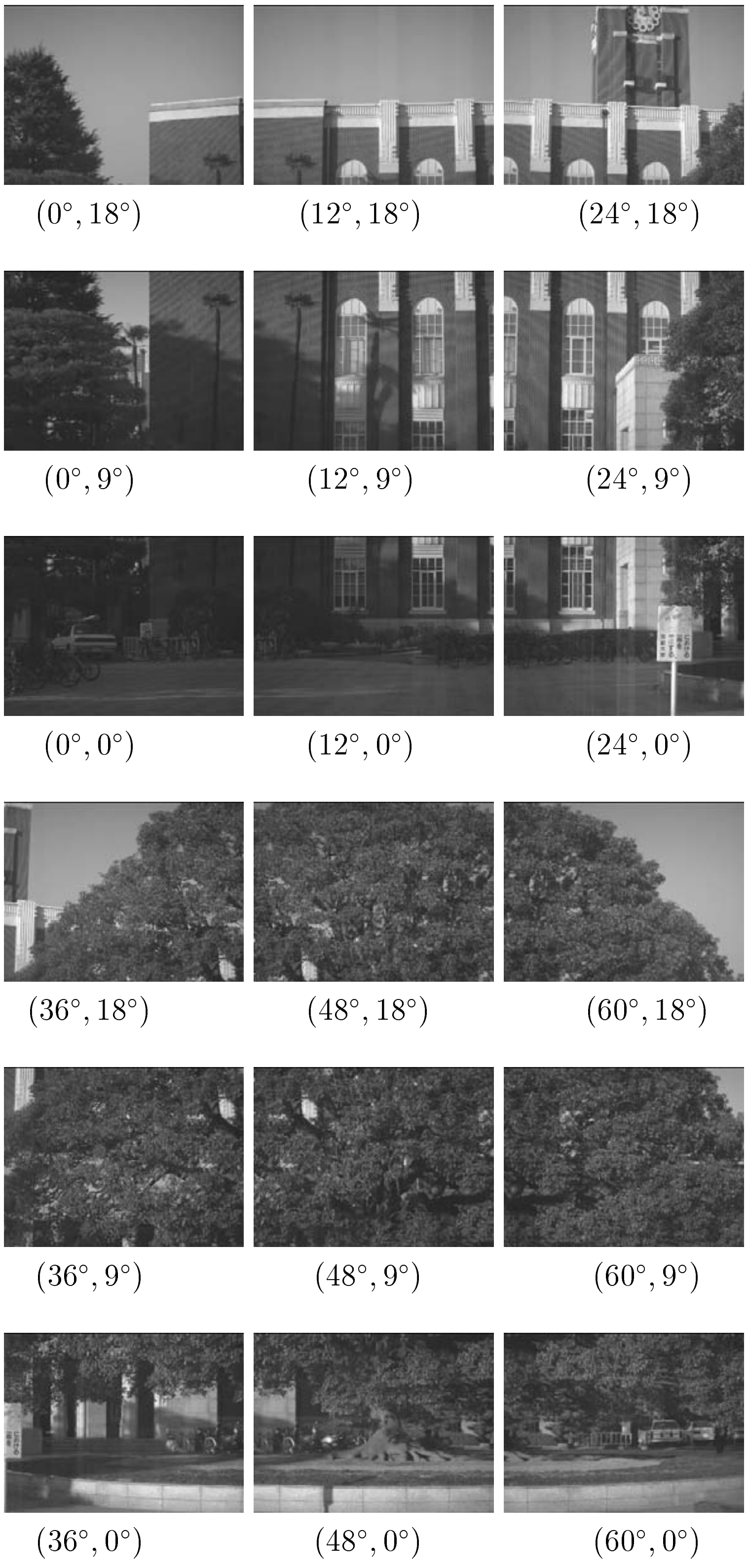


Figure 2.19: Observed images for calibrating the internal camera parameters.

onto a single planar screen (i.e., APP). During optimization of the parameters, the rotational angle of the APP screen is fixed $(0°, 0°)$. The distance from the projection center to the APP screen, on the other hand, is dynamically changed so that it is equal to the optimizing focal length of the observed image. Following are the given initial parameters:

- **Initial parameters:**

  Focal length ρ: 1090.0 [pixel]

  Radial distortion coefficient κ: 0.0[($\times 10^{-8}$) pixel^{-2}]

  Radial distortion center (x_0, y_0): (159.5 [pixel], 119.5 [pixel])

  Aspect ratio α: 1.0

  Slant angle of CCD $(\theta_c, \phi_c, \varphi_c)$: (0.0°, 0.0°, 0.0°)

The initial focal length (1090.0 [pixel]) is determined by the following procedures:

1. Determine a landmark in the observed scene.
2. Rotate the pan angle of the camera so that the landmark is projected in the right edge of the image. Let this angle be α.
3. Similarly, rotate the pan angle so that the landmark is projected in the left edge of the image. Let this angle be β.
4. The geometric configurations between the above two image planes (as illustrated in Figure 2.20) are represented by

$$\theta + \theta' = \beta - \alpha \ , \tag{2.20}$$

$$\tan\theta = \frac{S_i - C}{fl}, \tag{2.21}$$

$$\tan\theta' = \frac{C}{fl} \ , \tag{2.22}$$

   where S_i and C denote the image size and the coordinates of the FOE, respectively. We assume $C = S_i/2\,(= 320/2)$ and solve the above equations.

Following are the estimated optimal parameters:

- **Optimal parameters:**

  Focal length ρ: 1086.86 [pixel]

  Radial distortion coefficient κ: -1.24[($\times 10^{-8}$) pixel^{-2}]

  Radial distortion center (x_0, y_0): (158.55 [pixel], 118.67 [pixel])

  Aspect ratio α: 0.994

  Slant angle of CCD $(\theta_c, \phi_c, \varphi_c)$: (0.109°, 0.163°, 0.014°)

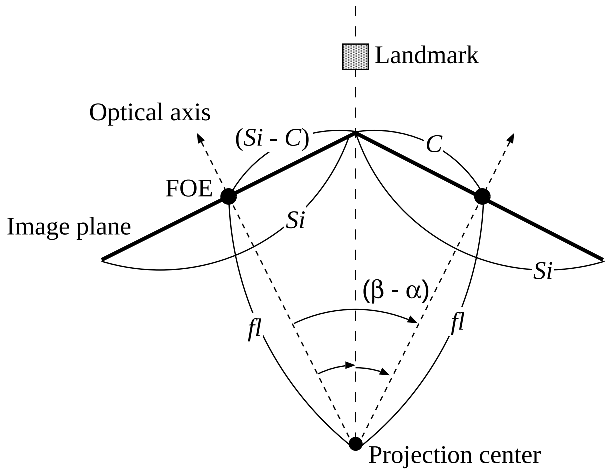


Figure 2.20: Determining the initial focal length.

After all of the observed images were corrected based on the above estimated parameters, we generated an APS (Figure 2.21). This APS image consists of 180 observed images. By employing it as a scene model, we can generate a scene image taken with arbitrary combinations of pan, tilt and zoom parameters[3] . Figure 2.22 shows the panoramic image generated by re-projecting all of the observed images onto an image plane.

In this experiment, from equations (2.18) and (2.19) and the estimated focal length ($\rho = 1086.86$), we can obtain the following values: $\delta(0) = 0.0527°$ at the image center, $\delta(256) = 0.0516°$ at the image frame, and $\omega = \pm 0.38°$ which corresponds to ± 5.574-pixel translation. That is, if the rotational stages have a small angle step of less than 0.06°, it is not required to correct the view direction. Furthermore, if the angular error is less than 0.38°, we can obtain the optimal scene image by translating a generated image.

2.5.1.4 Performance Evaluation

To verify the accuracy of camera calibrations and the generated APS, we compare the observed image with the image generated from the APS.

Figure 2.23 (a) shows an observed image at $(3°, 15°)$, (b) is the generated image at the same pan-tilt angles, and (c) is the difference between (a) and (b). This result shows that the generated image is approximately identical to the observed image.

To obtain the more accurate detected result, several shifted versions of the observed image are generated and their differences from the background image were computed. When the observed image was shifted to $(-1[pixel], -1[pixel])$, the least overall gray level difference was obtained. Figure 2.23 (d) shows the image with the least difference. As we can see, the shifting of the observed image is adequate to match it with the generated

[3] In this experiment, the rotational tilt angle of the camera head was limited. Therefore, we could not generate a fully spherical virtual screen.

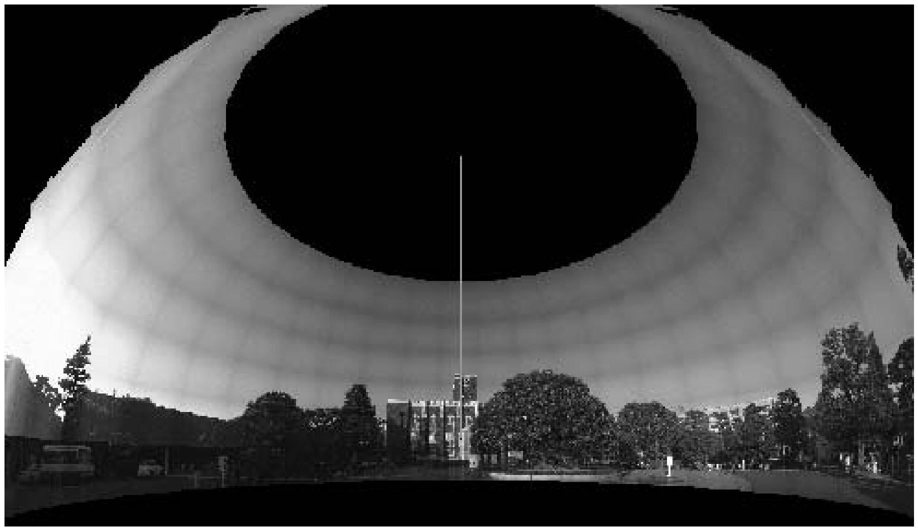


Figure 2.21: APS representation of Kyoto University Clock Tower scene (Dome-shaped APS).

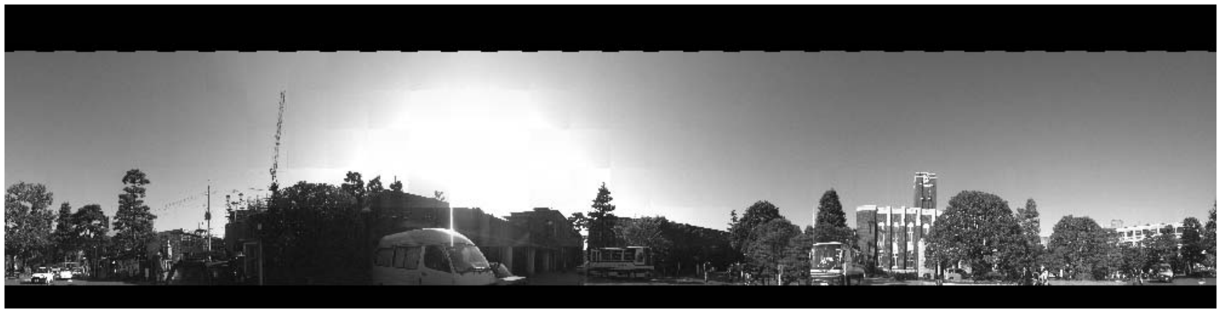


Figure 2.22: Panoramic representation of the APS image.

image when the view directions of these image are roughly the same.

2.5.2 Developing the FV-PTZ Camera

2.5.2.1 Hardware Specification

Here, we show experimental results for internal parameter calibration and image generation by using an off-the-shelf active video camera, SONY EVI-G20, which we found to be a good approximation of an FV-PTZ camera. Figure 2.24 shows the appearance of EVI-G20. Table 2.1 shows the mechanical specifications of EVI-G20. The rotational angles and zooming factor of EVI-G20 can be controlled via RS-232C by a computer. Pan-tilt rotations are designed as a gimbal mechanism (Figure 2.25).

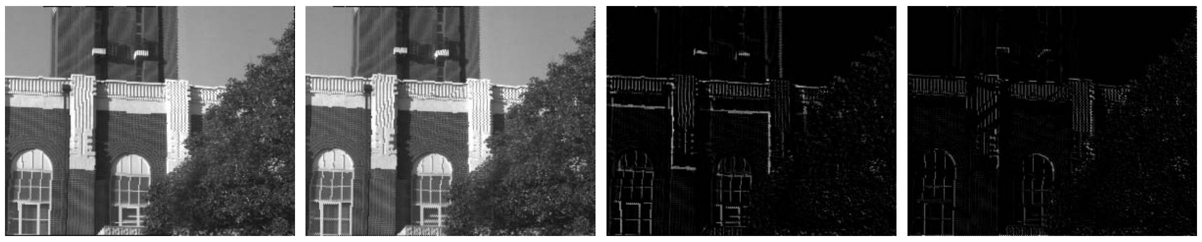

(a) Observed image $(3°, 15°)$ (b) Generated image $(3°, 15°)$ (c) Difference $(|(a) - (b)|)$ (d) Precise difference

Figure 2.23: Comparison between the observed image and the generated image

Table 2.2: Specification of SONY EVI-G20.

Pan rotation	$-30° \sim 30°$
Tilt rotation	$-15° \sim 15°$
Horizontal view angle	$15° \sim 44°$
Maximum rotational velocity	$245°$/sec
Maximum zooming (changing view angle) velocity	$4°$/sec
Angular resolution	$0.00193°$

The following descriptions give the differences between the experiments in this section and those in Section 2.5.1.

Bounded (narrow) horizontal view angle: The pan-rotation and horizontal-view angles of EVI-G20 are $-30° \leq$ pan rotation $\leq 30°$ and $15° \leq$ horizontal view $\leq 44°$, respectively. The total horizontal and vertical view angle of the camera are $104°$ and $59°$ ($< 180°$), respectively. This allows us to represent the entire scene model as a single APP.

Variable zoom: Since the zoom parameter of EVI-G20 can be controlled from a computer, we can obtain a high-resolution APP by controlling the zoom during observation.

With EVI-G20, we can realize a high-resolution APP.

2.5.2.2 Viewpoint Calibration

To utilize EVI-G20 as an FV-PTZ camera, we must first confirm that it is a good approximation of the FV-PTZ camera. We experimented to verify the geometric configuration of the projection center by the laser-beam-based calibration described in Section 2.4.1.

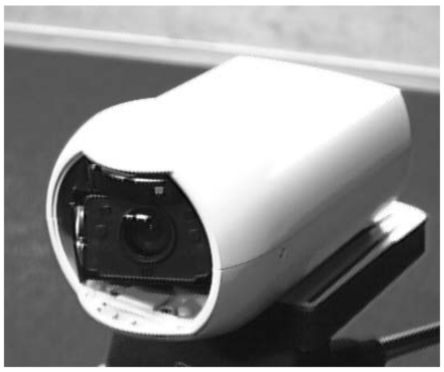


Figure 2.24: Off-the-shelf FV-PTZ camera: SONY EVI-G20.

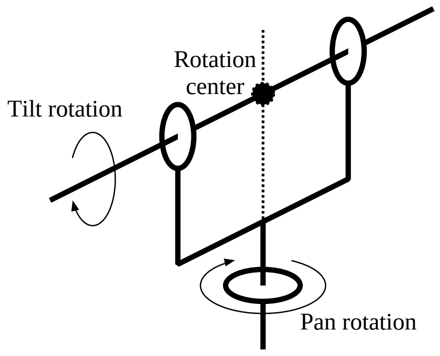


Figure 2.25: Gimbal mechanism.

The experimental results showed that the projection center is about 1.1 [cm] off the rotation center along the optical axis when the zooming factor is set smallest, and that as the zooming becomes large, the former comes closer to the latter. This displacement, however, does not cause serious problems in detecting anomalous regions in a wide area; image deformation is kept to less than 2 pixels when the observed scene is farther than 2.5 [m].

The above verification allows us to model EVI-G20 as an FV-PTZ camera.

2.5.2.3 Calibrating Internal Camera Parameters

Since we control the zoom parameter in this experiment, the position of the FOE is needed to generate the APP by integrating observed images taken with multiple zoom parameters. We estimated the FOE of EVI-G20 by the calibration method described in Section 2.4.2.

Figure 2.26 shows the estimated result of the FOE. From this figure, it is confirmed that 2D positions projected from a specific 3D point are in a straight line and that all of the straight lines cross each other at a point. From this result, we considered the position of the FOE to be $(322.825, 228.215)$.

Next, in order to estimate other internal camera parameters, we observed six images (Figure 2.27). The size of each image is 640×480 [pixel]. These images were taken at the following combinations of the pan and tilt parameters: $(-30°, 10°)$, $(0°, 10°)$, $(30°, 10°)$, $(-30°, -10°)$, $(0°, -10°)$ and $(30°, -10°)$. The focal length was fixed so that the view angle was the widest one. The vignetting distortions of all observed images were corrected by the procedure proposed in Section 2.5.1.3.

We re-projected the images onto an APP and performed the calibration method described in Section 2.4.2. In this experiment, we used the following initial values for internal camera parameters:

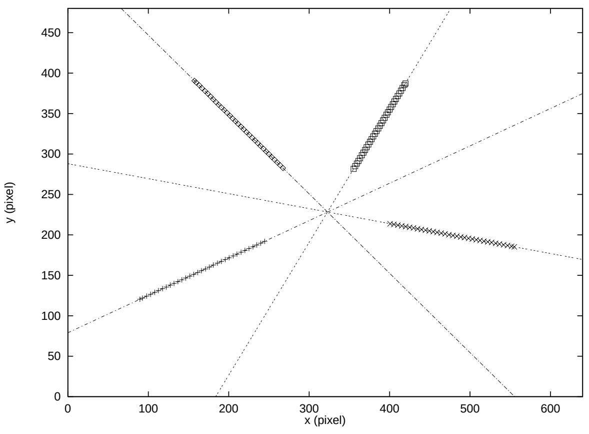


Figure 2.26: Estimated result of the FOE.

Focal length ρ: 800 $\sim$ 910 [pixel] (every 10 pixels)

Radial distortion coefficient κ: 0 [pixel^{-2}]

Radial distortion center (x_0, y_0): (319.5 [pixel], 219.5 [pixel])

Aspect ratio α: 1

Slant angle of CCD $(\theta_c, \phi_c, \varphi_c)$: (0°, 0°, 0°)

During optimization, the rotational angle of the APP screen is fixed at $(0°, 0°)$. The distance from the projection center to the APP screen, on the other hand, is dynamically changed to synchronize it with the optimizing focal length of the observed image. The optimization results are shown in Table 2.3.

The smallest evaluation value was obtained when the initial value of the focal length was 880 [pixel]. We considered the following results estimated from this initial value to be the optimal parameters for images taken with the widest view angle:

Focal length ρ: 830.746 [pixel]

Radial distortion coefficient κ: -10.26050 [$(\times 10^{-8})$ pixel^{-2}]

Radial distortion center (x_0, y_0): (295.594 [pixel], 222.853 [pixel])

Aspect ratio α: 0.99926

Slant angle of CCD $(\theta_c, \phi_c, \varphi_c)$: (0.16900°, -0.12219°, 0.20683°)

Figures 2.28 and 2.29 show the generated APPs and the subtraction images in the overlapping areas. The images in Figures 2.28 and 2.29 were obtained with the initial and

Table 2.3: Estimated results of the internal camera parameters (with the widest view angle).

	Initial value of ρ [pixel]					
	800.000	810.000	820.000	830.000	840.000	850.000
	Estimated parameters					
ρ [pixel]	824.167	824.628	822.651	825.712	827.856	827.189
κ [$(\times 10^{-8})$pixel^{-2}]	1.28871	–0.91411	–0.10969	–1.34124	–4.70199	–4.58066
x_0 [pixel]	317.832	318.823	319.529	319.335	330.391	318.939
y_0 [pixel]	233.228	238.544	239.475	239.982	242.843	238.924
φ_c [degree]	0.17505	0.21363	0.18359	0.19752	0.21267	0.21842
θ_c [degree]	–0.13269	–0.03613	0.06012	–0.02292	0.03743	–0.02453
ϕ_c [degree]	–0.08613	0.05345	–0.07940	–0.13657	–0.06581	0.03032
α	0.99847	0.99927	1.00138	0.99815	0.99845	0.99905
Value	0.000878	0.000685	0.000808	0.000677	0.000447	0.000447

	Initial value of ρ [pixel]					
	860.000	870.000	880.000	890.000	900.000	910.000
	Estimated parameters					
ρ [pixel]	828.236	829.776	830.746	869.297	885.749	902.773
κ [$(\times 10^{-8})$pixel^{-2}]	–5.75362	–8.94146	–10.26050	–4.16316	–2.94475	–1.90708
x_0 [pixel]	320.335	327.315	295.594	319.540	312.649	321.092
y_0 [pixel]	248.540	236.692	222.853	245.173	242.828	235.821
φ_c [degree]	0.21356	0.21781	0.20683	0.26423	0.22791	0.21102
θ_c [degree]	–0.01601	–0.06926	0.16900	0.02403	–0.07008	–0.13933
ϕ_c [degree]	0.06843	0.12875	–0.12219	–0.16378	0.11192	–0.94443
α	0.99876	0.99921	0.99926	0.95158	0.93295	0.91564
Value	0.000377	0.000252	0.000232	0.009550	0.011132	0.013109

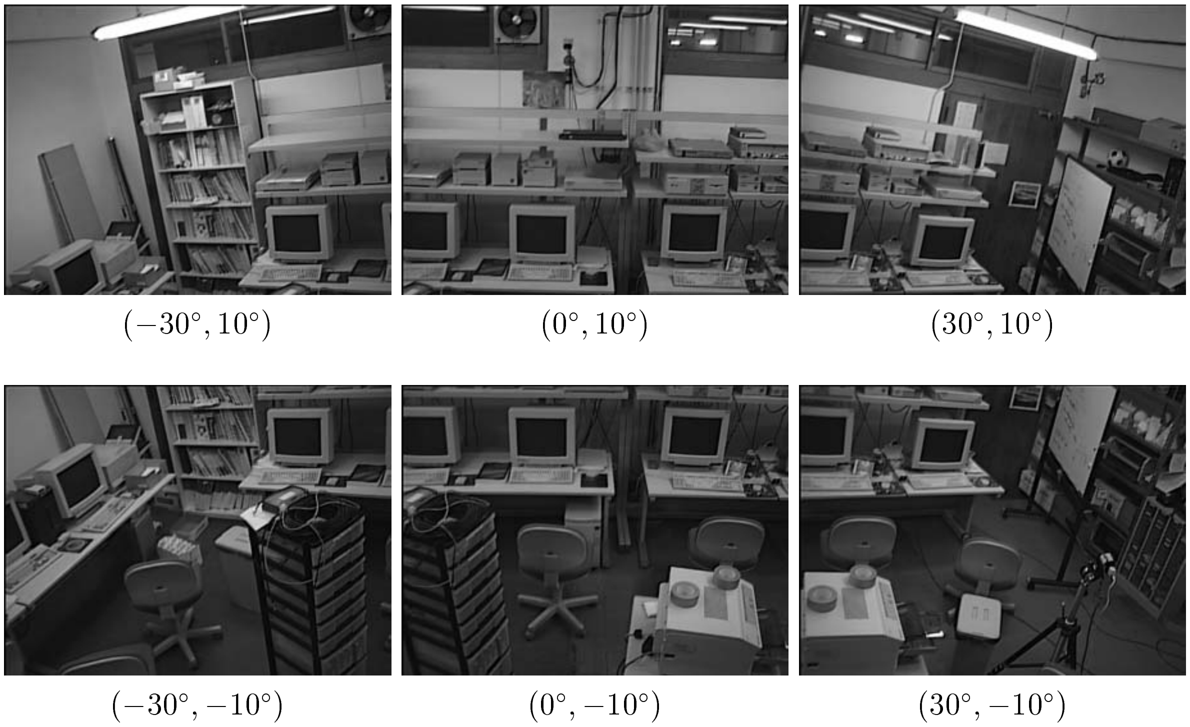


Figure 2.27: Observed images (with the widest view angle: $\rho = 830.746$).

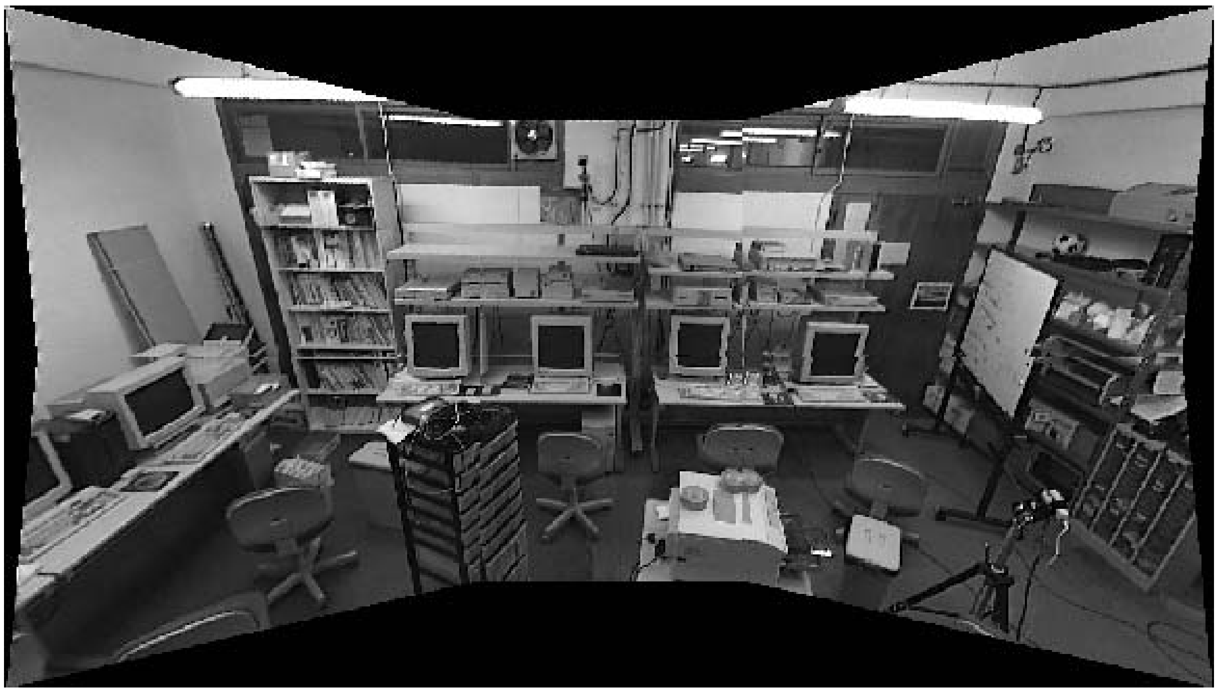
optimal parameters, respectively. Since the APP generated with the optimal parameters is seamless, the estimated parameters are confirmed as appropriate.

Similarly, as with the above procedure, we estimated the internal camera parameters for other zoom parameters. To control the zoom parameter of EVI-G20, we assign a value, which we call the zoom command parameter, to EVI-G20. The domain of the value is from 0 (the widest view angle) to 16384 (the narrowest view angle). The relationship between this value and the focal length, however, is not known. To control the view angle of the camera based on the image analysis (e.g., the region size of the detected object), the relationship between the physical value (i.e., the focal length) and the zoom parameter must be determined.

Table 2.4 shows the estimated optimal parameters for each zoom command parameter. We deduce the following properties of the internal camera parameters from this result:

- The focal length ρ changes linearly in proportion to the zoom command parameter as illustrated in Figure 2.30.
- The radial distortion coefficient κ becomes smaller[4] as the view angle narrows.

[4] If the radial distortion coefficient is less than 10^{-9}, the effect of the radial distortion can be disregarded.



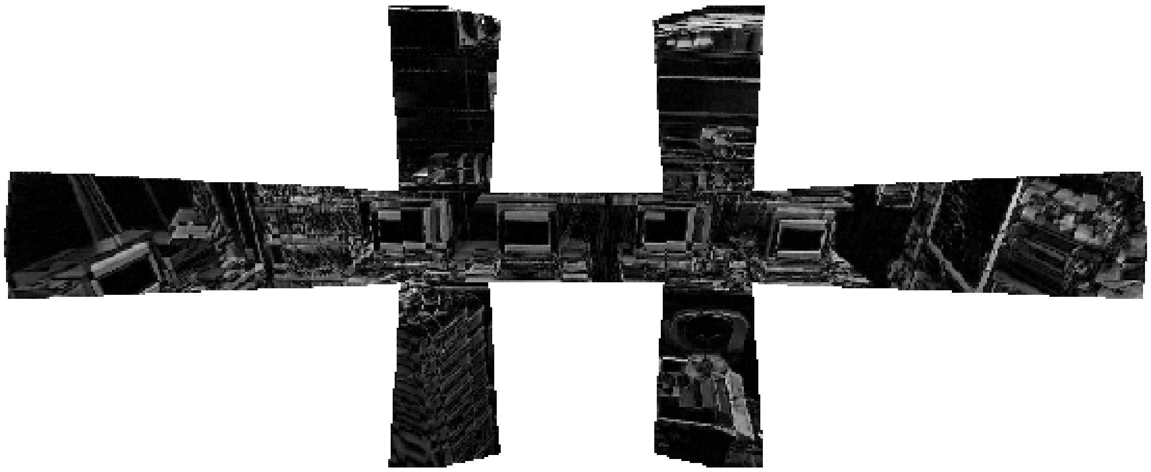
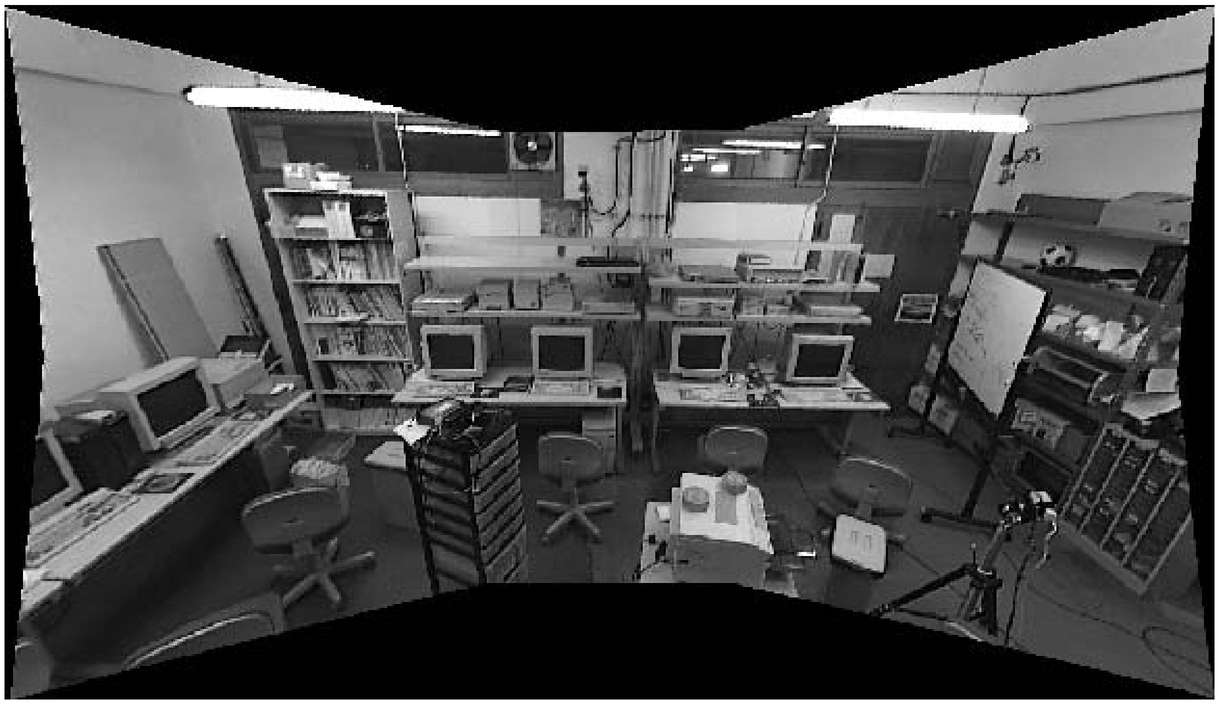


Figure 2.28: Upper: APP image generated with the initial parameters (ρ = 880 [pixel], image size = 1036 $\times$ 585 [pixel]). Six observed images taken with the widest view angle are re-projected. Lower: Gray level difference in overlapping area.



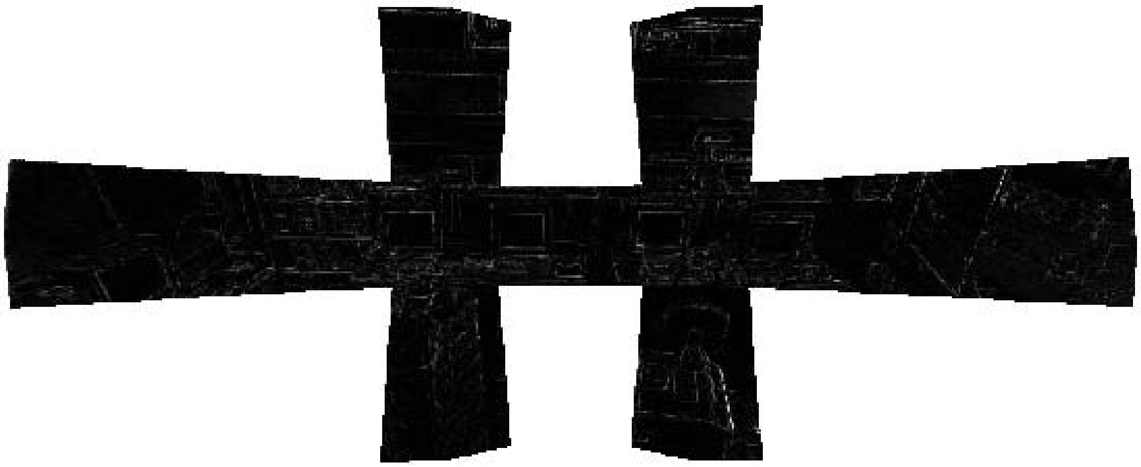


Figure 2.29: Upper: APP image generated with optimal parameters (ρ = 830.746 [pixel], image size = 1036 $\times$ 595 [pixel]). Six observed images taken with the widest view angle are re-projected. Lower: Gray level difference in overlapping area.

Table 2.4: Estimated results of the internal camera parameters (with nine kinds of zoom command parameters).

Zoom command parameter

	0	2048	4096	6144	8192
Initial value of ρ [pixel]					
	880.000	1060.000	1240.000	1450.000	1680.000
Estimated parameters					
ρ [pixel]	830.746	1034.436	1240.532	1450.434	1674.187
κ [$(\times 10^{-8})$pixel^{-2}]	-10.26050	-5.01263	-0.05896	-0.04233	-1.80669
x_0 [pixel]	295.594	329.148	319.499	319.472	319.977
y_0 [pixel]	222.853	245.286	239.499	239.513	241.053
φ_c [degree]	0.20683	0.20056	0.17184	0.20055	0.21095
θ_c [degree]	0.16900	-0.03004	0.09698	0.01854	-0.10951
ϕ_c [degree]	-0.12219	-0.00005	-0.10632	-0.03674	-0.26838
α	0.99926	0.99872	1.00059	1.00079	0.99563
Value	0.000232	0.000210	0.000145	0.000077	0.000035

Zoom command parameter

	10240	12288	14336	16384
Initial value of ρ [pixel]				
	1880.000	2080.000	2290.000	2500.000
Estimated parameters				
ρ [pixel]	1879.794	2080.520	2290.162	2500.622
κ [$(\times 10^{-8})$pixel^{-2}]	-0.08400	0.17395	0.13470	0.26613
x_0 [pixel]	319.490	319.497	319.519	319.535
y_0 [pixel]	239.510	239.517	239.513	239.570
φ_c [degree]	0.17008	0.17086	0.17986	0.16592
θ_c [degree]	-0.06010	-0.06500	-0.09137	-0.17498
ϕ_c [degree]	-0.14444	0.01277	0.15575	0.17341
α	0.99668	1.00092	1.00102	1.00280
Value	0.000034	0.000023	0.000017	0.000011

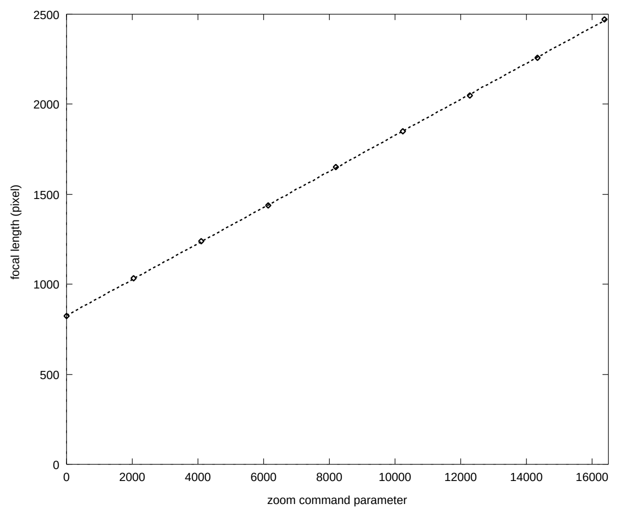


Figure 2.30: Relationship between the zoom command parameter and the focal length.

- The estimated result of the radial distortion center (x_0, y_0) becomes inaccurate with the decline in the view angle.
- Other parameters (i.e., θ_c, ϕ_c, φ_c and α) are nearly constant.

Based on the above properties, we define the variations in internal camera parameters while the zoom command parameter changes:

Focal length ρ: This changes linearly in proportion to the zoom command parameter as illustrated in Figure 2.30.

Radial distortion coefficient κ: The value estimated with the widest view angle is considered to be the optimal value, and this value changes linearly in proportion to the zoom command parameter.

Radial distortion center (x_0, y_0): The value estimated with the widest view angle is considered to be the optimal value, and the value is constant irrespective of zooming.

Other parameters θ_c, ϕ_c, φ_c and α: These parameters are constant irrespective of zooming. The average of the estimated values for all zoom command parameters is considered to be the optimal value.

By employing the above analysis of the internal camera parameters, we can generate an APP that is consistent with all zoom parameters.

Figure 2.31 shows the APP that is generated by integrating the observed images taken with nine kinds of zoom command parameters (i.e., 0, 2048, 4096, 6144, 8192, 10240, 12288, 14336 and 16384). Since the generated APP image is seamless, it is confirmed

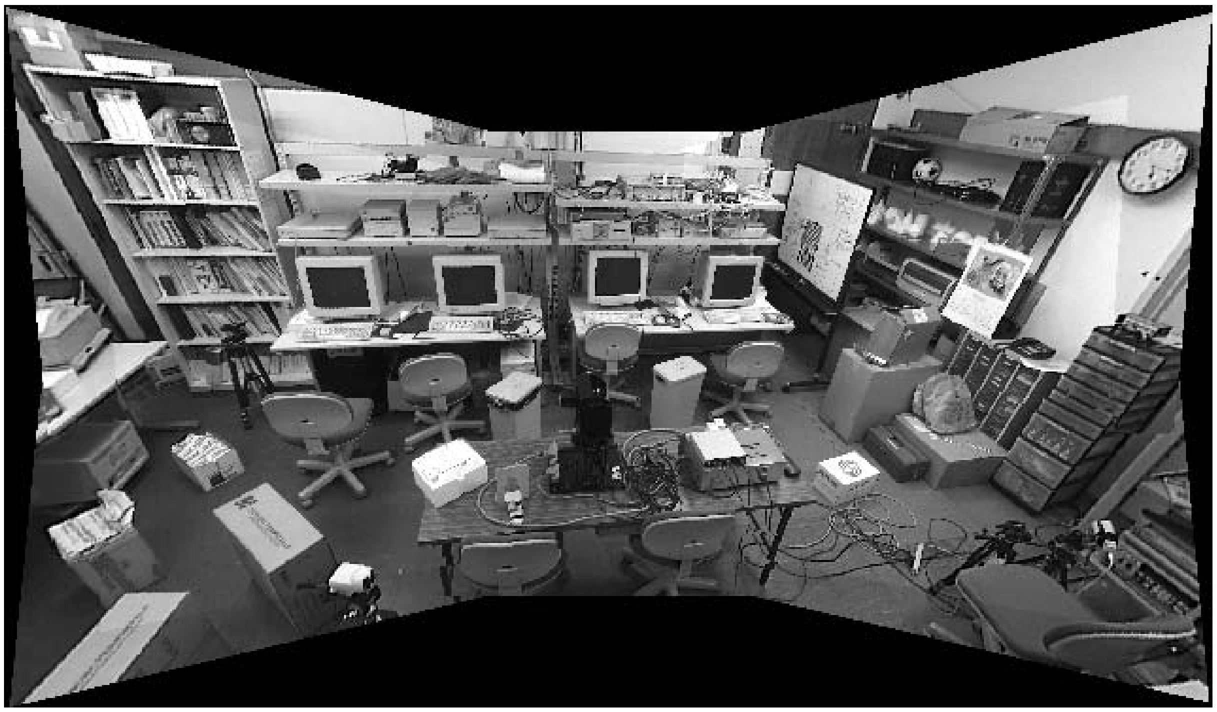


Figure 2.31: High resolution APP ($\rho = 2500.622$ [pixel], image size = 6332×3684 [pixel]): the images taken with various zoom parameters are re-projected onto an APP.

that the internal camera parameters of EVI-G20 are efficiently integrated among all zoom parameters. This APP is synthesized by re-projecting the images taken with the narrow view angle, namely when the camera zooms in. The high-resolution APP is, therefore, obtained.

2.5.2.4 Performance Evaluation

To verify the accuracy of the obtained camera model and the generated APP, we compared the observed image with the image generated from the APP.

These experiments were conducted with the following camera parameters:

Figure 2.32: Pan, tilt and zoom command parameters were $-20.0°$, $4.0°$ and 4096, respectively.

Figure 2.33: Pan, tilt and zoom command parameters were $-12.0°$, $6.0°$ and 12288, respectively.

Figures 2.32 and 2.33 show the observed images, generated images, and subtraction image. From these results, we can verify that (1) the estimated internal camera parameters were

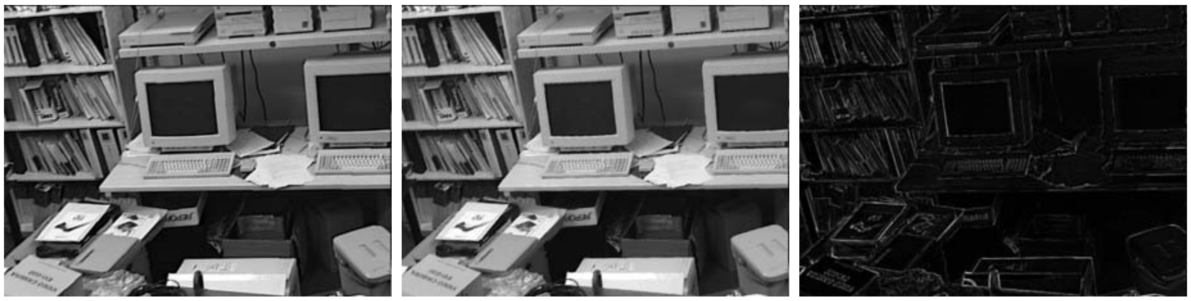


Figure 2.32: Comparison between the observed image and the generated image (Generated image: zoom = 4096, View direction = $(-20.0°, 4.0°)$).

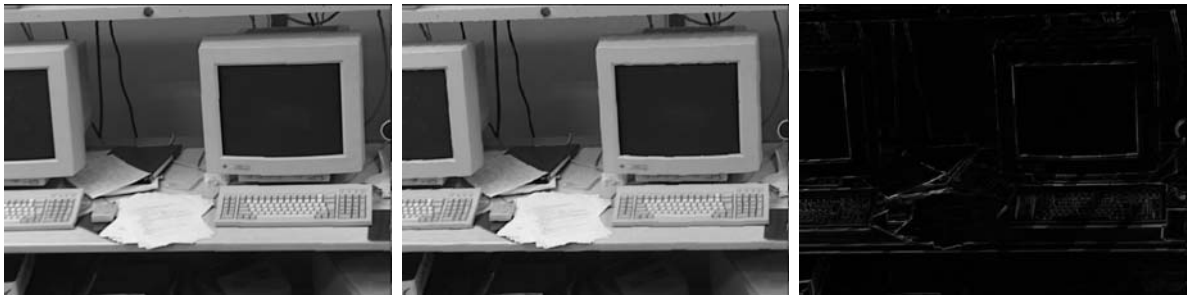


Figure 2.33: Comparison between the observed image and the generated image (Generated image: zoom = 12288, View direction = $(-12.0°, 6.0°)$).

appropriate and (2) images taken with arbitrary combinations of pan, tilt and zoom parameters could be generated from an APP.

However, the observed image was not completely identical to the generated image. The difference between the observed and generated images was caused by the following factors:

- Although the employed camera (SONY EVI-G20) is a good approximation of an FV-PTZ camera, the position of its projection center is not placed precisely at the rotation center as described in Section 2.5.2.2.

- Since the accuracy of the camera control from a computer was not adequate, the view direction of the observed image did not closely approximate that of the generated image. The influence of this difference can be canceled by translating the generated image as described in Section 2.4.3.

2.6 Concluding Remarks

We proposed an active camera for wide-area observation, which we call an FV-PTZ camera. With the FV-PTZ camera, we can realize the following functions.

- By changing the view direction, the camera can observe a wide area.
- By adjusting the zoom parameter, the camera can dynamically control the image resolution. This increases the adaptability and flexibility of the camera system.
- By mosaicing multiple images observed by changing pan, tilt and zoom parameters, the appearance model of the scene (i.e., APS and APP) can be easily generated.

While the instantaneous visual field of the FV-PTZ camera is limited, we can solve this problem by incorporating a group of distributed cameras.

Hereafter in this thesis, although we apply images generated from the scene model (i.e., APS/APP) only to the background subtraction method for object detection[5] , our method can be employed for many other vision tasks with active sensing. We, therefore, regard the FV-PTZ camera as a fundamental mechanism for Active Vision.

[5] We will address the object detection and tracking method using the APP in the next chapter.

Chapter 3

Active Background Subtraction for Object Tracking

3.1 Object Tracking using an Active Camera

3.1.1 Task of the Tracking System

This chapter proposes an active vision system for object detection and tracking using an active camera. We employ an FV-PTZ camera as an active camera.

The tasks of the tracking system using the FV-PTZ camera are defined as follows:

1. Detect an object that comes into the scene. This task is required to search for an object in the scene. In this chapter, we assume that there is only one object at most in the scene.

2. Track the object by controlling the pan-tilt parameters of the camera. To continue to track the focused target object, the system has to control the view direction of the camera towards the target object.

3. Capture images of the object in as high resolution as possible by controlling the zoom parameter of the camera. High resolution images are required to acquire the precise information about the object and achieve robust object identification.

To fulfill these tasks, the system has to incorporate image capturing, image processing, and camera control functions. This is because these functions need to be implemented based on the results of the other types of processing.

3.1.2 Object Detection Methods

Comparing a fixed camera, an active camera makes object detection difficult. This is because various appearance variations are caused in the observed image by varying camera parameters. The system, therefore, has to discriminate between these variations and object regions to properly detect object regions in the observed image.

We solve this problem with the FV-PTZ camera. Employing the FV-PTZ camera enables rectifying image variations caused by varying the camera rotation and zoom. With this ability, the system can detect object regions in the observed image taken while changing pan-tilt-zoom parameters by utilizing the same method as object detection with a fixed camera.

To detect object regions in the observed image, we can use the following methods:

Background subtraction: In this method, the observed image is compared with a stationary background image that is taken in advance. The regions different from the background image are considered to be object regions: if the gray level difference between pixels in these images is larger than a threshold, this pixel in the observed image is considered to be in object regions.

To utilize this method, the stationary background image has to been taken in advance.

In addition, the effectiveness of the background subtraction method is limited because the stationary background scene assumption does not always hold in the real world:

Variations in the scene

- Variations in an object's shape and posture: A fluttering leaf and flag, and the flickering of a CRT display (continuous small variations). Movements of a background object (intermittent large variations).
- Variations in the illumination: Sunlight fluctuations caused by sun and cloud movements and room-light variations.

To cope with variations in scenes, many works have been reported [SG99] [HHD00b] [SMKU00] [MOH00]. To cancel variations in a scene, [SG99] and [HHD00b] employ probability distributions to model the intensity variations at each pixel. For example, in [SG99], the pixel intensity is modeled as K Gaussian distributions (for continuous small variations). In [SMKU00], the background scene image is adaptively renewed by employing M-Estimation (for intermittent large variations). In [MOH00], non-stationary objects in the scene are modeled by (1) variations in the overall lighting conditions, and (2) local image pattern fluctuations, and so on. These methods solve the problem of variations in the scene.

Subtraction between consecutive images: By comparing images taken at t and $t+1$ with each other, the variations between the two images can be detected. The detected regions are considered to be moving objects.

This method cannot detect stationary objects.

In [MB94], the system 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 for the fixed camera to images taken with different camera rotations.

Optical flow: By comparing the window regions in the images taken at t and $t + 1$ with each other, the window regions projected from the same object in the scene are identified with each other between these consecutive images, and their motions between the images are estimated. The window regions showing different motions from those of the background region are are considered to be moving objects. Since appearance information has to be compared between images, it is difficult to detect objects with poor surface textures.

Similar to the above subtraction between consecutive images, this method cannot detect stationary objects.

In [MWM98b], moving objects are detected by an FV-PT camera. Since optical flow patterns caused by combinations of pan-tilt rotations can be estimated, the system can detect moving object regions by extracting those patterns that are different from the flow pattern of the background scene.

Template matching: This method is used for detecting the target region in the observed image. The target region is recorded as a target model called a template. By comparing the template with each region in the observed image, we can find the target region.

The template needs to be acquired in advance. Therefore, only the target recorded as the template can be detected.

In [HB96], a deformable template image and illumination basis for the target appearance are employed to cope with changes in the geometry and shading of the target object. With this knowledge, the system can track the target object without interference by variations in (1) the object's location and posture and (2) illuminations. In [WADP97], a multi-class statistical color and 3D shape model is used to obtain a 2D representation of a person's head and hands in a wide range of viewing conditions. The obtained 2D appearance is compared with the region in the observed image, and the object shape and posture can be recognized.

As mentioned above, each method has its own advantages and disadvantages, and we should select the appropriate method depending on the task.

In our system, we employ the background subtraction method for object detection. This is because if the background image is taken in advance, the background subtraction method can detect all object regions without any prior knowledge about target objects whether they are in motion or still.

To detect object regions while changing the camera rotation and zoom, the system has to perform the background subtraction method with input images taken with arbitrary combinations of pan, tilt and zoom parameters. We realize such an active background subtraction method by comparing the input image with the background image generated from a background image model (i.e., APS/APP).

In what follows, we first present our active object detection and tracking method with the FV-PTZ camera (Section 3.2). Then, experimental results are shown to demonstrate

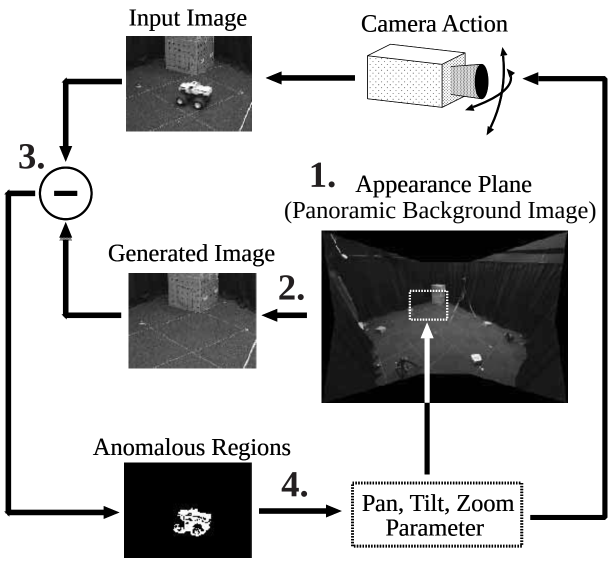


Figure 3.1: Object detection and tracking by the active background subtraction with an FV-PTZ camera.

that the proposed object detection and camera control method allows the system to detect object motions and keep tracking a moving object in a wide area (Section 3.3).

3.2 Active Background Subtraction with the FV-PTZ camera

3.2.1 Basic Scheme

Figure 3.1 shows a basic scheme for moving object detection and tracking using the FV-PTZ camera.

Step 1: Generate a background image model; with the FV-PTZ camera, a panoramic background image (i.e., APP) can be easily generated by integrating multiple images observed by changing pan-tilt-zoom parameters.

Step 2: Extract a window image from the panoramic background image according to the current pan-tilt-zoom parameters and regard it as the background image; with the FV-PTZ camera, one-to-one mappings exist between the positions in the panoramic background image and pan-tilt-zoom parameters of the camera.

Step 3: Compute the differences between the generated-background image and an observed image.

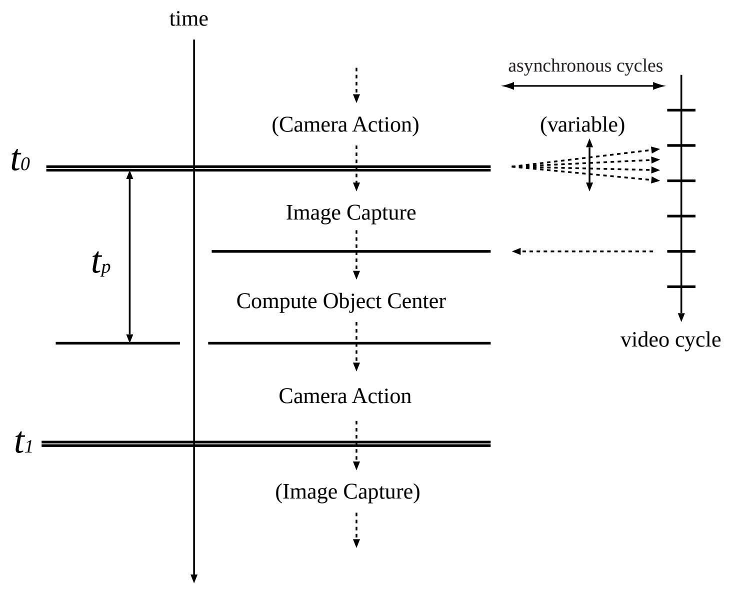


Figure 3.2: Time chart of the system cycle.

Step 4: If object regions are detected in the difference image, select one and control the camera parameters to track the selected target (the system is then in the tracking mode). Otherwise, move the camera along the predefined trajectory to search for an object (the system is then in the search mode).

Figure 3.2 shows the time chart of the system cycle. Suppose the image acquisition is initiated at t_0. The right vertical bar in Figure 3.2 illustrates the video cycle, which is not synchronized with the system; the camera repeats its own video cycle[1] .

Note that the image capturing should be done after the camera stops. The reasons why the camera should stop for image capturing are as follows:

Exact alignment between the observed image and the background image: It is necessary for object detection by the background subtraction method to align the observed image exactly with the background image. As we mentioned in Section 2.4.3, however, it is hard to attain pixel-wise exact alignment between the observed image and background image generated from the APS/APP under a smooth (non-stop) camera motion. To guarantee good conditions for the background subtraction method, therefore, the camera should be made to stop to observe the scene and maintain its pan-tilt-zoom parameters until the image capturing is finished.

Motion blur avoidance: A fast camera motion causes motion blurs in the observed image and they incur many false alarms in the background subtraction method.

[1] If the camera can accept external triggers, the system can capture images whenever required.

The degree of motion blurs depends on the shutter speed and the rotational velocity of the camera:

- A fast shutter speed can avoid motion blurs in the observed image. To perform such image acquisition, (1) the camera has to possess a mechanism for controlling the shutter speed and (2) bright illumination is required to observe the scene with a short exposure time. That is, the restrictions about the camera function and the scene condition are required.
- The rotational velocity in the observed image is determined depending on the following factors: (1) the focus length of the camera, (2) the rotational velocity of the camera, (3) the distance between the camera and the object, and (4) the velocity of the object motion. Only the above factors 1 and 2 (i.e., the focal length of the camera) can be adjusted by the system. If the focal length is short (namely, the view angle of the camera is wide), the rotational velocity of the camera decreases because it is enough to keep tracking the target object. Then, motion blurs can be prevented. However, the camera cannot observe a high resolution image of the object. If the focal length is long (namely, the view angle of the camera is short), on the other hand, the rotational velocity of the camera increases to keep tracking the target object, and then motion blurs appear. Motion blurs, therefore, cannot be prevented to keep obtaining the meaningful target image as long as the camera captures the image while changing the camera parameters.

While we can solve the problem of motion blurs by analyzing them and removing their effects from the observed image[WGCM96], this analysis unfortunately makes real-time processing difficult.

Hence, in our tracking system, the camera repeats a ‘stop and motion’ process to stably detect object regions by the background subtraction method without the strong restrictions about the camera and the scene.

In the basic scheme of active background subtraction, image capturing, image processing, and camera control are, in turn, activated by the information managed by the other functions:

Image capturing: After the camera stops, the image is captured.

Image processing: After the newest image is captured, the system detects objects in the captured image.

Camera action: After the information of the detected object is obtained, the system determines and starts the next camera action.

3.2.2 Object Detection Algorithm

3.2.2.1 Generating the Panoramic Background Image

For active background subtraction, we utilize our earlier proposed APS/APP image model as the background image model. The reliability of object detection with the APS/APP image depends on the following factors:

1. Accuracy of the fixed-viewpoint.

2. Precision of the estimated (geometric and photometric) internal camera parameters.

3. Differences in the camera parameters between the observed image and generated-background image.

To guarantee the reliability of the detected result, we proposed (1) a calibration method for fixing the viewpoint, (2) an internal parameter calibration method, and (3) active camera parameter estimation and correction functions in chapter 2.

In addition, to increase the accuracy of object detection, we adaptively change the threshold for the background subtraction method depending on the coordinates in the APS/APP. The practical procedure for the variable threshold is as follows:

1. Generate the APS/APP image.

2. Capture the stationary background image while changing the pan-tilt-zoom parameters.

3. Re-project all captured images onto the APS/APP screen, and compute the gray level differences between the re-projected images and the APS/APP image for each pixel.

4. Record the maximum difference for each pixel into a virtual screen whose data structure is identical to that of the APS/APP screen.

The pixel values in the above new virtual screen are considered to be the thresholds for the background subtraction method. We call this newly generated image the *Threshold APS/APP* image.

3.2.2.2 Image Capturing and Correction

A captured image can be affected by various image distortion factors. On the other hand, the background image generated from the APS/APP image is a good approximation of the perspective projection, because the APS/APP image is corrected by the estimated internal camera parameters. This difference between the captured and generated images incurs detection errors.

To solve this problem, the captured image also needs to be corrected by the estimated internal camera parameters. The image correction is implemented by the same method as the generation of the APS/APP image.

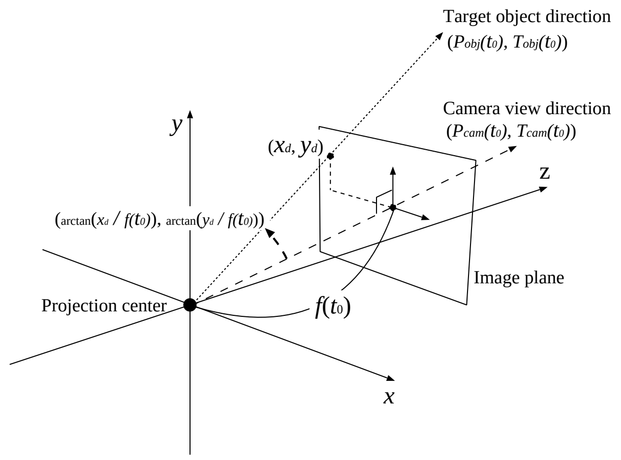

Target object direction
$(P_{obj}(t_0), T_{obj}(t_0))$
Camera view direction
$(P_{cam}(t_0), T_{cam}(t_0))$
y
(x_d, y_d)
z
$(\arctan(x_d / f(t_0)), \arctan(y_d / f(t_0)))$
Image plane
Projection center
$f(t_0)$
x

Figure 3.3: Directions of the Camera view and the target object.

3.2.2.3 Background Subtraction

The system repeats the following steps for object detection:

1. Capture the image just after the camera motion is stopped. Let the image capture time be t_0.

2. Correct the distortions of the captured image.

3. Generate the background and threshold images from the APS/APP and threshold APS/APP images, respectively, according to the current pan-tilt-zoom parameters.

4. Compute the gray level difference between the captured and generated-background images.

5. Compare the difference with the threshold image. If a pixel value of the difference is larger than that of the threshold image, this pixel is considered to be in an object region.

6. Compute (1) the centroid of the detected object region and (2) the number of pixels considered to be in the object region (this number is denoted by NP_d). The 2D vector from the image center to the computed centroid is denoted by (x_d, y_d).

7. Based on the information of the target object (i.e., (x_d, y_d) and NP_d), the system determines the next camera parameters (mentioned later).

3.2.3 Camera Control

3.2.3.1 View Direction Control

The system determines the next view direction of the camera based on the detected object information (i.e., (x_d, y_d)). We define a straight line from the projection center to the FOE on the image plane as the view direction of the camera[2] . To gaze at the target object, the system turns the view direction of the camera towards (x_d, y_d) in the observed image taken at t_0. Let $(P_{cam}(t), T_{cam}(t))$ and $(P_{obj}(t), T_{obj}(t))$ denote the pan-tilt directions of the camera view and those of the target object, respectively, and $f(t)$ denote the focal length of the camera at t. The direction from the projection center to the target object at t_0 is represented by

$$\begin{pmatrix} P_{obj}(t_0) \\ T_{obj}(t_0) \end{pmatrix} = \begin{pmatrix} P_{cam}(t_0) \\ T_{cam}(t_0) \end{pmatrix} + \begin{pmatrix} \arctan(x_d/f(t_0)) \\ \arctan(y_d/f(t_0)) \end{pmatrix} \tag{3.1}$$

as illustrated in Figure 3.3.

Then, to capture the target object at the image center in the next capturing time (denoted by t_1), the current view direction of the camera $(P_{cam}(t), T_{cam}(t))$ should be changed by $(\arctan(x_d/f(t_0)), \arctan(y_d/f(t_0)))$. That is,

$$\begin{pmatrix} P_{cam}(t_1) \\ T_{cam}(t_1) \end{pmatrix} == \begin{pmatrix} P_{cam}(t_0) \\ T_{cam}(t_0) \end{pmatrix} + \begin{pmatrix} \arctan(x_d/f(t_0)) \\ \arctan(y_d/f(t_0)) \end{pmatrix} = \begin{pmatrix} P_{obj}(t_0) \\ T_{obj}(t_0) \end{pmatrix} \tag{3.2}$$

For the background subtraction to obtain a successful result, the camera has to stop before the image is captured. The system, therefore, controls the view direction and holds it at $(P_{obj}(t_0), T_{obj}(t_0))$. To confirm the camera motion, the system continues inquiring about the camera's current view direction (i.e., $(P_{cam}(t), T_{cam}(t))$) until the difference between $(P_{cam}(t), T_{cam}(t))$ and $(P_{obj}(t_0), T_{obj}(t_0))$ is less than the threshold. When the current camera parameter satisfies this criterion, the system starts image capturing and object detection again. Let the time when the system captures the next image $t_0 + t_p$[3] . The time spent in controlling the view direction (i.e., $t_1 - (t_0 + t_p)$) is determined by $(\arctan(x_d/f(t_0)), \arctan(y_d/f(t_0)))$ and the motion characteristic of the camera.

3.2.3.2 Zoom Control

The objective of view direction control is to track the target object while keeping its captured silhouette at the center of the observed image. On the other hand, the zoom is controlled to accomplish the following two tasks:

Stabilization of tracking: To keep capturing the object's silhouette in the image without failure, the zooming factor should be controlled so that the view angle becomes wider.

[2] Namely, the view line is identical to the optical axis of the camera.
[3] See, Figure 3.2.

Acquisition of precise object information: To observe a high resolution image of the object, the zooming factor should be controlled so that the view angle becomes narrower.

The zoom controlling method is designed to take into account the above conflicting tasks.

We evaluate the degree of achievement of the above two tasks by the centroid of the detected region (i.e., (x_d, y_d)) and the pixel count of the detected region (i.e., NP_d). If the distance between the image center and the centroid of the detected region is smaller than the threshold Const_d (namely, $||(x_d, y_d)|| < \text{Const}_d$), the system considers that the current view angle is wide enough to keep tracking the target object and controls the zoom parameter to zoom in. Let the total number of pixels in the observed image be NP_I. The zoom parameter is then adjusted so that the rate of NP_d (namely, NP_d/NP_I) becomes the predefined constant. Otherwise (namely, $||(x_d, y_d)|| \geq \text{Const}_d$), the camera zooms out to increase the tracking stability.

In the above discussion, we do not address the time spent in changing the zoom parameter. Although all of the camera parameters are controlled at the same time ($t_0 + t_p$ in Figure 3.2), the intervals spent by them might differ from one another. To track the target object persistently, the view direction control should have a higher priority than the zoom control. Accordingly, the system stops zooming when the view direction reaches the destination (i.e., $(P_{obj}(t_0), T_{obj}(t_0))$) even if the zooming has not finished.

3.3 Experiments

3.3.1 System Organization

We conducted experiments with the following architecture:

Active camera: SONY EVI-G20: When the camera angle and the zooming factor were changed for tracking, both of them were controlled at the maximum speed[4] .

PC: Sun Microsystems Sun Ultra2-1300.

Image capturing board: Active Imaging Snapper: We calibrated the internal camera parameters of EVI-G20 for a 640 × 480 [pixels] image as discussed in chapter 2. In this experiment, however, we resized the gray image from 640 × 480 [pixels] to 320 × 240 [pixels] in order to cancel the effect of interlace scanning. The internal camera parameters were adjusted to the image size:

- ρ, (x_0, y_0) and FOE: a half of the original values.
- κ: four times of the original value.
- $(\theta_c, \phi_c, \varphi_c)$ and α: No transformation.

[4] The mechanical specifications of EVI-G20 are shown in Section 2.5.2.1.

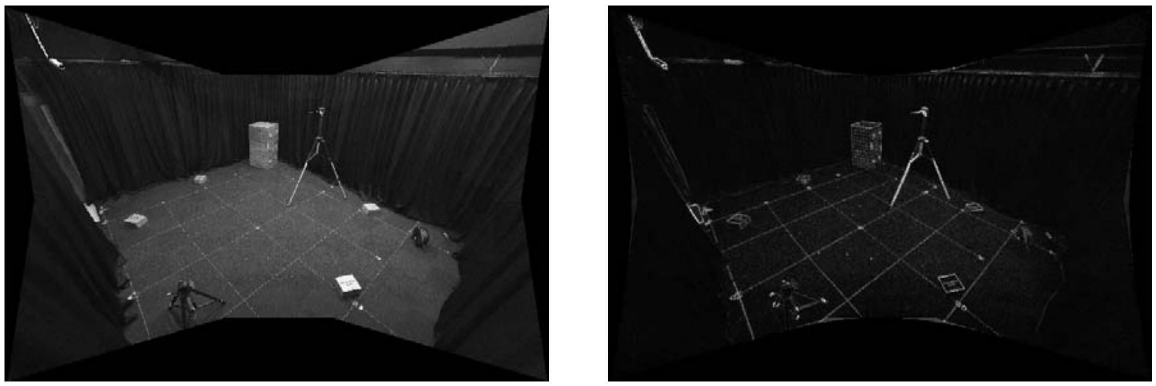


Figure 3.4: Generated APP image (left) and threshold APP image (right).

3.3.2 Tracking Results

To demonstrate the effectiveness of our active background subtraction method, we conducted experiments to detect and track a radio-controlled toy car. The car was manually controlled by a human; it moved around on a 4[m] $\times$ 4[m] flat floor while avoiding several obstacles and sometimes stopped and changed directions. The FV-PTZ camera was placed about 2.5[m] above the floor corner looking downward obliquely.

We first generated the APP and threshold APP images. Figure 3.4 shows these images. In the threshold APP image, high thresholds are given especially at the edges of objects. The threshold image, therefore, enables the system to cancel quantization errors caused by the A-D conversion in the image capturing.

Figure 3.5 and Figure 3.6 show sequences of observed images and detected target silhouettes. We gave a number to each observed image in the order of capture (from No. 1 to No. 40). In this experiment, the zooming factor was controlled so that the ratio of the pixel count of the detected object region to the total pixel count of the image was 0.15 to 1. Figure 3.7 and Figure 3.8 illustrate the histories of the pan-tilt and pan-tilt-zoom controls during tracking, respectively. The type of the history line denotes the system state. The dotted and solid lines denote the search and tracking modes, respectively.

In images No. 1 $\sim$ 4, the system moved the camera view along the predefined trajectory with the widest view angle to search for an object. After the system detected an object in the 5th observed image, the system considered it to be the target object, and the camera was controlled to capture the centroid of the detected region at the image center during the 6th $\sim$ 11th observations. When the system observed the 12th image, the target object stopped its motion. The system then started capturing the target object's image in a high resolution by controlling the zoom. The view angle of the camera became the closest in the 22nd observed image. When the target object started moving again, the system zoomed out to track it stably. In particular, from the 36th to 40th observations, the view angle became wider because the target object moved at a higher speed.

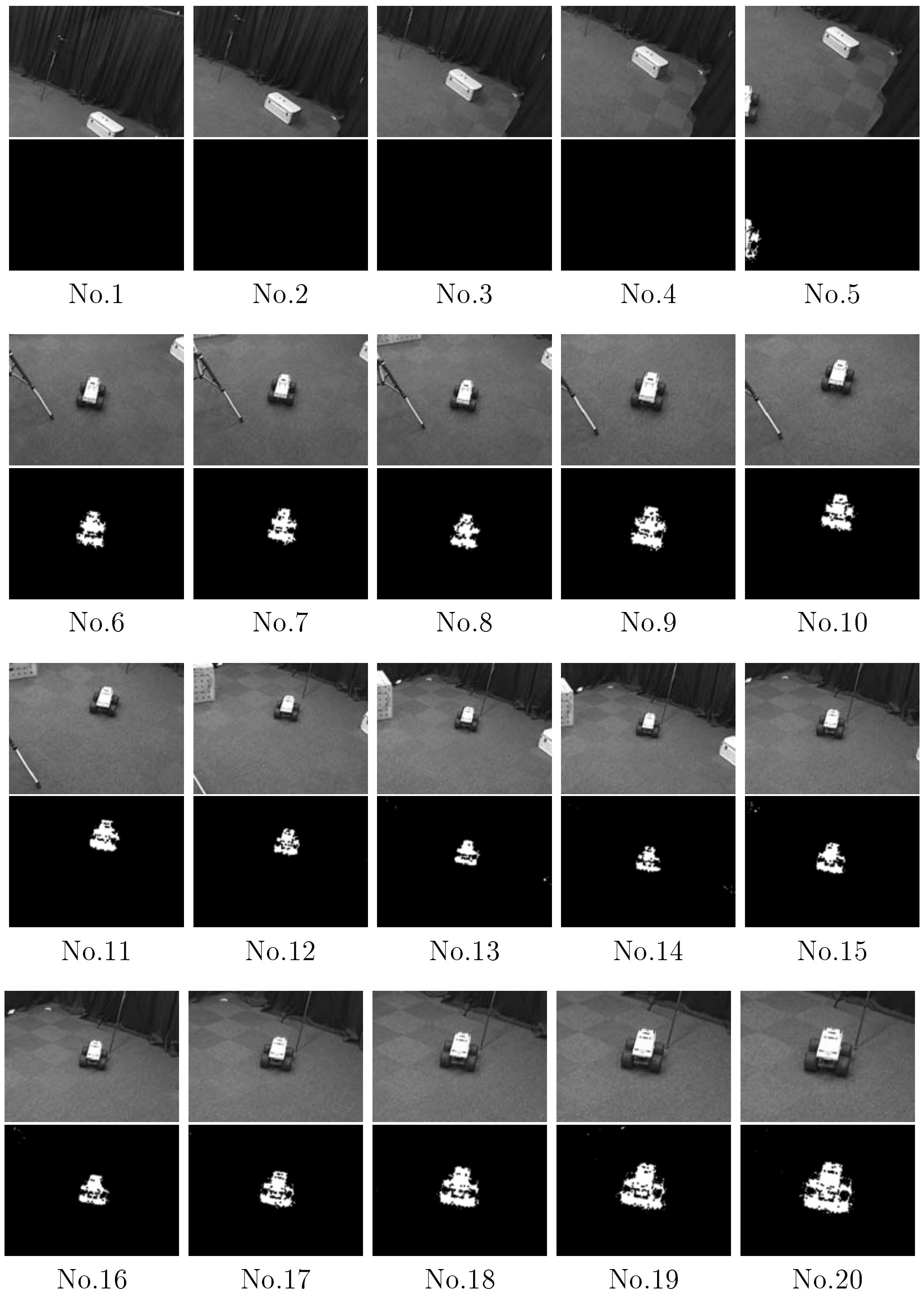


Figure 3.5: Images observed during tracking (upper: input images, lower: detected object silhouette).

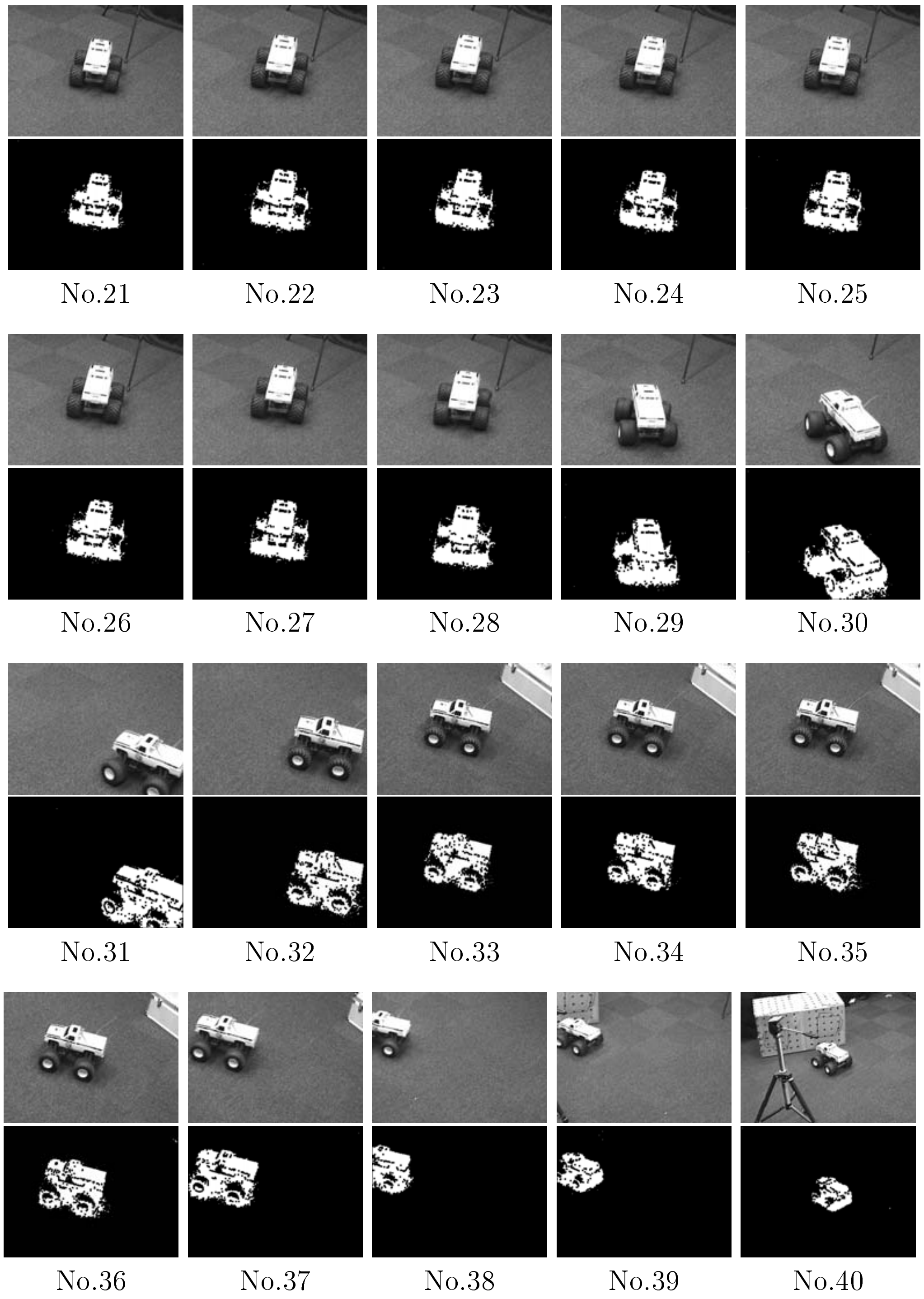


Figure 3.6: Images observed during tracking (upper: input images, lower: detected object silhouette).

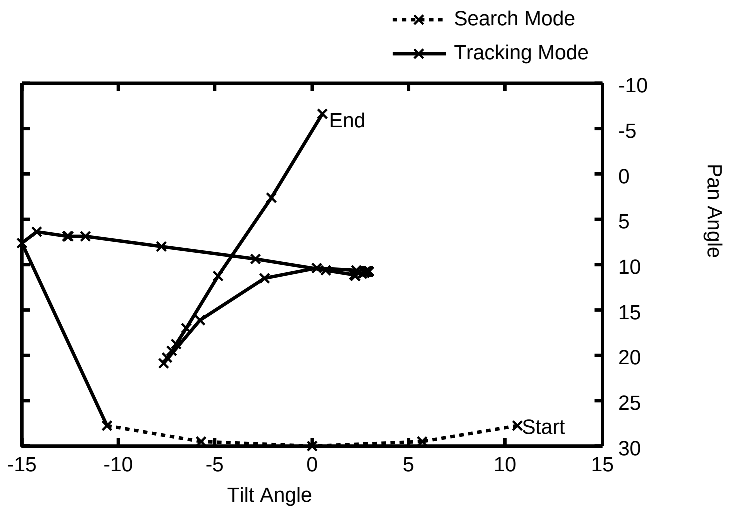


Figure 3.7: History of pan-tilt control.

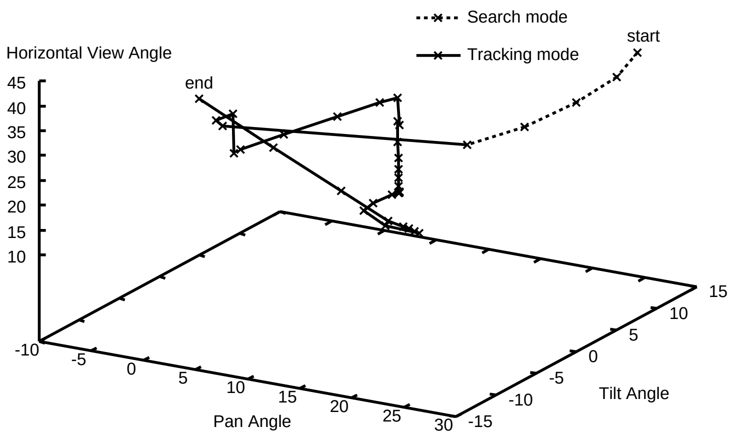


Figure 3.8: History of pan-tilt-zoom(horizontal view angle) control.

The entire tracking period was 22 seconds (i.e., about 1.8 image-observations/second on average). The interval between observations was not short enough to stably track the target object and satisfactorily acquire the object information. The reason why the interval was this long was because the image capturing could not be done while the camera was moving in order to avoid motion blurs. That is, the long interval between observations was caused by the mechanical characteristic of EVI-G20 ($T_a \gg T_p$)[5] .

3.4 Concluding Remarks

We proposed an object tracking system with an active camera. We employ an active background subtraction method with the FV-PTZ camera for object detection and tracking. The system keep tracking a target object persistently as follows:

- The system detects object regions in the observed image taken with any combinations of pan-tilt-zoom parameters by the active background subtraction method.
- The system controls the camera not only to keep tracking the target object but also to capture the object image in as high resolution as possible while taking into account the information of the detected target object.

In order to augment this tracking method, the following technical problems remain:

- Study on an object detection method that can work independent of the camera's actions (e.g., employing optical flow analysis[MWM98b]).
- Consideration of the target motion and camera motion. As mentioned in Section 3.3.2, the time spent in controlling the camera is longer than the time for the other processes. This long interval makes reactive tracking difficult. To solve this problem, a type of prediction-based camera control that takes into account the target motion and camera motion might be effective. Such a method is described in the next chapter.
- Realization of more flexible system dynamics. A sequential perception-action cycle is too simple to reactively cope with dynamic object motion. For the flexible system dynamics, therefore, both the perception and action functions should be implemented as parallel modules. In the next chapter, we propose a *Dynamic Vision*, where the perception and action modules work together while dynamically interacting with each other.

[5] In this system, the image capturing, generation and subtraction processes are completed in less than one video cycle (i.e., 1/30 [sec]).

Chapter 4

Real-time Object Tracking with Dynamic Memory

4.1 Dynamic Integration of Visual Perception and Camera Action

In the last chapter, we proposed an active background subtraction method for object detection and tracking. The basic scheme for this method is to simply repeat the steps described in Section 3.2.1.

To improve the dynamics of this tracking system, a prediction-based active vision system using an active camera is proposed in [MWM99]. The system incorporates a sophisticated temporal coordination mechanism among image capturing, image processing, and camera control. That is, the functions of the system's perception and action modules are well-coordinated to work together. Figure 4.1 shows the dynamics of the system developed in [MWM99]. For object tracking, the system incorporates a prediction-based dynamic control method (1) to cope with delays involved in the image processing and physical camera motion and (2) to synchronize the image acquisition and camera motion. With this prediction-based method, the system adaptively controls both the camera parameters and the next observation timing so that a 'best-looking' object image can be captured. While this system employs a sophisticated prediction-based camera control, its fundamental dynamics is still limited to the sequential one; the activations of the perception and action modules are just interleaved on the temporal axis (Figure 4.1). That is, one module stays idle while the other is activated.

Such sequential control introduces non-smooth camera motions and intermittent observations. Furthermore, the system dynamics realized by the sequential steps is too simple to cope with dynamically changing situations as found in the real world. To overcome these disadvantages, a flexible system dynamics to control visual perception and camera action should be incorporated into the system. That is, we should introduce a dynamic system architecture, where perception (image capturing and image processing) and action (camera control) modules run in parallel. Such an architecture would allow the system to adapt itself to dynamically changing situations in the real world.

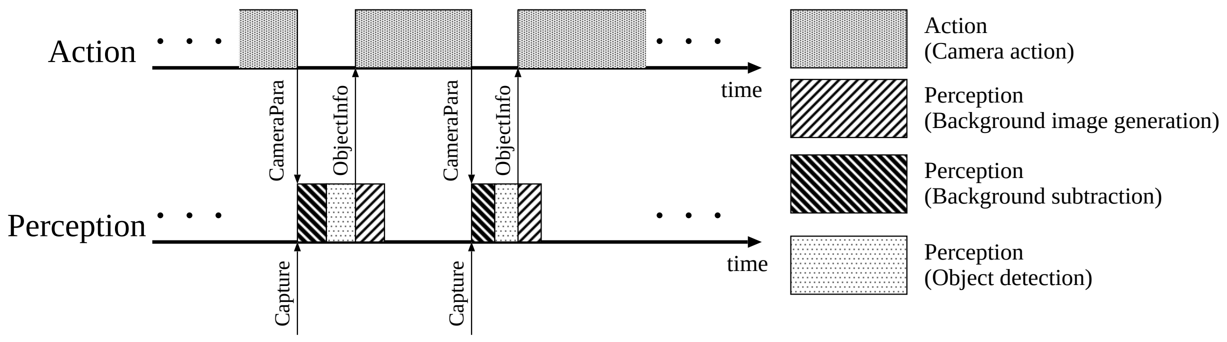


Figure 4.1: Dynamics of the object tracking system proposed in [MWM99].

4.1.1 Dynamic Vision

The fundamental functions of the tracking system are summarized as follows:

Image capturing: When a meaningful image can be observed, the system captures the image.

Image processing: The system analyzes the captured image to detect an object.

Camera Action: Based on the information of the detected object, the system determines the next camera parameters (i.e., pan-tilt angles and zooming factor). The system then starts controlling the camera.

Since the camera parameters at the time when the image is captured are required to analyze the observed image taking into account the camera motion, the perception module has to require them from the action module. Similarly, in order to determine the next camera parameters to gaze at the target object, the action module has to request the target object information from the perception module.

The above information flows between the perception and action modules are illustrated in Figure 4.2. The perception and action modules are activated alternately, and their functions depend on each other. We therefore have to design temporal interaction between the two modules.

The integration of visual perception and camera action has been studied in Active Vision [AWB88] [Bal89] and Visual Servo [Bro90] [WSN87]. In the former, although many studies have been done on the *Where to Look* problem (i.e., geometric camera motion planning based on image analysis), only a few analyses have been done on the system dynamics. Weiss–Sanderson–Neuman [WSN87] called this dynamics the '*static* look and move structure' where visual perception and camera control modules are activated sequentially.

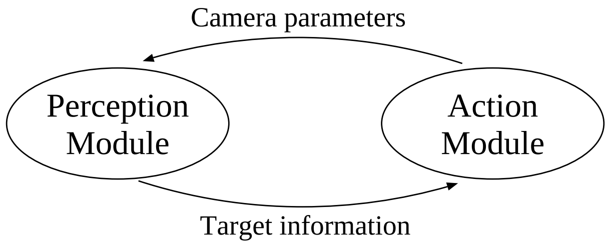


Figure 4.2: Information flows between the perception and action modules for object detection and tracking.

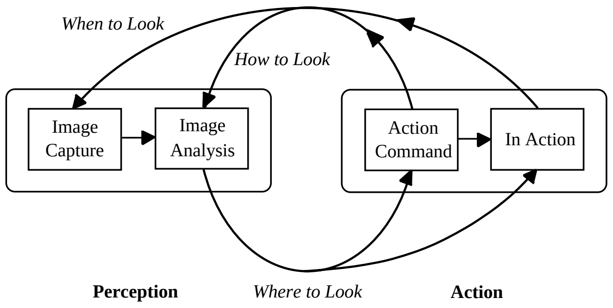


Figure 4.3: Information flows in dynamic vision.

In visual servo, on the other hand, various dynamic control methods have been studied based on the control theory. For example, Brown[Bro90] showed that prediction-based control is effective for coping with delays.

In visual servo systems, visual perception and camera control modules work in parallel and the information flows continuously through the signal lines connecting the modules. However, the inter-module interactions are rather simple and fixed. First of all, the types of information exchanged between the modules are exactly the same as those of active vision. Second, the interactions are continuously synchronized by analog and discrete time parameters and no asynchronous interaction mechanisms are incorporated. In fact, asynchronous events usually happen in the real world. That is, the world itself has its own dynamics, which exhibits asynchronous features as its complexity increases. To make a system work adaptively in such complexity, we should develop more flexible dynamic interaction mechanisms between visual perception and camera control modules.

Based on the above discussions, Matsuyama[Mat98] proposed a novel scheme named *Dynamic Vision*, where event-driven asynchronous interactions are realized between visual perception and action modules. The distinguishing characteristics of dynamic vision are as follows.

- In a dynamic vision system, complicated information flows are formed between visual perception and action modules to solve the *When to Look* and *How to Look* problems as well as the common *Where to Look* problem (Figure 4.3).

Where to look problem: Based on the result of an image analysis, the system decides the next gazing location and the viewing trajectory. The system then starts controlling the camera.

When to look problem: The next camera action gives the system the subsequent image capturing timings. These timings are determined by taking into account both the object motion and the mechanical characteristic of the camera. For example, the image capturing timings should be determined depending on the camera motions, because quick motions can degrade observed images.

How to look problem: When the image capturing timings are determined, the system is also endowed with the appropriate way to detect object regions. Actual examples are given in the following:

- If the interval until the next capturing is long enough for the camera to gaze the goal direction and stop its motion, the background subtraction method is suitable for object detection.
- If the camera cannot achieve the above stop-and-sensing observation due to mechanical limitations, another object detection method is required[1] .

That is, the image analysis should be facilitated by the camera parameters (focus, iris, zoom as well as motion parameters).

In this thesis, however, we do not focus on the 'how to look problem' in order to concentrate on designing the system dynamics, and detect object regions only by the background subtraction method.

- The system dynamics is represented by a pair of parallel time axes, on which the dynamics of visual perception and action modules are represented. The dynamic interactions between the modules are represented by inter-time-axes coordinations.

4.1.2 Dynamic Memory Architecture: Asynchronous Inter-Module Interaction

To realize dynamic vision systems, we have to make the modules run in parallel and develop an asynchronous dynamic interaction mechanism between the modules. The asynchronous interactions cause the following problems:

Problem 1: A reactive response from the other module is not guaranteed due to the asynchronization between the modules; when a module requires the other module to transmit its information, the requested module might not reactively respond to the requirement because it is executing its own task autonomously.

[1] In [MWM98b], object regions are detected while a camera is rotated based on the optical flow analysis.

Problem 2: The required information at a certain moment is not necessarily obtained because each module repeats its own cycle autonomously; if a module requires the information at T from the other module, the requested module might not have the information at T.

To support such asynchronous inter-module interactions, Matsuyama *et al.*[MHW$^+$00] proposed a novel dynamic system architecture named the *Dynamic Memory Architecture*, where parallel modules 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 modules are implemented as parallel processes that dynamically read from and write into the dynamic memory according to their own individual dynamics. The dynamic memory supports asynchronous dynamic interactions (i.e., data exchanges between the modules) without wasting time for synchronization. This no-wait asynchronous module interaction capability greatly facilitates the implementation of real-time reactive systems such as a moving object tracking system.

We, therefore, can solve the above two problems by employing the dynamic memory:

1. The dynamic memory mediates between all of the modules for the asynchronous information exchanges. Each module writes/reads shared information to/from the dynamic memory.

2. Reading the values from the dynamic memory enables the system to obtain information at an arbitrary time.

Based on the above discussion, we define the organization of the dynamic vision system with the dynamic memory as follows (Figure 4.4):

- The parallel perception and action modules read the image from the camera and control the camera, respectively, according to their own individual dynamics.

- The system contains the dynamic memory, which records the information about the system (e.g., the state of a camera action) and the scene (e.g., an object location).

- Each module interacts with the other modules through the dynamic memory.

In this chapter, we propose an active vision system for real-time object tracking, where perception and action modules are dynamically integrated with the dynamic memory. While the basic contents of the information exchanged between the modules are the same as those of the tracking system proposed in the last chapter, the modules exchange their information through the dynamic memory as illustrated in Figure 4.4. The system controls the camera to reactively adapt itself to dynamic variations in the scene by employing not only the prediction-based control method but also flexible information exchange without synchronization.

In what follows, we first introduce the general concept and functions of the dynamic memory in Section 4.2. Section 4.3 proposes a real-time object tracking system with

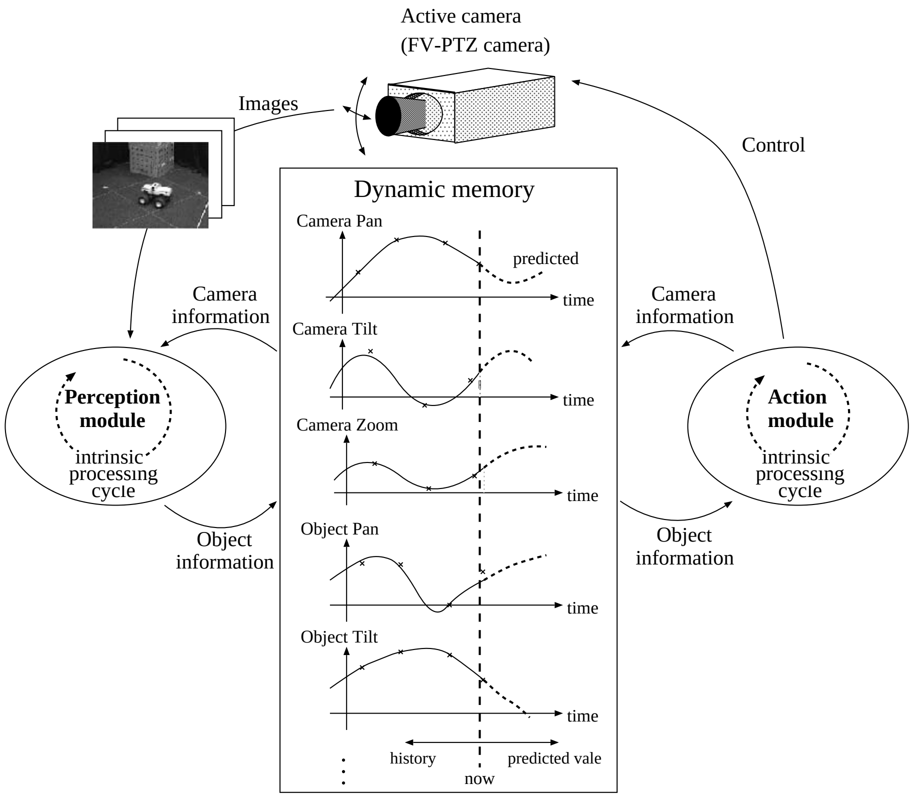


Figure 4.4: Real-time object tracking system using the dynamic memory.

the dynamic memory. In Section 4.3.3, a practical implementation of the dynamic memory for object tracking is presented. In Section 4.3.4 and Section 4.3.5, we describe an object detection method that copes with dynamic camera motions and the design of a sophisticated prediction-based camera control method, respectively. A quantitative dynamic characteristic of our system is given to demonstrate its performance in Section 4.4. Our experimental results demonstrate that the proposed dynamic control method greatly improves the performance of the system in terms of stability and flexibility for object tracking.

4.2 Dynamic Memory

In general, an intelligent system such as an AVA consists of multiple modules with different functionalities and dynamics. Thus, a key issue to design and implement an intelligent system rests in the functional and dynamic integrations of the modules. Here we focus on the dynamic integration of the modules, since the functional decomposition of an AVA is rather straightforward: visual perception, camera action, and network communication

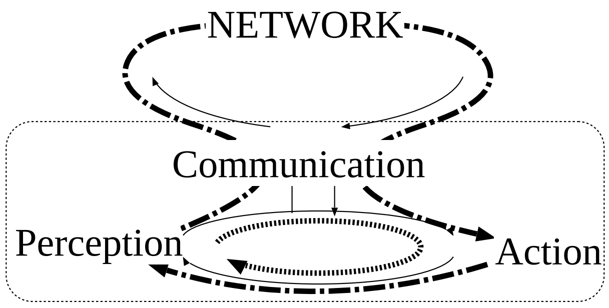


Figure 4.5: Module organization of an AVA and information flows among the modules.

modules.

To implement an AVA, we have to integrate three modules with different intrinsic dynamics:

1. **Visual Perception**: video rate periodic cycle,
2. **Camera Action**: mechanical motions involving rather large inconstant delays, and
3. **Network Communication**: asynchronous message exchanges, where inconstant delays are incurred depending on communication activities over the network.

Figure 4.5 illustrates information flows among these modules. The problem we study here is how we can design and implement flexible dynamic information flows, i.e., dynamic interactions among the modules.

4.2.1 Basic Operations

In the dynamic memory architecture, multiple parallel processes such as perception, action, and communication modules, share the dynamic memory. Each module writes its state variable such as pan-tilt angles of the camera and the target object location. This information is shared among all the modules through the dynamic memory. Since the shared information is written as a temporal history (i.e., time-series data) and shared among the modules, the time information in all the modules have to been consistent with each other. To guarantee the time consistency among the modules, one of the following conditions is required:

- All the modules share a single clock.
- Each module has its own time: all of the clocks are set with each other, or the offset between them are known.

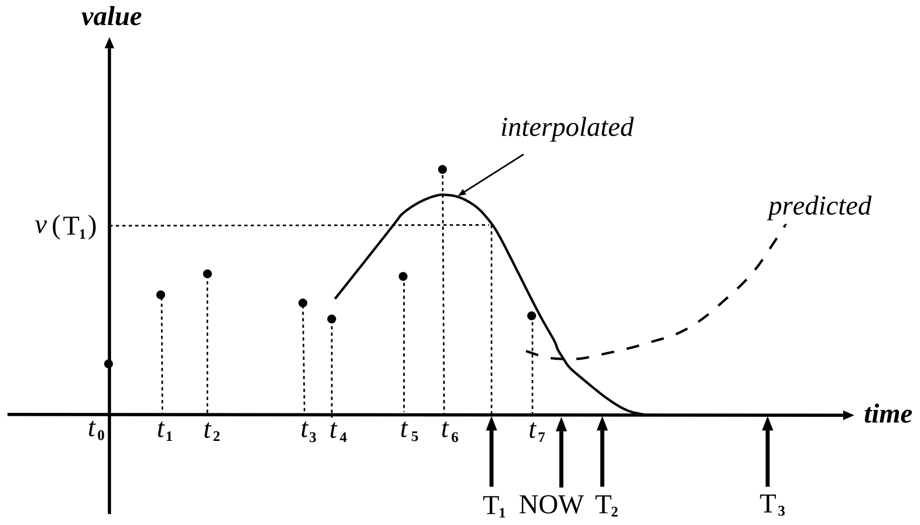


Figure 4.6: Representation of a time varying variable in the dynamic memory.

In our system, all the modules work in a single processor and share the same clock in the processor.

The read/write operations from/to the dynamic memory are defined as follows (Figure 4.6):

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 (a sequence of black dots in Figure 4.6). 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 from the dynamic memory the value of the variable at a certain moment according to its own dynamics: for example, the value at T_1 in Figure 4.6. 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 which are not written yet by the writer process (for example, the value at T_3 in Figure 4.6). 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. Note that as illustrated in Figure 4.6, multiple values may be defined by the interpolation and prediction functions, for example, at NOW and T_2 in Figure 4.6. We have to define the functions to avoid such multiple value generation.

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. That is, the dynamic memory integrates parallel processes into a unified system while decoupling their dynamics; each module can run according to its own dynamics without being disturbed by the others. This no-wait asynchronous module interaction capability greatly facilitates the implementation of real-time reactive systems.

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 well model the dynamics of the object. As will be shown later, therefore, off-line modeling and calibration of the object dynamics should be done *a priori* to define the functions.

4.2.2 Comparison with Previous Works

While the system architecture consisting of multiple parallel processes with a common shared memory looks similar to the 'white-board architecture[SST86]', the critical difference rests in that the dynamic memory maintains variables whose values change dynamically along the temporal axis spanning the period *continuously* from the past to the *future*.

Little–Kam[LK93] proposed an idea of the *smart buffer*, where virtual values are synthesized to dynamically coordinate parallel processes with different processing speeds. Their idea, however, does not include variables with dynamically changing values or their temporal interpolation and prediction.

Zhang–Mackworth[ZM95], on the other hand, proposed *constraint nets*, where variables with dynamically changing values were introduced. Their major interest, however, was in designing dynamic systems and did not refer to the dynamic integration of multiple parallel modules like the white-board system. Thus, the dynamic memory architecture can be regarded as an advanced dynamic system architecture integrating the white-board, the smart buffer, and the constraint nets.

4.3 Real-Time Object Tracking System using the Dynamic Memory

We design a real-time object tracking system based on the idea of dynamic vision. Basically, this system is an extension of the active background subtraction system proposed in the last chapter, and the tasks of the system are the same as those defined there:

1. Detect an object that comes into the scene.

2. Track it by controlling the pan-tilt parameters.
3. Capture its images in as high resolution as possible by controlling the zoom parameter.

In our system, the camera control (i.e., tasks 2 and 3 above) is implemented as smooth camera motions. This is a difference of the system behavior from the system proposed in the last chapter.

4.3.1 System Organization

The system consists of an image processor (PC) with an active camera (FV-PTZ camera). We employ SONY EVI-G20 as an active camera. The perception and action modules and the dynamic memory are implemented by threads on a PC.

Figure 4.4 illustrates the system organization, where the pan-tilt angles of the camera and target object location are dynamically exchanged between the perception and action modules through the dynamic memory. The information flows between the modules are summarized as follows:

Information	*Writer*		*Reader*
Pan-tilt angles and zooming factor	Action module	→	Perception module
Object information	Perception module	→	Action module

4.3.2 Basic Scheme of Real-time Active Background Subtraction

The basic scheme of active background subtraction is divided into two types of processes, each of which is executed by the perception and action modules, as illustrated in Figure 4.7.

Perception module: Detect object regions in the observed image.

Action module: Control the camera rotation and the zooming factor to capture a silhouette of the target object at the image center.

Figure 4.8 illustrates the real-time active background subtraction implemented by the dynamic interactions between the perception and action modules.

4.3.3 Implementation of the Dynamic Memory for Real-time Object Tracking

Here, we present a practical implementation of the dynamic memory for our real-time active background subtraction system.

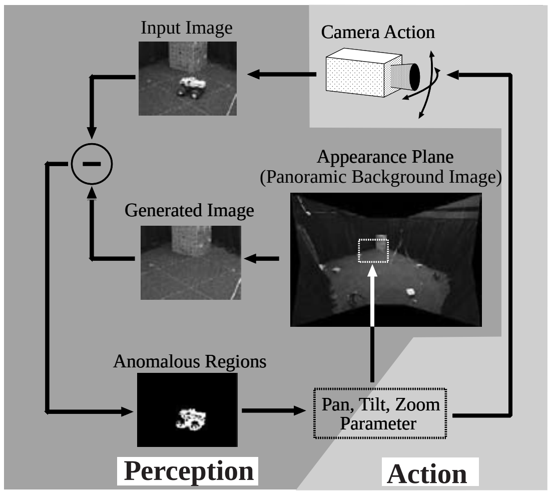


Figure 4.7: Role assignment to the perception and action modules.

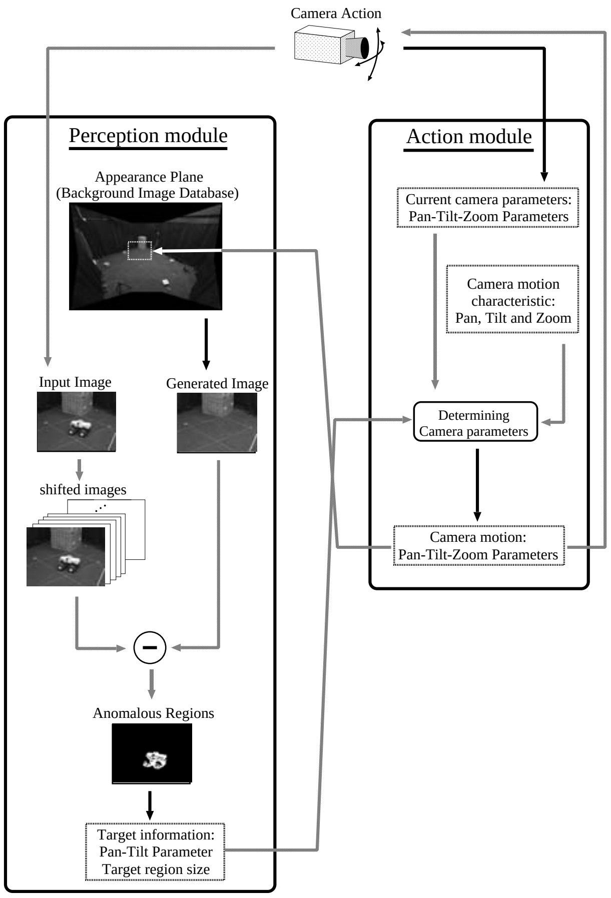


Figure 4.8: Object detection and tracking by a real-time active background subtraction: the perception and action modules dynamically interact to work together.

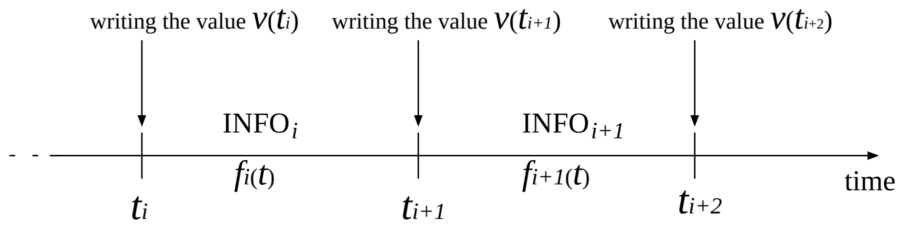

writing the value $V(t_i)$
writing the value $V(t_{i+1})$
writing the value $V(t_{i+2})$
INFO$_i$
INFO$_{i+1}$
$f_i(t)$
$f_{i+1}(t)$
t_i
t_{i+1}
t_{i+2}
time

Figure 4.9: Representation of the time-series information in the dynamic memory.

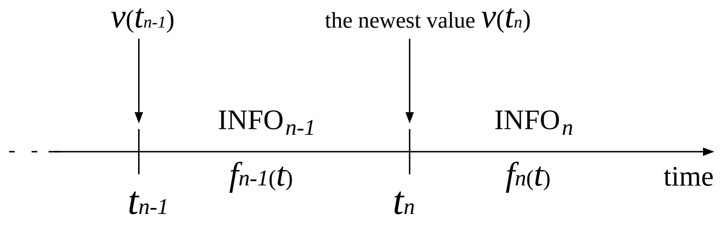


Figure 4.10: Representation of the future information in the dynamic memory.

4.3.3.1 Describing Time Varying Information

Based on the idea illustrated in Figure 4.6, the following descriptive method is employed in the system. The information written into the dynamic memory is a set of temporal discrete values. We represent this information as a set of functions, each of which is defined for every interval between the consecutive values. The temporal information INFO$_i$, which is valid from t_i to t_{i+1} (where i denotes the ith updating cycle), is represented by the interpolation function $f_i(t)$ (Figure 4.9):

$$\begin{aligned} \text{INFO}_i : [t_i, t_{i+1}) \\ f_i(t) \end{aligned}$$

Let t_n be the time when the newest value is written into the dynamic memory. To implement future predictions in the dynamic memory, the prediction function $f_n(t)$ that is valid after t_n is required (Figure 4.10):

$$\begin{aligned} \text{INFO}_n : [t_n, \infty) \\ f_n(t) \end{aligned}$$

Since each module writes the newest value into the dynamic memory while working, the information should be renewed. Suppose that a module has observed data about information INFO at t_i $(i = 1, \cdots, n)$ and the new $(n+1)-$th data is obtained at t_{n+1}. The module writes it into the dynamic memory to update the information maintained in the dynamic memory as follows:

$$\begin{array}{rcl} \mathrm{INFO}_i : [t_i, t_{i+1}) & & \mathrm{INFO}_j : [t_j, t_{j+1}) \\ f_i(t) & & g_j(t) \\ (i = 1, \cdots, n-1) & \Longrightarrow & (j = 1, \cdots, n) \\ \mathrm{INFO}_n : [t_n, \infty) & & \mathrm{INFO}_{n+1} : [t_{n+1}, \infty) \\ f_n(t) & & g_{n+1}(t), \end{array}$$

where $f_i(t)$ $(i = 1, \cdots, n-1)$ and $g_j(t)$ $(j = 1, \cdots, n)$ denote interpolation functions, and $f_n(t)$ and $g_{n+1}(t)$ prediction functions.

4.3.3.2 Target Object Information

The information of the target object that is exchanged through the dynamic memory should satisfy the following properties:

- The shared information has to be obtained from the object image observed by the perception module.
- To write the information into the dynamic memory as time-series data, the information has to be numerical values.
- The object information is utilized by the action module to control the camera for object tracking[2] . The shared target object information should, therefore, be meaningful for the action module to determine the next pan-tilt-zoom parameters.

In this system, the direction and the scale of the object are regarded as the object information shared by the dynamic memory (Figure 4.11):

Object direction ($(P_{\mathrm{obj}_n}, T_{\mathrm{obj}_n})$): The direction is determined by the projection center of the camera and the centroid of the detected object region in the observed image. This information is required for controlling the pan-tilt angles towards the target object.

Object scale ($(P_{S\mathrm{obj}_n}, T_{S\mathrm{obj}_n})$): The scale is determined by the projection center of the camera and the rectangular window that includes the detected object region in the observed image. This information is required for controlling the zoom parameter. By representing the object size in the observed image as the angular values, the represented values can be consistent even if the zoom parameter is changed.

[2] For a multi-target tracking system, the perception module also requires the object information for object identification. We will address this problem in the next chapter.

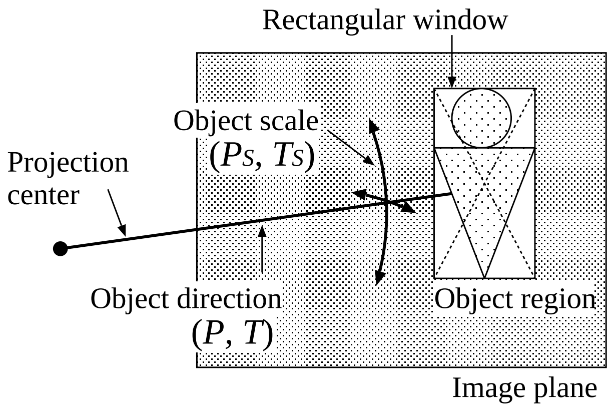


Figure 4.11: Representation of the object information.

4.3.3.2.1 Target Motion Model and Motion Estimation and Prediction using Kalman Filter

In the dynamic memory, the observed information is written as discrete values. Since these values include errors, estimating a reliable value from them is important. We solve this problem by the Kalman filter. The Kalman filter (1) estimates a true value at the current time and (2) predicts a value in the future, from a set of measured values based on the predefined state model.

Motion and measurement models of the target

To employ the Kalman filter for representing the object direction and scale, the state equation of the target motion is required. We model the target motion during a processing cycle of the perception module as follows:

Direction model: The direction of the object is modeled to be a constant angular velocity motion around the projection center of the camera.

Scale model: The varying rate of the object scale is modeled to be a constant.

Since a processing cycle of the perception module is short enough, these assumptions about the target motion are valid.

Let $\left(\dot{P}_{\mathrm{obj}_n}, \dot{T}_{\mathrm{obj}_n}\right)$ and $\left(\dot{P}_{S\mathrm{obj}_n}, \dot{T}_{S\mathrm{obj}_n}\right)$ denote the varying rates (i.e., differential values) of the object direction and scale, respectively. If the system state is represented by

$$\boldsymbol{x}_n = \left(P_{\mathrm{obj}_n}, T_{\mathrm{obj}_n}, P_{S\mathrm{obj}_n}, T_{S\mathrm{obj}_n}, \dot{P}_{\mathrm{obj}_n}, \dot{T}_{\mathrm{obj}_n}, \dot{P}_{S\mathrm{obj}_n}, \dot{T}_{S\mathrm{obj}_n}\right)^{\mathrm{T}}, \tag{4.1}$$

the state equation of the target motion is represented as follows:

$$\boldsymbol{x}_{n+1} = \boldsymbol{F}\boldsymbol{x}_n + \boldsymbol{G}\boldsymbol{\omega}, \tag{4.2}$$

provided that

$$\boldsymbol{F} = \begin{pmatrix} \boldsymbol{I}_4 & \Delta t \cdot \boldsymbol{I}_4 \\ \boldsymbol{0}_4 & \boldsymbol{I}_4 \end{pmatrix}, \quad \boldsymbol{G} = \begin{pmatrix} \boldsymbol{0}_4 \\ \boldsymbol{I}_4 \end{pmatrix}, \quad \boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3, \omega_4)^{\mathrm{T}},$$

where $\boldsymbol{I}_4$: a 4×4 unit matrix, $\boldsymbol{0}_4$: a 4×4 zero matrix, Δt: the interval between consecutive frames, and $\omega_i \; \{i|i = 1, 2, 3, 4\}$: white noise whose average and variance are 0 and $\sigma^2_{w_i}$, respectively. On the other hand, the measured value $\boldsymbol{y}_n = \left(P'_{\mathrm{obj}_n}, T'_{\mathrm{obj}_n}, P'_{S\mathrm{obj}_n}, T'_{S\mathrm{obj}_n}\right)^{\mathrm{T}}$, which includes the error $\boldsymbol{v}$, is represented by

$$\boldsymbol{y}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}, \tag{4.3}$$

provided that

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{I}_4 & \boldsymbol{0}_4 \end{pmatrix}, \quad \boldsymbol{v} = (v_1, v_2, v_3, v_4)^{\mathrm{T}},$$

where $v_i \; \{i|i = 1, 2, 3, 4\}$ denote white noise whose average and variance are 0 and $\sigma^2_{v_i}$, respectively.

Estimation and prediction from measured values

If the motion and measurement models of the target are represented by equations (4.2) and (4.3), the estimated value $\hat{\boldsymbol{x}}_n$ and the predicted value $\bar{\boldsymbol{x}}_{n+1}$ are computed by the following equations:

$$\begin{aligned} \hat{\boldsymbol{x}}_n &= \bar{\boldsymbol{x}}_n + \boldsymbol{K}_n \left(\boldsymbol{y}_n - \boldsymbol{H}\bar{\boldsymbol{x}}_n\right), && (4.4) \\ \bar{\boldsymbol{x}}_{n+1} &= \boldsymbol{F}\hat{\boldsymbol{x}}_n, && (4.5) \\ & && (4.6) \end{aligned}$$

where $\boldsymbol{K}_n$ denotes the Kalman gain:

$$\begin{aligned} \boldsymbol{K}_n &= \widehat{\boldsymbol{P}}_{n|n-1}\boldsymbol{H}^T \left(\boldsymbol{I}_4 + \boldsymbol{H}\widehat{\boldsymbol{P}}\boldsymbol{H}^T\right)^{-1}, \\ \widehat{\boldsymbol{P}}_{n|n} &= \widehat{\boldsymbol{P}}_{n|n-1} - \boldsymbol{K}_n\boldsymbol{H}\hat{P}_{n|n-1}, \\ \widehat{\boldsymbol{P}}_{n+1|n} &= \boldsymbol{F}\widehat{\boldsymbol{P}}_{n|n}\boldsymbol{F}^T + \boldsymbol{Q} \end{aligned}$$

where a 8×8 matrix $\boldsymbol{Q}$ is represented by

$$\boldsymbol{Q} = \left(\begin{array}{cc} \boldsymbol{0}_4 & \boldsymbol{0}_4 \\ \boldsymbol{0}_4 & \begin{array}{cccc} \frac{\sigma_{w_1}}{\sigma_{v_1}} & 0 & 0 & 0 \\ 0 & \frac{\sigma_{w_2}}{\sigma_{v_2}} & 0 & 0 \\ 0 & 0 & \frac{\sigma_{w_3}}{\sigma_{v_3}} & 0 \\ 0 & 0 & 0 & \frac{\sigma_{w_4}}{\sigma_{v_4}} \end{array} \end{array} \right).$$

In the above discussion, we assume that a processing cycle of the perception module (i.e., Δt) is constant. In general, however, Δt varies depending on the observed image. We, therefore, regard Δt as a variable value.

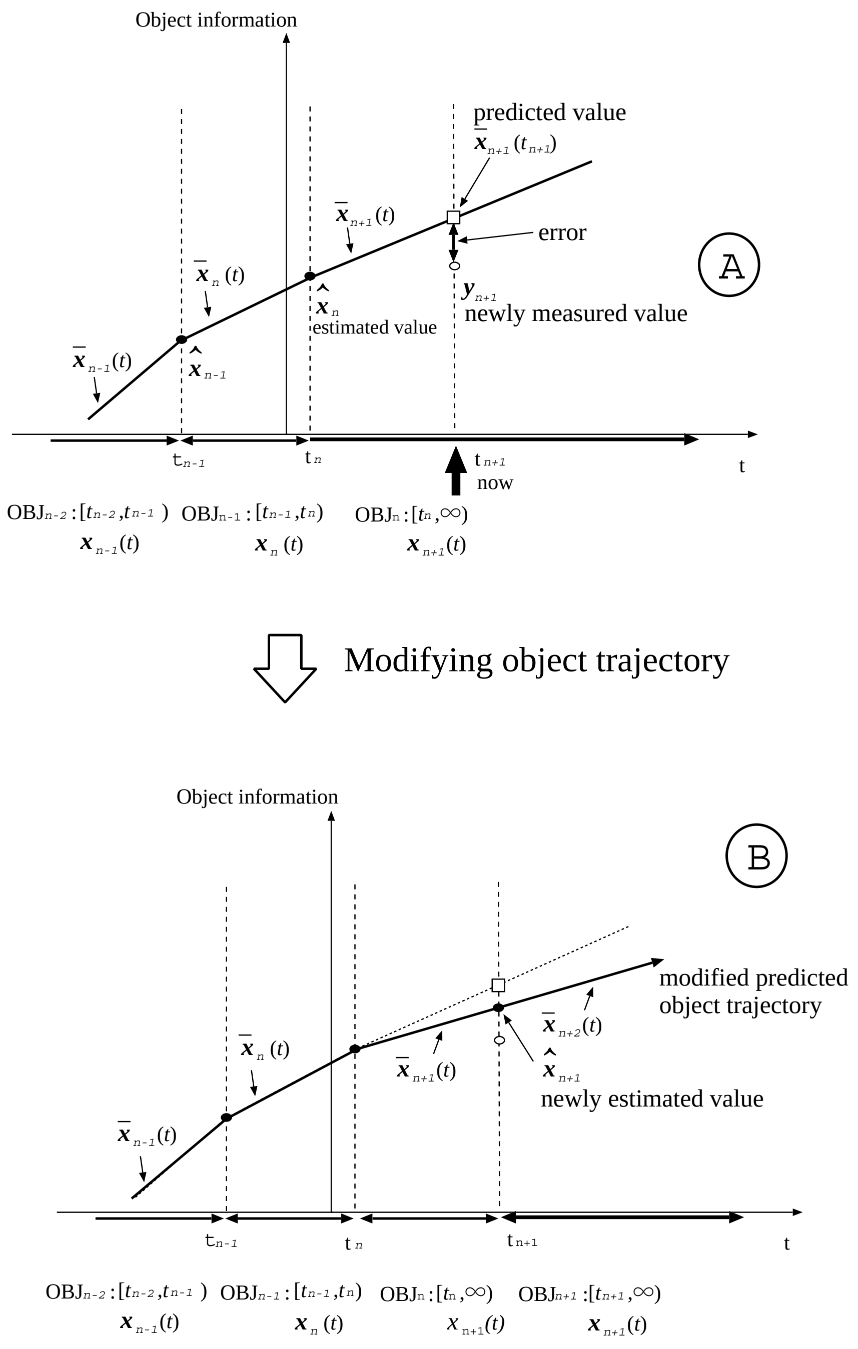


Figure 4.12: Updating the object motion trajectory.

4.3.3.2.2 Describing Target Motion

We here show how the Kalman filter updates the information in the dynamic memory. Let OBJ_i denote the information about the target object at t ($t \in [t_i, t_{i+1})$, where i denotes the ith observation).

Suppose that observations at t_i ($i = 1, \cdots, n$) have been done. In the dynamic memory, the target information modified at t_n (i.e., OBJ_n) is represented as follows:

$$\begin{aligned}\mathrm{OBJ}_n \; : \; & [t_n, \infty) \\ & \left(P_{\mathrm{obj}_n}, T_{\mathrm{obj}_n} P_{S\mathrm{obj}_n}, T_{S\mathrm{obj}_n}, \dot{P}_{\mathrm{obj}_n}, \dot{T}_{\mathrm{obj}_n}, \dot{P}_{S\mathrm{obj}_n}, \dot{T}_{S\mathrm{obj}_n}\right) \; = \; \bar{\boldsymbol{x}}_{n+1}(t).\end{aligned} \tag{4.7}$$

If a new data at t_{n+1} (i.e., $\boldsymbol{y}_{n+1} = \left(P'_{\mathrm{obj}_{n+1}}, T'_{\mathrm{obj}_{n+1}}, P'_{S\mathrm{obj}_{n+1}}, T'_{S\mathrm{obj}_{n+1}}\right)^{\mathrm{T}}$) is detected by the perception module, the dynamic memory rewrites the data (Figure 4.12). Let $\Delta t_{n+1} = t_{n+1} - t_n$. Initially, the predicted value at t_{n+1} is computed from equation (4.6)[3] :

$$\bar{\boldsymbol{x}}_{n+1}(t_{n+1}) = \boldsymbol{F}(\Delta t_{n+1}) \bar{\boldsymbol{x}}_{n+1}(t_n). \tag{4.8}$$

Then, the estimated value $\widehat{\boldsymbol{x}}_{n+1}$ is obtained from equation (4.5):

$$\widehat{\boldsymbol{x}}_{n+1} = \bar{\boldsymbol{x}}_{n+1}(t_{n+1}) + \boldsymbol{K}_{n+1}\left(\boldsymbol{y}_{n+1} - \boldsymbol{H}\bar{\boldsymbol{x}}_{n+1}(t_{n+1})\right). \tag{4.9}$$

From equations (4.8) and (4.9), the information of the target object at and after t_n is updated as follows:

$$\begin{aligned}\mathrm{OBJ}_n : \; [t_n, \infty) \quad \Longrightarrow \quad \mathrm{OBJ}_n : \quad & [t_n, t_{n+1}) \\ & \bar{\boldsymbol{x}}_{n+1}(t) \; = \; \frac{\widehat{\boldsymbol{x}}_{n+1} - \widehat{\boldsymbol{x}}_n}{t_{n+1} - t_n}(t - t_n) + \widehat{\boldsymbol{x}}_n, \\ \mathrm{OBJ}_{n+1} : \quad & [t_{n+1}, \infty) \\ & \bar{\boldsymbol{x}}_{n+2}(t) \; = \; \boldsymbol{F}(t - t_{n+1})\widehat{\boldsymbol{x}}_{n+1}.\end{aligned} \tag{4.10}$$

4.3.3.3 Camera Information

4.3.3.3.1 Camera Motion Model

Controlling the pan, tilt and zoom parameters of the camera can be classified into two types, namely specifying the direction/focal-length and the velocity[4] . As mentioned at the beginning of this section, we implement a smooth camera motion. For this motion, specifying the velocity is appropriate because specifying the direction/focal-length forces the camera to suspend its motion after controlling the camera parameters. We, therefore, control the camera parameters by specifying their velocities.

The management of the camera motion history in the dynamic memory involves some complications, because the action module controls the speed of each camera parameter and, at the same time, measures the pan-tilt angles and the zoom of the camera.

[3] In equation (4.7), $\bar{\boldsymbol{x}}_{n+1}(t)$ denotes the predicted value at and after t_n. If $t = t_n$, however, $\bar{\boldsymbol{x}}_{n+1}(t)$ is equal to the estimated value at t_n (i.e., $\widehat{\boldsymbol{x}}_n$). Consequently, $\boldsymbol{x}_{n+1}(t_n) = \widehat{\boldsymbol{x}}_n$.

[4] SONY EVI-G20 accepts both the direction/focal-length and velocity specifications.

First of all, we conducted experiments to model the dynamics of the FV-PTZ camera (EVI-G20) and found that it can be well described by 'the first-order lag and dead time model' and 'the first-order lag model' in the pan-tilt angles and zoom controls, respectively. The following represents the obtained dynamics of the FV-PTZ camera.

Pan-tilt angles

Suppose that the pan-tilt velocities at $t = 0$ are $(\dot{P}_{\text{cam}_0}, \dot{T}_{\text{cam}_0})$ and the pan-tilt velocities' control commands $(\dot{P}_{\text{com}}, \dot{T}_{\text{com}})$ are sent to the camera at the same time. Then, the pan-tilt velocities (denoted by $(\dot{P}_{\text{cam}}(t), \dot{T}_{\text{cam}}(t))$) are represented by

$$
\begin{aligned}
\dot{P}_{\text{cam}}(t) &= \begin{cases} \dot{P}_{\text{cam}_0} & (0 \le t < \tau) \\ \left(\dot{P}_{\text{com}} - \dot{P}_{\text{cam}_0}\right)\left(1 - \exp\left(-\frac{t-\tau}{T}\right)\right) + \dot{P}_{\text{cam}_0} & (\tau \le t) \end{cases} \\
\dot{T}_{\text{cam}}(t) &= \begin{cases} \dot{T}_{\text{cam}_0} & (0 \le t < \tau) \\ \left(\dot{T}_{\text{com}} - \dot{T}_{\text{cam}_0}\right)\left(1 - \exp\left(-\frac{t-\tau}{T}\right)\right) + \dot{T}_{\text{cam}_0} & (\tau \le t) \end{cases}
\end{aligned}
\tag{4.11}
$$

where τ and T denote the dead time and the time constant, respectively. By integrating the above equations, the pan-tilt angles at t (denoted by $(P_{\text{cam}}(t), T_{\text{cam}}(t))$) can be obtained:

$$
\begin{aligned}
P_{\text{cam}}(t) &= \begin{cases} P_{\text{cam}_0} + t\dot{P}_{\text{cam}_0} & (0 \le t < \tau) \\ P_{\text{cam}_0} + \tau\dot{P}_{\text{cam}_0} + (t-\tau)\,\dot{P}_{\text{com}} & \\ +T\left(\exp\left(-\frac{t-\tau}{T}\right) - 1\right)\left(\dot{P}_{\text{com}} - \dot{P}_{\text{cam}_0}\right) & (\tau \le t) \end{cases} \\
T_{\text{cam}}(t) &= \begin{cases} T_{\text{cam}_0} + t\dot{T}_{\text{cam}_0} & (0 \le t < \tau) \\ T_{\text{cam}_0} + \tau\dot{T}_{\text{cam}_0} + (t-\tau)\,\dot{T}_{\text{com}} & \\ +T\left(\exp\left(-\frac{t-\tau}{T}\right) - 1\right)\left(\dot{T}_{\text{com}} - \dot{T}_{\text{cam}_0}\right) & (\tau \le t) \end{cases}
\end{aligned}
\tag{4.12}
$$

where $(P_{\text{cam}_0}, T_{\text{cam}_0})$ denotes the pan-tilt angles at $t = 0$.

The dead time $\tau(= 44[\text{msec}])$ and the time constant $T(= 63[\text{msec}])$ were determined by intensive experiments. Figure 4.13 illustrates a pan angle history; the pan velocity control command ($166°/sec$) is sent to the camera when the pan angle is 20°. The solid line and the dots denote the first-order lag and dead time function and the measured values, respectively. Figure 4.13 shows that the angular dynamics of EVI-G20 is well described as the first-order lag and dead time model.

Zoom

We represent the dynamics of the zooming factor as the history of the view angle. Then, the dynamics of the zooming factor can be described as the first-order lag model. The velocity and position of the view angle at t (denoted by $\dot{Z}_{\text{cam}}(t)$ and $Z_{\text{cam}}(t)$, respectively) are represented as follows:

$$
\dot{Z}_{\text{cam}}(t) = \left(\dot{Z}_{\text{com}} - \dot{Z}_{\text{cam}_0}\right)\left(1 - \exp\left(-\frac{t}{T}\right)\right) + \dot{Z}_{\text{cam}_0}, \tag{4.13}
$$

$$
Z_{\text{cam}}(t) = Z_{\text{cam}_0} + (t)\,\dot{Z}_{\text{com}} + T\left(\exp\left(-\frac{t}{T}\right) - 1\right)\left(\dot{Z}_{\text{com}} - \dot{Z}_{\text{cam}_0}\right), \tag{4.14}
$$

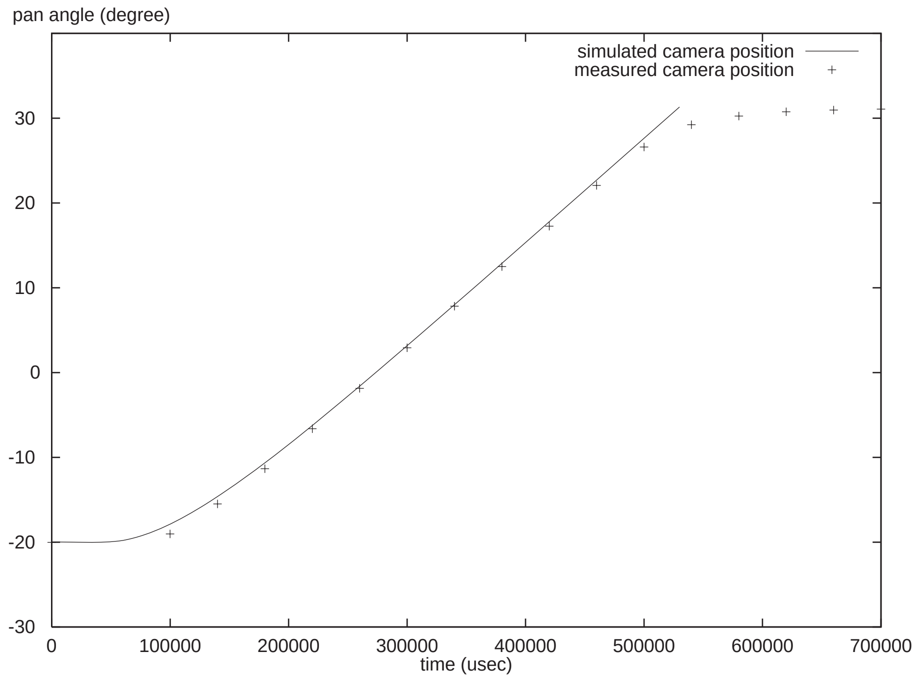


Figure 4.13: Angular dynamics of the FV-PTZ camera.

where $\dot{Z}_{\mathrm{com}}$ denotes the zoom velocity control command sent at $t = 0$, and $\dot{Z}_{\mathrm{cam}_0}$ and Z_{cam_0} denote the velocity and the position of the view angle at $t = 0$, respectively. The time constant $T(= 30[\mathrm{msec}])$ was determined by intensive experiments.

4.3.3.3.2 Describing Camera Motion

Suppose that we have the following information about the camera motion in the dynamic memory (Figure 4.14 [5] A):

$$\begin{array}{rcl} \mathrm{CAM}_n & : & [t_n, \infty) \\ & & P_{\mathrm{cam}_n}(t) \\ & & T_{\mathrm{cam}_n}(t) \\ & & Z_{\mathrm{cam}_n}(t) \quad , \\ & & \dot{P}_{\mathrm{cam}_n}(t) \\ & & \dot{T}_{\mathrm{cam}_n}(t) \\ & & \dot{Z}_{\mathrm{cam}_n}(t) \end{array} \tag{4.15}$$

where t_n denotes the time when the nth camera control is done, and $(P_{\mathrm{cam}_n}(t), T_{\mathrm{cam}_n}(t))$, $Z_{\mathrm{cam}_n}(t)$ and $(\dot{P}_{\mathrm{cam}_n}(t), \dot{T}_{\mathrm{cam}_n}(t))$, $\dot{Z}_{\mathrm{cam}_n}(t)$ denote the position and the velocity of pan-tilt-zoom parameters, respectively. Equations (4.11) and (4.14) express that the camera

[5] Although Figure 4.14 illustrates only the pan angle history, other parameters are also updated and modified by the same method.

velocities converge in the specified control command (i.e., $\dot{P}_{\text{com}}$, $\dot{T}_{\text{com}}$ and $\dot{Z}_{\text{com}}$). That is, the velocity error does not accumulate. The position errors, on the other hand, accumulate while the camera parameters are controlled. To solve this problem, the dynamic memory has to update the camera motion history while correcting the positions of the pan-tilt-zoom parameters.

When the camera speed control command $(\dot{P}_{\text{com}_{n+1}}, \dot{T}_{\text{com}_{n+1}}, \dot{Z}_{\text{com}_{n+1}})$ is sent to the camera at t_{n+1}, CAM_n (equation (4.15)) is changed to equations (4.16) and (4.17) using the camera dynamics in equations (4.11) $\sim$ (4.14) (Figure 4.14 B):

$$\text{CAM}_n \; : \; \begin{array}{l} [t_n, t_{n+1}) \\ P_{\text{cam}_n}(t) \\ T_{\text{cam}_n}(t) \\ Z_{\text{cam}_n}(t) \\ \dot{P}_{\text{cam}_n}(t) \\ \dot{T}_{\text{cam}_n}(t) \\ \dot{Z}_{\text{cam}_n}(t) \end{array} , \tag{4.16}$$

$$\begin{aligned} \text{CAM}_{n+1} \; &: \; [t_{n+1}, \infty) \\ P_{\text{cam}_{n+1}}(t) &= \begin{cases} P_{\text{cam}_n}(t) & (t_{n+1} \le t < t_{n+1} + \tau) \\ P_{\text{cam}_n}(t_{n+1}+\tau) + (t - t_{n+1} - \tau)\,\dot{P}_{\text{com}_{n+1}} & \\ \quad +T\left(\exp\left(-\frac{t-t_{n+1}-\tau}{T}\right) - 1\right)\left(\dot{P}_{\text{com}_{n+1}} - \dot{P}_{\text{cam}_n}(t_{n+1}+\tau)\right) & \\ & (t_{n+1} + \tau \le t) \end{cases} \\ T_{\text{cam}_{n+1}}(t) &= \begin{cases} T_{\text{cam}_n}(t) & (t_{n+1} \le t < t_{n+1} + \tau) \\ T_{\text{cam}_n}(t_{n+1}+\tau) + (t - t_{n+1} - \tau)\,\dot{T}_{\text{com}_{n+1}} & \\ \quad +T\left(\exp\left(-\frac{t-t_{n+1}-\tau}{T}\right) - 1\right)\left(\dot{T}_{\text{com}_{n+1}} - \dot{T}_{\text{cam}_n}(t_{n+1}+\tau)\right) & \\ & (t_{n+1} + \tau \le t) \end{cases} \\ Z_{\text{cam}_{n+1}}(t) &= Z_{\text{cam}_n}(t_{n+1}) + (t - t_{n+1})\,\dot{Z}_{\text{com}_{n+1}} \\ &\quad +T\left(\exp\left(-\frac{t-t_{n+1}}{T}\right) - 1\right)\left(\dot{Z}_{\text{com}_{n+1}} - \dot{Z}_{\text{cam}_n}(t_{n+1})\right) \\ \dot{P}_{\text{cam}_{n+1}}(t) &= \begin{cases} \dot{P}_{\text{cam}_n}(t) & (t_{n+1} \le t < t_{n+1} + \tau) \\ \dot{P}_{\text{cam}_n}(t_{n+1}+\tau) & \\ + \left(\dot{P}_{\text{com}_{n+1}} - \dot{P}_{\text{cam}_n}(t_{n+1}+\tau)\right)\left(1 - \exp\left(-\frac{t-t_{n+1}-\tau}{T}\right)\right) & \\ & (t_{n+1} + \tau \le t) \end{cases} \\ \dot{T}_{\text{cam}_{n+1}}(t) &= \begin{cases} \dot{T}_{\text{cam}_n}(t) & (t_{n+1} \le t < t_{n+1} + \tau) \\ \dot{T}_{\text{cam}_n}(t_{n+1}+\tau) & \\ + \left(\dot{T}_{\text{com}_{n+1}} - \dot{T}_{\text{cam}_n}(t_{n+1}+\tau)\right)\left(1 - \exp\left(-\frac{t-t_{n+1}-\tau}{T}\right)\right) & \\ & (t_{n+1} + \tau \le t) \end{cases} \\ \dot{Z}_{\text{cam}_{n+1}}(t) &= \left(\dot{Z}_{\text{com}_{n+1}} - \dot{Z}_{\text{cam}_n}(t_{n+1})\right)\left(1 - \exp\left(-\frac{t-t_{n+1}}{T}\right)\right) + \dot{Z}_{\text{cam}_n}(t_{n+1}) \end{aligned} \tag{4.17}$$

After sending the command, the action module reads the current pan-tilt angles (P', T') and focal length (Z') at $t'(> t_{n+1})$ from the camera. Then, some discrepancies ($P_{\text{err}_{n+1}}$, $T_{\text{err}_{n+1}}$, $Z_{\text{err}_{n+1}}$) may be found between the predicted and observed pan-tilt angles (Figure

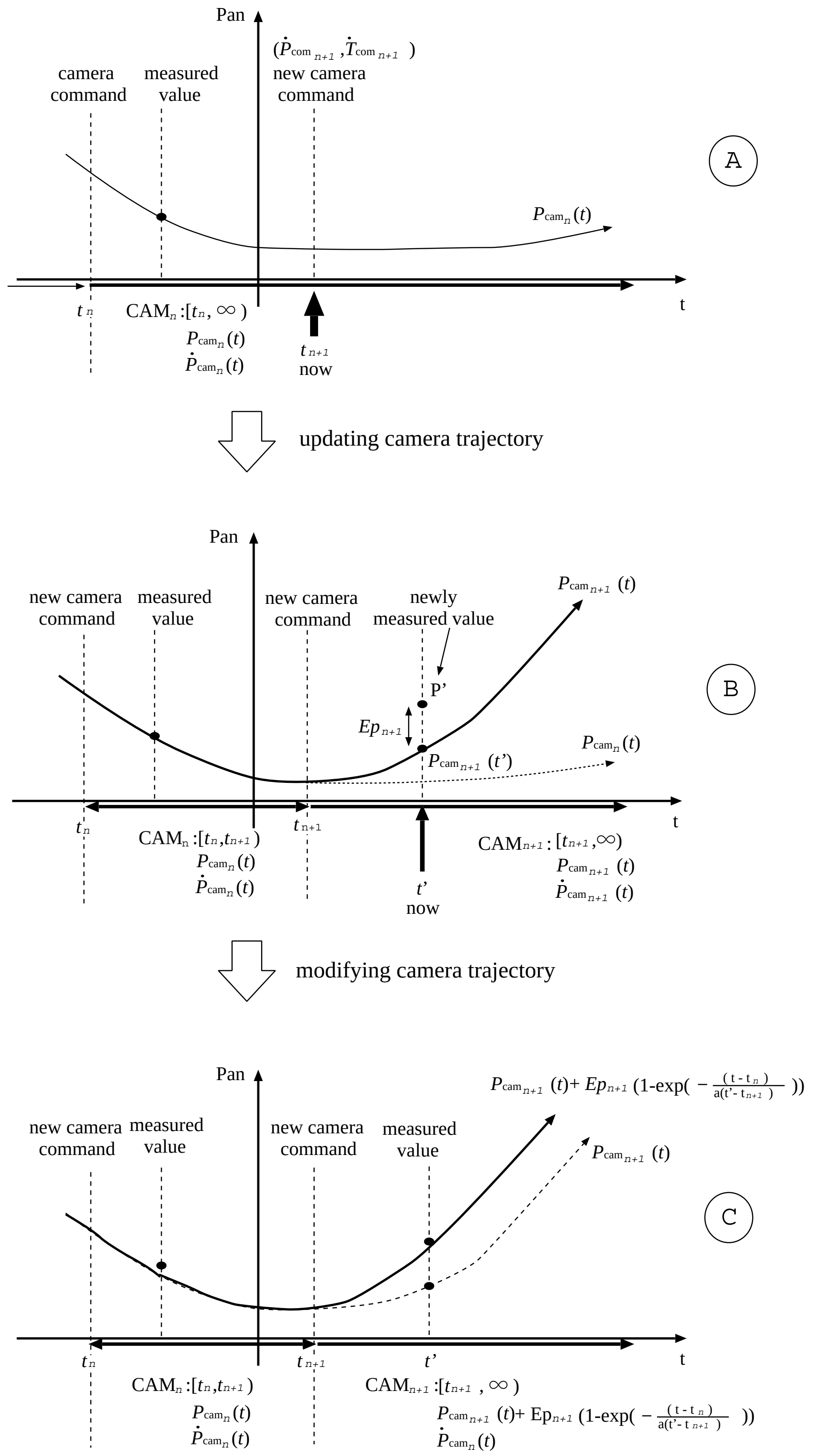


Figure 4.14: Updating an modifying the camera motion history.

4.14 B):

$$\begin{aligned} P_{\mathrm{err}_{n+1}} &= P' - P_{\mathrm{cam}_{n+1}}(t'). \\ T_{\mathrm{err}_{n+1}} &= T' - T_{\mathrm{cam}_{n+1}}(t'). \\ Z_{\mathrm{err}_{n+1}} &= Z' - Z_{\mathrm{cam}_{n+1}}(t'). \end{aligned}$$

To reduce these discrepancies and generate a smooth camera motion trajectory in the dynamic memory, we modify CAM_{n+1} as follows (Figure 4.14 C):

$$\mathrm{CAM}_{n+1} \;:\; \begin{array}{l} [t_{n+1}, \infty) \\ P_{\mathrm{cam}_{n+1}}(t) = P_{\mathrm{cam}_{n+1}}(t) + P_{\mathrm{err}_{n+1}}\left(1 - \exp(-\frac{t-t_{n+1}}{a(t'-t_{n+1})})\right) \\ T_{\mathrm{cam}_{n+1}}(t) = T_{\mathrm{cam}_{n+1}}(t) + T_{\mathrm{err}_{n+1}}\left(1 - \exp(-\frac{t-t_{n+1}}{a(t'-t_{n+1})})\right) \\ Z_{\mathrm{cam}_{n+1}}(t) = Z_{\mathrm{cam}_{n+1}}(t) + Z_{\mathrm{err}_{n+1}}\left(1 - \exp(-\frac{t-t_{n+1}}{b(t'-t_{n+1})})\right) \\ \dot{P}_{\mathrm{cam}_{n+1}} = \dot{P}_{\mathrm{cam}_{n+1}} \\ \dot{T}_{\mathrm{cam}_{n+1}} = \dot{T}_{\mathrm{cam}_{n+1}} \\ \dot{Z}_{\mathrm{cam}_{n+1}} = \dot{Z}_{\mathrm{cam}_{n+1}} \end{array} \quad , \qquad (4.18)$$

where a $(= 1.25)$ and b $(= 0.22)$ denote the time constants.

4.3.4 Perception Module: Dynamic Object Detection Method

The perception module repeats the following tasks:

1. Read the current pan-tilt-zoom parameters from the dynamic memory.
2. Generate the background and threshold images from the APS and threshold APS images, respectively.
3. Obtain the subtraction image between the background and observed images, and compare it with the threshold image for object detection.
4. Compute the centroid and the rectangular region of the detected object in the observed image.
5. Write them into the dynamic memory.

To make this background subtraction work well, we have to align the synthesized background image exactly with the observed image. To attain the accurate alignment, in turn, the perception module has to obtain the *current* pan-tilt angles since the camera is moving continuously. This is exactly the place where the dynamic memory plays a crucial role; the pan-tilt angles are measured by the action module and recorded into the dynamic memory, based on which the *current* pan-tilt angles are interpolated or predicted by the dynamic memory to answer the read request from the perception module.

Note that even with the dynamic memory, pixel-wise exact alignment between the background and observed images is hard to attain. The perception module compensates for such a small misalignment (the left column in Figure 4.8): several shifted versions of the observed image are generated and their differences from the background image are computed. The image with the least overall gray level difference is regarded as the result of the background subtraction. The effectiveness and limitations of the comparison between shifted images are described in Section 2.4.3.

Note also that since an image is captured during the camera motion, it is corrupted with motion blurs. To cope with motion blurs, the perception module has to establish object detection taking into account the following two factors:

- The sensitivity of the subtraction should be lowered around high contrast edges in the observed and background images. The threshold image proposed in Section 3.2.2.1 is useful for suppressing any sensitive subtraction around edges.
- The image should be captured when the camera moves slow enough. To determine the proper capturing timing, the perception module has to inquire of the dynamic memory about not only the pan-tilt angles but also the camera speed.

With the above fine image alignment and sensitivity control, a silhouette of the object can be stably extracted.

4.3.5 Action Module: Prediction-Based Camera Control Method

4.3.5.1 View Direction Control

The action module controls the view direction of the camera to capture the object image at the image center. We first present the control method based on the PID control method.

PID method

Proportional-Integral-Derivative (PID, in short) control is a major control scheme. The controller, which consists of three terms (namely, the proportion, integral, and derivative terms), examines any instantaneous error between the process value and the set point. The proportional term causes a larger control action to be taken for a larger error. The integral term adds to the control action if the error has persisted for some time and the derivative term supplements the control action if the error is changing rapidly with time.

Practical view-direction controlling steps for object tracking based on the PID method are as follows.

1. Let $(P_{\mathrm{cam}}(t_n), T_{\mathrm{cam}}(t_n))$ be the camera pan-tilt angles read by the action module at t_n, where n denotes the nth control cycle.
2. Read from the dynamic memory the location of the target object at t_n. Let $(P_{\mathrm{obj}}(t_n), T_{\mathrm{obj}}(t_n))$ be the location.
3. Compute the displacement:

$$\begin{aligned} P_{\mathrm{dis}}(t_n) &= P_{\mathrm{obj}}(t_n) - P_{\mathrm{cam}}(t_n), \\ T_{\mathrm{dis}}(t_n) &= T_{\mathrm{obj}}(t_n) - T_{\mathrm{cam}}(t_n). \end{aligned}$$

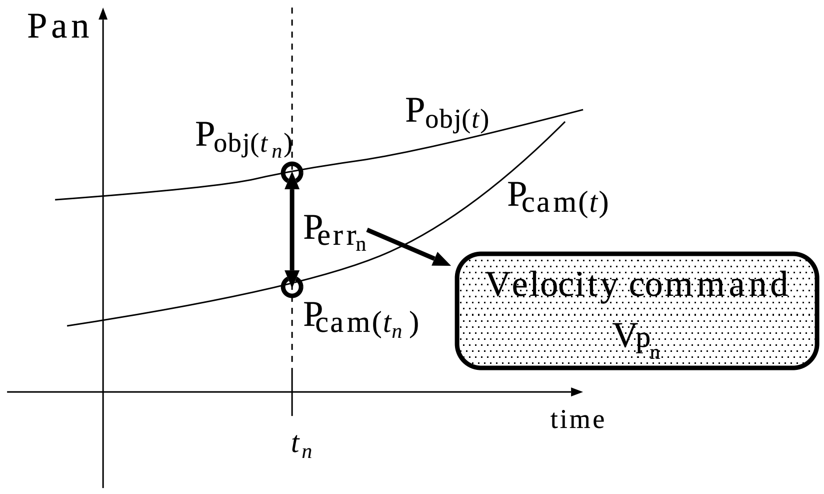


Figure 4.15: View direction control based on the PID method.

4. Determine the camera velocity control command $(V_P(n), V_T(n))$ by the following equations and send it to the camera.

$$\begin{aligned} V_P(n) &= K\left\{\frac{P_{\text{dis}}(t_n)}{\Delta t} + \frac{\alpha}{(\Delta t)^2}(P_{\text{dis}}(t_n) - P_{\text{dis}}(t_{n-1})) + \frac{1}{\beta}\sum_{i=n-10}^{n} P_{\text{dis}}(t_i)\right\}, \\ V_T(n) &= K\left\{\frac{T_{\text{dis}}(t_n)}{\Delta t} + \frac{\alpha}{(\Delta t)^2}(T_{\text{dis}}(t_n) - T_{\text{dis}}(t_{n-1})) + \frac{1}{\beta}\sum_{i=n-10}^{n} T_{\text{dis}}(t_i)\right\}, \end{aligned} \quad (4.19)$$

where $\Delta t = t_n - t_{n-1}$ and K, α, β are the predefined constants.

Here again, the dynamic memory plays a crucial role in realizing stable physical camera control. Firstly, whereas the control timing t_n is determined according to the autonomous dynamics of the action module, the target object location at the specified timing can be obtained from the dynamic memory. Secondly, while the gains for the PID control are determined by purely off-line stand-alone experiments without taking into account the dynamics of the perception module, the same stable camera control can be realized even after integrating the perception and action modules. This is because the dynamic memory guarantees that the autonomous dynamics of a module is not disturbed even if the module is integrated with other modules.

This method controls the pan-tilt angles of the camera as follows:

- If the displacement between the current positions of the target and the camera angle (i.e., $(P_{\text{dis}}(t_n), T_{\text{dis}}(t_n))$) is large, the high-speed control command is sent.
- As the displacement becomes smaller, the velocity gets further suppressed.

Although this control method realizes smooth camera motion, the time spent in controlling the pan-tilt angles is not taken into account. That is, the target motion causes some

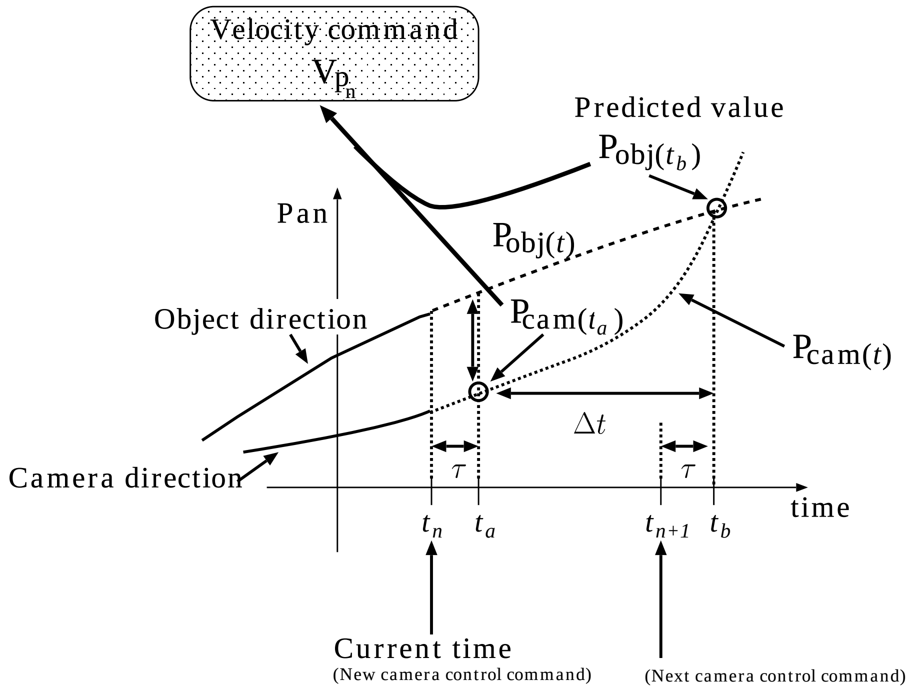


Figure 4.16: Prediction-based view direction control.

displacement between the directions of the target object and the camera view. This disadvantage is fatal for real-time moving object tracking. Accordingly, the view-direction control method that takes into account both the camera motion and the target motion is required.

Prediction Based Control

Suppose the action module sends the camera control command at t_n. Let τ be the constant latency for the camera to accept the camera control command from the action module, and Δt be a processing cycle of the action module. We first consider Δt also to be constant. Since the camera has the latency τ, the last command $(P_{\mathrm{cam}_{n-1}}(t), T_{\mathrm{cam}_{n-1}}(t))$ is actually valid until $t_n + \tau$, and the newest command changes the camera action from $t_n + \tau$. The action module, therefore, determines the camera control command so that the displacement between the target direction and the camera angle at $t_a(= t_n + \tau)$ vanishes at $t_b(= t_n + \tau + \Delta t)$, as illustrated in Figure 4.16. The object direction at t_b (i.e., $P_{\mathrm{obj}}(t_b), T_{\mathrm{obj}}(t_b)$) and the view direction of the camera at t_a (i.e., $P_{\mathrm{cam}}(t_a), T_{\mathrm{cam}}(t_a)$) can be read from the dynamic memory. From these parameters and $(P_{\mathrm{cam}_{n+1}}(t), T_{\mathrm{cam}_{n+1}}(t))$ in equation (4.17), the following equation is obtained:

$$\begin{aligned} P_{\mathrm{cam}_n}(t_b) &= P_{\mathrm{cam}_n}(t_a) + \Delta t P_{\mathrm{com}_n} + T\left(\exp\left(-\frac{\Delta t}{T}\right) - 1\right)(P_{\mathrm{com}_n} - \dot{P}_{\mathrm{cam}_n}(t_a)), \\ T_{\mathrm{cam}_n}(t_b) &= T_{\mathrm{cam}_n}(t_a) + \Delta t T_{\mathrm{com}_n} + T\left(\exp\left(-\frac{\Delta t}{T}\right) - 1\right)(T_{\mathrm{com}_n} - \dot{T}_{\mathrm{cam}_n}(t_a)). \end{aligned} \quad (4.20)$$

To align the view angle of the camera with the object direction at t_b,

$$\begin{aligned} P_{\text{obj}}(t_b) &= P_{\text{cam}_n}(t_b), \\ T_{\text{obj}}(t_b) &= T_{\text{cam}_n}(t_b), \end{aligned} \tag{4.21}$$

have to be satisfied. $(P_{\text{com}_n}, T_{\text{com}_n})$ follows from equations (4.20) and (4.21) that

$$\begin{aligned} P_{\text{com}_n} &= \dot{P}_{\text{cam}_n}(t_a) + \frac{P_{\text{obj}}(t_b) - P_{\text{cam}_n}(t_a) - \dot{P}_{\text{cam}_n}(t_a)\Delta t}{\Delta t + T\left(\exp\left(-\frac{\Delta t}{T}\right) - 1\right)}, \\ T_{\text{com}_n} &= \dot{T}_{\text{cam}_n}(t_a) + \frac{T_{\text{obj}}(t_b) - T_{\text{cam}_n}(t_a) - \dot{T}_{\text{cam}_n}(t_a)\Delta t}{\Delta t + T\left(\exp\left(-\frac{\Delta t}{T}\right) - 1\right)}. \end{aligned} \tag{4.22}$$

These control commands enable the system to capture the target object at the image center without delay.

In the above discussion, Δt is considered to be constant. In general, however, Δt varies in each processing cycle:

- The processing cycle of the action module is summarized as follows:

  1. Read the values from the dynamic memory at t_n.
  2. Compute $(P_{\text{com}_n}, T_{\text{com}_n})$ at t_n.
  3. Send $(P_{\text{com}_n}, T_{\text{com}_n})$ to the camera at t_n.
  4. Inquire the current camera motion of the camera to modify the camera trajectory recorded in the dynamic memory at t_n.

  While the former three tasks can be finished immediately, the action module has to wait the reply from the camera for task 4. The time spent in inquiring for the camera motion varies depending on the states of the camera and network. The action module, therefore, computes the average over the past m times and regards it as Δt:

$$\Delta t = \frac{\sum_{i=n-m+1}^{n} (t_i - t_{i-1})}{m}. \tag{4.23}$$

  Provided that an initial Δt is determined by intensive experiments in advance.

4.3.5.2 Zoom Control

To control the view direction towards the target object, the action module (1) estimates the dynamics of the camera and the target object and (2) analyzes the observed image. In these processes, many uncertain factors are involved:

- **Target motion:** Since the target object moves freely, its motion cannot be precisely estimated. Moreover, as long as the target motion is modeled as the constant angular velocity motion, this model just gives an approximation.

- **Camera motion:** Whereas the camera dynamics is modeled *a priori*, its physical motion can vary depending on its internal mechanical and electronic states.
- **Image analysis:** The computed position of the target object can fluctuate due to noise and varying photographing conditions.

The action module controls the zoom under the existence of these uncertainties. That is, when the degree of uncertainties is low, the action module zooms in to acquire high resolution object images. When some unexpected events happen and the prediction deviates largely from the observed data, on the other hand, the action module zooms out so as not to lose track of the target object . In what follows, we describe such a zoom control method.

All of the uncertainties mentioned above are reflected in the prediction error of the target position, namely, the distance between the image center and the centroid of the target region in the observed image. The action module records this prediction error to learn the degree of uncertainties involved in the task. The action module then determines the optimal zoom parameter based on the learned degree of uncertainties:

Criterion for zooming: From the recorded prediction errors, the uncertainty degree at each observation is computed. The maximum uncertainty degree can then be obtained. The action module estimates the zoom parameter that allows the camera to capture a required silhouette of the target object even in the worst case. This zoom parameter is considered to be optimal to realize both stable tracking and high-resolution capturing.

To employ the above criterion, we first have to evaluate the uncertainty degree from the prediction error. The prediction error in the observed image depends on the following factors:

Target motion: Since we model the target motion as a simple constant angular velocity motion, the difference between the model and the actual target motion incurs the prediction error.

Interval between observations: The longer the observation interval becomes, the larger the prediction error becomes.

Area size of the object: When the object is close to the camera and/or the action module zooms in, the object is projected largely onto the observed image. In this case, even a small movement of the object produces a big variation in the observed image. This makes accurate prediction difficult.

Among these factors, the interval between observations and the area size of the object region can be exactly obtained from the system clock and the detection result, respectively. We normalize the prediction error by these two factors to evaluate the uncertainty degree.

Normalization by the interval between observations:

We consider the prediction error to be in proportion to the interval between observations. The prediction error is, therefore, divided by the interval to normalize the uncertainty degree.

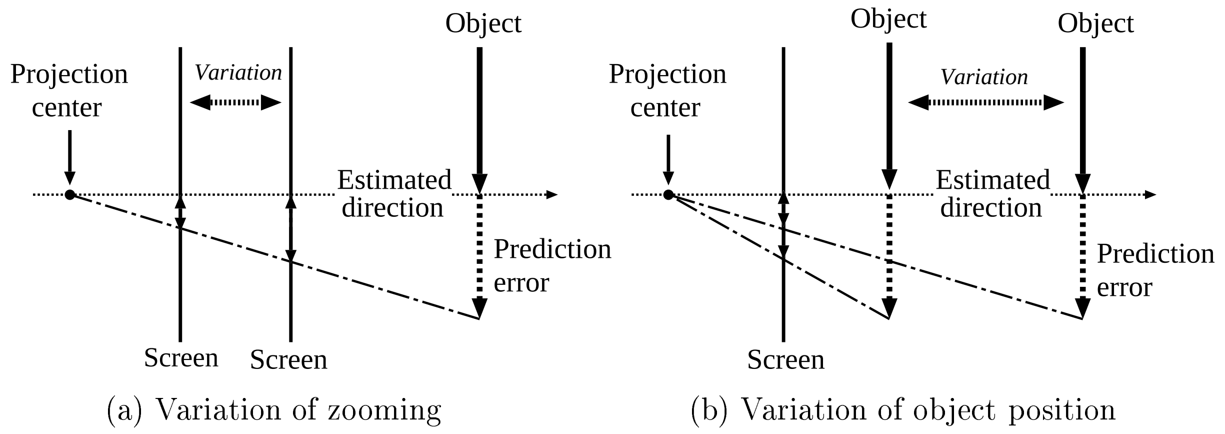


Figure 4.17: Normalization of the estimation error due to the area size of the object region.

Normalization by the area size of the object:

Figure 4.17 illustrates the relationship between the estimation error and its primary factors (i.e., the zooming factor and the distance between the projection center and the object in the scene). Each factor affects (1) the area size of the projected object and (2) the estimation error as follows:

Zooming factor: The area size is linearly in proportion to the square of the focal length, while the estimation error is proportional to the focal length.

Distance between the projection center and the object: The area size and the estimation error are in inverse proportion to the square of the distance and the distance, respectively.

The area size of the target object at t_i (denoted by $AREA(t_i)$) and the estimation error at t_i (denoted by $POS_{error}(t_i)$) are then represented by

$$AREA(t_i) = C_{area} \times \frac{f^2(t_i)}{DIS^2(t_i)}, \tag{4.24}$$

$$POS_{error}(t_i) = C_{pos} \times \frac{f(t_i)}{DIS(t_i)}, \tag{4.25}$$

where C_{area} and C_{pos} are coefficients, $f(t_i)$ the focal length at t_i, and $DIS(t_i)$ the 3D distance between the projection center and the object at t_i. In our system, since the perception module writes (1) the projected size of the detected object (i.e., $(P_{S\mathrm{obj}}, T_{S\mathrm{obj}})$) and (2) the history of the view angle (i.e., Z_{cam}) into the dynamic memory, the action module can read these values at any time. From these values, the rectangular window size of the detected object in the observed image at an arbitrary time can be estimated. The action module regards it as $AREA(t_i)$.

From equations (4.24) and (4.25), the following relationship is obtained between the estimation error and the area size of the object:

$$POS_{error}(t_i) \propto AREA(t_i). \tag{4.26}$$

The prediction error is, therefore, divided by the area size to normalize the uncertainty degree.

Based on the analysis of the prediction error described above, we now define the instantaneous uncertainty degree, $\Delta UD(t_i)$, at the ith observation time t_i as follows:

$$\Delta UD(t_i) = \frac{POS_{\mathrm{error}}(t_i)}{T(t_i) \times \sqrt{AREA(t_i)}}. \tag{4.27}$$

With the above definition of the uncertain degree, the action module determines the optimal zoom factor by the following steps:

1. At the nth observation time t_n, obtain the maximum possible uncertainty degree

$$\Delta UD_{\max} = \max\{\Delta UD(t_n)\}. \tag{4.28}$$

2. Determine the focal length $f(t_{n+1})$ for the next observation so that the maximum possible position error, $POS^{\max}_{\mathrm{error}}(t_{n+1})$, defined below, becomes less than the prefixed threshold.

$$POS^{\max}_{\mathrm{error}}(t_{n+1}) = \Delta UD_{\max} \times (t_{n+1} - t_n)\sqrt{AREA(t_{n+1})}, \tag{4.29}$$

where

$$AREA(t_{n+1}) = \frac{AREA(t_n)}{f^2(t_n)} \times f^2(t_{n+1}).$$

We remark here again that $(t_{n+1} - t_n) = \Delta t$ is determined by equation (4.23).

3. Compute the optimal view angle at t_{n+1} (denoted by Z_{n+1}) from the determined optimal focal length.

4. Substitute Z_{n+1} for $Z_{\mathrm{cam_{n+1}}}(t)$ in equation (4.17). Then, determine the zoom control command $\dot{Z}_{\mathrm{com}}$:

$$\dot{Z}_{\mathrm{com}_n} = \dot{Z}_{\mathrm{cam}_n}(t_n) + \frac{Z_{n+1} - Z_{\mathrm{cam}_n}(t_n) - \dot{Z}_{\mathrm{cam}_n}(t_n)\Delta t}{\Delta t + T\left(\exp\left(-\frac{\Delta t}{T}\right) - 1\right)}. \tag{4.30}$$

4.3.5.3 Camera Control Process

Here, we summarize the tasks of the action module. The action module repeats the following steps:

1. Read the object direction and area size from the dynamic memory.
2. Determine the next camera control command by equations (4.22) and (4.30).
3. Write it into the dynamic memory. (Then, the dynamic memory updates the camera motion history.)
4. Send it to the camera.
5. Inquire of the camera the current pan-tilt-zoom parameters.
6. Write the parameters obtained in step 5 into the dynamic memory. (Then, the dynamic memory modifies the camera motion history.)

4.4 Experiments

We conducted experiments to verify the effectiveness of the proposed system for real-time object tracking. Following are system resources.

Active camera: SONY EVI-G20.

PC: PentiumIII 600MHz $\times$ 2.

The perception and action modules and the dynamic memory were implemented by threads on a PC, and run in parallel.

4.4.1 Suppressing Motion Blurs

We first examined the degree of motion blurs with various camera velocities. Figure 4.18 shows the experimental results:

- The images in the left column were captured while the pan angle was rotating at a constant velocity.
- The images in the middle column were generated from the APP. To generate these images, the camera angle was inquired when the image was captured.
- The images in the right column were subtraction images between the captured and generated images.

Since the observed scene was stationary, no region appeared in a subtraction image if motion blurs were not included in the captured image.

From the subtraction results, we can confirm that motion blurs were not caused unless the rotational velocity exceeded $60°$/sec. Accordingly, we defined the maximum rotational velocity as $60°$/sec.

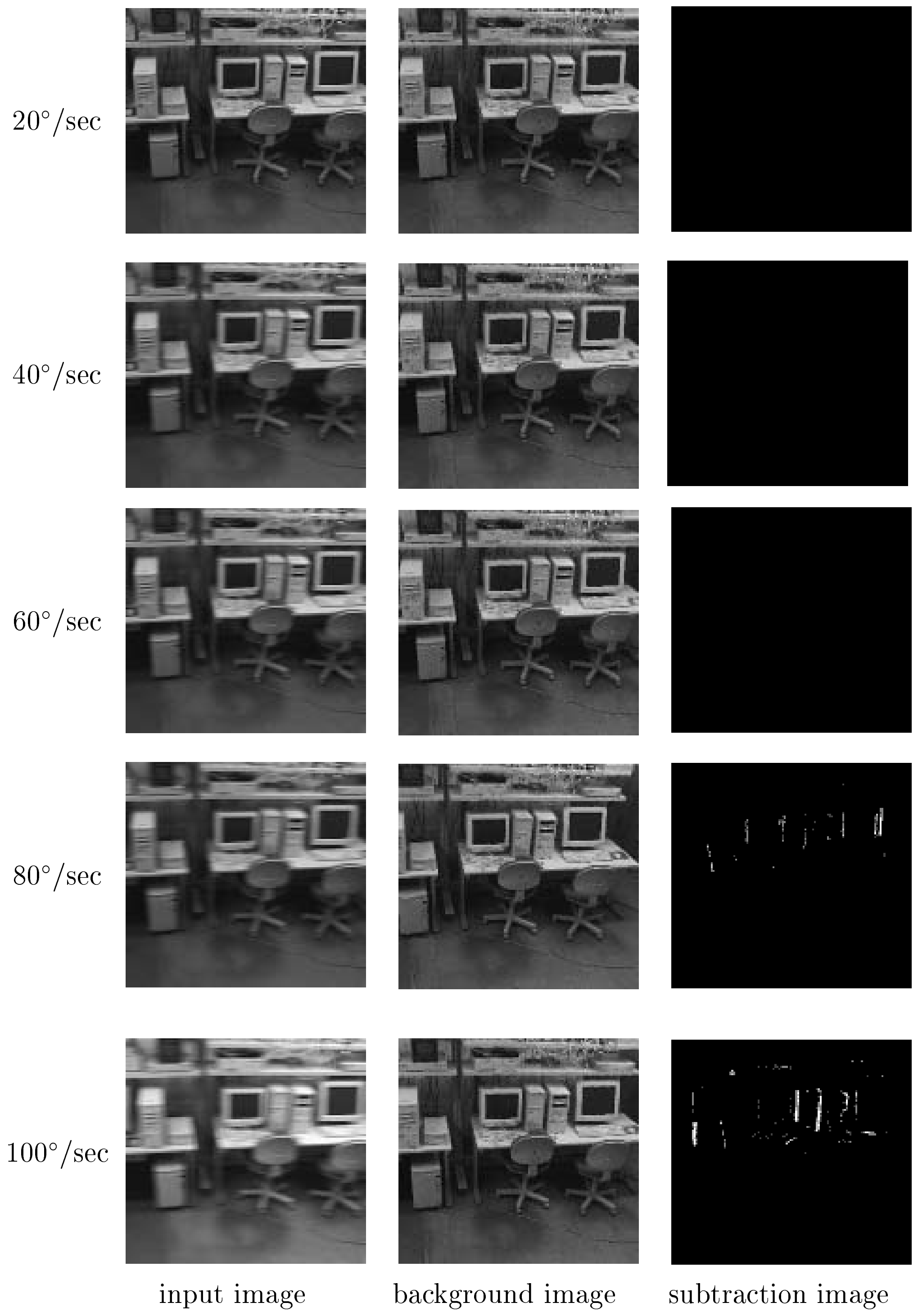


Figure 4.18: Motion blurs in the observed images: these images were captured while the camera was rotating at different speeds.

4.4.2 Tracking Results

A computer-controlled mobile robot moved in the scene. The camera was placed about 2.0[m] above the floor. Figure 4.19 shows an example of observed image sequence (the input and detection images). The size of each image is 320 $\times$ 240 [pixels]. The system captured the images in about 0.1 [sec] intervals on average.

Figure 4.20 shows the read/write access timings from/to the dynamic memory by the perception and action modules. Each vertical line denotes a read/write timing. The upper graph is for the object location data, which was written by the perception module and read by the action module. The lower one is for the pan-tilt camera position data, written by the action module and read by the perception module. We can obtain the following observations.

1. Both modules work asynchronously while keeping their own intrinsic dynamics.
2. The perception module runs almost twice as fast as the action module (about 100[msec/cycle]).
3. Irrespective of these mutually independent dynamics, smooth dynamic information flows through the dynamic memory are realized without introducing any idle time for synchronization.

Figure 4.21 illustrates object and camera motion trajectory data written into and read from the dynamic memory, where

graph 1 (upper right)	:	pan-tilt camera positions measured from the camera,
graph 2 (upper left)	:	pan-tilt camera positions read from the dynamic memory,
graph 3 (lower left)	:	object locations estimated from observed images, and
graph 4 (lower right)	:	object locations read from the dynamic memory.

Each graph includes a pair of trajectories: a larger amplitude is about pan and a smaller amplitude is about tilt. Note that the object locations as well as the camera positions are described in terms of (pan, tilt).

We see the following observations.

1. Comparing graph 1 with graph 2, the data density of the latter is higher than that of the former. This is because the perception module runs faster and hence reads the pan-tilt camera position data more frequently. This also holds true for graph 3 and graph 4.
2. The camera control is well synchronized with the object motion. Figure 4.22 shows the overlapped pan trajectories of graph 1 and graph 4.

4.4.3 Performance Evaluation

4.4.3.1 Effectiveness of Smooth Camera Motion

Here, we point out the effectiveness of smooth camera motion by the following comparative study. Figure 4.23 shows observed images taken by the simple active background

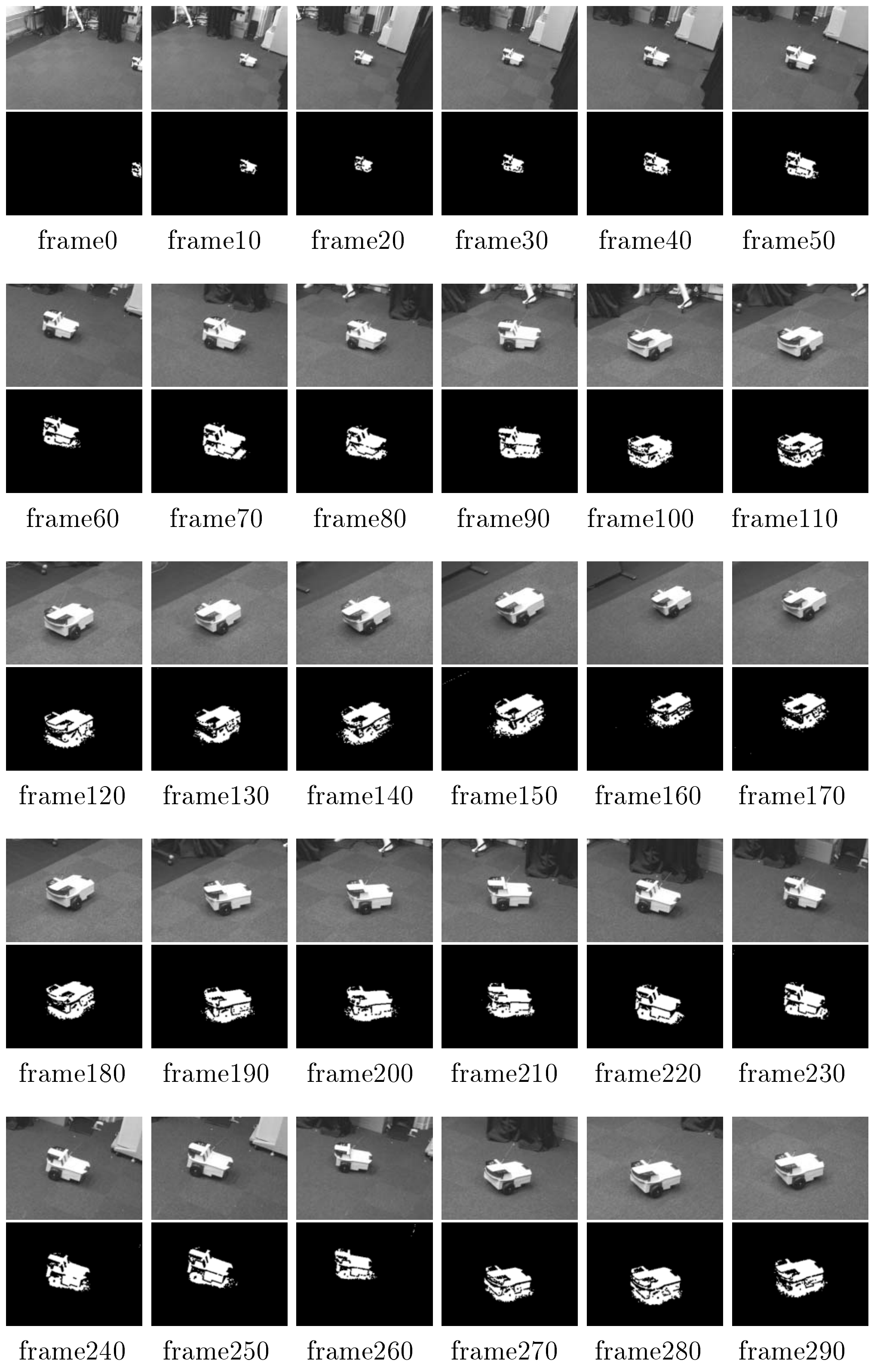


Figure 4.19: Example of observed image sequence taken by the proposed system. Upper: input images, Lower: detection images.

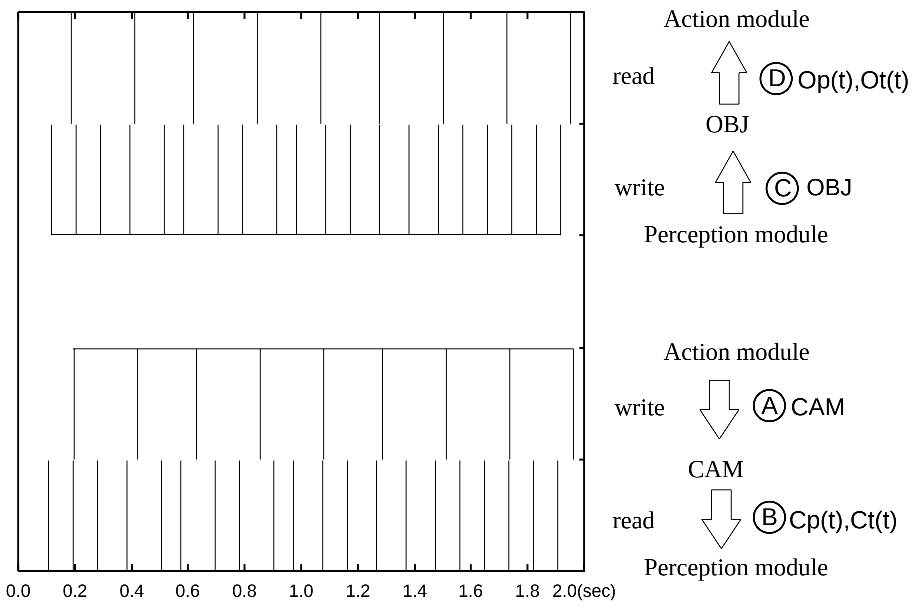


Figure 4.20: Access timing to the dynamic memory by the perception and action modules. Upper object information, Lower: camera information.

subtraction system without modular (perception and action) functions mentioned in the last chapter. We call this system **system A**, and the system with the perception-action modules and the dynamic memory (i.e., proposed system) **system B**. Similarly as with the experiment by system B in Section 4.4.2, a computer-controlled mobile robot moved in the scene. When each system worked, the robot moved along the same trajectory at the same speed. Note that the frame intervals shown in Figure 4.19 and Figure 4.23 differ from one another (system A: 2 frame intervals, system B: 10 frame intervals). Systems A and B captured images in about 0.5 and 0.1 [sec] intervals on average.

The average and the 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 4.1. Table 4.1 indicates that the proposed system improves the stability of tracking with active background subtraction.

In addition, the proposed system drastically shortens the capturing interval and controls the camera smoothly. These advantages not only produce quantitative improvements but also make it easy to understand the trajectory of the object motion. Figure 4.24 shows observed images taken by systems A and B. As we can see, the observed images taken by system B were captured while the view direction of the camera was smoothly changed. System A, on the other hand, captured images intermittently. The intermittent observations caused difficulty in understanding the situation in the scene. In particular, when the camera zoomed in, it was hard to follow the object motion because the observed image included little information about the background scene.

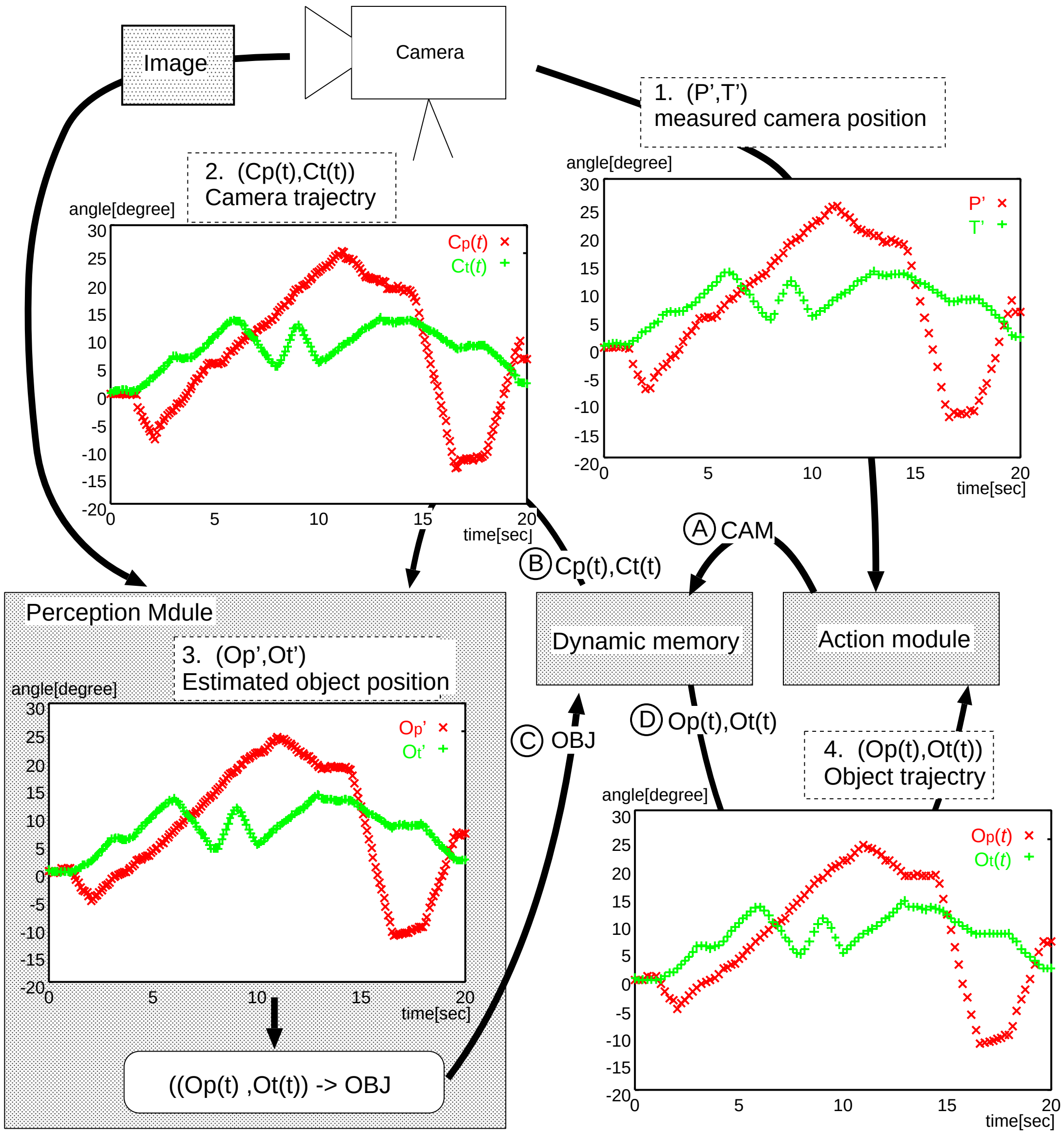


Figure 4.21: Dynamic data exchanged between the perception and action modules. Large amplitude: pan, Small amplitude: tilt. $(O_p(t), O_t(t))$ and $(C_p(t), C_t(t))$ in the figure denote object location $(P_{\rm obj}(t), T_{\rm obj}(t))$ and camera gaze direction $(P_{\rm cam}(t), T_{\rm cam}(t))$ in the text, respectively.

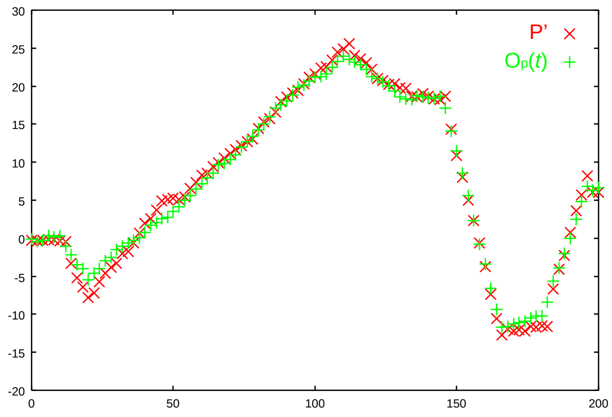


Figure 4.22: Overlapped pan trajectories of graph 1 (camera position measured by the action module) and graph 4 (object position read by the action module).

Table 4.1: (1) Distance 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[pixel]	6.7[pixel]	5083[pixel]	145[pixel]
System B	16.7[pixel]	1.5[pixel]	5825[pixel]	108[pixel]

4.4.3.2 Target Motion Estimation and Prediction using Kalman Filter

In this experiment, $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3, \omega_4)$ in equation (4.2) and $\boldsymbol{v} = (v_1, v_2, v_3, v_4)$ in equation (4.3) were determined so that $(\sigma^2_{w_i}/\sigma^2_{v_i}) = 10^{-11}$ $(i = 1, 3)$ and $(\sigma^2_{w_i}/\sigma^2_{v_i}) = 10^{-13}$ $(i = 2, 4)$. Figure 4.25 illustrates the measured (solid line), predicted (broken line), and estimated (dotted line) values of the target object direction. We can see that the estimation and prediction using the Kalman filter worked well.

To verify the effectiveness of (1) smooth camera motion by employing the dynamic memory and (2) prediction-based tracking, we conducted experiments with the following three systems:

System 1: Simple active background subtraction system without modular functions.

System 2: Real-time active background subtraction system with the dynamic memory using the PID control method.

System 3: Real-time active background subtraction system with the dynamic memory using the prediction-based control method.

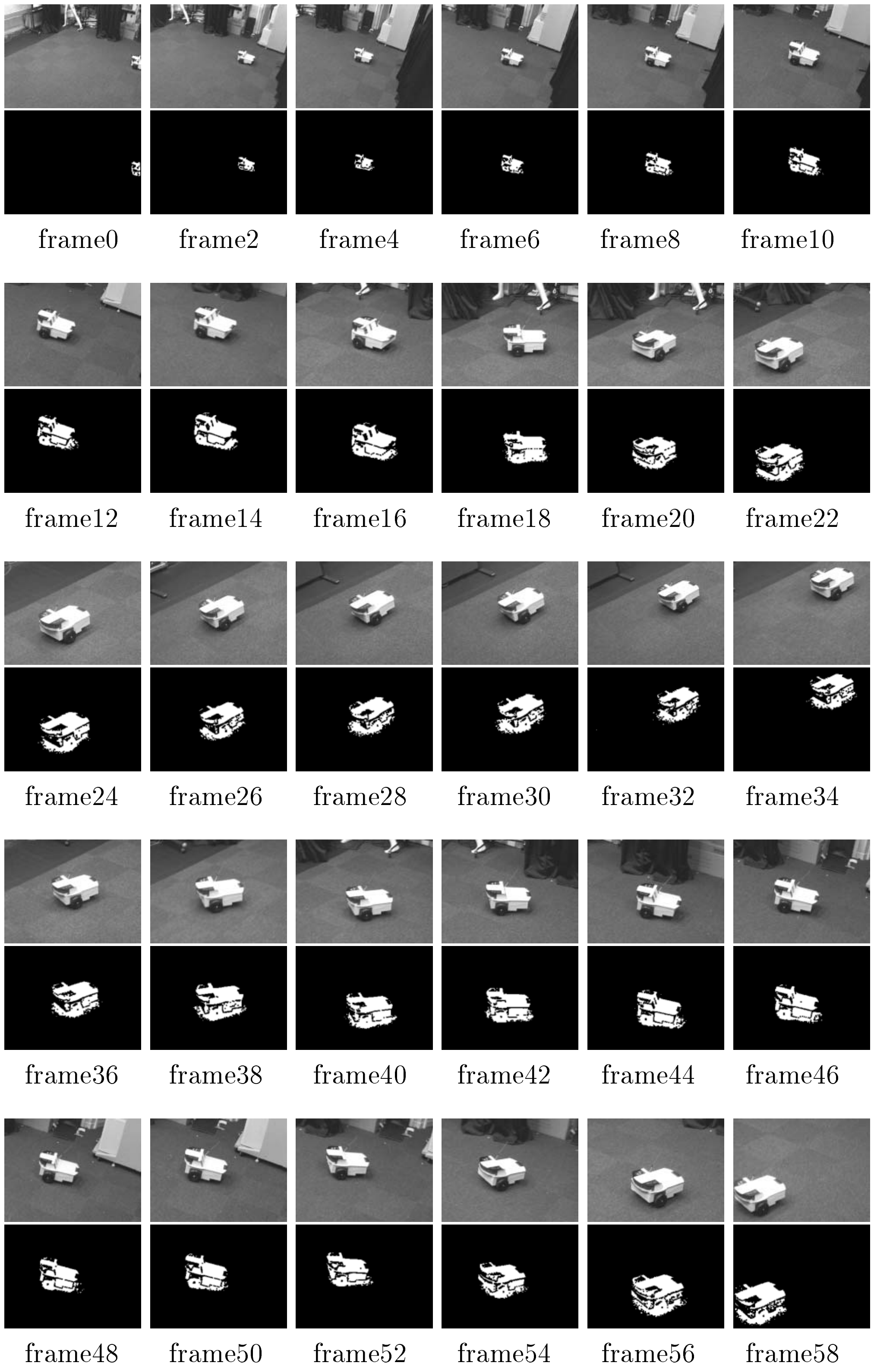


Figure 4.23: Example of observed image sequence taken by the simple active background subtraction system (system A). Upper: input images, Lower: detection images.

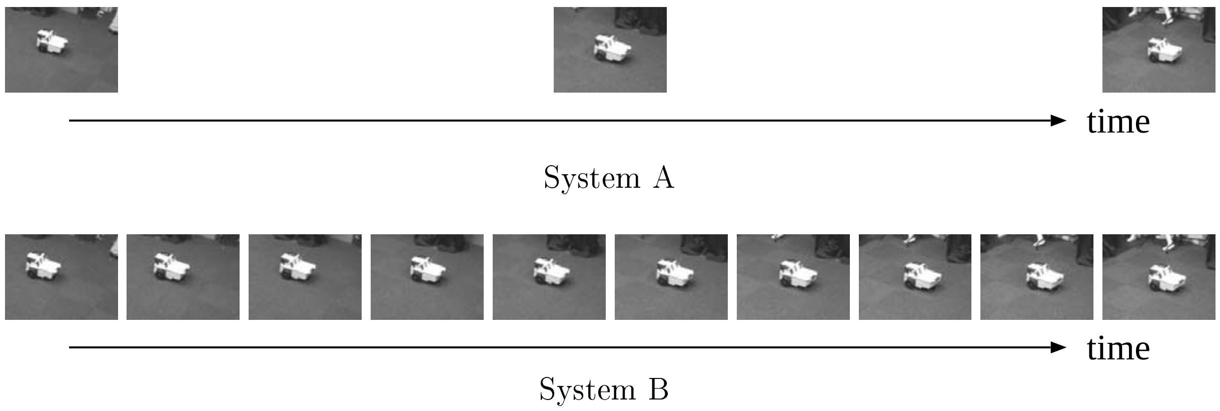


Figure 4.24: Comparison between images taken by systems A and B. Each sequence shows the images observed for the same period (about one second).

Figure 4.26 illustrates the pan angle history of each system. In the figure, non-smooth camera motion controlled by system 1 (small dotted line) incurs a large camera-control delay; the average of the delay was 0.89[sec]. System 1 captured images in about 0.5[sec] intervals on average. This interval was five-fold slower than that of the system with the dynamic memory. For system 2 (broken line), the view direction of the camera followed the object direction after a delay of 0.47[sec]. For system 3 (large dotted line), on the other hand, the delay was shortened to 0.08[sec]. As we can see, the proposed system (system 3) greatly improves the tracking ability.

4.5 Concluding Remarks

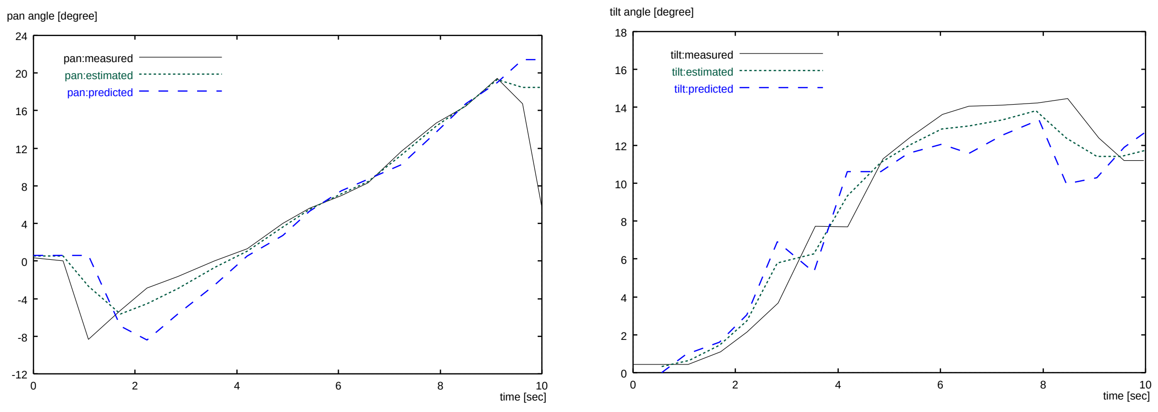
In this chapter, we proposed a real-time moving object tracking system with a dynamic memory. Although the basic scheme for object detection and tracking is the same as that of the active background subtraction method presented in the last chapter, the system has increased flexibility and adaptability by the following properties:

- In the system, the perception and action modules run in parallel and work together for object detection and tracking.

  Perception: Capture the image and detect the object in the captured image.

  Action: Control the camera parameters to track the target object while keeping its silhouette in the image.

- These modules exchange their information with each other through the dynamic memory. The dynamic memory allows the modules (1) to exchange the information asynchronously without disturbing their own intrinsic dynamics and (2) to obtain the available information at any time.



(a) Pan angle (b) Tilt angle

Figure 4.25: Measured, predicted and estimated values of the target object direction.

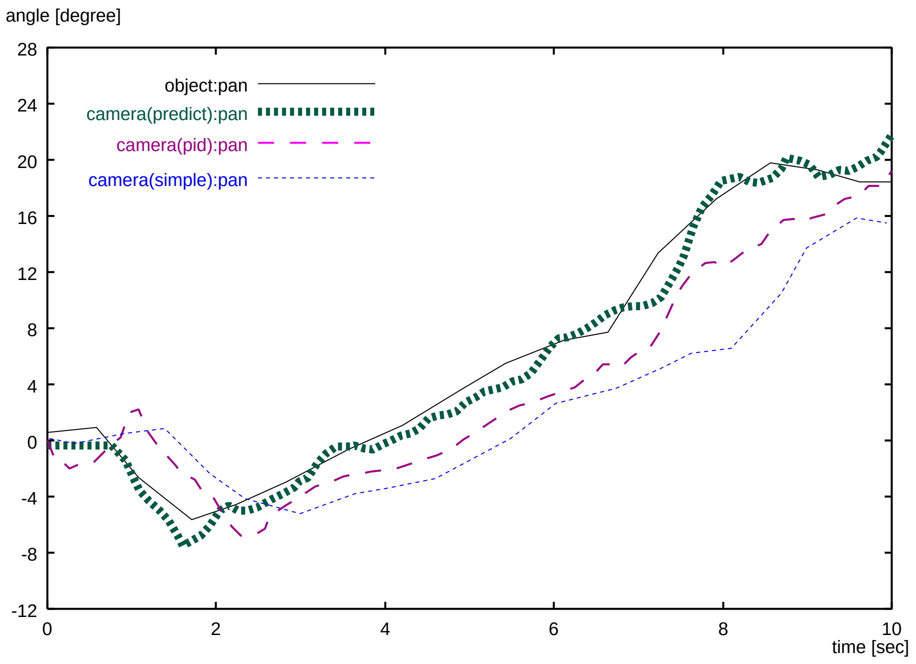


Figure 4.26: Camera control results with the PID and prediction-based methods.

- The camera parameters are controlled based on a sophisticated prediction-based method that takes into account both the target and camera motion for efficient tracking. This method enables the system to keep tracking the target object without any delays.

These properties increase the reactiveness of the system as required for real-time processing.

The practical effectiveness of our system has been demonstrated by several experiments.

Note that in a CDV system, asynchronous interactions play a crucial role in realizing the dynamic integration of the visual perception, action, and communication functions. This is because message exchanges among AVAs are asynchronous in nature. That is, such flexible behaviors are required to make good use of the CDV system's functions. For example,

- Since the computational resources of an AVA are limited, the observation of meaningless images wastes the resources that could possibly be used for other processes such as communication.
- Each AVA should adaptively determine its own dynamics in accordance with the result of its image analysis and interactions with other AVAs.

Therefore, the realization of the dynamic coordinations between the functions of the AVA has generality.

Chapter 5

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

5.1 Real-time Cooperative Multi-target Tracking

In this chapter, we propose a real-time cooperative tracking system that gazes at multiple objects simultaneously. The system consists of *Active Vision Agents* (AVAs, in short), where an AVA is a logical model of the active vision system that is capable for communicating each other through network.

For real-time object tracking by multiple AVAs, we have solved

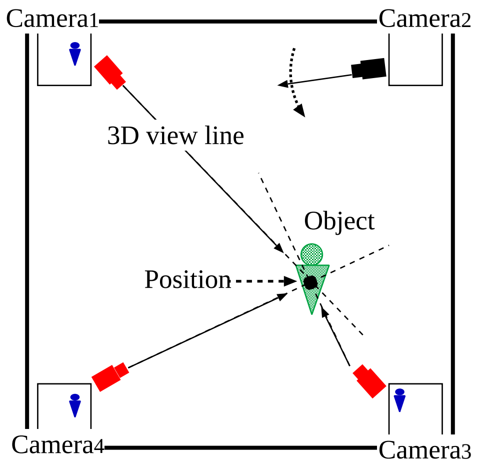
- how to design an active camera for dynamic object detection (chapter 2) and
- how to realize real-time object tracking with an active camera (chapters 3 and 4).

Here, we put our focus upon how to realize a real-time cooperation among AVAs.

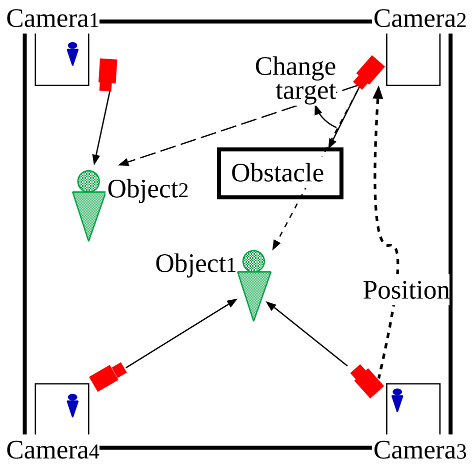
In order to implement the real-time cooperation among AVAs, we propose a three-layered interaction architecture. In each layer, parallel processes exchange different kinds of object information for effective cooperation. To realize real-time information exchange and processing, we employ the dynamic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

By employing multiple pan-tilt-zoom cameras, we aim at designing a system that can not only simply track trajectories of target objects but also acquire their detailed information. The detailed information of the object is required to apply the system to face recognition and volume reconstruction methods and so on.

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



(a) 3D reconstruction



(b) Continuous wide-area observation, Adaptive role assignment

Figure 5.1: Advantages of target tracking by multiple cameras.

5.1.1 Advantages of Multi-camera Tracking System

In general, tracking target objects by multiple cameras allows the system to acquire the following abilities:

3D reconstruction: If the external camera parameters (i.e., the 3D position of all cameras) are calibrated, the 3D information of the object can be reconstructed from the 2D information of the object observed by multiple cameras.

In the case of Figure 5.1 (a), the 3D position of the object is reconstructed from the object information detected by camera$_1$, camera$_3$, and camera$_4$ based on the triangulation.

Continuous wide-area observation: By exchanging the object information between cameras, the system can keep tracking the focused target object without interference of obstacles and other moving objects.

In the case of Figure 5.1 (b), camera$_2$ cannot observe object$_1$ due to an obstacle. Camera$_2$ can, however, gaze at object$_1$ by receiving the 3D position of object$_1$ from another camera.

Adaptive Role assignment: By dynamically assigning the appropriate role to each camera so that each target object is tracked by the camera that is suitable for gazing at, the system can adapt itself to the object motion.

In the case of Figure 5.1 (b), camera$_2$ changes its target object from object$_1$ to object$_2$ because 1) camera$_2$ cannot observe object$_1$ and 2) only camera$_1$ tracks object$_2$. Consequently, all objects are observed by multiple cameras, and the 3D information of all the objects can be reconstructed.

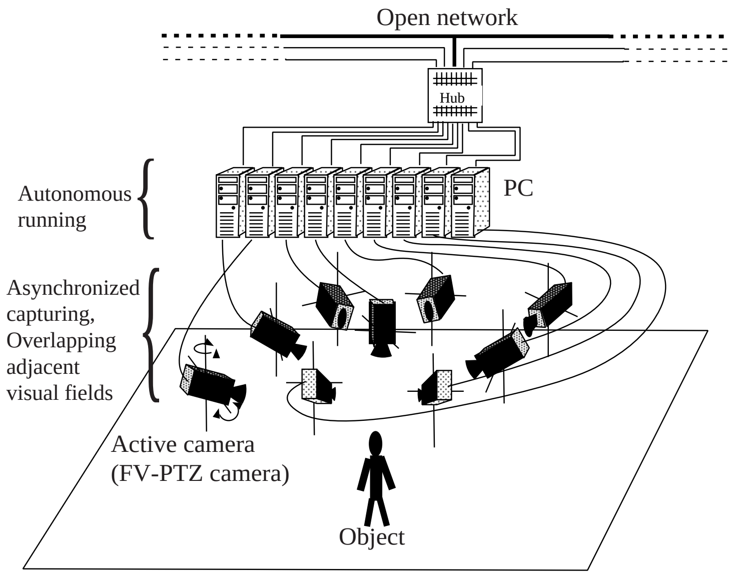

Open network
Hub
Autonomous running
PC
Asynchronized capturing, Overlapping adjacent visual fields
Active camera (FV-PTZ camera)
Object

Figure 5.2: System organization.

The above abilities allows the system to track moving objects persistently in the real world.

5.1.2 System Organization

Our system consists of a group of network-connected computers, each of which possesses an active camera, as illustrated in Figure 5.2. A group of spatially distributed active cameras enables continuous wide-area observation as well as detailed measurement of 3D object information. We impose the following constraints about the camera configuration on the system:

- Visual fields of cameras are overlapping with each other in order to keep tracking a target object in the observation scene without a break. That is, in our system, the area of the observation scene 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. This is because every space has to be observed by multiple cameras to reconstruct 3D information of an object being there.

A pair of network-connected computer and an active camera is called an Active Vision Agent (AVA). We employ an FV-PTZ camera as an active camera. By employing the properties of the FV-PTZ camera, each AVA can detect and track the moving object independently. We have proposed the dynamic object detection and tracking method using the FV-PTZ camera in this thesis.

With the above architecture and functions, each AVA works autonomously while keeping its own intrinsic dynamics and cooperates with other AVAs by exchanging the information through the network. The network is not a special close network (e.g., high-speed PC cluster) but an open network. Each AVA captures images asynchronously because it works autonomously keeping 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 order 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 the AVAs are synchronized. For example, by comparing the time stamps of images captured by different AVAs with each other, the system can identify images taken at the almost same time with one another.

5.1.3 Related Works

Our system is designed as an asynchronized distributed camera system, where each camera corresponds to a single agent. It differs from other similar systems in the following points:

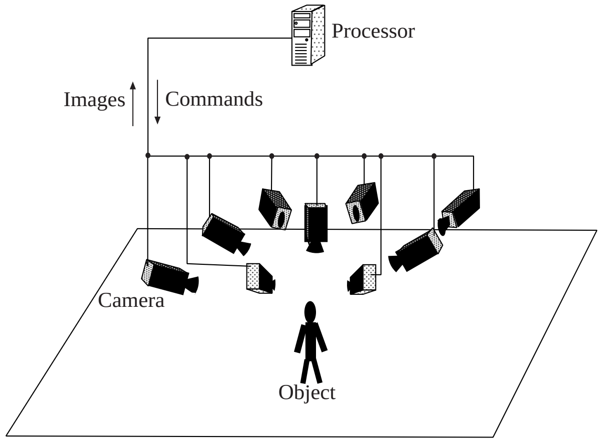
Image capturing timing: While images captured by different cameras are asynchronized in our system, several other multi-camera systems synchronize all cameras by employing a synchronization mechanism. Yonemoto–Arita–Taniguchi [YAT00] proposed a fully synchronized multi-camera system, where all cameras are synchronized by a common sync-generator. In this system, each camera is coupled to a PC and the internal clocks of all the PCs are synchronized. Since each image includes a time stamp when it is captured, the system can identify images taken at the same time by different cameras with one another. In the system with widely distributed cameras, proposed by Kitahara–Ohta–Kanade[KOK00], all cameras are also synchronized by a sync-generator. In this system, each image obtains its time stamp from a time-code generator shared by the entire system.

In the above synchronized multi-camera systems, each camera cannot capture images depending on its own internal state. However, for realizing a flexible multi-camera system adaptable to various tasks, asynchronized image capturing are required as follows:

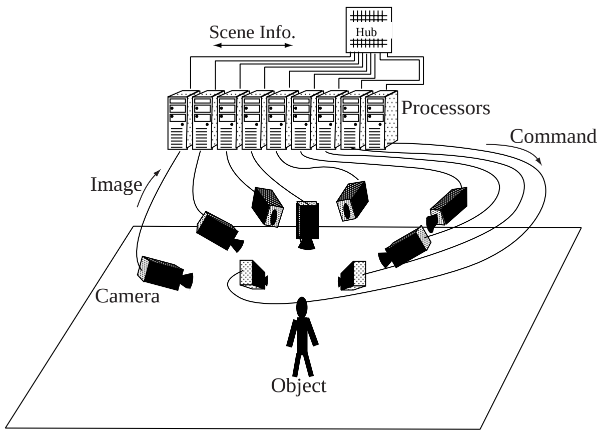
- Each camera should observe the scene keeping its intrinsic dynamics to reactively adapt itself to dynamic situations in the scene.
- Depending on a task given to each camera, its motion dynamics, and so on, the internal states of all the cameras are different from each other. This makes difficult for all the cameras to synchronize with each other.

In our system, therefore, all cameras are controlled independently and observe the scene asynchronously.

Processing and controlling mechanism: Our system consists of multiple agents, each of which corresponds to a computer with an active camera. Each agent analyzes its observed image independently. For the entire system to work as a cooperative



(a) Centralized processing system



(b) Distributed processing system

Figure 5.3: Processing and controlling mechanism.

distributed system, all the agents exchange their analyzed information with each other (Figure 5.3 (a)).

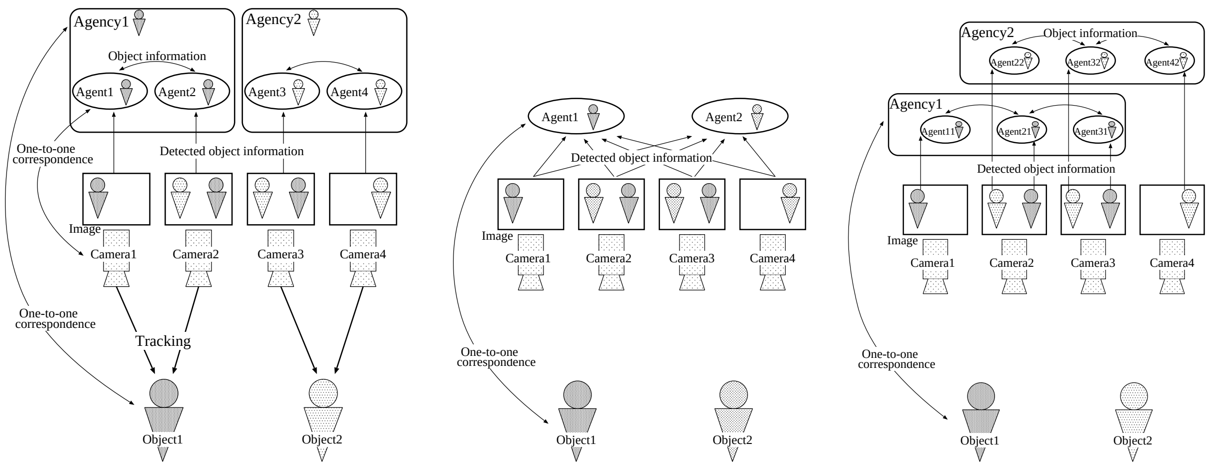
Sogo–Ishiguro–Trivedi[SIT00], on the other hand, proposed the centralized processing tracking system (Figure 5.3 (a)). In this system, a single computer gathers all the captured images through the network. This computer then analyzes them and integrates the object information observed by each camera. This system has the following two problems:

- **Increasing the network-load and computational complexity:** Since images captured by all cameras pass through the network, a huge network-load is caused. Besides the network problem, a single computer has to cope with all complicated situations in the real world by itself. This expands the computational complexity of the process that is conducted by a single processor.
- **Complex design of the system behavior:** We have to design a complex behavior of the entire system taking into account all combinations of predictable situations. The complexity of the system increases dramatically by introducing active cameras; each camera should be dynamically controlled depending on object motions.

These problems result in difficulty in realizing a real-time flexible multi-camera system. This is the reason why we adopt a distributed processing system.

Definition of an agent: In general, there are two kinds of agents:

Software agent: This is a virtual agent without any physical body in the real world. Each agent corresponds to a logical data (e.g., the information of the detected object) in the system.



(a) Our proposed system (b) Nishio–Ohta[NO92] (c) Nakazawa[Nak01]

Figure 5.4: Definition of an agent.

Real-world agent: This is an agent with its own physical body (e.g., an active camera and a mobile robot) that can be controlled by itself. There exists one-to-one correspondence between an agent in the system and the body in the real world. The agent can (1) interact with the real world by utilizing its physical body and (2) be affected by events in the real world through its physical body.

We believe that an intelligent system has to possess its own body to mutually interact with the real world. The concept of such a real-world agent is described in [Mat98]. We, therefore, define an *agent* to correspond to each physical body in the real world. That is, we call (1) the above real-world agent an agent simply and (2) an agent without its body a software agent.

For real-time and reactive processing to cope with complicated dynamic situations in the real world, every agent should be modular with the dynamic memory as well as the real-time object tracking system in chapter 4. In multi-agent system, a communication module is required for the message exchange among agents in addition to perception and action modules. Therefore, the above two kinds of agents consist of the following compositions:

Software agent: The perception and communication modules with the dynamic memory.

Real-world agent: The perception, action and communication module with the dynamic memory.

The difference between the compositions of two agents is the existence of the action module that is required to control the physical body.

Although several object tracking systems with multi-camera and multi-agent systems are reported, most of them employ only software agents. In these systems, each software agent corresponds to the information of the detected object as follows:

Nishio–Ohta[NO92] (Figure 5.4 (b)): They defined an agent to correspond to each object detected by the system. Each agent examines the object information detected by all cameras and keeps tracking its target object.

Nakazawa[Nak01] (Figure 5.4 (c)): An agent is created for all objects detected by each camera . A single camera can, therefore, correspond to multiple agents. Each agent communicates with agents in other cameras and forms an agency (i.e., a group of AVAs) with agents which track the same object in the scene.

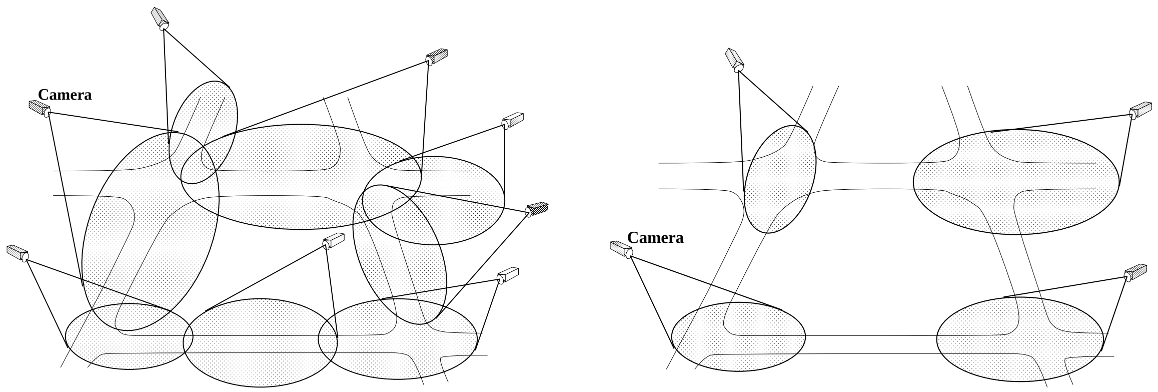
In the above systems, all cameras are shared by software agents, each of which manages the information of each detected object. The above definitions force each software agent to examine the object information detected by all cameras for tracking its target. Besides this technological problem, the above definitions have an essential limitation: multiple software agents may control a camera inconsistently in tracking their target objects (i.e., controlling pan, tilt and zoom parameters), if the system employs active cameras[1] . Since our objective is not only estimating the trajectory of the target object but also acquiring its detailed information, each camera has to control the zoom parameter to obtain a high-resolution object image. This results in difficulty for a camera to gaze at multiple objects simultaneously. Accordingly, the above essential limitation is fatal for realizing the system which we aim at.

In our system, on the other hand, an agent (i.e., AVA) corresponds to a single active camera (Figure 5.4 (a)). That is, each agent monopolizes its own camera. All AVAs can, therefore, control their own cameras to gaze at the target object. As we can see, our definition of the agent has the advantage in that it has the one-to-one correspondence between the agent and the camera.

In our system also, however, the information of each target object should be managed intensively in order to (1) compare the object information detected by cameras and (2) record the information of each object severally. To realize this function, a software agent, which has the one-to-one correspondence with a target object, gathers the object information detected by cameras and manage the information of its target object. 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 (the details of the agency and its software agent will be mentioned later).

Object tracking has a large variety of categories and implies many problems. Although we do not focus on the following problems, they are also major topics on object tracking.

[1] In [NO92] and [Nak01], they employed stationary cameras. This problem, therefore, did not become apparent.



(a) Overlapped configuration (b) Isolated configuration

Figure 5.5: Camera configuration for object tracking.

Camera configuration planning: While we do not consider how to arrange the cameras to effectively gaze at the target objects, there are many researches about the effective camera configuration for realizing the given task. Cowan–Kovesi[CK88] proposed an automatic camera placement method for object feature detection. In this method, each camera is placed so that all surface points be in focus, all surfaces lie within the visual field of the camera, and no surface points be occluded. Tarabanis–Tsai–Allen[TTA95] proposed the MVP sensor planning system. This system determines the optimal settings of the camera and illumination by virtually synthesizing desirable camera views based on geometric models of the environment, optical models of the cameras ,and models of the task.

Tracking with isolated camera configuration: In our system, we impose the constraint about the camera configuration on the system: visual fields of cameras are overlapping with each other in order to keep tracking a target object in the observation scene without a break (Figure 5.5 (a)).

On the other hand, several works on object tracking with isolated camera configuration (Figure 5.5 (b)) are reported. In this camera configuration, the system has to reconstruct the paths taken by moving objects that are temporarily visible from multiple non-overlapping cameras. Distributed Vehicle Monitoring Testbed (DVMT)[LC83] is famous for target motion estimation in a wide-spread area. The system integrates pieces of target motions observed by each distributed sensor for estimating a global target motion. Wada–Tamura–Matsuyama[WTM96] proposed a multi-agent system that solves a global object identification by estimating the optimal combination of local identification results obtained by each agent. Although object identification in a wide area cannot be solved generally based on a common combinatorial optimization method, a spatiotemporal constraint about object motion and path reduces the search space and allows the system to obtain the opti-

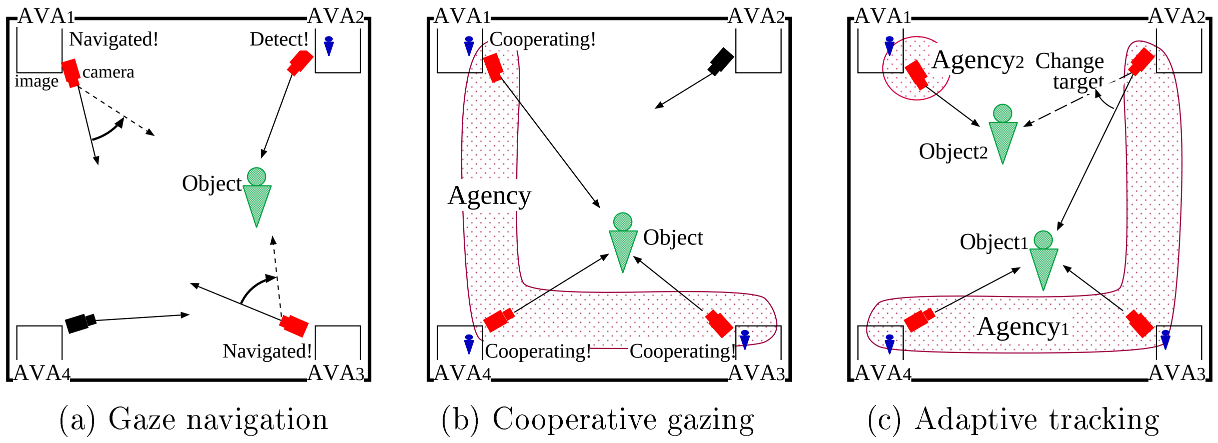


Figure 5.6: Basic scheme for cooperative tracking. A mesh region denotes an agency, and AVAs within the same mesh region belong to the same agency.

mal result. Kettnaker–Zabih[KZ99] presented a Bayesian formalization of this task, where the optimal solution is the set of object paths with the highest posterior probability given the observed data. They showed how to efficiently approximate the maximum a posteriori solution by linear programming. While the former two systems (i.e., [LC83] and [WTM96]) are non-real-time systems, this system tracks a target object in real-time.

5.1.4 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 target objects. Followings are the tasks of the system:

1. Initially, each AVA independently searches for an object that comes into the observation scene.

2. When an AVA detects an object, the AVA examines whether or not the information of the detected object is required to the given task. If the information is required, the AVA regards the detected object as a target object.

3. If the AVA detects the target object, the AVA navigates the gaze of other AVAs towards the target object as illustrated in Figure 5.6 (a).

4. An AVA, which is required to gaze at the target object by another AVA, decides whether it accepts the navigation or continues its current role depending on the situation.

5. AVAs, all of which gaze at the same object, keep tracking the focused target object cooperatively as illustrated in Figure 5.6 (b). A group of AVAs that track the same object is called an *Agency.* In our system, there exists the one-to-one correspondence between an agency and a target object in the scene.

6. Depending on the target motion, each AVA dynamically changes its target object as illustrated in Figure 5.6 (c).

7. When the target object gets out of the scene, the AVA decides whether it searches for an object again or tracks another target object that is tracked by other AVAs depending on the situation.

5.1.5 Issues in Real-time Cooperative Multi-target Tracking

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 discriminate between multiple objects in the scene.

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 a new object even while tracking target objects.
2. For each target object to be tracked by the AVA that is suitable for gazing at, the system has to adaptively assign AVAs to their target objects in accordance with the target motion.

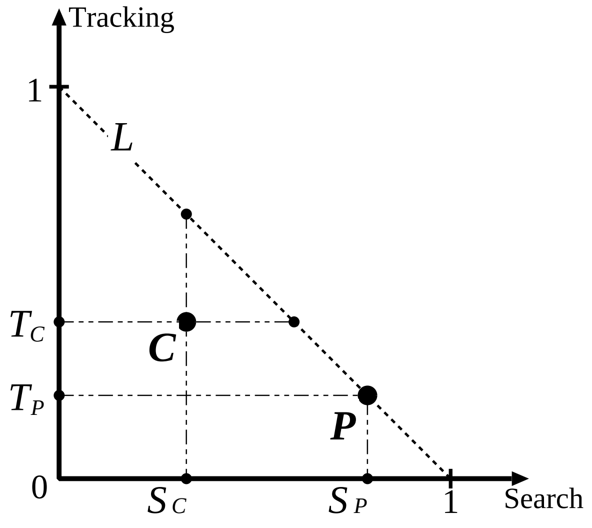
We solve these problems with real-time cooperative communication among AVAs and agencies.

5.2 Task Specification

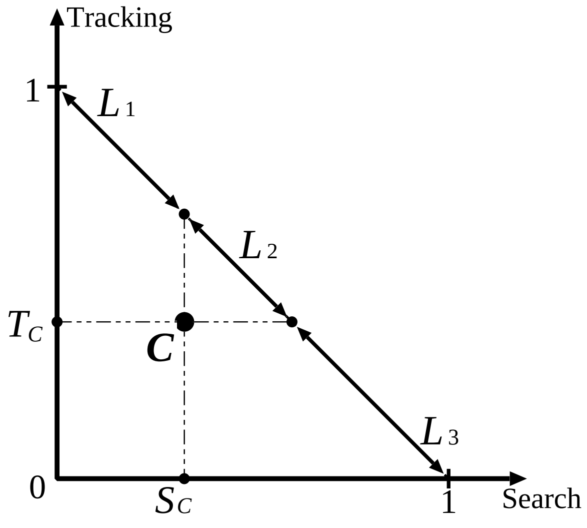
Tracking multiple objects includes a large variety of behaviors depending on the task given to the system. This is because each applied system requires different kinds of object information. We, therefore, design the system that is adaptable to various tasks by specifying parameters.

First of all, the tracking system need to search for an object in the scene. This role is called *Search*[2] . Once a target object is detected, the system then gazes at the target

[2] Hereafter, slanted *search* denotes the role for searching a new object.



(a) Current state, Task-constraint



(b) Three types of the system states

Figure 5.7: System state graph (system state representation with *search* and *tracking*).

object to obtain its information. This role is called *Tracking*[3] . In addition, the system is required to selectively gaze at the object whose information is necessary for the given task because the importance of each object information may be different from each other depending on a given task.

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

Task-constraint: This parameter represents the number of AVAs that execute *search* and *tracking*.

Object-importance: This parameter specifies the priority of each object.

Goal-function: This parameter specifies the aptitude of an AVA for each task.

5.2.1 Task-constraint

The number of AVAs that execute *search* and *tracking* are adjusted in accordance with the given task-constraint. An AVA that searches for a new (undetected) object is called a *Freelancer*-AVA. A freelancer-AVA observes a wide area independently, and undertakes *search*. AVAs that cooperatively track the same object form a group (i.e., agency). Each agency corresponds to a specified object detected in the real world. 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 Figure 5.7. We call this graph a *System State Graph*.

[3] Hereafter, slanted *tracking* denotes the role for tracking a target object.

The horizontal and vertical axes indicate the rates of AVAs that perform *search* and *tracking*, respectively. We call values of the horizontal and vertical axes *Search-level* and *Tracking-level.*

Definition 1 (Search-level and Tracking-level)

$$\text{Search-level} = \frac{\text{The number of AVAs searching an object}}{\text{The total number of AVAs}} \tag{5.1}$$

$$\text{Tracking-level} = \frac{\text{The number of AVAs tracking target objects}}{\text{The total number of AVAs}} \tag{5.2}$$

A domain of each value is, therefore, determined as follows:

$$0 \leq \text{Search-level} \leq 1 \tag{5.3}$$

$$0 \leq \text{Tracking-level} \leq 1 \tag{5.4}$$

$$0 \leq (\text{Search-level} + \text{Tracking-level}) \leq 1 \tag{5.5}$$

That is, a combination of search-level and tracking-level has to be within a triangle determined by the horizontal and vertical axes and the line L in Figure 5.7.

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 Figure 5.7 (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 Figure 5.7 (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 Figure 5.7 (a)) represents the minimum search-level (S_C) and tracking-level (T_C), which the system has to keep while working. 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 system has to, therefore, adjust the current state so that its search-level and tracking-level are not less than those of the task-constraint.*

Followings 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 $S_C > S_P$. Namely, the current state $\boldsymbol{P}$ is on L_1 in Figure 5.7 (b).

Task satisfaction: $T_C \leq T_P$ and $S_C \leq S_P$. Namely, the current state $\boldsymbol{P}$ is on L_2 in Figure 5.7 (b).

Deficiency of tracking-level: $T_C > T_P$ and $S_C < S_P$. Namely, the current state $\boldsymbol{P}$ is on L_3 in Figure 5.7 (b).

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

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

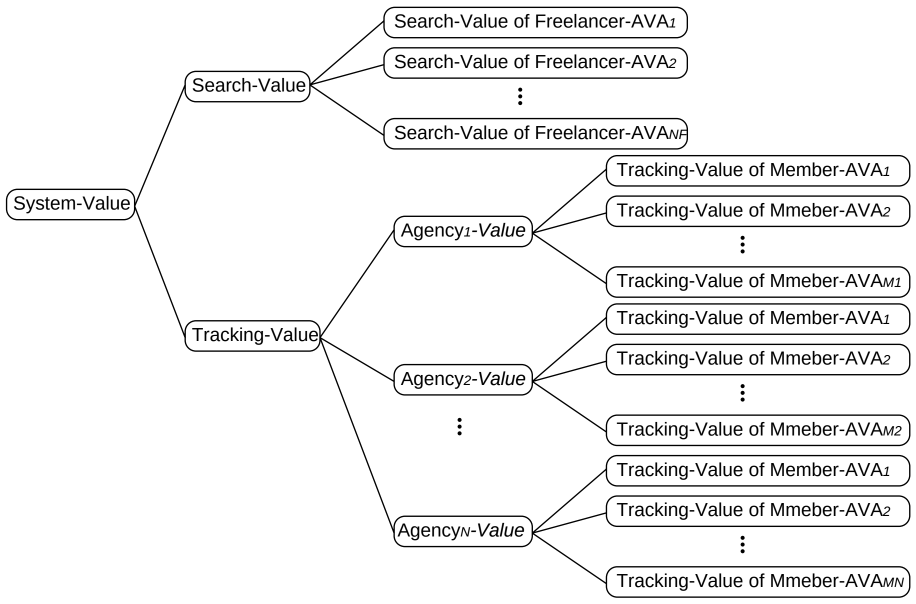

System-Value
Search-Value
Search-Value of Freelancer-AVA1
Search-Value of Freelancer-AVA2
Search-Value of Freelancer-AVANF
Tracking-Value
Agency1-Value
Tracking-Value of Member-AVA1
Tracking-Value of Mmeber-AVA2
Tracking-Value of Mmeber-AVAM1
Agency2-Value
Tracking-Value of Member-AVA1
Tracking-Value of Mmeber-AVA2
Tracking-Value of Mmeber-AVAM2
AgencyN-Value
Tracking-Value of Member-AVA1
Tracking-Value of Mmeber-AVA2
Tracking-Value of Mmeber-AVAMN

Figure 5.8: Hierarchy of the goal-function.

5.2.2 Object-importance

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

Definition 4 (Object-importance I_P) *Let I_P denote the object-importance of the target object 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 object:

$$M_P = (\text{The total number of the AVAs}) \times \frac{I_P}{S}, \tag{5.6}$$

$$S = \sum_{i=1}^{A} I_i, \tag{5.7}$$

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

5.2.3 Goal-function

In our system, each AVA has to decide its own role according to the given task-constraint and object-importance. These two parameters give the system the restriction about the numbers of AVAs that execute each role (i.e., *search* and *tracking* of each target object). Each AVA can, however, freely change its role under this restriction. The AVA changes its role taking into account what we call a goal-function. Each AVA decides its role to increase the value of the goal-function under the restriction of the task-constraint and object-importance. The value of the goal-function is determined by the roles of all the AVAs in the system. That is, the value of the goal-function varies depending on (1) whether each AVA is searching for an object or tracking a target object and (2) which object each AVA is tracking.

The goal-function of our tracking system has a hierarchical structure shown in Figure 5.8.

- **System-value:** The goal-function of the entire system. This function is the sum of the following search-value and tracking-value.
  - **Search-value:** The goal-function of *search*. This function is the sum of all the search-values of freelancer-AVAs.
    - **Search-values of freelancer-AVAs$_{1,\cdots,NF}$:** The goal-function of each freelancer-AVA. The value of this function is determined by the search ability of the freelancer-AVA.
  - **Tracking-value:** The goal-function of *tracking*. This function is the sum of all the agency-values.
    - **Agencies$_{1,\cdots,N}$-values:** The goal-function of the agency that tracks each target object. Each value is the sum of all the tracking-value of member-AVAs belonging to this agency.
      - **Tracking-values of member-AVAs$_{1,\cdots,M}$:** The goal-function of each member-AVA. The value of this function is determined by the tracking ability of the member-AVA.

That is, the value of the goal-function for the entire system is the total sum of the search-values of the freelancer-AVAs and the tracking-values of the member-AVAs. This goal-function can be designed to be adapt itself to the task given by users[4] .

5.2.4 Summary of the task specification

Here, we summarize the dynamic role assignment of AVAs based on the task specification. The task-constraint determines the number of freelancer-AVAs and member-AVAs, each of which performs *search* and *tracking*, respectively. Next, the number of member-AVAs belonging to each agency is determined by the object-importance of each target object.

These two parameters give the system the restriction about the numbers of AVAs that execute each role (i.e., *search* and *tracking* of each target object). Under this restriction, each AVA changes its role to increase the value of the goal-function. To give the system the proper goal-function, each AVA can work in accordance with dynamic situations in the real world (i.e., object motions) depending on the given task.

[4] We give an example in Section 5.5.1.3.

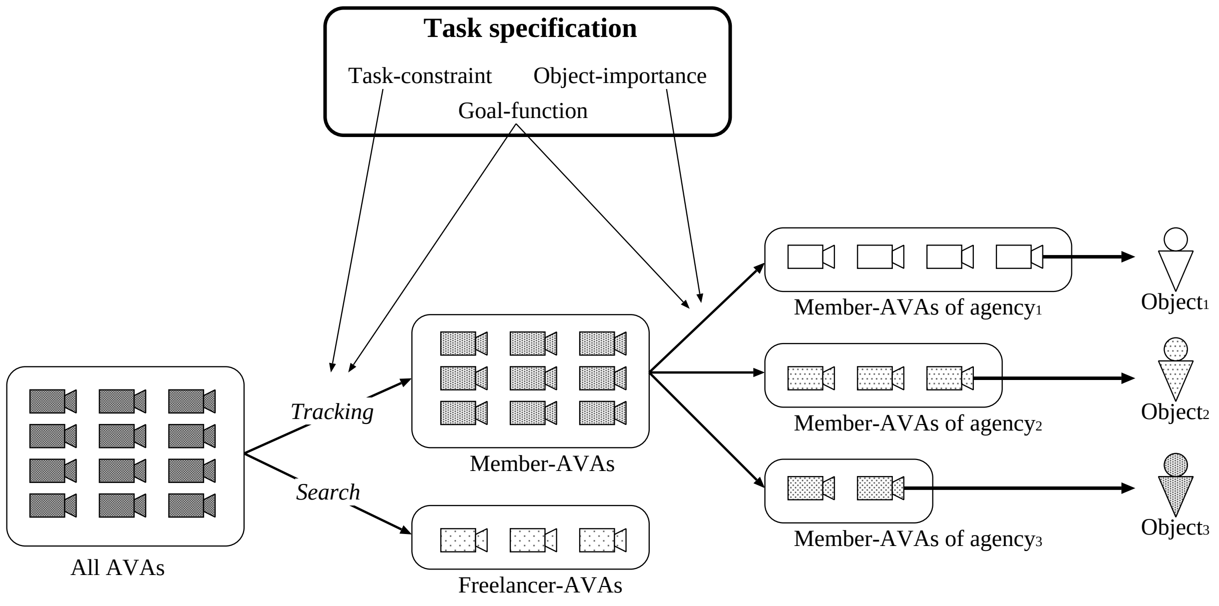


Figure 5.9: Dynamic role assignment of AVAs based on the task specification.

5.3 Dynamic Interaction for Cooperative Tracking

In our system, multiple 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 the complex behavior of the total system through the interactions 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 dynamic situations in the real world.

5.3.1 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 the elements in the system as follows (Figure 5.10):

Intra-AVA layer (the lowest layer): An AVA.

Intra-agency layer (the middle layer): An agency.

Inter-agency layer (the highest layer): The total system.

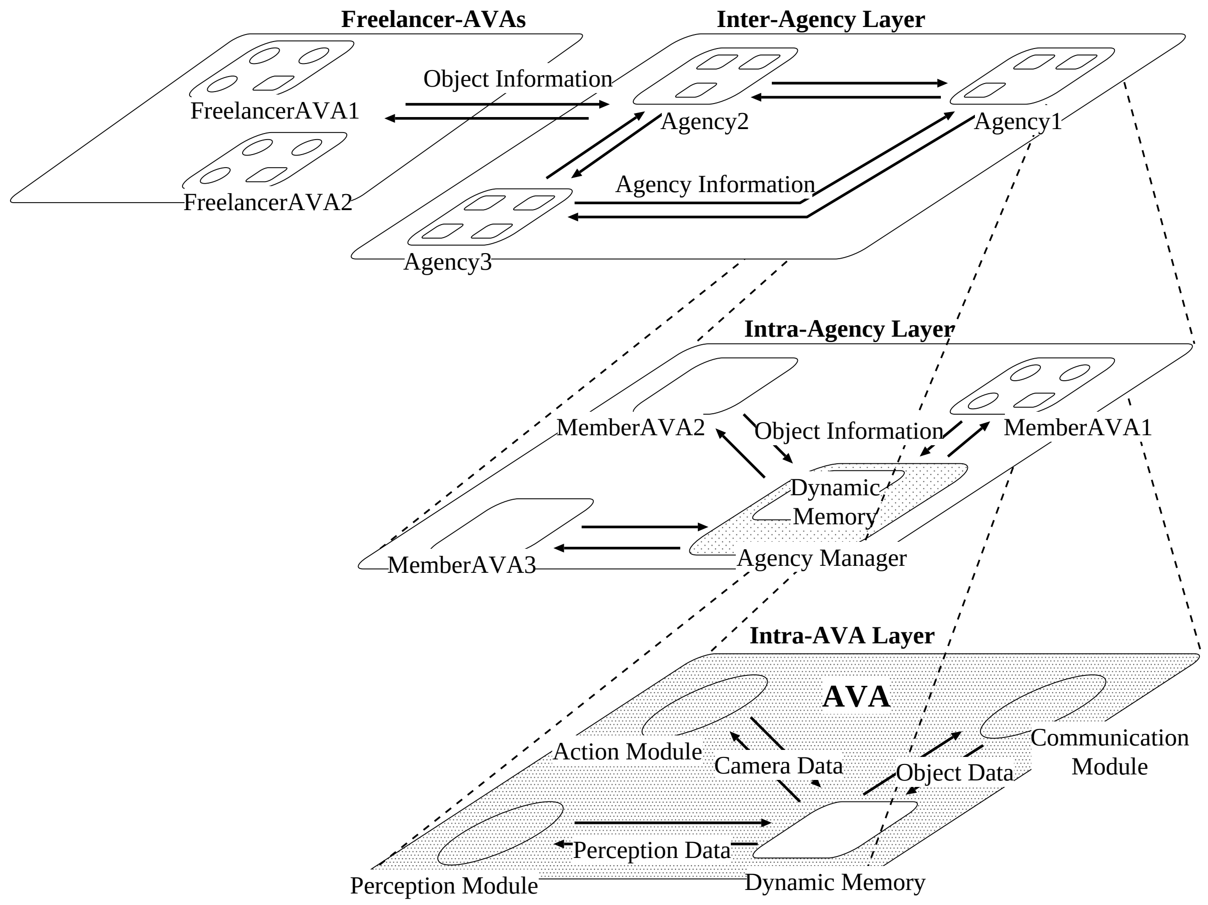


Figure 5.10: Three layers in the system (lowest: intra-AVA, middle: intra-agency and highest: inter-agency layers).

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 activated.

In what follows, we address the interactions in each layer.

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

In the intra-AVA layer, perception, action and communication modules interact with each other through the dynamic memory. That is, an AVA consists of a group of three modules and a dynamic memory. The interactions among modules materialize the functions of the AVA.

5.3.2.1 Perception Module

Followings are the tasks of the perception module:

1. Capture an image.
2. Detect anomalous regions in the captured image.
3. Estimate the number of the detected objects, and obtain the information of each object.
4. Find out its own target object from the detected objects.

While the above tasks 1 and 2 are similar to the tasks of the single-target tracking system proposed in chapter 4, the multi-target tracking system has to establish object identification (the above tasks 3 and 4). Note that a freelancer-AVA performs only the above tasks 1 and 2 because it should observe a wide area without gazing at a specific object. A member-AVA, on the other hand, has to establish object identification to keep tracking its target object, namely the above tasks 3 and 4. In what follows, actual problems for realizing the tasks 3 and 4 are described.

After the perception module detects anomalous regions in the observed image by the background subtraction[5] , it then obtains the following information:

The number of the detected objects: Since there might be multiple objects in the observed image, the perception module has to discriminate between the observed objects.

The information of each object: To represent the object information, the object direction (i.e., the 3D view line from the projection center of the camera to the centroid of the detected object region in the observed image) and the size of the object region (i.e., the pixel number included in the detected object region) are computed. With this information, object identification and camera control are implemented.

To discriminate between the observed objects, the adjoining relations between the detected anomalous pixels are examined. The result of the background subtraction is recorded in the image that is called a *Detection Image*. Let $D(x, y)$ be a pixel value at the image coordinates (x, y) in the detection image:

- If $D(x, y) = 1$, the pixel (x, y) is considered to be the anomalous region.
- If $D(x, y) = 0$, the pixel (x, y) is considered to be the background region.

If $D(x, y) = 1$ and $D(\boldsymbol{n}) = 1$, where $\boldsymbol{n}$ denotes the 8-neighbor pixels of (x, y) (i.e., $\boldsymbol{n} \in \{(x-1, y-1), (x, y-1), (x+1, y-1), (x-1, y), (x+1, y), (x-1, y+1), (x, y+1), (x+1, y+1)\}$), $D(x, y)$ and $D(\boldsymbol{n})$ are considered to be the same object region. Figure 5.11 illustrates the object discrimination based on the adjoining relations.

[5] The object detection method by the perception module are mentioned in chapter 4.

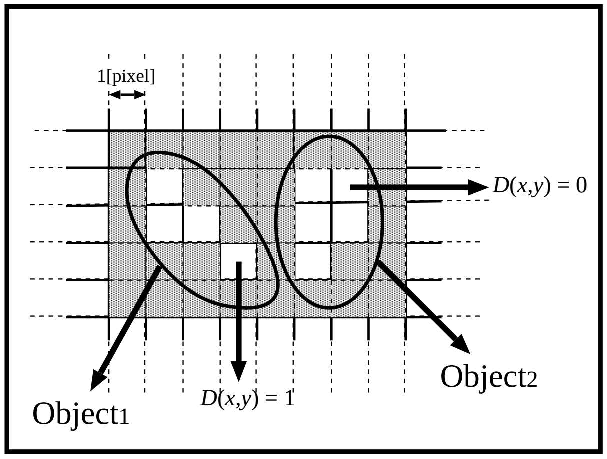


Figure 5.11: Object discrimination based on the adjoining relations between the detected anomalous regions (pixels).

However, since the result of the background subtraction includes image processing errors, the examination for each pixel incurs a misunderstanding about the adjoining relations. To increase the robustness for object discrimination, we decrease the resolution of the detection image as illustrated in Figure 5.12:

1. $C(i,j)$ is a group of $L \times L$ pixels: $C(i,j)$ consists of $\{(x,y)|\ x = (L \times i) + 1, \cdots, L \times (i+1),\ y = (L \times j) + 1, \cdots, L \times (j+1)\}$.

2. Let $R(i,j)$ be a rate of pixels, whose pixel values are 1, in $C(i,j)$: $R(i,j) = N_D(i,j)/(L \times L)$, where $N_D(i,j)$ denotes the number of pixels, whose pixel values are 1, in $C(i,j)$.

3. If $R(i,j)$ is larger than a predefined threshold, the pixel value at (i,j) in the coarse detection image is 1 (i.e., $D(i,j) = 1$ in the coarse detection image).

4. Otherwise, $D(i,j) = 0$ in the coarse detection image.

The object discrimination algorithm mentioned above is applied to the coarse detection image.

Based on the result of the object discrimination in the coarse detection image, the centroid and size of each detected object are obtained in the original detection image. First, each object region in the coarse detection image is superimposed on the original object detection image, and the superimposed region is regarded as the object region in the original detection image (Figure 5.13). The centroid and size of detected object$_p$ are denoted by (x^p_d, y^p_d) and $AREA^p$, respectively. Next, the pan-tilt angles of object$_p$ at the

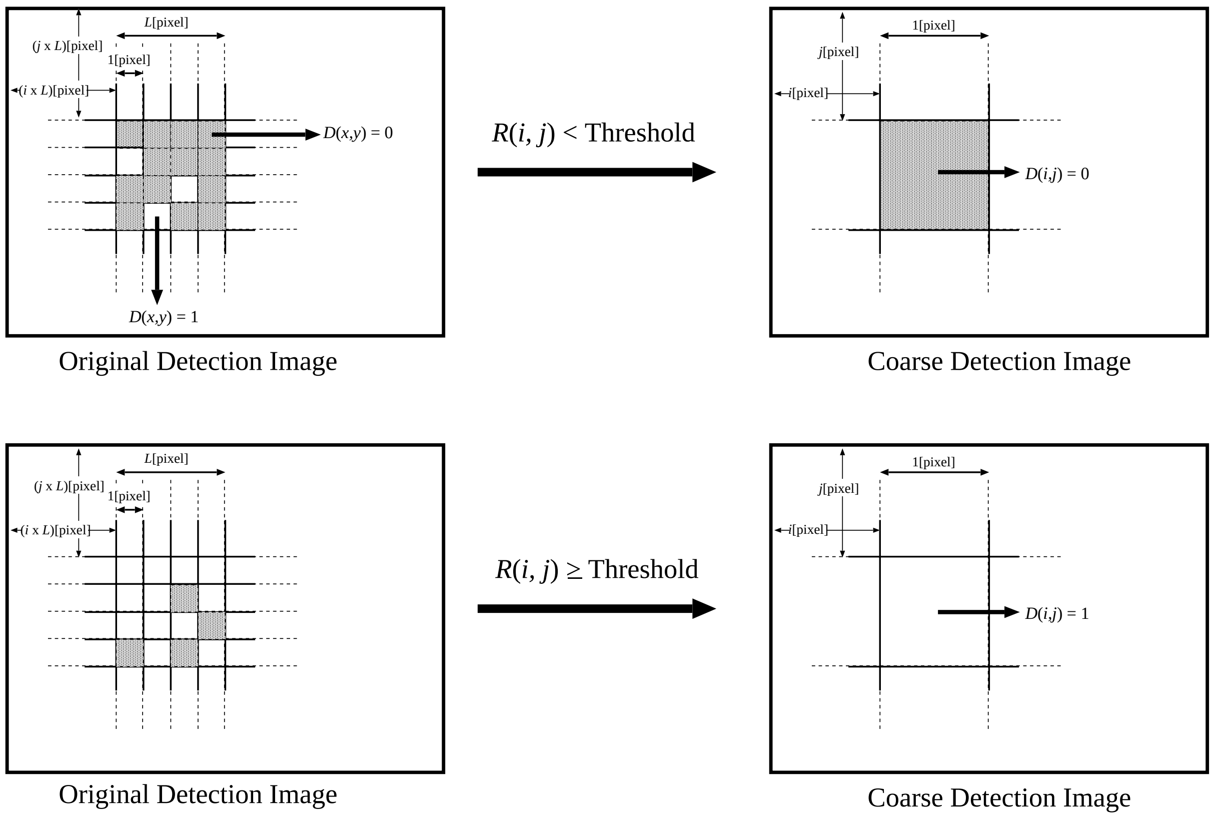


Figure 5.12: Coarsening the detection image.

image capturing time t (denoted by $(P^p_{\rm obj}(t), T^p_{\rm obj}(t))$) is represented by

$$\begin{pmatrix} P^p_{\rm obj}(t) \\ T^p_{\rm obj}(t) \end{pmatrix} = \begin{pmatrix} P_{\rm cam}(t) \\ T_{\rm cam}(t) \end{pmatrix} + \begin{pmatrix} \arctan(x^p_d/f(t)) \\ \arctan(y^p_d/f(t)) \end{pmatrix}, \tag{5.8}$$

where $(P_{\rm cam}(t), T_{\rm cam}(t))$ and $f(t)$ denote the pan-tilt angles and focal length of the camera at t. That is, $(P^p_{\rm obj}(t), T^p_{\rm obj}(t))$ denotes the 3D view direction from the projection center to object$_p$ at t. Hereafter, we call this 3D view direction a *3D view line* $L^p(t)$. The 3D view line $L^p(t)$ and the region size $AREA^p(t)$ are regarded as the information of object$_p$ at t.

When the above object information is obtained at $t+1$, the perception module compares the 3D view lines of objects$_{1,\cdots,N_{t+1}}$ detected at $t+1$ (denoted by $L^1(t+1), \cdots, L^{N_{t+1}}(t+1)$) with those of objects$_{1,\cdots,N_t}$ detected at t for object identification. Let (1) $L^p(t+1)$, where $p \in \{1, \cdots, N_{t+1}\}$, have the shortest angle between $L^q(t)$, where $q \in \{1, \cdots, N_t\}$, and (2) the angle between $L^p(t+1)$ and $L^q(t)$ be shorter than a threshold. The perception module then identifies $L^p(t+1)$ with $L^q(t)$.

Next, this module compares the 3D view line of its target at t (denoted by $\widehat{L}(t)$) with $L^1(t+1), \cdots, L^{N_{t+1}}(t+1)$. Let (1) $L^x(t+1)$ have the shortest angle between $\widehat{L}(t)$,

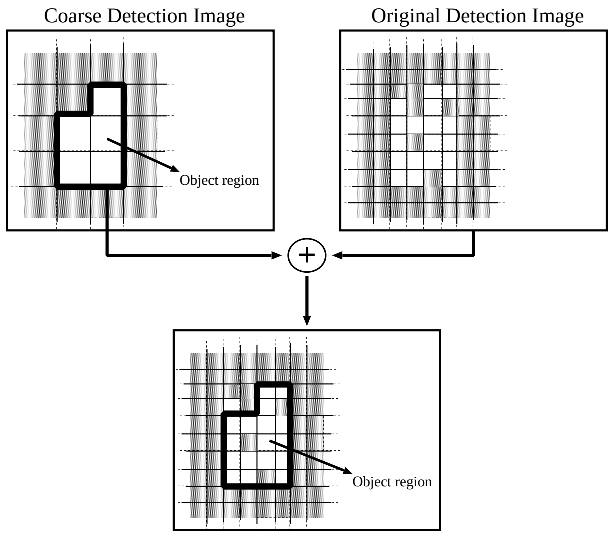


Figure 5.13: Superimposing the detected object region.

where $x \in \{1, \cdots, N_{t+1}\}$, and (2) the angle between $\widehat{L}(t)$ and $L^x(t+1)$ be shorter than a threshold. The perception module considers object$_x$ corresponding to $L^x(t+1)$ as the target object at t+1. Consequently, $\widehat{L}(t{+}1) = L^x(t{+}1)$ and $\widehat{AREA}(t{+}1) = AREA^x(t{+}1)$.

If the interval between t and $t+1$ is small enough and the angle between 3D view lines of multiple objects is large enough, the above identification method is useful. In general, however, this assumption does not always hold. To increase the reliability of object identification, the perception module reads the 3D view line of the target object at $t+1$ from the dynamic memory (future prediction), and compares it with the 3D view lines detected at $t+1$. This procedure makes identification reliable.

The perception module writes the information of all the detected objects at $t+1$ (i.e., $(L^1(t{+}1), \cdots, L^N(t{+}1)$ and $AREA^1(t{+}1), \cdots, AREA^N(t{+}1))$ into the dynamic memory. Note that the information of the non-target objects are also recorded as time-series data in the dynamic memory. This is because object identification by the perception module might be corrected by the agency: when the member-AVA tracks a non-target object due to failing object identification by itself, the agency informs the member-AVA of the target information by sending the 3D position of the target object (denoted by $\widehat{P}(t)$)[6] . In this case, the target information in the dynamic memory is modified so that the detected 3D view line $L_y(t)$, where $y \in \{1, \cdots, N\}$, identified with $\widehat{P}(t+1)$ is regarded as the target information $\widehat{L}(t)$. Therefore, the information of all the detected objects has to be recorded in the dynamic memory.

[6] The details of the process for (1) correcting the target information in the AVA and (2) detecting the wrong object identification by the agency will be mentioned in Section 5.3.2.2 and Section 5.3.3.6, respectively.

5.3.2.2 Action Module

The action module in the freelancer-AVA moves its camera along the predefined trajectory to search for an object.

In the member-AVA, on the other hand, the action module controls the camera to gaze at the target object according to (1) the 3D view line and the area size of the target (i.e., $\widehat{L}(t+1)$ and $\widehat{AREA}(t+1)$) detected by the perception module or (2) the 3D position of the target (i.e., $\widehat{P}(t+1)$) received by the communication module. In each processing cycle, the action module first read $\widehat{P}(t+1)$ from the dynamic memory. If $\widehat{P}(t+1)$ is valid[7] , the action module controls the camera based on $\widehat{P}(t+1)$, otherwise based on $\widehat{L}(t+1)$ and $\widehat{AREA}(t+1)$. When the action module controls the camera based on $\widehat{L}(t+1)$ and $\widehat{AREA}(t+1)$, the camera is controlled by the prediction-based tracking method with the dynamic memory proposed in chapter 4. When the view direction is controlled based on $\widehat{P}(t+1)$, on the other hand, the following rules are employed:

1. **Object identification:** Let the 3D view line determined by $\widehat{P}(t+1)$ and the projection center be $\widehat{L}'(t+1)$. The action module compares $\widehat{L}'(t+1)$ with the object information $L^1(t+1)$, $\cdots$, $L^N(t+1)$ detected by the perception module for object identification. Depending on the result of object identification, the target information is modified as follows:

   Case A: If $L^y(t+1)$ has the enough small angle between $\widehat{L}'(t)$, where $y \in \{1, \cdots, N\}$, $L^y(t+1)$ is considered to be the target information at $t+1$ (successful identification).

   Case B: If all the 3D view lines $L^1(t+1)$, $\cdots$, $L^N(t+1)$ are distant from $\widehat{L}'(t)$, the target is considered to be lost at $t+1$ (unsuccessful identification).

2. **Camera control:** Based on the result of object identification, the action module controls the camera as follows:

   Case A: If object identification is successful, the pan-tilt-zoom parameters are controlled by the prediction-based tracking method.

   Case B: If object identification is unsuccessful, the pan-tilt angles are changed towards $L^y(t+1)$. To search for the target object, the zoom parameter is controlled so that the view angle of the camera becomes the widest.

5.3.2.3 Communication Module

If necessary, the communication module transmits the object information detected by the perception module (i.e., $L^1(t+1), \cdots, L^N(t+1)$) to agencies. This module also receives the information of the target object (i.e., $\widehat{P}(t+1)$) from agencies.

Object information sent from an AVA is listed in Table 5.1. This information is transmitted through the network as a message. Depending on the role of the AVA, this message is transmitted in the different ways:

[7] If the interval between $t+1$ and the current time is shorter than the predefined threshold, $\widehat{P}(t+1)$ is considered to be valid.

Table 5.1: Object information sent from an AVA to 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 when an AVA observed
	Object$_1$	View line	3D view line from a 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 0.

- If the AVA works as a freelancer-AVA, the message is broadcasted. While this message is accepted by all agencies, all AVAs ignore it.
- If the AVA works as a member-AVA, the message is sent only to its agency.

An agency extracts the necessary information from this message and establish object identification between its own target object and the object information included in the received message.

5.3.2.4 Dynamic Memory: Interaction between the Modules

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

Perception: The 3D view lines of the detected objects.

Action: The camera parameters (i.e., pan, tilt and zoom).

Communication: The received information of the target object.

Table 5.2 shows the reader and writer of the information exchanged between the modules.

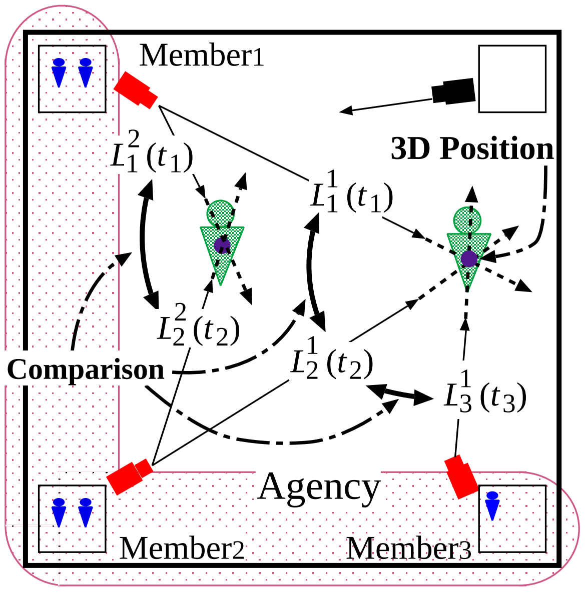
This information exchange is realized through the dynamic memory. The contents of the dynamic memory in the intra-AVA layer are shown in Table 5.3. The functions of the dynamic memory and the dynamic interactions between modules are identical to those of the single-target tracking system proposed in chapter 4. With the dynamic memory, the

Table 5.2: Information exchanged in the intra-AVA layer.

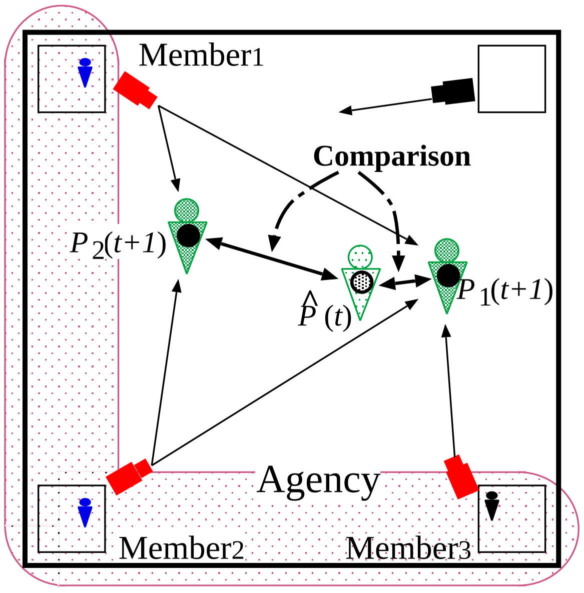
		Reader		
		Perception	Action	Communication
Writer	Perception		The 3D view lines of the detected objects	The 3D view lines of the detected objects
	Action	Camera parameters		ϕ
	Communication	The corrected information about the target object	The 3D position of the target object	

Table 5.3: Entries of the dynamic memory in the intra-AVA layer.

Entry			*Information*
AVA information	AVA-ID		ID of an AVA
	External parameters		External camera parameters (3D position and view direction)
	Role		Current role of an AVA (i.e., freelancer or member)
Camera information	Pan		Pan positions are recorded as time-series data ($P_{\text{cam}}(t)$).
	Tilt		Tilt positions are recorded as time-series data ($T_{\text{cam}}(t)$).
	Zoom		Zoom positions are recorded as time-series data ($Z_{\text{cam}}(t)$).
Target information	3D position		3D positions of the target object 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.
	$\vdots$		$\vdots$
	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 0.



(a) Spatial object identification



(b) Temporal object identification

Figure 5.14: Object identification established in the intra-agency layer.

modules can exchange their information asynchronously at an arbitrary time. Therefore, each module can work autonomously without damaging the reactiveness required for a real-time system.

5.3.3 Intra-agency Layer: Interaction between AVAs

The intra-agency layer consists of member-AVAs belonging to the same agency simultaneously. In this layer, the member-AVAs in the same agency exchange the information of the detected objects for object identification. Two kinds of object identification are required in this layer.

5.3.3.1 Spatial Object Identification

When the member-AVAs$_{1,\ldots,M}$ of the agency capture the images, the agency has to establish object identification between the 3D view lines detected by each member-AVA $\{L_1^i(t_1)|i=1,\cdots,N_1\}$, $\cdots$, $\{L_M^i(t_M)|i=1,\cdots,N_M\}$, where $\{L_m^i(t_m)|i=1,\cdots,N_m\}$ denotes the 3D view lines detected by member-AVA$_m$ at t_m (Figure 5.14 (a)). These 3D view lines are compared between AVAs, and the 3D distance between each view line is computed. If the 3D distance between the view lines is less than a threshold, these view lines are considered to be the information of the same object. In addition, an intersection of the identified 3D view lines is regarded as the 3D position of the object.

In an example shown in Figure 5.14 (a),

- member-AVA$_1$ detected two objects at t_1 (the 3D view lines are denoted by $L_1^1(t_1)$, $L_1^2(t_1)$, and
- member-AVA$_2$ detected two objects at t_2 (the 3D view lines are denoted by $L_2^1(t_2)$, $L_2^2(t_2)$, and
- member-AVA$_3$ detected one objects at t_3 (the 3D view line is denoted by L_3^1).

These 3D view lines are compared between the AVAs. '$L_1^1(t_1)$, $L_2^1(t_2)$ and $L_3^1(t_3)$' and '$L_1^2(t_1)$ and $L_2^2(t_2)$' are identified with each other, respectively. 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 this spatial identification method is effective only if the interval between $t_1, \cdots, t_M$ is short enough.

5.3.3.2 Temporal Object Identification

To gaze at the target object continuously, the agency compares the information of the target object at t with the information of the objects observed at $t+1$ after spatial object identification (Figure 5.14 (b)). Depending on the types of the observed object information, temporal object identification includes the following four cases:

- When spatial object identification at $t+1$ has reconstructed the 3D positions of detected objects (denoted by $\{P_i(t+1)|i = 1, \cdots, N\}$), the information of the target object at t is compared with $\{P_i(t+1)|i = 1, \cdots, N\}$. Note that $\{P_i(t+1)|i = 1, \cdots, N\}$ are reconstructed from the 3D view lines observed by each member-AVA at the time closest to $t+1$:

  Case 1. When the 3D position of the target object at t (i.e., $\widehat{P}(t)$) is compared with $\{P_i(t+1)|i = 1, \cdots, N\}$: Let (1) $P_x(t+1)$ have the shortest distance between $\widehat{P}(t)$, where $x \in \{1, \cdots, N\}$, and (2) the distance between $\widehat{P}(t)$ and $P_x(t+1)$ be shorter than a threshold. The agency then regards $P_x(t+1)$ as the 3D position of the target object at $t+1$.

  If this identification fails, namely none of $\{P_i(t+1)|i = 1, \cdots, N\}$ are identified with $\widehat{P}(t)$, the following case 3 is applied.

  Case 2. When the 3D view line of the target object at t (i.e., $\widehat{L}(t)$) is compared with $\{P_i(t+1)|i = 1, \cdots, N\}$: Let (1) $P_x(t+1)$ have the shortest distance between $\widehat{L}(t)$, where $x \in \{1, \cdots, N\}$, and (2) the distance between $\widehat{L}(t)$ and $P_x(t+1)$ be shorter than a threshold. The agency then regards $P_x(t+1)$ as the 3D position of the target object at $t+1$.

  If this identification fails, namely none of $\{P_i(t+1)|i = 1, \cdots, N\}$ are identified with $\widehat{P}(t)$, the following case 4 is applied.

  In these cases, $\widehat{P}(t+1) = P_x(t+1)$.

- When spatial object identification at $t+1$ has not reconstructed 3D position of any detected object, namely none of 3D view lines detected by member-AVAs are identified with each other, the information of the target object at t is compared with the 3D view lines of the objects, each of which is observed by each member-AVA$_{1,\cdots,M}$ at the time closest to $t+1$ (denoted by $\{L_1^i(t_1)|i=1,\cdots,N_1\}$, $\cdots$, $\{L_M^i(t_M)|i=1,\cdots,N_M\}$):

  Case 3. When the 3D position of the target object at t (i.e., $\widehat{P}(t)$) is compared with $\{L_1^i(t_1)|i=1,\cdots,N_1\}$, $\cdots$, $\{L_M^i(t_M)|i=1,\cdots,N_M\}$: Let (1) $L_x^y(t_y)$ have the shortest distance between $\widehat{P}(t)$, where $x \in \{1,\cdots,N_y\}$ and $y \in \{1,\cdots,M\}$, and (2) the distance between $\widehat{P}(t)$ and $L_x^y(t_y)$ be shorter than a threshold. The agency then regards $L_x^y(t_y)$ as the 3D view line of the target object at $t+1$. In addition, $\widehat{P}(t)$ is projected onto $L_x^y(t_y)$, and this projected 3D point is regarded as the 3D position at $t+1$.

  Case 4. When the 3D view line of the target object at t (i.e., $\widehat{L}(t)$) is compared with $\{L_1^i(t_1)|i=1,\cdots,N_1\}$, $\cdots$, $\{L_M^i(t_M)|i=1,\cdots,N_M\}$: Let (1) $\widehat{L}(t)$ be observed by member-AVA$_y$, and (2) $L_x^y(t_y)$ be identified with $\widehat{L}(t)$ by the perception module of member-AVA$_y$. The agency then regards $L_x^y(t_y)$ as the 3D view line of the target object at $t+1$.

  In these cases, $\widehat{L}(t+1) = L_x^y(t+1)$.

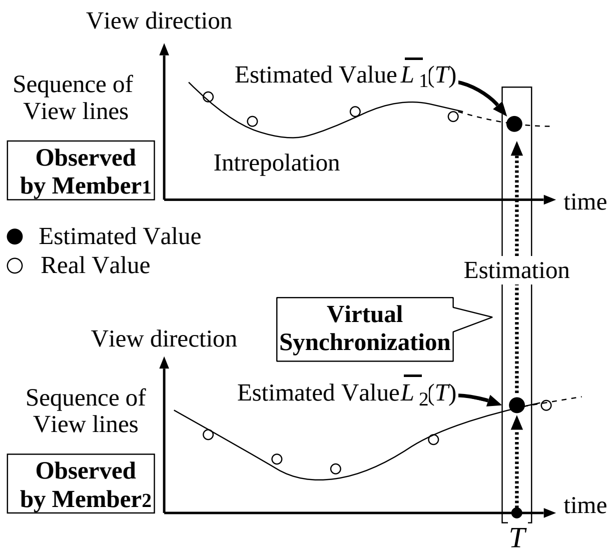
In an example shown in Figure 5.14 (b), the system estimated the 3D positions of two objects at $t+1$ (denoted by $P_1(t+1)$ and $P_2(t+1)$) by spatial object identification. The system, then, compares these 3D positions with the 3D position of the target object at t (denoted by $\widehat{P}(t)$), namely this situation is the above case 1. As a result, $P_1(t+1)$ is identified with $\widehat{P}(t)$.

Note that this temporal identification method is effective only if the interval between t and $t+1$ is short enough.

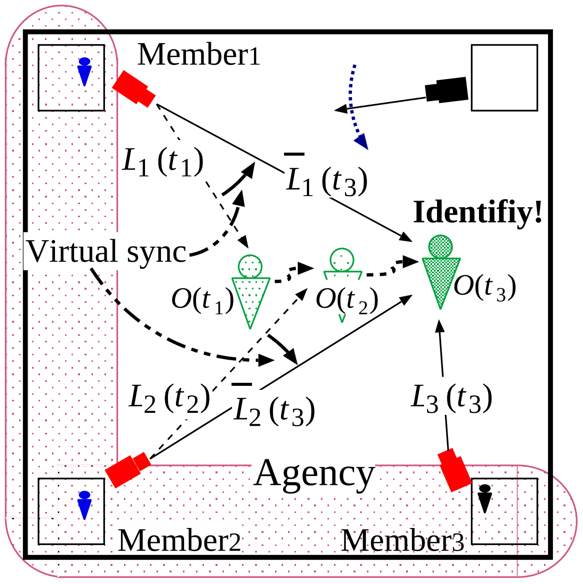
5.3.3.3 Virtual Synchronization for Spatial Object Identification

Since AVAs capture images autonomously, the member-AVAs in the same agency observe the 3D view lines of the objects ($\{L_m^i(t_m)|i=1,\cdots,N_m\}$) at different times (i.e., $t_m \neq t_n$). Furthermore, the message delay via network makes the interval between t_m and t_n larger. The result of object identification is, therefore, unreliable if these asynchronized object information from each camera is compared with each other. The unreliable identification results in difficulty in a continuous observation of the target object.

Other distributed systems that consist of autonomous cameras coped with this problem as follows: In [Nak01] and [KMS00], the newest information gathered from each camera is considered to be observed at the same time. In [Ste99], each object information includes its 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 small enough, as the simultaneous information. These approximate methods break down under complicated situations and network congestion.



(a)Readout the value from the dynamic memory



(b)Spatial object identification

Figure 5.15: Virtual synchronization for spatial object identification among member-AVAs in the same agency.

To solve this problem, we put the dynamic memory in the intra-agency layer as we do in the intra-AVA layer (Figure 5.10). By employing the dynamic memory, a 3D view line at an arbitrary time can be estimated from asynchronous discrete time-series data recorded in the dynamic memory. We can, therefore, solve this problem to estimate a 3D view line observed by each camera at the same time. We call this procedure a *Virtual Synchronization*. Since each agency works while keeping its intrinsic dynamics, it can establish object identification at an arbitrary time by reading 3D view lines detected by all the member-AVAs from the dynamic memory.

Figure 5.15 (a) shows an example of the virtual synchronization with the dynamic memory. In this example, the object information (i.e., the 3D view line) detected by member-AVA$_1$ and member-AVA$_2$ (indicated by white points in the figure) is written in the dynamic memory shared by the member-AVAs. To establish spatial object identification at T, the agency can obtain the 3D view lines detected by both member AVAs at T (denoted by $\bar{L}_1(T)$ and $\bar{L}_2(T)$, both of which are indicated by black points in the figure) by estimating the values from the dynamic memory. The agency can, therefore, establish spatial object identification between the 3D view lines observed at the same time.

In our system, spatial object identification is practically realized as follows. When an agency is formed (mentioned in Section 5.3.3.5), an *Agency Manager* is generated at the same time. An agency manager is an autonomous software agent[8] independent of AVAs, and performs the following tasks as a delegate of an agency.

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

Table 5.4: Entries of the dynamic memory in the intra-agency layer.

Entry			*Information*
Target information	3D position		3D positions of the target object are recorded as time-series data ($\widehat{P}(t)$).
	3D view line		3D view lines of the target object are recorded as time-series data ($\widehat{L}(t)$).
	Object-importance		The object-importance of the target object
Member information $\{1, \cdots, M\}$	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 information $\{1, \cdots, M\}$	Detected Information of member-AVA$_1$		Detected information (shown in Table 5.1) sent from member-AVA$_1$ is recorded as time-series data.
	$\vdots$		$\vdots$
	Detected Information of member-AVA$_M$		Detected information sent from member-AVA$_M$ is recorded as time-series data.

- Management of the dynamic memory in each agency. The contents of the dynamic memory in the intra-agency layer are shown in Table 5.4.

- Object identification. This task is performed by the perception module in the agency manager.

- Communication with other agencies and AVAs. This task is performed by the communication module in the agency manager.

That is, the agency is a conceptual group, and the agency manager is an entity of the agency. All the information of the target object is managed by each agency manager. There exists one-to-one correspondence between the target object (and its information) and the agency (and its agency manager). To manage the information of the target object with each agency intensively, the system handle the object information as follows:

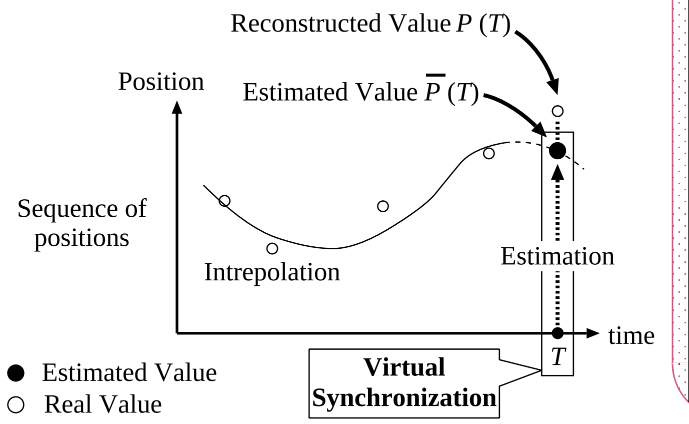
- All the member-AVAs send the information of the detected objects to its agency manager.
- Even if the agency disappears once because the target object cannot be observed, its recorded information will be managed by the agency that tracks the same target object when it is detected again (mentioned later).
- In the proposed system, the system detects only the position and direction as the information of the detected objects. If multi-modal information (e.g., appearance, sound and so on) is observed, all of them are managed intensively by the agency manager and written into its dynamic memory.

Member-AVA$_m$ sends the 3D view lines of the detected objects (i.e., $\{L^i_m(t_m)|i = 1, \cdots, N_m\}$) to the agency manager in the same agency. The agency manager then writes the received object information into its dynamic memory. When spatial object identification is required, the agency manager compares the object information observed by member-AVAs$_{1,\ldots,M}$, which are obtained by the virtual synchronization (i.e., $\{\bar{L}^i_1(T)|i = 1, \cdots, N_1\}, \cdots, \{\bar{L}^i_M(T)|i = 1, \cdots, N_M\}$). If the distance between $\bar{L}^i_p(T)$ and $\bar{L}^j_q(T)$ is small enough, the agency manager regards the ith detected result of AVA$_p$ and jth detected result of AVA$_q$ as the 3D view lines both of which go towards the same object (denoted by object$_x$) in the scene. The intersection of $\bar{L}^i_p(T)$ and $\bar{L}^j_q(T)$ (denoted by $P_x(T)$) is regarded as the 3D position of object$_x$ at T.

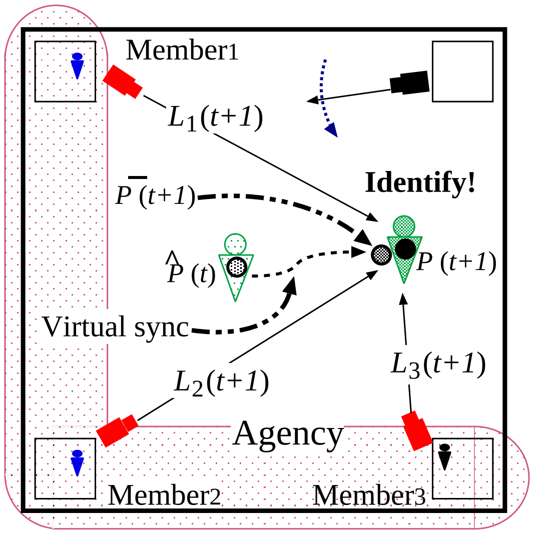
Object information sent from the member-AVA to the agency is listed in Table 5.1. For object identification in the intra-agency layer, only several parts of the detected information $\{1, \cdots, N\}$ are utilized. The other information is, however, required to other purposes:

- The target flag is referred by the agency manager to confirm whether or not the member-AVA successfully tracks the target object of the agency (mentioned later (in Section 5.3.3.6)).
- The AVA information is employed for the interaction in the inter-agency layer (mentioned later (in Section 5.3.4)).

Figure 5.15 (b) shows an example of spatial object identification with the virtual synchronization. In this example, AVA$_1$, AVA$_2$ and AVA$_3$ capture the 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, and then acquires $\bar{L}_1(t_3)$, $\bar{L}_2(t_3)$ and $L_3(t_3)$. By comparing these values synchronized virtually, the agency manager can realize reliable spatial object identification.



(a)Readout the value from the dynamic memory



(b)Temporal object identification

Figure 5.16: Virtual synchronization for temporal object identification.

5.3.3.4 Virtual Synchronization for Temporal Object Identification

For temporal object identification, the agency manager has to compare the 3D position of the target object at t (i.e., $\hat{P}(t)$) with the 3D positions of the detected objects$_{1,\cdots,N}$ at $t+1$ (i.e., $\{P_i(t+1)|i = 1, \cdots, N\}$). The result of object identification is, however, unreliable because the object information obtained at different times is compared with each other.

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

Figure 5.16 shows an example of temporal object identification with the virtual synchronization. By interpolating the reconstructed 3D positions of the target object, the agency manager can estimates the target position at T (Figure 5.16 (a)). In Figure 5.16 (b), the 3D position $P(t+1)$ is reconstructed at $t+1$. The agency manager then estimates the 3D position of the target object at $t+1$ (i.e., $\bar{P}(t+1)$), and compares $\bar{P}(t+1)$ with $P(t+1)$.

As mentioned above, the information exchange in the intra-agency layer through the dynamic memory allows the system to stabilize both spatial and temporal object identification. Note that all object identification in the intra-agency layer, namely object

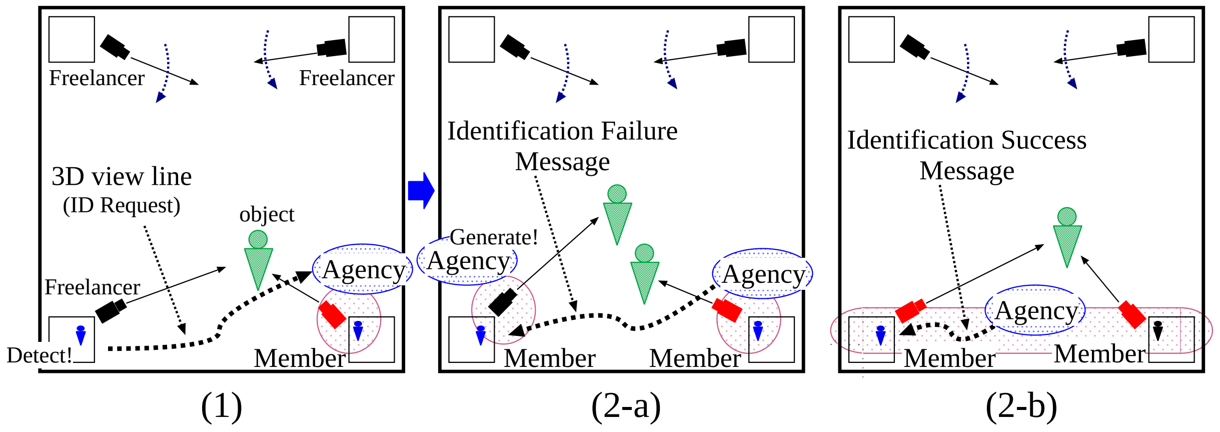


Figure 5.17: Agency formation.

identification between an agency and a member-AVA/freelancer-AVA, is established by the agency (agency manager).

Depending on whether or not spatiotemporal object identification is successful, the dynamic interactions for a generation and maintenance of an agency are activated. These dynamic interactions are defined by the following three cooperative-tracking protocols:

Agency Formation: This protocol defines

- a new agency generation by a freelancer-AVA and
- a participation of a freelancer-AVA in an agency.

Agency Maintenance: This protocol defines

- a secession of a member-AVA from an agency and
- an elimination of an agency.

Agency Spawning: This protocol defines a new agency generation from an existing agency.

5.3.3.5 Agency Formation Protocol

Initially, each AVA independently searches for an object. When a freelancer-AVA finds a new object, it requests from the existing agencies object identification between the newly detected object and the target object of each agency[9] (Figure 5.17 (1)). Depending on whether or not the result of object identification is successful, the freelancer-AVA works as follows:

[9] The details of a communication between the freelancer-AVA and the agency will be mentioned in Section 5.3.5.

Case A: No agency established a successful identification:

The freelancer-AVA that finds the new object starts a new agency manager. The agency that tracks the newly detected object is then formed. The freelancer-AVA joins into this agency (Figure 5.17 (2-a)).

Case B: An agency established a successful identification:

The freelancer-AVA joins into the agency that has made successful identification if requested by the agency (Figure 5.17 (2-b)).

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: if there are not enough freelancer-AVAs for *search* (i.e., in the case of deficiency of search-level), the number of freelancer-AVAs must not decrease. Then, the freelancer-AVA and agency works as follows:

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

Case B: The agency manager, which established a successful identification, examines the goal-functions of the freelancer-AVA and its member-AVAs to determine (1) whether or not include the freelancer-AVA in the agency instead of the current member-AVA and (2) if the freelancer-AVA joins into the agency, which member-AVA should be released.

Since all the information about the same object should be managed together, a newly generated agency manager (1) reads the object information obtained in the past and (2) compares its own target information with the read information for object identification. If this identification is successful, the agency manager considers the read information as the information of the own target object and records it into the dynamic memory. For this purpose, the agency manager need to record its own target information when the agency disappears (the details of the disappearance of the agency are mentioned in Section 5.3.3.6).

5.3.3.6 Agency Maintenance Protocol

After an agency is generated, the agency manager continues spatial and temporal object identification for cooperative tracking (Figure 5.18 (1)). If temporal object identification between the target objects of the agency and member-AVA$_m$ fails[10] , the agency manager reports the 3D position of the target object to member-AVA$_m$. This information navigates the gaze of member-AVA$_m$ towards the target object (Figure 5.18 (2)). Nevertheless, if the failure of identification continues for a long time, the agency manager puts member-AVA$_m$ out of the agency (Figure 5.18 (3)).

[10] The member-AVA sends the information of all detected objects to the agency. The agency, however, can find the target information of this member-AVA by checking the target flag (shown in Table 5.1).

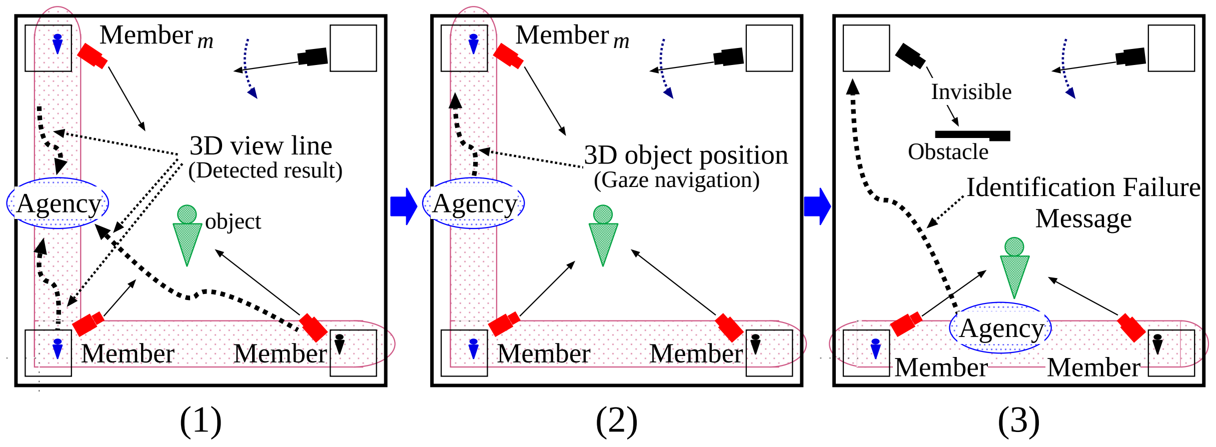


Figure 5.18: Agency maintenance.

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

In addition, depending on the relationship between the current state of the system and the task-constraint, the agency manager adjusts the number of member-AVAs. If there are not enough freelancer-AVAs (i.e., in the case of deficiency of search-level), the agency manager has to release its member-AVA to increase freelancer-AVAs.

The agency manager records its object information when the agency is eliminated[11] . As mentioned above, in order to unify the information of the same object, this recorded object information is read by a newly generated agency.

5.3.3.7 Agency Spawning Protocol

The agency manager can distinguish its own target object from other objects detected by member-AVA$_n$ (Figure 5.19 (1)). If the agency manager finds the 3D view line of the newly detected object (denoted by L_n), it requires other agencies to compare L_n with their own target objects for object identification. Then, the results of object identification are returned from other agency managers. If none of identification is successful (namely, it seems that there is not an agency that tracks the newly detected object in the system), the agency manager orders member-AVA$_n$ to generate a new agency (Figure 5.19 (2)). As a result, member-AVA$_n$ generates an agency for tracking L_n and joins it (Figure 5.19 (3)).

An 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.

[11] In our system, the object information is written into the disk drive.

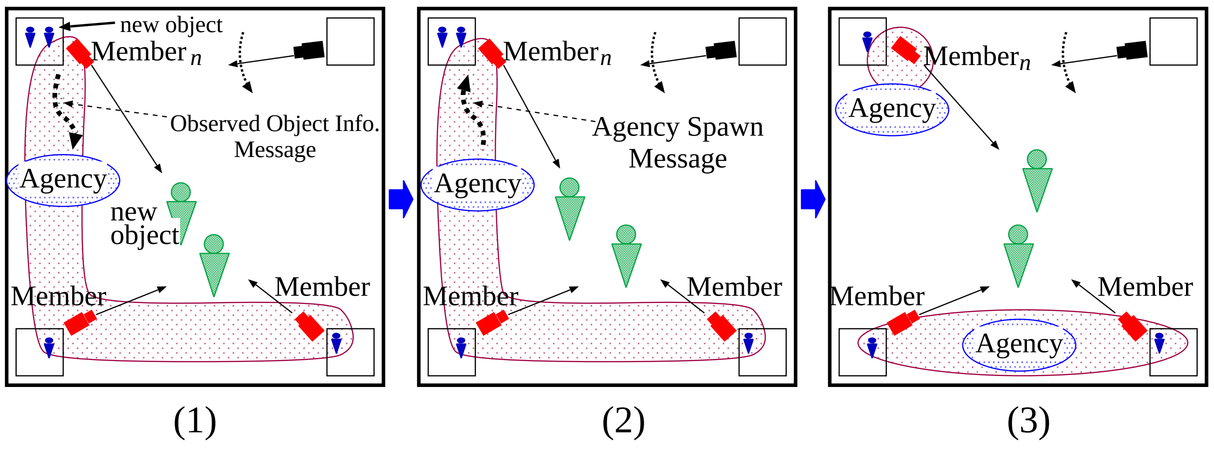


Figure 5.19: Agency spawning.

5.3.4 Inter-agency Layer: Interaction between Agencies

The inter-agency layer consists of all existing agencies. The fundamental task of an agency is to keep tracking its own target object. In order to keep tracking the target object in the complicated wide area, agencies need to exchange their member-AVAs with each other in accordance with the target motions. To realize such a dynamic reconstruction of the agency, the following two kinds of information are exchanged between agencies.

- Information of the target object.
- Information of the member-AVAs.

We call this information *Agency Information*. The contents of the agency information are listed in Table 5.5.

In our system, the agency reports its agency information not only to other agencies but also to all freelancer-AVAs through broadcast messages. The freelancer-AVA also broadcasts the information of the detected objects to all agencies. The freelancer-AVA has to exchange the object information between agencies in order to achieve the following functions:

- Maintain a consistency of one-to-one correspondence between the agency and the target object. That is, if there is not the agency that is tracking the object detected by a freelancer-AVA, the system can generate a new agency. The system should not, however, generate a new agency if there is an agency that is tracking this object.
- 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 is newly received, the tracking-level of the system increases based on the number of the member-AVAs belonging the agency.

Table 5.5: Agency information.

Entry			*Information*
Target information	3D information		3D position of the target object $\widehat{P}$ or 3D view line of the target object $\widehat{L}$
	Object-importance		The object-importance of the target object
	Time		The time when the target is observed
Member information $\{1, \cdots, M\}$	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

We will give a full account of the interaction between the agencies and the freelancer-AVAs in Section 5.3.5.

5.3.4.1 Virtual Synchronization for Object Identification between Agencies

An agency that has received the agency information from another agency (agency$_i$) compares the 3D position of its own target with that of agency$_i$'s target. This object identification is not reliable if these 3D positions are observed at different times. This problem can be solved with the virtual synchronization in the same way as temporal object identification in the intra-agency layer. With the 3D positions of the target object recorded as time-series data in the dynamic memory, the agency manager can synchronize the 3D position of its own target with the received 3D position of agency$_i$'s target. This virtual synchronization of the 3D positions realizes reliable object identification between agencies.

Depending on whether or not the result of object identification between agencies is successful, the dynamic interactions in the inter-agency layer defined by the following two cooperative-tracking protocols are performed:

Agency Unification: This protocol defines a merge of agencies, both of which track the same object.

Agency Restructuring: This protocol defines a reformation of the member-AVAs between agencies.

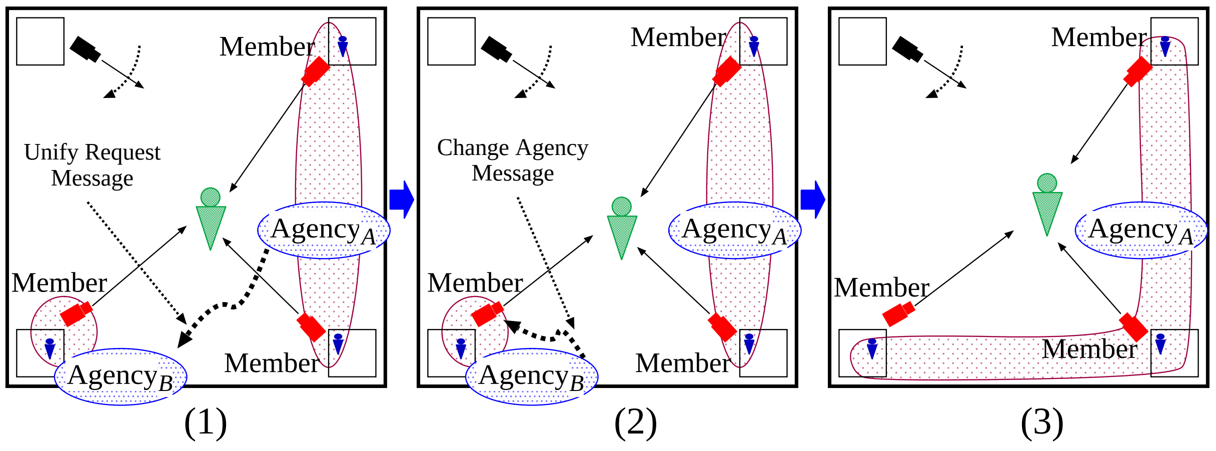


Figure 5.20: Agency unification.

5.3.4.2 Agency Unification Protocol

The *Agency Unification* protocol is executed when the result of object identification between agencies is successful. With this protocol, the agencies that are tracking the same object are merged into one.

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

- When the agency considers multi-objects in the scene as a single object because of the failure in object identification. For example, this situation is caused when different objects become close enough to be identified as the same object. In this case, the agencies that make successful identification merge together temporally to keep one-to-one correspondence between the agency and the detected target object.

- When a single object is first regarded as multiple objects because of the failure of object identification, and then multiple agencies are formed for the same object by mistake. This error is recovered by the subsequent observation. That is, multiple agencies that are made due to the mistake merge together when object identification between these agencies is successful.

Figure 5.20 shows an example of the agency unification. Agency manager$_A$, which has made successful object identification with the target object of agency$_B$, sends the message to agency manager$_B$. This message asks agency$_B$ to join agency$_A$ (Figure 5.20 (1)). In order to order the member-AVAs of agency$_B$ to transfer to agency$_A$, agency manager$_B$ sends messages to its member-AVAs after it receives the unification-request message from agency$_A$ (Figure 5.20 (2)). Agency manager$_B$ then eliminates itself. Thus, two agencies merge together (Figure 5.20 (3)).

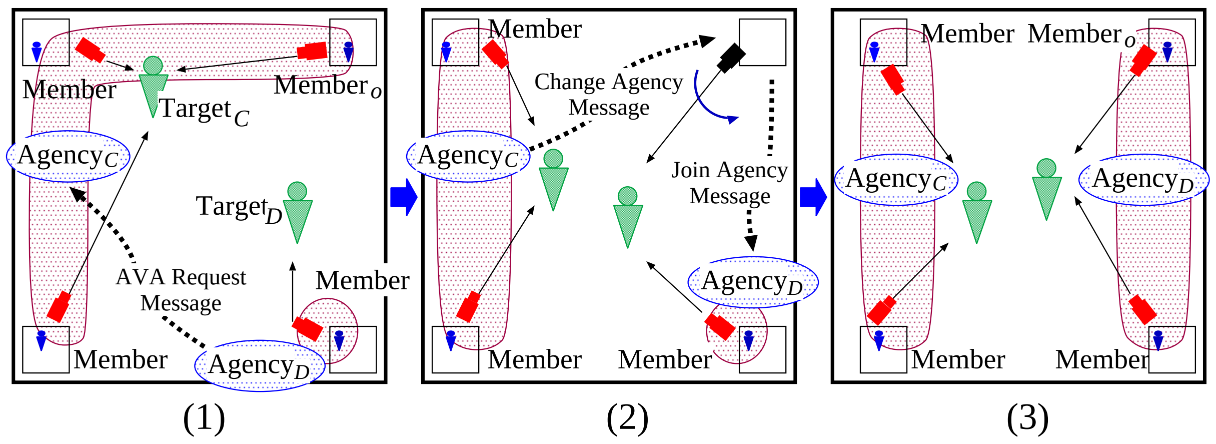


Figure 5.21: Agency restructuring.

5.3.4.3 Agency Restructuring Protocol

The *Agency Restructuring* protocol is executed when the result of object identification between agencies is unsuccessful. The agency manager performs the agency restructuring taking into account the following factors:

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

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

Due to object motion: Depending on the 3D position of the target object, the AVAs that are competent enough to gaze at the target object are determined.

Due to new agency generation: When a new agency is generated, this agency requires member-AVAs enough to track the target object. This agency, therefore, requests AVAs from other agencies.

We have various factors in determining the aptitude of each AVA for *tracking* (e.g. the 3D distance between the camera and the target, visibility from the camera), namely the criterion for the agency restructuring. Users can settle down this criterion depending on the task of the system as the goal-function.

Figure 5.21 shows an example of the agency restructuring.

1. If agency manager$_D$ makes unsuccessful object identification between agency$_C$, it examines (1) the number of the member-AVAs between agency$_C$ and agency$_D$ and (2) the tracking ability of each member-AVA. Based on this examination, agency

manager$_D$ decides whether or not to request a member-AVA from agency$_C$. If it requests, it sends the AVA-request message to agency$_C$ (Figure 5.21 (1)).

2. When agency manager$_C$ is requested to transfer its member-AVA to agency$_D$, agency manager$_C$ examines the aptitude of own member-AVAs for tracking the target object of each agency. Agency manager$_C$ determines the member-AVA that is more suitable for tracking target$_D$ rather than target$_C$, where target$_C$ and target$_D$ indicate the target object of agency$_C$ and agency$_D$, respectively. If agency manager$_C$ considers member-AVA$_o$ to be suitable for tracking target$_D$, agency manager$_C$ orders member-AVA$_o$ to transfer to agency$_D$ ("Change agency message" in Figure 5.21 (2)). Member-AVA$_o$ then informs agency$_D$ that it joins into agency$_C$ ("Join agency message" in Figure 5.21, 2.).

3. Member-AVA$_o$ then starts working as a member of agency$_D$ (Figure 5.21 (3)).

5.3.4.4 Exclusive Interaction in Inter-agency Layer

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

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

Agency Unification: If two agencies decide to join into one side at the same time, both agencies may fail in the agency unification because the agency of the destination has disappeared.

Agency Restructuring: If multiple agencies exchange their member-AVAs at the same time, there is some possibility that AVAs will swing between agencies due to a radical reformation of agencies.

To solve these problems, each agency performs a dynamic interaction in the inter-agency layer only with a single agency simultaneously. In addition, each inter-agency cooperative-tracking protocol is executed depending on the following conditions.

Agency Unification: Agency$_P$ that has made successful object identification with agency$_Q$ requires agency$_Q$ to join into agency$_P$.

When agency$_Q$ is requested to join into agency$_P$, it decides whether or not to accept the request according to its own 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 with 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: Agency$_P$ that has made unsuccessful object identification with another agency can require a member-AVA only if agency$_P$ is not executing any other inter-agency interaction with other agencies.

When the agency is requested a member-AVA, it decides whether or not to accept the request according to its own condition as follows:

- If the agency is not executing any inter-agency interaction, it determines whether or not to release its member-AVA. Depending on the determination, the agency sends the message of acceptance or rejection in reply.
- If the agency is interacting with another agency, it rejects the request.

5.3.5 Communication with the Freelancer-AVA

A freelancer-AVA communicates only with agencies. The freelancer-AVA sends the information of the detected objects to the agencies by the broadcast message. The contents of this message are listed in Table 5.1.

The agency that has received the object information from the freelancer-AVA establishes spatial object identification between its target information and the received object information. Then, the agency sends the result of identification to the freelancer-AVA in reply. The freelancer-AVA that has received this reply performs the agency formation depending on the received issue.

On the other hand, the agency broadcasts the agency information. The freelancer-AVA, that has received the agency information, refers to the target information included in the received message. This freelancer-AVA decides the next role depending on the condition of 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 object. If this freelancer-AVA starts tracking the target object, it points its gaze towards the 3D position of the target object that the agency sends.

(Search-level of the current state) $\leq$ (Search-level of the task-constraint)

The freelancer-AVA should continue to search for a new object.

5.4 Completeness and Soundness of the System

5.4.1 Completeness for Persistent Tracking

In general, it is hard to guarantee that the system can always track all target objects under every situation in the real world. The possibility of tracking depends on various factors (e.g., the number of cameras and target objects, mechanical limitations of the camera, the speed of the target object, and so on). Here, we mention the relations between the numbers of cameras and target objects, and show the upper limitation of the target objects to be tracked simultaneously in the proposed system.

To track a target object, an agency is generated. We define an agency as a representation of a target object in the system. Because of this definition, the maximum number of target objects to be tracked is equal to the maximum number of agencies. An agency is generated by an AVA that detects a target object in the scene, and the agency has to be attended by at least one member-AVA for tracking its target object. The maximum number of agencies, therefore, is the total number of AVAs in the system. In this case, each agency has only one member-AVA.

In the proposed system, however, an agency reconstructs 3D information of its target object from 2D information of the object observed by multiple member-AVAs. The 3D information of the target object greatly assists the agency to persistently keep tracking the target object as follows[12] :

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

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

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

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

Case 2: $\lceil \frac{n_a}{2} \rceil < n_t \leq n_a$: Although the system can track all the target objects, $(n_t - \lceil \frac{n_a}{2} \rceil)$ or more target objects 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.

This limitation about the number of target objects results because (1) the agency receives the information of objects only from its member-AVAs and (2) the agency has to be attended by at least one member-AVA. To avoid this problem, we can modify the system as follows:

3D reconstruction method: If the agency receives the object information from AVAs except for its member-AVAs as well, it can possibly reconstruct the 3D information of the target object even when it has a single member-AVA. That is, this information

[12] In addition, to apply the proposed system to various vision systems (e.g., navigation and motion capturing systems), the 3D information of the target object has to be reconstructed.

exchange allows the agency to stably track the target object even in $\lceil \frac{n_a}{2} \rceil < n_t \leq n_a$ (the above case 2). In this case, however, member-AVAs have to send the information of the detected objects not only to their own agency manager but also to other agency managers that require the object information. This increases a network-load. In particular, if each member-AVA broadcasts the object information to report it to all agencies, a huge network-load is produced. To avoid a breakdown of the system due to increasing the network-load, each member-AVA should examine which agency requires the information, and send it based on the result of the examination.

Definition of an agency: If the agency can exist without any member-AVA, the system can track the target objects even in the case of $n_a < n_t$ (the above case 3). To obtain the information of the target object, the agency without the member-AVA has to gather the object information from non-member-AVAs. We have to, therefore, solve the problem about increasing the network-load mentioned above. In addition, this definition has an essential problem: since the vacuous agency cannot control any camera, it is not guaranteed that this agency (1) keeps tracking the target object and (2) acquires its meaningful image.

In addition, here again note that we aim at designing a system that can not only track trajectories of target objects but also acquire their detailed information. If the system is required only to track the trajectory of the target object, each AVA can survey a wide area and observe multiple objects simultaneously by adjusting the zooming factor of its camera at the wide view angle. To acquire the detailed information of the object, however, each AVA should control the pan, tilt and zoom parameters of its camera to keep obtaining the high-resolution image of the target object. In this case, it is hard for each AVA to keep observing multiple objects simultaneously. Accordingly, each agency has to be attended by at least one member-AVA in order to acquire the detailed information of its target object continuously by controlling the gaze of its member-AVA.

Thus, to avoid several problems mentioned above, we design the system so that (1) the agency reconstructs the 3D information of its target object only from the information received from its member-AVAs and (2) each agency has at least one member-AVA.

5.4.2 Necessity and Sufficiency of Cooperative-tracking Protocols

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

All the cooperative-tracking protocols are activated by the agency depending on whether or not the result of object identification is successful. Object identification is established when the agency received the message including the information of the objects from the freelancer-AVA, member-AVA and other agencies. Table 5.6 shows the

Table 5.6: Cooperative-tracking protocols activated by the agency depending on the result of object identification.

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

types of the cooperative-tracking protocols that are activated according to the relations between the type of the received object information and the result of object identification.

As we can see, the cooperative-tracking protocols are designed just enough in accordance with the situations in the real-world.

5.4.3 Soundness of Communicating with Other Processes

In each layer, multiple parallel processes dynamically exchange their information with each other for cooperation. This dynamic interaction has to be realized without causing deadlock.

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

Intra-agency layer: Similarly, each agency has its own dynamic memory managed by the agency manager. All member-AVAs send their observed information to the agency manager. The agency manager continues spatiotemporal object identification while keeping its own intrinsic dynamics. In addition, several information is reported from the agency manager to its member-AVAs by the message transmission. The member-AVA accepts only the message from its agency manager not to be affected inconsistently by multiple agencies: for example, message delays incur the invalid communication between the agency and the member-AVA belonging to another agency.

Inter-agency layer: The information exchange in this layer is implemented by a direct communication using the message transmission. To avoid a conflict of different interactions between agencies, (1) each agency performs an inter-agency cooperative-tracking protocol only with one agency simultaneously and (2) a timeout processing for the inter-agency interaction is adopted to cope with message delays, dynamic agency generation and elimination, and other unpredictable factors.

Thus, the dynamic interaction in each layer can be reactively realized without inconsistency and deadlock.

5.4.4 Soundness of State Transitions of the System

Here, we show that each module, AVA and agency can continue to work persistently without causing deadlock. To confirm this, we summarize (1) the functions of all the cooperative-tracking protocols and (2) the transitions of the system state caused by these protocols.

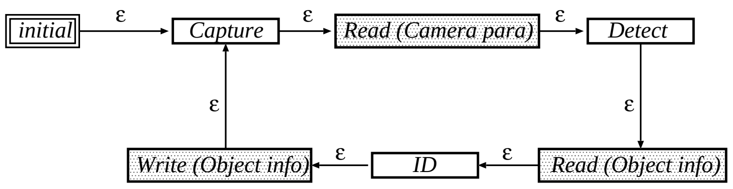
5.4.4.1 State Transitions of the Modules

Perception module: Starting at *initial*, a perception module repeats the following steps (Figure 5.22 (a)):

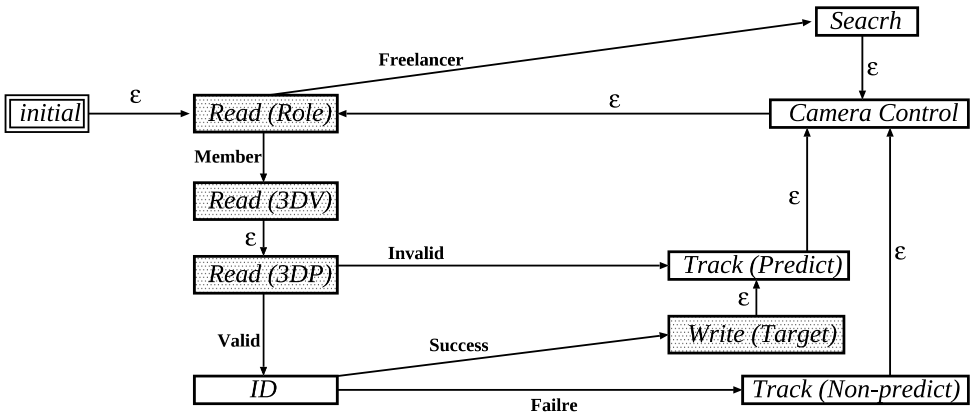
1. *Capture*: Capture an image (at t_0).
2. *Read (Camera para)*: Read the pan-tilt-zoom parameters at t_0 from the dynamic memory.
3. *Detect*: Detect object regions in the captured image.
4. *Read (Object info)*: Read the histories of the object information detected in the past from the dynamic memory.
5. *ID*: Identify the newly detected object information with the histories of the object information.
6. *Write (Object info)*: Write the result of object identification into the dynamic memory.

Action module: Starting at *initial*, an action module works at each state as follows (Figure 5.22 (b)):

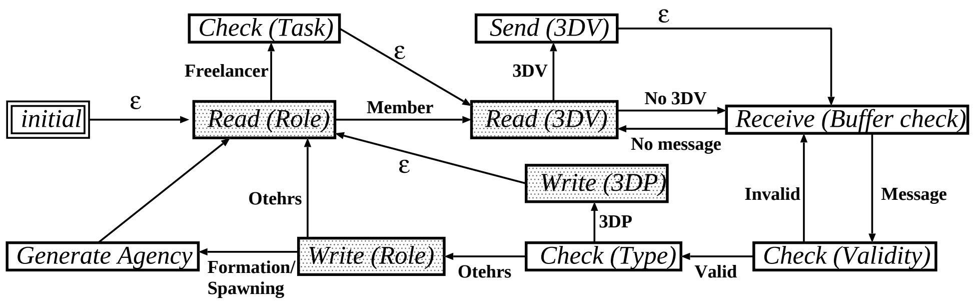
- *Read (Role)*: Read the current role (i.e, freelancer-AVA or member-AVA) from the dynamic memory:
  - If the current state is a freelancer-AVA, the state changes to *Search*.
  - If the current state is a member-AVA, the state changes to *Read (3DP)*.
- *Search*: Determine the next camera parameters based on the predefined trajectory. The state changes to *Camera Control*.
- *Read (3DV)*: Read the 3D view lines of the detected objects from the dynamic memory. The state changes to *Read (3DP)*.



(a) State transition network of the perception module



(b) State transition network of the action module



(c) State transition network of the communication module

Figure 5.22: State transition networks of the modules. Each box and arrow indicate a state and state transition, respectively. A meshed box shows that the module accesses the dynamic memory at this state. Each arrow has a condition for activation, provided that an automatic state transition (denoted by ϵ) occurs immediately.

- *Read (3DP)*: Read the 3D position of the target object from the dynamic memory. Depending on whether or not the information of the 3D position is valid, the state changes as follows:
  - If the information is valid, the state changes to *ID*.
  - If the information is invalid, the state changes to *Track (Predict)*.
- *ID*: Identify the 3D position of the target object with the 3D view lines of the detected objects:
  - If identification is successful, the state changes to *Write (Target)*.
  - If identification is unsuccessful, the state changes to *Track (Non-predict)*.
- *Write (Target)*: Write the information of the newly identified target object into the dynamic memory. The state changes to *Track (Predict)*.
- *Track (Predict)*: Determine the next camera parameters by the prediction-based control method. The state changes to *Camera Control*.
- *Track (Non-predict)*: Determine the next camera parameters to gaze at the 3D position of the target object. The state changes to *Camera Control*.
- *Camera Control*: Control the camera parameters. The state changes to *Read (Role)*.

Communication module: All messages sent to the AVA are stored in the message buffer managed by the communication module. Starting at *initial*, a communication module works at each state as follows (Figure 5.22 (c)):

- *Read (Role)*: Read the current role from the dynamic memory:
  - If the current role is a freelancer-AVA, the state changes to *Check (Task)*.
  - If the current role is a member-AVA, the state changes to *Read (3DV)*.
- *Check (Task)*: Compare the task-constraint and the current state of the system. The result of the comparison is valid until the state is changed to *Check (Task)* again.
- *Read (3DV)*: Read the 3D view lines of the detected objects from the dynamic memory. Depending on whether or not there exists the newest 3D view lines that has not been sent to the agency manager, the state changes as follows:
  - If the newest information has been detected, the state changes to *Send (3DV)*.
  - If the newest information has not been detected, the state changes to *Receive (Buffer check)*.
- *Send (3DV)*: Depending on the current role of the AVA, the communication module works as follows:
  - Send the newest 3D view lines to the agency manager if the AVA is a member-AVA.
  - Broadcast the newest 3D view lines, if the AVA is a freelancer-AVA.

The state changes to *Receive (Buffer check)*.

- *Receive (Buffer check)*: Read the message buffer: all the messages are put into the message buffer, and the communication module read them in the order of FIFO:
  - If there exists a message in the buffer, the state changes to *Check (Validity)*.
  - Otherwise, the state changes to *Read (3DV)*.
- *Check (Validity)*: Examine whether or not the read message is valid[13] , the state changes as follows:
  - If the message is valid, the state changes to *Check (Type)*.
  - If the message is invalid, the state changes to *Receive (Buffer check)*.
- *Check (Type)*: Depending on the type of the message, the state changes as follows:
  - If the message is the 3D position of the target object, the state changes to *Write (3DP)*.
  - Otherwise, the state changes to *Write (Role)*.
- *Write (3DP)*: Write the received 3D position of the target object into the dynamic memory. The state changes to *Read (Role)*.
- *Write (Role)*: Write the new role if the received message makes the AVA change its state. Depending on the type of the received message and the result of the comparison in *Check (Task)*, the state changes as follows:
  - If the message requests the AVA to generate the new agency (i.e., the agency formation and agency spawning protocols) and the task-constraint allows the system to increase member-AVAs, the state changes to *Generate Agency*.
  - Otherwise, the state changes to *Read (Role)*.
- *Generate Agency*: Generate an agency. The state changes to *Read (Role)*.

As mentioned above, all the modules can work while keeping their own intrinsic dynamics. In addition, since the information exchange between the modules is implemented through the dynamic memory, all the modules can work continuously without conflicting with another module when they exchange the information.

5.4.4.2 State Transition of the AVA

Figure 5.23 (a) shows the state transition model of the AVA. All the state transitions of the AVA are caused by the cooperative-tracking protocols mentioned in Section 5.3, provided that the state transitions indicated by arrows with ϵ occur automatically and immediately.

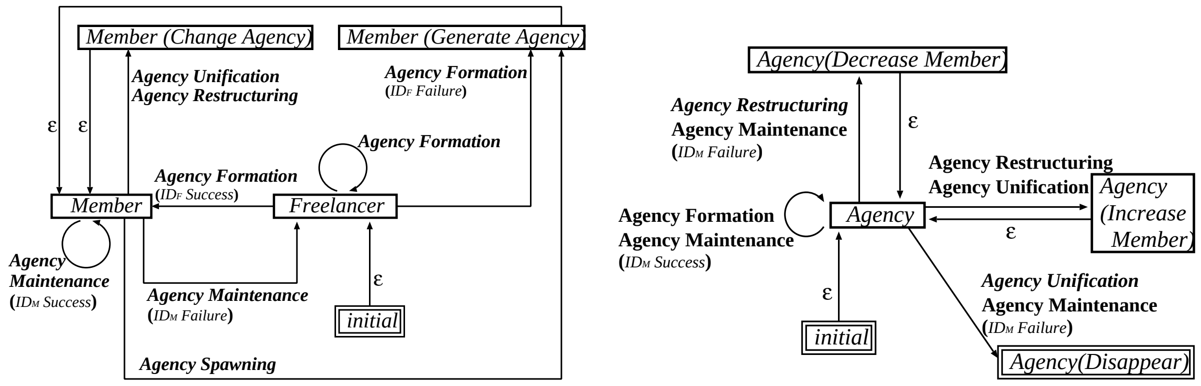
[13] For example, a freelancer-AVA neglects the object information depending on the current state of the system. A member-AVA neglects all messages from other agencies.

1. Starting at *initial*, an AVA works as a freelancer-AVA at *Freelancer*.
2. If the AVA at *Freelancer* finds an object, it requests the existing agencies to identify the detected object with their own target objects. Depending on the result of object identification, the AVA changes its state as follows:
   - ***Agency Formation*:** If this AVA cannot start tracking the object due to the relations between the task-constraint and the current state of the system, its state stays at *Freelancer* regardless of the result of object identification.
   - ***Agency Formation*** (*ID_F Success*), ***Agency Formation*** (*ID_F Failure*)**:** If this AVA can start tracking the object, its state is changed to *Member* or *Member(Generate Agency)* depending on the result of object identification. Its state is immediately changed to *Member* after it generates an agency manager even if the state has been changed to *Member(Generate Agency)*. This AVA, then, starts working as a member-AVA at *Member*.
3. If the AVA receives the message that induces the state transition from its agency manager when the AVA works as a member-AVA, its state is changed depending on the kind of the executed cooperative-tracking protocol as follows:
   - ***Agency Unification***, ***Agency Restructuring*:** The state is returned to *Member* via *Member(Change Agency)*. This AVA then transfers to another agency.
   - ***Agency Spawning*:** The state is returned to *Member* via *Member(Generate Agency)*. A new agency is generated at *Member(Generate Agency)*, and this AVA belongs to the new agency.
   - ***Agency Maintenance*** (*ID_M Failure*)**:** This AVA exits from the agency, and then works as a freelancer-AVA again. The state is changed to *Freelancer*.
   - ***Agency Maintenance*** (*ID_M Success*)**:** The AVA stays in the current agency, and its state remains at *Member*.

Note that although the states *Freelancer* and *Member* have the reiterative state transitions (***AgencyFormation*** and ***Agency Maintenance*** (*ID_M Success*), respectively), the state of the AVA changes constantly in the lower level as follows:

- Even if the AVA continues to be a freelancer-AVA, its camera parameters are controlled to search for an object. That is, its internal state always keeps changing in the action level.
- The member-AVA fixes its camera parameters when the target object stands. The AVA, however, keeps observing the target object, and all the observation results are written into the dynamic memory in the AVA. The internal state of the AVA, therefore, keeps changing by updating the contents of the dynamic memory.

Based on the above discussion, it is confirmed that each AVA work continuously without falling into the steady condition.



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

Figure 5.23: State transition networks of the AVA and agency. Each box and arrow indicate a state and state transition, respectively. A cooperative-tracking protocol with each arrow causes a state transition. Protocols shown by bold and italic fonts are caused by object identification and a message that reports the result of object identification established by another agency, respectively. An automatic state transition (denoted by ϵ) occurs immediately. ID_F and ID_M denote object identification of the agency with the freelancer-AVA and member-AVA, respectively.

5.4.4.3 State Transition of the Agency

Figure 5.23 (b) shows the state transition model 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 with that of the AVA.

1. Just after an agency is generated (at *initial*), it starts working at *Agency*.

2. The agency receives the following two kinds of messages from AVAs and other agencies:

   Object information: When the agency receives the object information, it establishes object identification between its target object and the received object information. Each cooperative-tracking protocol executed by this object identification is shown by a bold font in Figure 5.23 (b).

   Result of object identification: The result of object identification between agencies, which is established by another agency, is replied. This message is sent based on two kinds of cooperative-tracking protocols (i.e., the agency unification and agency restructuring). Each result of these cooperative-tracking protocols is shown by a italic bold font in Figure 5.23 (b).

Depending on the result of object identification, the state of the agency is changed as follows:

- **Agency Formation**, **Agency Maintenance** (*ID_M Success*)**:** The agency waits the next message. Its state stays at *Agency*.
- ***Agency Restructuring***, **Agency Maintenance** (*ID_M Failure*)**:** If the agency is requested to release and give its member-AVA from another agency (i.e., ***Agency Restructuring***) or object identification with its member-AVA fails for a long time (i.e., ***Agency Maintenance*** (*ID_M Failure*)), the number of its member-AVAs decreases. The state of the agency changes to *Agency (Decrease Member)* and returns to *Agency*.
- **Agency Restructuring**, **Agency Unification:** If the agency requests the member-AVA from another agency, its state changes to *Agency (Increase Member)* and returns to *Agency*.
- **Agency Maintenance** (*ID_M Failure*), ***Agency Unification*:** Because the agency does not have any member-AVA, the state of the agency moves to *Agency (Disappear)*. The agency then eliminates itself.

Although the state transition network of the agency has the reiterative state transition as well as that of the AVA, the internal state of the agency also keeps changing: since the agency receives the information of the detected objects from its member-AVAs, the target information managed by the agency is always updated.

To realize the above state transition of the agency, the perception and communication modules in the agency manager (of agency$_A$) works as follows. Figure 5.24 shows their state transition models.

Perception module: Starting st *initial*, the state changes as follows:

- *Read (Object info)*: Read the object information received from freelancer-AVAs and agencies, which object identification has not been established with its own target information. The state changes to *Select ID-type*.
- *Select ID-type*: Select the type of object identification that will be established next:
  - If spatiotemporal object identification among the 3D view lines detected by different member-AVAs is selected, the state changes to *Read (Virtual sync$_M$)*.
  - If object identification between its own target information and the object information received from the freelancer-AVA or the other agency is selected, the state changes to *Read (Virtual sync$_T$)*.
- *Read (Virtual sync$_M$)*: Read (1) the 3D view lines detected by the member-AVAs and (2) the 3D information of its target object, each of which is observed at the same time (T_M) with the virtual synchronization. If the 3D position of its target object at T_M in the dynamic memory is valid, the 3D position is

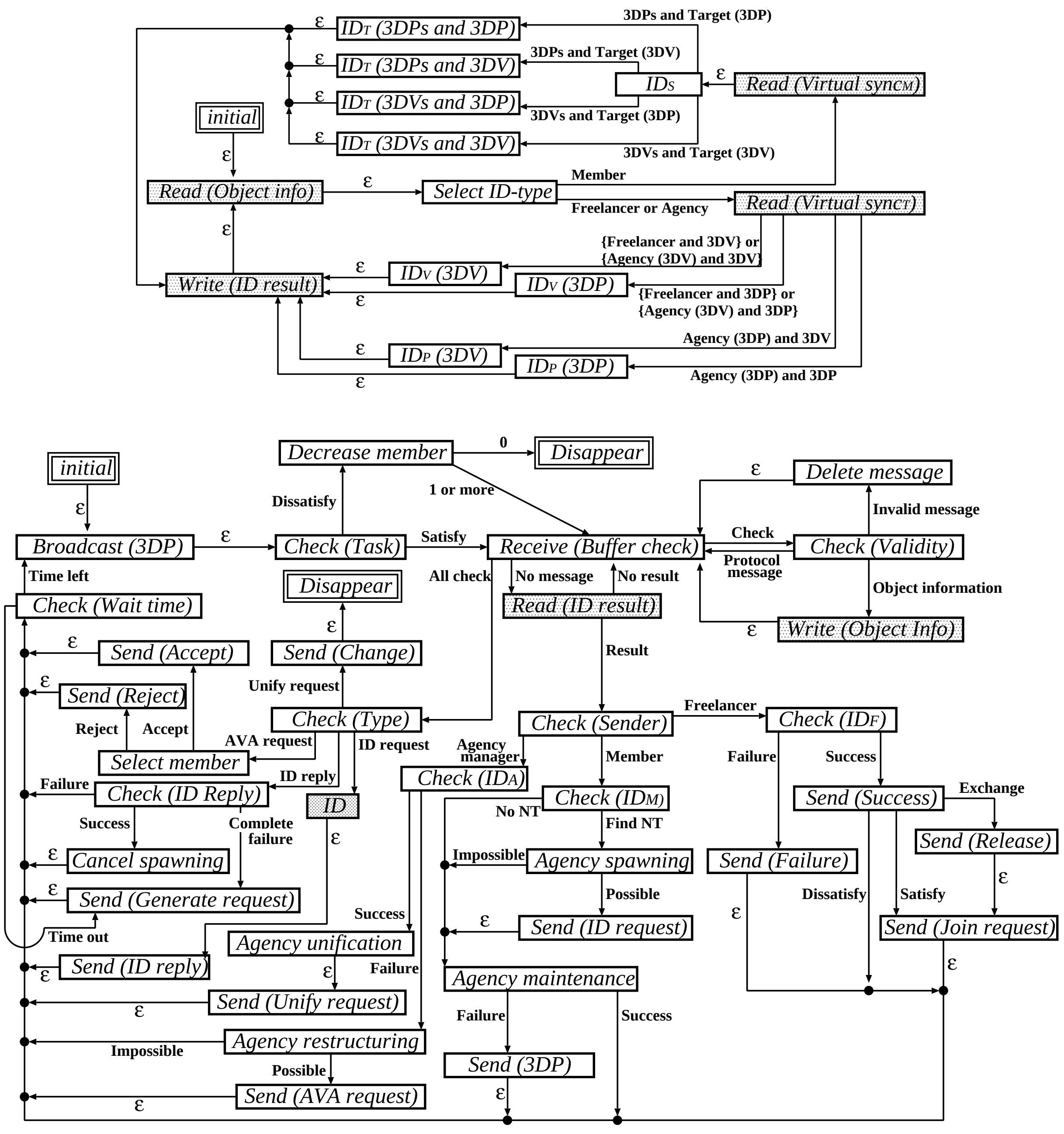


Figure 5.24: State transition network of the perception and communication modules in the agency manager (Upper: perception, Lower: communication). Each box and arrow indicate a state and state transition, respectively. A meshed box shows that the module accesses the dynamic memory at this state. Each arrow has a condition for activation, provided that an automatic state transition (denoted by ϵ) occurs immediately.

regarded as the target information (denoted by $\widehat{P}$). Otherwise, the 3D view line of the target object is regarded as the target information (denoted by $\widehat{L}$). The state changes to ID_S.

- ID_S: Establish spatial object identification among the 3D view lines detected by the member-AVAs (denoted by $L_{1,\cdots,l}$). Depending on (1) whether or not spatial object identification reconstructs the 3D positions of the objects (denoted by $P_{1,\cdots,p}$) and (2) the type of its target information read at *Read (Virtual sync$_M$)*, the state changes as follows:
  - If (1) spatial object identification reconstructs $P_{1,\cdots,p}$ and (2) $\widehat{P}$ is read, the state changes to *ID_T (3DPs and 3DP)*.
  - If (1) spatial object identification reconstructs $P_{1,\cdots,p}$ and (2) $\widehat{L}$ is read, the state changes to *ID_T (3DPs and 3DV)*.
  - If (1) spatial object identification cannot reconstruct any 3D positions and (2) $\widehat{P}$ is read, the state changes to *ID_T (3DVs and 3DP)*.
  - If (1) spatial object identification cannot reconstruct any 3D positions and (2) $\widehat{L}$ is read, the state changes to *ID_T (3DVs and 3DV)*.
- *ID_T (3DPs and 3DP)*: Select the 3D position closest to $\widehat{P}$ from $P_{1,\cdots,p}$. The state changes to *Write (ID result)*.
- *ID_T (3DPs and 3DV)*: Select the 3D position closest to $\widehat{L}$ from $P_{1,\cdots,p}$. The state changes to *Write (ID result)*.
- *ID_T (3DVs and 3DP)*: Select the 3D view line closest to $\widehat{P}$ from $L_{1,\cdots,p}$. The state changes to *Write (ID result)*.
- *ID_T (3DVs and 3DV)*: Select the 3D view line closest to $\widehat{L}$ from $L_{1,\cdots,p}$. The state changes to *Write (ID result)*.
- *Read (Virtual sync$_T$)*: Read the 3D information of its target object, which is virtually synchronized with the object information received from the freelancer-AVA or the other agency. Suppose all the object information is synchronized at T_T. If the 3D position of its target object at T_T in the dynamic memory is valid, the 3D position is regarded as the target information (denoted by $\bar{P}$). Otherwise, the 3D view line of the target object is regarded as the target information (denoted by $\bar{L}$).
  - If (1) the received information is the 3D position and (2) $\bar{P}$ is read, the state changes to *ID_P (3DP)*.
  - If (1) the received information is the 3D position and (2) $\bar{L}$ is read, the state changes to *ID_P (3DV)*.
  - If (1) the received information is the 3D view line and (2) $\bar{P}$ is read, the state changes to *ID_V (3DP)*.
  - If (1) the received information is the 3D view line and (2) $\bar{L}$ is read, the state changes to *ID_V (3DV)*.
- *ID_P (3DP)*: Compare $\bar{P}$ with the received 3D position for object identification.

- *ID_P (3DV)*: Compare $\bar{L}$ with the received 3D position for object identification.
- *ID_V (3DP)*: Compare $\bar{P}$ with the received 3D view line for object identification.
- *ID_V (3DV)*: Compare $\bar{L}$ with the received 3D view line for object identification.
- *Write (ID result)*: Write the result of object identification into the dynamic memory.

Communication module: All messages sent to the agency manager are stored in the message buffer managed by the communication module. Starting st *initial*, the state changes as follows:

- *Broadcast (3DP)*: Broadcast the 3D position of its own target object. The state changes to *Check (Task)*.
- *Check (Task)*: Compare the task-constraint and the current state of the system. The result of the comparison is valid until the state is changed to *Check (Task)* again. Depending on the result of the comparison, the state changes as follows:
  - If the task-constraint is satisfied, the state changes to *Receive (Buffer check)*.
  - If the task-constraint is dissatisfied, the state changes to *Decrease member*.
- *Decrease member*: Release a member-AVA that is determined based on the goal-function. Depending on the number of the member-AVAs, the state changes as follows:
  - If agency$_A$ has no member-AVAs, the state changes to *Disappear*.
  - Otherwise, the state changes to *Receive (Buffer check)*.
- *Receive (Buffer check)*: Read the message buffer:
  - If there is not any messages in the buffer, the state changes to *Read (ID result)*.
  - If there is a message in the buffer, the state changes to *Check (Validity)*.
  - If all the messages in the buffer are read, the state changes to *Check (Type)*.

  Note that all the messages in the buffer are read, they are not deleted from the buffer.
- *Check (Validity)*: Depending on whether or not the read message is valid[14] , the state changes as follows:
  - If the message is invalid, the state changes to *Delete message*.
  - If the message includes object information, the state changes to *Write (Object info)*.
  - If the message reports the result of the cooperative tracking protocol activated by another agency, the state changes to *Receive (Buffer check)*.

[14] For example, an agency manager neglects the message about inter-agency cooperative-tracking protocol depending on the result of the comparison between the task-constraint and the current state of the system.

- *Delete message*: Delete the invalid message from the buffer. The state changes to *Receive (Buffer check)*.
- *Write (Object info)*: Write the object information into the dynamic memory, and delete it from the buffer. The state changes to *Receive (Buffer check)*.
- *Read (ID result)*: Read all the results of object identification from the dynamic memory, and select the one which has the biggest priority:
  - If there is not any results in the dynamic memory, the state changes to *Receive (Buffer check)*.
  - If the identification result is selected, the communication module deletes it from the dynamic memory. The state changes to *Check (Sender)*.
- *Check (Sender)*: Depending on the type of the identification result, the state changes as follows:
  - If object identification is established with the object information detected by a freelancer-AVA (freelancer$_f$), the state changes to *Check (ID$_F$)*.
  - If object identification among the object information detected by member-AVAs is established, the state changes to *Check (ID$_M$)*.
  - If object identification is established with the object information detected by another agency (agency$_a$), the state changes to *Check (ID$_A$)*.
- *Check (ID$_F$)*: Depending on the result of identification, the state changes as follows:
  - If identification is successful, the state changes to *Send (Success)*.
  - If identification is unsuccessful, the state changes to *Send (Failure)*.
- *Send (Success)*: Report the successful result of object identification to freelancer-AVA$_f$. Depending on the current state of the system, the state changes as follows:
  - If the current state satisfies the task-constraint, the state changes to *Send (Join request)*.
  - If the current state does not satisfy the task-constraint, the communication module examines the goal-function whether or not it should exchange the roles of its member-AVA and freelancer-AVA$_f$. If it should exchange, the state changes to *Send (Release)*.
  - Otherwise, the state changes to *Check (Wait time)*.
- *Send (Join request)*: Request freelancer-AVA$_f$ to join agency$_A$. The state changes to *Check (Wait time)*.
- *Send (Release)*: Order the selected member-AVA to exit from agency$_A$. The state changes to *Send (Join request)*.
- *Send (Failure)*: Report the unsuccessful result of object identification to freelancer-AVA$_f$. The state changes to *Check (Wait time)*.
- *Check (ID$_M$)*:

  - If there exists a newly detected non-target object in the object information detected by the member-AVAs, the state changes to *Agency spawning*.
  - Otherwise, the state changes to *Agency maintenance*.

- *Agency spawning*: Depending on the number of the member-AVAs, the state changes as follows:
  - If (1) agency$_A$ has multiple member-AVAs or (2) the object importance of the newly detected object is larger than that of its target object, the state changes to *Send (ID request)*.
  - Otherwise, the state changes to *Agency maintenance*.
- *Send (ID request)*: Request other agencies to identify the newly detected object with their target objects. The state changes to *Agency maintenance*.
- *Agency maintenance*: Depending on the result of object identification between the 3D view lines of the target object detected by the member-AVAs and the 3D position of its own target object, the state changes as follows:
  - If identification is successful, the state changes to *Check (Wait time)*.
  - If identification with member-AVA$_m$ is unsuccessful, the state changes to *Send (3DP)*.
- *Send (3DP)*: Send the 3D position of the target object to member-AVA$_m$. The state changes to *Check (Wait time)*.
- *Check (ID$_A$)*: Depending on the result of object identification with agency$_a$, the state changes as follows:
  - If identification is successful, the state changes to *Agency unification*.
  - If identification is unsuccessful, the state changes to *Agency restructuring*.
- *Agency unification*: The state changes to *Send (Unify request)*.
- *Send (Unify request)*: Request agency$_a$ to join into agency$_A$. The state changes to *Check (Wait time)*.
- *Agency restructuring*: Examine the object-importance of the target object, the number of the member-AVAs, and the goal-function of each agency. Depending on the result of the examination, the communication module works as follows:
  - If agency$_A$ does not have enough member-AVAs or agency$_a$ has the AVA that is suitable for gazing at the target object of agency$_A$, the state changes to *Send (AVA request)*.
  - Otherwise, the state changes to *Check (Wait time)*.
- *Send (AVA request)*: Request a member-AVA from agency$_a$. The state changes to *Check (Wait time)*.
- *Check (Type)*: Select the message in the buffer, which has the biggest priority, and delete it from the buffer. Suppose the selected message is sent from agency$_b$. Depending on the type of the selected message, the state changes as follows:

  - If the message requests agency$_A$ to merge into agency$_b$ (i.e., the agency restructuring protocol), the state changes to *Send (Change)*.
  - If the message requests agency$_A$ to give the member-AVA to agency$_b$ (i.e., the agency restructuring protocol), the state changes to *Select member*.
  - If the message requests object identification between the target object of agency$_A$ and the object information included in the selected message (i.e., the agency spawning protocol), the state changes to *ID*.
  - If the message includes the reply of object identification (i.e., the agency spawning protocol), the state changes to *Check (Reply)*.

- *Select member*: Examine the object-importance of the target object, the number of the member-AVAs, and the goal-function of each agency. Depending on the result of the examination, the communication module works as follows:
  - If agency$_A$ should transfer its member-AVA to agency$_b$, the communication module selects the member-AVA that transfers to agency$_b$. The state changes to *Send (Accept)*.
  - Otherwise, the state changes to *Send (Reject)*.
- *Send (Accept)*: Report the acceptance of the request to agency$_b$, and order the selected member-AVA to transfer to agency$_b$.
- *Send (Reject)*: Report the rejection of the request to agency$_b$.
- *Send (Change)*: Order all the member-AVAs to transfer to agency$_b$. The state changes to *Disappear*.
- *ID*: Identify the received object information with its target information by the virtual synchronization. The state changes to *Send (ID reply)*.
- *Send (ID reply)*: Send the result of object identification to agency$_b$. The state changes to *Check (Wait time)*.
- *Check (ID reply)*: Depending on the received result of object identification, the state changes as follows:
  - If all the existing agencies establishes unsuccessful object identification, the state changes to *Send (Generate request)*.
  - If the received message reports the successful result of object identification, the state changes to *Cancel spawning*.
  - Otherwise, the state changes to *Check (Wait time)*.
- *Send (Generate request)*: Request the member-AVA, which has detected the newly detected object, to generate a new agency, and release this member-AVA from the agency. The state changes to *Check (Wait time)*.
- *Cancel spawning*: Finish waiting the replies of object identification from other agencies, namely the agency spawning.
- *Check (Wait time)*:

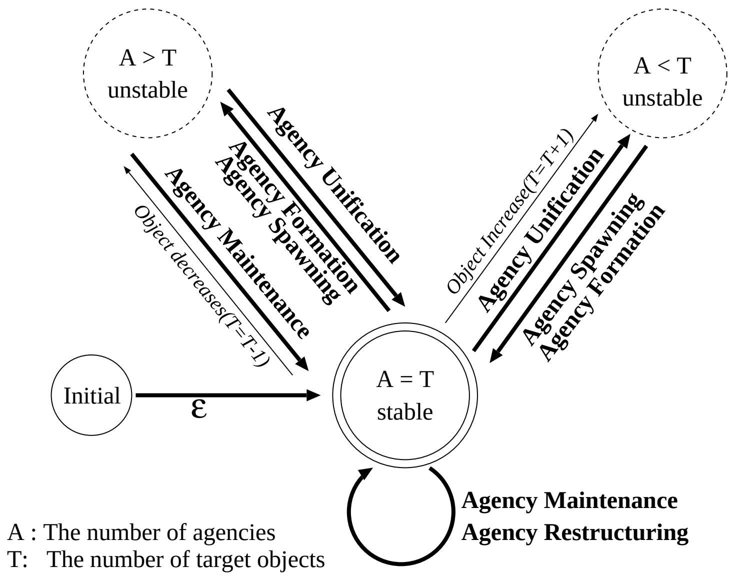


Figure 5.25: State transition network of the system. Each circle and arrow indicate a state and state transition, respectively. Varying situations in the scene are shown by an italic font. The results of cooperative-tracking protocols are shown by a bold font.

  - If (1) the communication module has requested object identification from other agencies at T_r based on the agency spawning protocol, (2) the agency spawning has not finished, and (3) the difference between the current time and t_r is larger than the threshold, the state changes to *Send (Generate request)*.
  - Otherwise, the state changes to *Broadcast (3DP)*.

- *Disappear*: Eliminate itself.

5.4.4.4 State Transition of the System

Finally, we show the state transition network of the total system. Changing behaviors of each AVA and agency brings the state transition of the system. With various cooperative-tracking protocols, the system can track multiple target objects. The system dynamically changes its state while tracking. Figure 5.25 shows the state transition of the system.

When (1) the situation in the real world changes (namely, the number of the target objects in the scene changes) and (2) a wrong cooperative-tracking protocol is activated, the system returns to the stable state by properly executing the cooperative-tracking protocol as follows:

1. When the situation in the real world changes.

When the number of objects increases: If a freelancer-AVA finds a new object, the agency formation is executed. If a member-AVA finds a new object, on the other hand, the agency spawning is executed.

When the number of objects decreases: Since temporal object identification in the intra-agency layer is unsuccessful, the agency maintenance is executed. Then, the agency that has tracked the disappeared object is eliminated.

2. When the system has executed a cooperative-tracking protocol by mistake.

Due to the wrong agency formation and agency spawning: Then, multiple agencies track a single object simultaneously. In this case, the agency unification corrects the system state.

Due to the wrong agency unification: Then, only a single agency tracks multiple objects. In this case, the agency formation or agency spawning restores the system to the stable state.

That is, the system keeps working with repeating stable and unstable conditions.

5.5 Experiments

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

5.5.1 Specifications of the System

5.5.1.1 System Organization

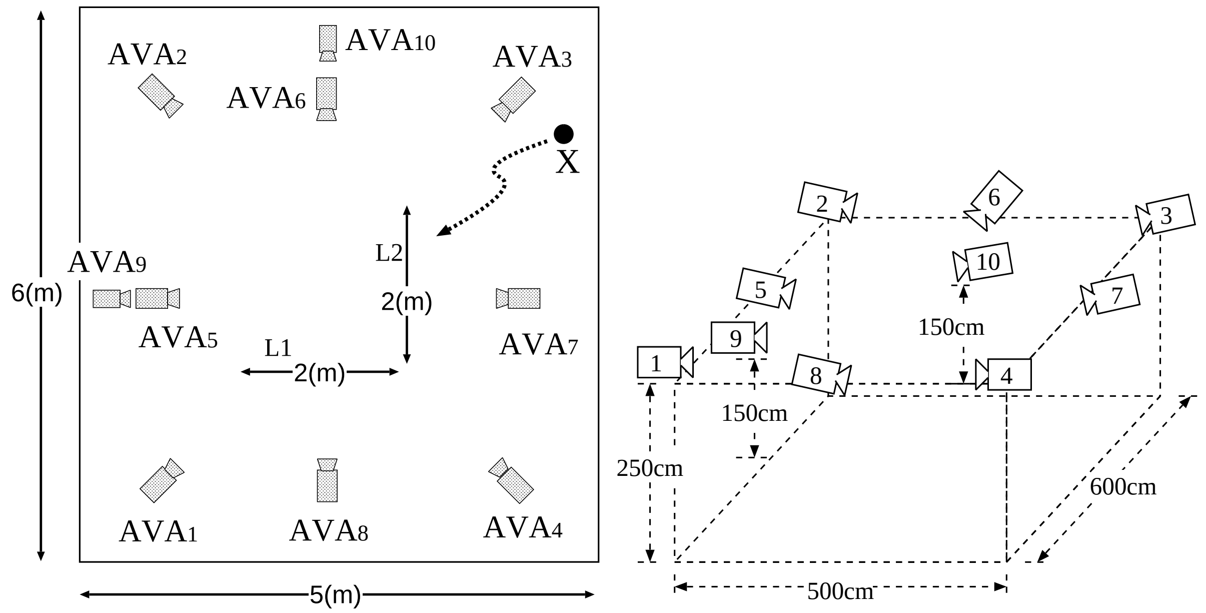
In our experiments, we employed ten AVAs. Each AVA consists of a network-connected PC with an active camera.

PC: PentiumIII 600MHz $\times$ 2 and 256MB memories with Linux operating system.

Active camera: FV-PTZ camera (SONY EVI-G20). The camera parameters can be controlled via RS-232C.

Network: 100M-base Ethernet.

The perception, action and communication modules are implemented by threads on a PC. The dynamic memory is also implemented by a thread on the same PC. The communication module exchanges information by UDP messages. In addition, the internal clocks of all the PCs are synchronized by Network Time Protocol (NTP)[Mil91]. With these resources, the perception module can capture images and detect objects in the observed image at about 0.1[sec] intervals on average.



(a) Top view (b) Camera settings

Figure 5.26: Experimental environment.

5.5.1.2 Calibrating External Camera Parameters

We conducted experiments in the environment illustrated in Figure 5.26. Camera$_9$ and camera$_{10}$ are placed about 1.5[m] above the floor. All other cameras are placed about 2.5[m] above the floor (Figure 5.26 (b)). The external camera parameters (i.e., the 3D position and view direction of each camera) were calibrated. We acquired the external camera parameters of all the cameras based on a homography between image planes of cameras. In the experimental environment illustrated in Figure 5.26, the flat floor can be observed from every camera. Therefore, the projection between an image plane and the floor can be represented by a homography. By utilizing this property, the external camera parameters are acquired as follows:

1. From the homography matrices $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$, each of which is determined by the floor and the image planes of two cameras, we can obtain a collineation matrix $\boldsymbol{M} = \boldsymbol{H}_2\boldsymbol{H}_1^{-1}$. The matrix $\boldsymbol{M}$ can be estimated by giving four corresponding points projected from the floor onto the two image planes.

2. By decomposing the matrix $\boldsymbol{M}$, we can obtain relative rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$ matrices between two cameras, as well as the surface normal vector of the floor $\vec{n}$.

3. If we have m cameras, the above relative external camera parameters (i.e., $\boldsymbol{R}$ and $\boldsymbol{T}$) between 'camera$_1$ and camera$_2$', $\cdots$, 'camera$_{m-1}$ and camera$_m$' are estimated.

4. By integrating all the relative external camera parameters, we can acquire the external camera parameters of all the cameras.

5.5.1.3 Designing Goal-function

The goal-function is examined by the agency manager (1) when the agency releases its member-AVA to satisfy the task-constraint (i.e., *Decrease member* in Figure 5.24), (2) when object identification with a freelancer-AVA is successful (i.e., *Send (Success)* in Figure 5.24), (3) when the agency restructuring protocol is activated (i.e., *Agency restructuring* and *Select member* in Figure 5.24), and (4) when the agency is required a member-AVA from another agency (i.e., *Select member* in Figure 5.24).

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

Search-value of freelancer-AVA$_f$: Let W_f denote the area size of the floor that is visible from AVA$_f$. In this experiment, W_f is computed from the external parameters of camera$_f$ (i.e., the 3D position and view direction of the camera). The value of this function (denoted by V_{S_f}) is determined as follows:

$$V_{S_f} \quad = \quad \alpha_S \times W_f, \tag{5.9}$$

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 object 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 value of this function (denoted by $V_{T_m^n}$) is determined as follows:

$$V_{T_m^n} \quad = \quad \frac{1}{D_m^n} \times \frac{1}{A_m^n}. \tag{5.10}$$

5.5.2 Performance Evaluation

We conducted experiments with the systems with/without the virtual synchronization. In order to verify the effectiveness of the virtual synchronization against not only the asynchronized observations but also the network delay and the lost packet, we broadcasted vain packets over the network to adjust the network load.

The system tracked two computer-controlled mobile robots. Both the robots repeated a straight-line motion at a speed of 50[cm/sec] in the observation scene. L1 and L2 in Figure 5.26 (a) show the trajectories of the robots.

Figure 5.27 (a) shows variations of network conditions when the packet size of the vain broadcast messages is changed. The error of spatial identification in Figure 5.27 (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 Figure 5.27 (c) denotes the average distance between the 3D positions of the target, each of which are reconstructed at t and $t+1$ (i.e., $\widehat{P}(t)$ and $\widehat{P}(t+1)$). Since temporal identification was

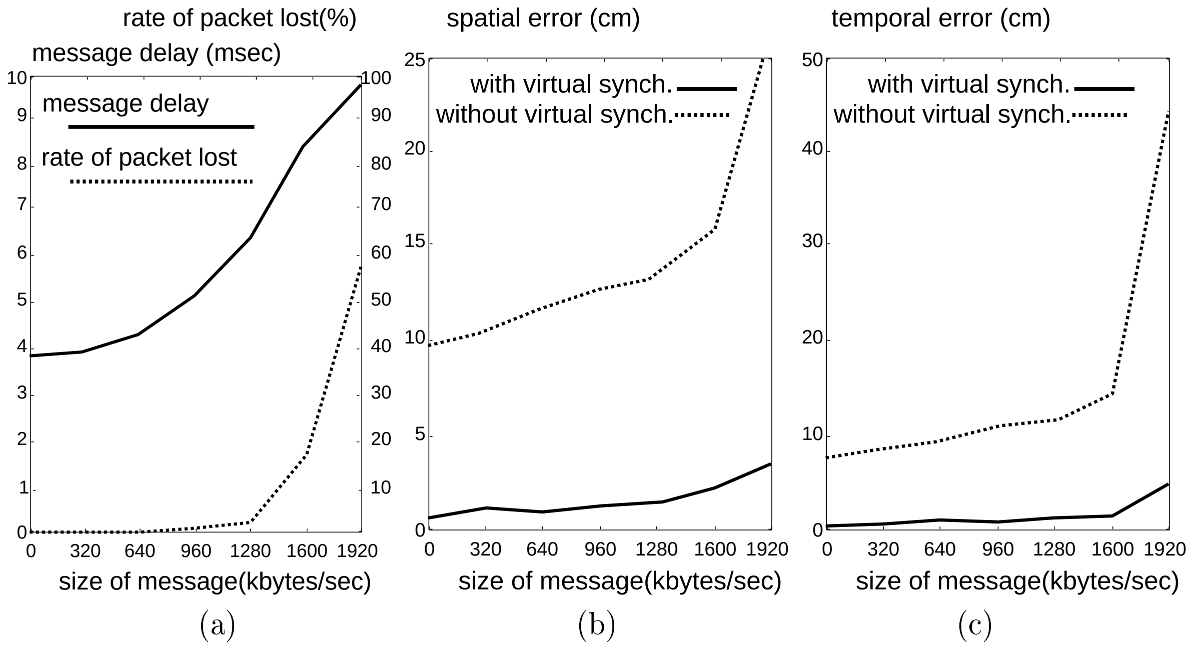


Figure 5.27: Performance evaluation of the virtual synchronization: (a) Delay of the message (solid line) and Rate of lost packet (dotted line), (b) Error in spatial object identification, (c) Error in temporal object identification. The horizontal axis indicates the total size of the vain broadcast messages per second.

established at 0.1[sec] intervals in this experiment, the robot moved for 5[cm] between each temporal identification.

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

5.5.3 Verifying Cooperative-tracking Protocols

Next, we experimented to verify the effectiveness of the cooperative-tracking protocols. Our experimental results demonstrated flexible and reliable multi-target tracking by co-operation among AVAs.

In order to verify the effectiveness of the task representation with the task-constraint and object-importance, we made two experiments in the same environment. In each experiment, we gave the following parameters as the task representation.

Experiment 1

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

Object-importance: Values for all objects is 1.0.

Experiment 2

Task-constraint: Search-level was 0.3. Tracking-level was 0.7.

Object-importance: Values for object$_1$ and object$_2$ were 1.0 and 0.5, respectively[15] .

In both experiments, the system tracked two people. Object$_1$ first came into the observation space from the location 'X' (shown in Figure 5.26 (a)). Next, object$_2$ came into the observation space. Both objects, then, moved freely in the observation space.

At first, we show the result of the first experiment.

Figure 5.28 shows the partial image sequences observed by each AVA. The size of each image is 320×240 [pixel]. The images on the same row were taken by the same AVA. Figure 5.28 shows images taken by AVA$_2$, AVA$_4$, AVA$_5$, AVA$_7$, AVA$_8$, AVA$_9$ and AVA$_{10}$ as examples. The images on the same column were taken at almost the same time. The enclosed regions within the red and blue lines in the images indicate the detected regions of object$_1$ and object$_2$, respectively.

Figure 5.29 shows the role of each AVA and the formation of each agency. The image in Figure 5.29 indicates the situation at the time when the observed image (in Figure 5.28) on the same column was captured. A green circle indicates a freelancer-AVA. Circles of other colors indicate member-AVAs. A square indicates a target object tracked by an agency. The color of the object is the same with that of member-AVAs which was tracking each target object. A line from the camera to the target object indicates correspondence between the agency and the target object. 'X' shown in Figure 5.29 indicates location X in Figure 5.26 (a).

In this experiment, the system worked as follows:

a: First of all, each AVA was searching for an object independently (Figure 5.29 (a)).

b: AVA$_5$ first detected object$_1$ (Figure 5.28, 5-b), and then agency$_1$ was formed (Figure 5.29 (b)).

c: All the AVAs, except for AVA$_5$, were tracking object$_1$ as the member-AVAs of agency$_1$. AVA$_5$ was searching for a new object as the freelancer-AVA (Figure 5.28, 5-c).

d: Next, AVA$_5$ detected a new object (Figure 5.28, 5-d). AVA$_5$ then regarded this object as the target (object$_2$), and generated agency$_2$ (Figure 5.29 (d)).

e: In this experiment, the object-importance of both object$_1$ and object$_2$ were equal to each other. As a result of the agency restructuring, therefore, the number of the member-AVAs in agency$_2$ became equal to that in agency$_1$ (Figure 5.29 (e)).

f: Since object$_1$ came close to object$_2$, no AVA could divide these objects by the background subtraction (Figure 5.28, 2-f, 4-f, 5-f, 7-f, 8-f, 9-f). Then, the newest target information (i.e., the 3D position) of both agency$_1$ and agency$_2$ became identical,

[15] In this experiment, the system gave 1.0 and 0.5 as the object-importance to the object that was detected first and second, respectively

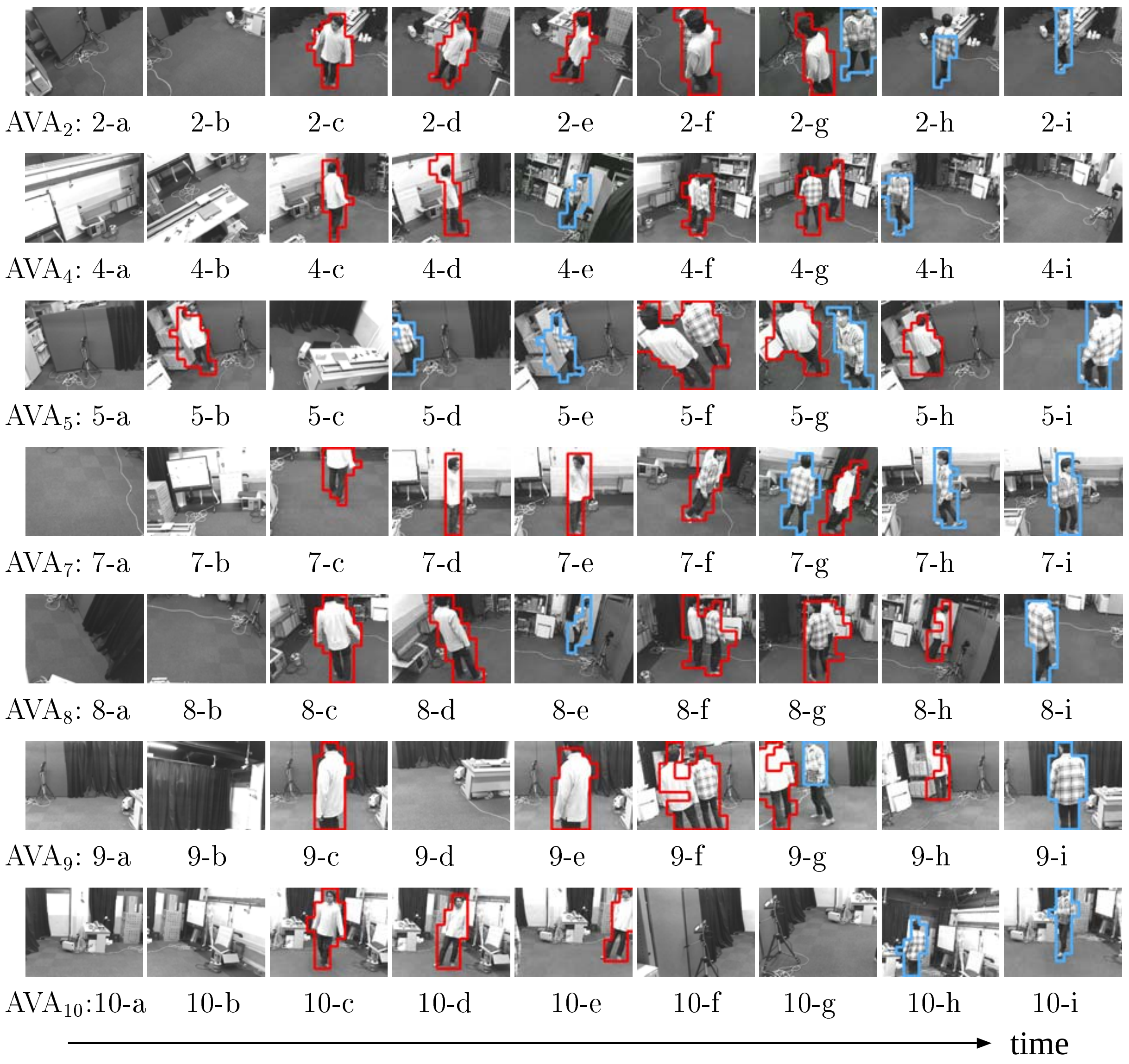

AVA$_2$: 2-a 2-b 2-c 2-d 2-e 2-f 2-g 2-h 2-i
AVA$_4$: 4-a 4-b 4-c 4-d 4-e 4-f 4-g 4-h 4-i
AVA$_5$: 5-a 5-b 5-c 5-d 5-e 5-f 5-g 5-h 5-i
AVA$_7$: 7-a 7-b 7-c 7-d 7-e 7-f 7-g 7-h 7-i
AVA$_8$: 8-a 8-b 8-c 8-d 8-e 8-f 8-g 8-h 8-i
AVA$_9$: 9-a 9-b 9-c 9-d 9-e 9-f 9-g 9-h 9-i
AVA$_{10}$:10-a 10-b 10-c 10-d 10-e 10-f 10-g 10-h 10-i
time

Figure 5.28: Experiment 1: Partial image sequences observed by AVAs.

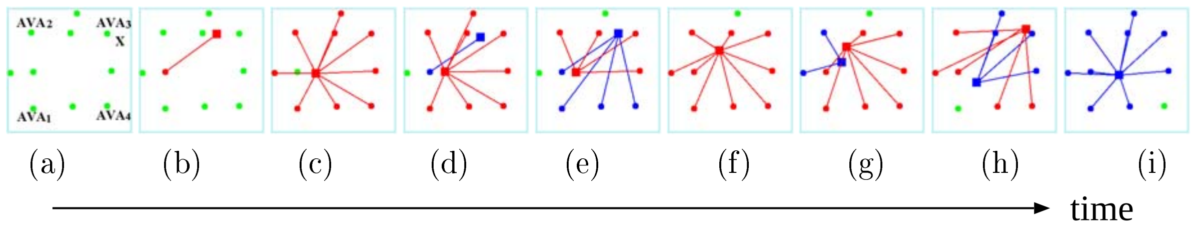
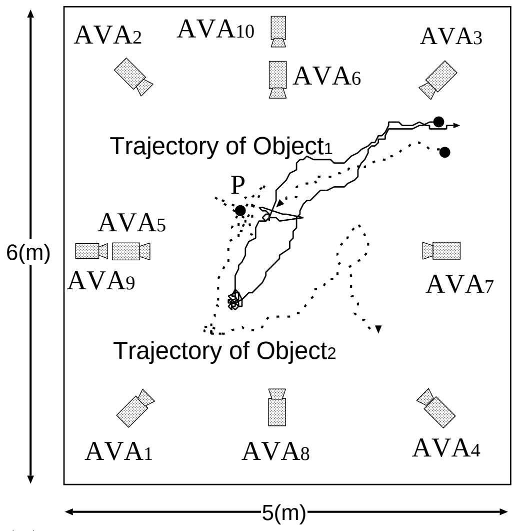
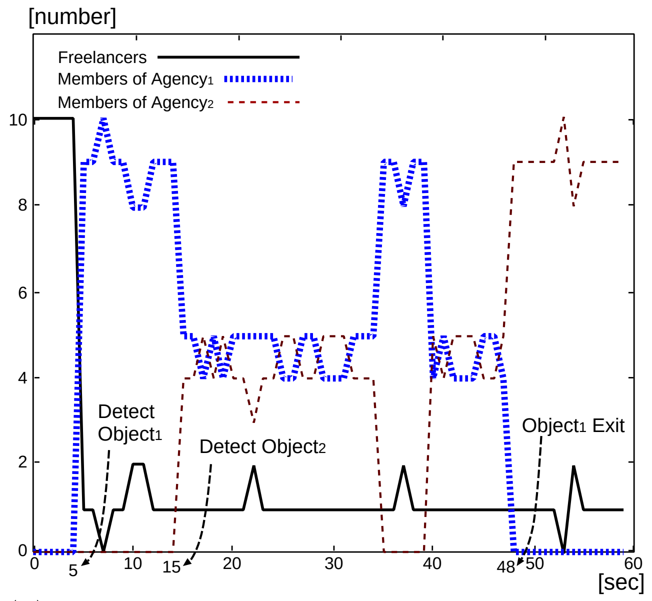


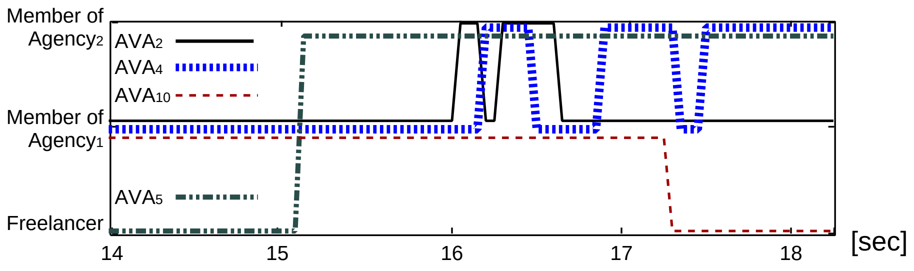
Figure 5.29: Experiment 1: Transitions of AVA roles and agency formations.



(a) Target motion trajectories reconstructed by agencies



(b) The number of AVAs that work for each role (‘*search*’ and ‘*tracking* of each object’)



(c) Histories of AVAs’ roles

Figure 5.30: Experiment 1: Experimental results.

and object identification between two agencies was successful. As a result, two agencies merged together by the agency unification, and became agency$_1$ (Figure 5.29 (f)).

g: After two objects were apart from each other, some AVAs could detected object$_1$ and object$_2$ severally (Figure 5.28, 2-g, 5-g, 7-g, 9-g). Then, the agency that tracked a newly detected object was generated. At this time, the information of object$_2$ obtained in the past was read. The newly generated agency 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$_2$ (Figure 5.29 (g)).

h: Object$_1$ came to location X again (Figure 5.29 (h)), and would disappear.

i: After agency$_1$ dissolved, all the AVAs except for AVA$_4$ was tracking object$_2$ as the member-AVAs of agency$_2$ (Figure 5.29 (i)).

Figure 5.30 (a) shows the trajectories of the target objects, which are reconstructed by the agencies. In this figure, the reconstructed 3D positions are projected onto the floor. When the agency spawning for object$_2$ caused, tracking of object$_2$ was started at location P.

Figure 5.30 (b) shows the histories of the number of (1) the freelancer-AVAs, (2) the member-AVAs tracking object$_1$ and (3) the member-AVAs tracking object$_2$. The horizontal and vertical axes indicate the time and the number of AVAs undertaking each role, respectively. This graph shows the system states at 1[sec] intervals. Figure 5.30 (c) shows the histories of the AVA's roles. The horizontal and vertical axes indicate the time and the AVA's role. Bumps in Figure 5.30 (b) and (c) indicate the temporal states of the system while each AVA was changing its role depending on the target motions. From these results, we can see that the system as a whole worked to cope with the dynamic situations in the scene by dynamically changing the role of each AVA.

Note that each member-AVA did not always gaze at the target object of its agency as we can see in Figure 5.28. That is, just after the AVA transferred to the new agency, it still observes the target object of the former agency because it has not finished changing its gazing direction towards the new target object. For example, AVA$_2$ and AVA$_4$ belonged to agency$_1$ at (h) in Figure 5.29. They had to, therefore, gaze at object$_1$. Both of them, however, observed object$_2$. This is because it was immediately after AVA$_2$ and AVA$_4$ transferred from agency$_2$ to agency$_1$. Figure 5.31 shows images observed by AVA$_4$ before and after (h) in Figure 5.29. 4-H in Figure 5.31 is the same image with 4-h in Figure 5.28. When AVA$_4$ captured 4-A, AVA$_4$ was ordered to transfer to agency$_1$. We can see that AVA$_4$ was changing its gazing direction towards object1.

Next, we show the result of the second experiment. In this experiment, we verified the efficacy of the task representation.

Figure 5.32 shows the partial image sequences observed by each AVA. The arrangement of images is the same as that of the first experiment. Figure 5.28 shows images taken by AVA$_1$, AVA$_2$ and AVA$_8$ as examples. Figure 5.33 shows the role of each AVA and the formation of each agency.

Figure 5.34 (a) shows the trajectories of the target objects, which are reconstructed by the agencies. Figure 5.34 (b) shows the histories of the number of (1) the freelancer-AVAs, (2) the member-AVAs tracking object$_1$ and (3) the member-AVAs tracking object$_2$. This graph shows the system states at 1[sec] intervals. The system detected object$_1$ and object$_2$ at 7 seconds and 28 seconds, respectively. The system kept tracking both objects. Figure 5.34 (c) shows the histories of the AVA's roles. The horizontal and vertical axes indicate the time and the AVA's role. Since two objects had the different object-importances, the numbers of the member-AVAs in agency$_1$ and agency$_2$ were different from each other.

As we can see, the dynamic interactions among AVAs and agencies enable the system to persistently track multiple objects taking into account the given task.

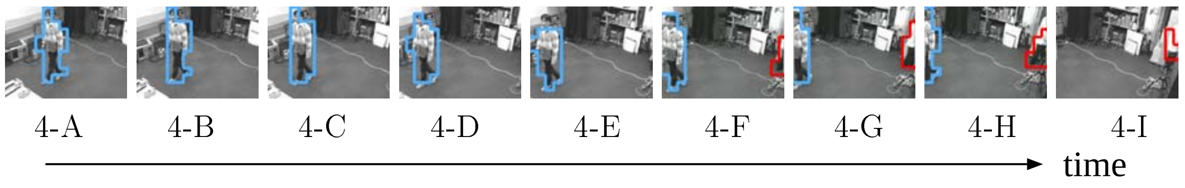


Figure 5.31: Experiment 1: AVA$_4$ was changing its gazing direction to detect the new target object (i.e., object$_1$ enclosed with red line). These images were taken at about 0.1 [sec] intervals on average.

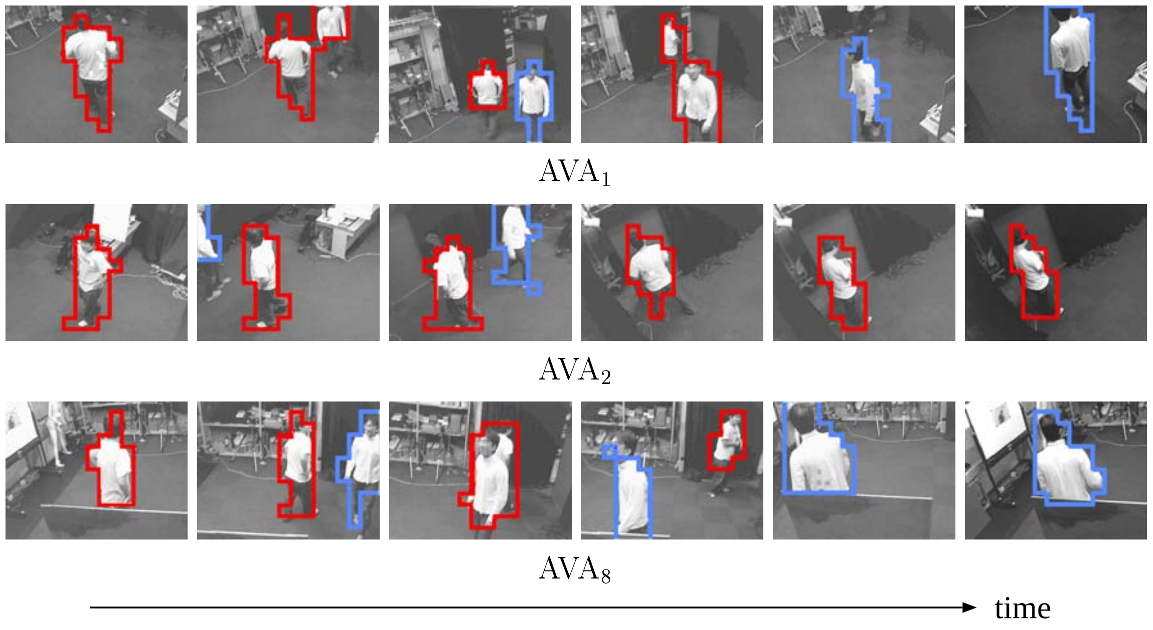


Figure 5.32: Experiment 2: Partial image sequences observed by AVAs. These images were taken at about 1[sec] intervals.

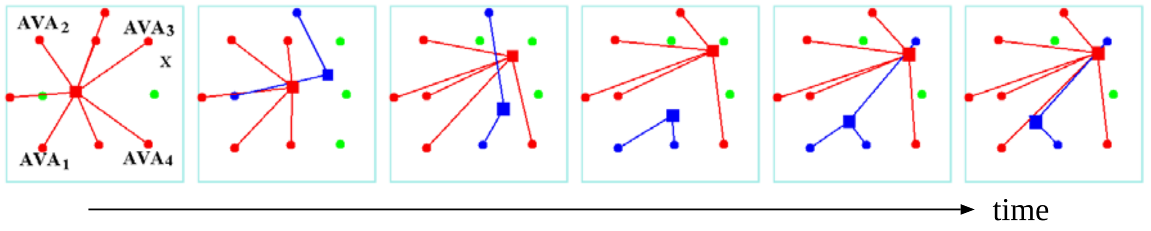
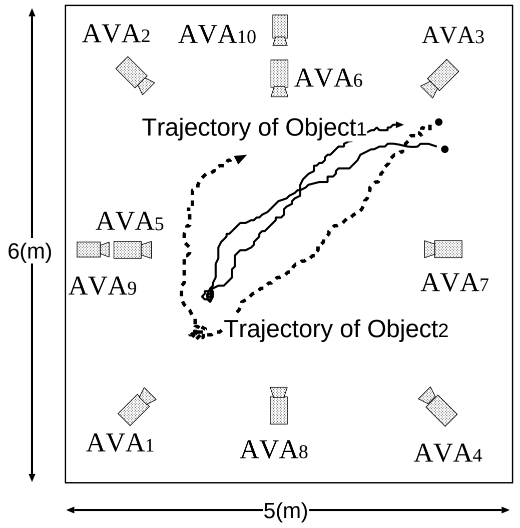
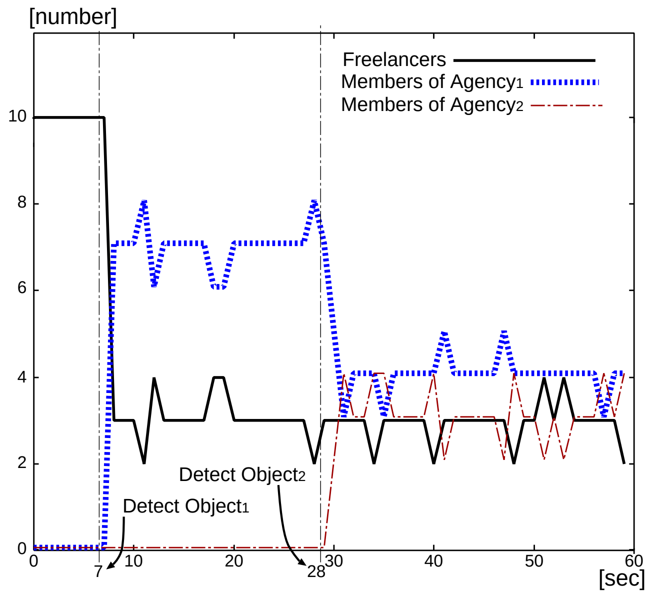


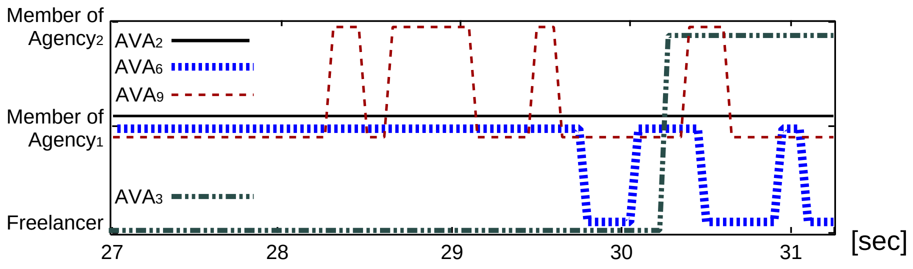
Figure 5.33: Experiment 2: Transitions of AVA roles and agency formations.



(a) Target motion trajectories reconstructed by agencies



(b) The number of AVAs that work for each role (‘*search*’ and ‘*tracking* of each object’)



(c) Histories of AVAs’ roles

Figure 5.34: Experimental results 2: Experimental results.

5.6 Concluding Remarks

This chapter proposed a real-time multi-target tracking by employing the concept of CDV. Our system has the following properties.

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

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

Inter-agency layer: In order to adaptively restructure agencies taking into account target motions, the agencies exchange the agency information with each other.

- Employing the dynamic memory architecture realizes the dynamic interactions in each layer without synchronization. The system is endowed with a high reactiveness.

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

Experimental results demonstrated the practical effectiveness of our system.

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. Followings are justifications of each layer for its existence.

Intra-AVA layer: To perform versatile and complex behaviors, an intelligent autonomous agent should consist of several functional modules required for the task

Intra-agency layer: Agents, all of which aim at the same purpose, should form an agency to cooperatively work together. The agency should be personified as a delegate of a corporate group in order to work without contradictions between agents.

Inter-agency layer: For agencies to cooperatively work by negotiations, they should interact with each other.

Chapter 6

Incremental Observable-area Modeling for Cooperative Tracking

6.1 Sharing Information for Cooperative Tracking

For multi-agent systems, the knowledge of partners' abilities is required to realize cooperative action among the agents whatever task is defined. In particular, for cooperative object tracking, every agent should know the area in the scene that is observable by each agent. Each agent should then decide its target object and gazing direction taking into account the adaptive role assignment among all the agents.

In this chapter, in order to augment our cooperative tracking system with multiple AVAs, we put our focus upon the sharing knowledge of all the AVAs' abilities (i.e., observable area in the scene) for the efficient cooperative object tracking and scene observation. Our system incrementally acquires the observable-area information of each AVA, and enables the AVAs to dynamically and appropriately change their roles by taking into account all the AVAs' observable areas.

6.1.1 Adaptive Role Assignment

In our cooperative tracking system, all member-AVAs can keep tracking a focused target object without being disturbed by obstacles or other moving objects through the compulsory gaze navigation by the agency (i.e., the agency maintenance protocol in the intra-agency layer). They continue to obey the gaze navigation even if they cannot observe the target object due to obstacles. The AVA that cannot observe the target object, however, should change its role for increasing the efficiency of the total system. For example, the following functions can be considered for such an AVA (Figure 6.1):

1. AVA_1 predicts the position where the target object will appear within its observable area, then changes its gazing direction to ambush the target object (Figure 6.1, 1.).

2. AVA_1 gazes the area where none of other AVAs observes to find another object (Figure 6.1, 2.).

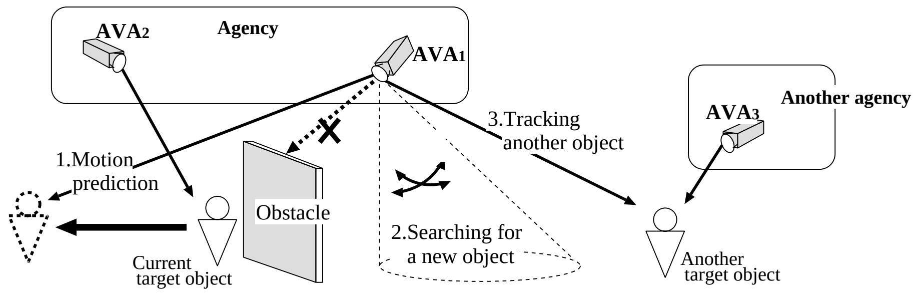


Figure 6.1: Dynamic role assignment to AVA.

3. If AVA$_1$ can observe the target object of another agency, it joins this agency (Figure 6.1, 3.).

Our system realizes the dynamic role assignment by the following cooperative tracking protocols:

- Agency maintenance: When a member-AVA loses track of the target object, the agency navigates the gaze of the member-AVA towards the target object. Nevertheless, if the member-AVA has not detected the target object for a long time, the agency releases it. In this case, the AVA is assigned to search for a new object.
- Agency restructuring: The agencies exchange their member-AVAs to cope with the target motions.

To implement these protocols, the target information of each AVA is employed.

Shared information 1 (Target information): (1) The target object of each AVA. (2) 3D position of each target object.

In our system, this information is dynamically exchanged between agencies. The target information, however, is not enough to effectively realize the above functions 1, 2 and 3 because the system does not take into account the visible/invisible information of the target object in the scene.

The above problem is caused by lack of the information about the 3D geometric configurations of the scene. That is, the system cannot know whether the target object is interfered by obstacles or object detection fails due to processing errors, when the target object is not detected in the observed image. To solve this problem we expand our cooperative tracking system to identify each AVA's visible/invisible area in the scene and employs this information for the adaptive role assignment. In the proposed system, the system gathers the visible/invisible area information of all AVAs during tracking. This information is shared by AVAs and agencies.

Shared information 2 (Observable-area information): The information about the observable area of each AVA.

By referring the observable-area information, (1) an agency can dynamically assign the appropriate role to each member-AVA and (2) a freelancer-AVA can search its observable area for an object. These properties allow the system to realize the functions 1, 2 and 3 mentioned above for increasing the efficiency of the total system.

In the proposed system, the above shared information 1 and 2 is taken into account simultaneously. Hence, the system cooperatively tracks multiple objects with referring the geometric configurations between each target object and the environment of the observation scene.

In what follows, we first describe (1) how to acquire the visible/invisible information of each AVA and (2) how to manage the acquired information in Section 6.2. We then present communication protocols for cooperative tracking with the observable-area information in Section 6.3. Finally, experimental results demonstrate the effectiveness of the cooperation among the AVAs with the help of the proposed observable-area information in Section 6.4.

6.2 Observable-area Model

6.2.1 Observable-area Model for Adaptive Role Assignment

In the proposed system, an agency refers both the visible/invisible area information and the trajectory of the target object in order to examine whether or not each AVA can observe the target object. Based on the result of this examination, each AVA is assigned to appropriate role.

To realize efficient cooperative tracking by the total system, an appropriate role of each AVA should be determined by considering the visible/invisible information of all the AVAs comprehensively. The system, therefore, manage the visible/invisible area information of all the AVAs collectively as a scene model. We call this scene model an *Observable-area Model*.

6.2.2 Data Structure of the Observable-area Model

We adopt the octree representation[AV89] for the data structure of the observable-area model (Figure 6.2).

Octree representation: An octree is a tree data structure. Starting with an upright cubical region of space, the octree space is recursively decomposed into eight cubes called octants if the label in each octant should be different. The information of each 3D area (e.g., visible/invisible information) is shown by the label in the each octant.

We have the following advantages in employing the octree representation as a visible/invisible area model:

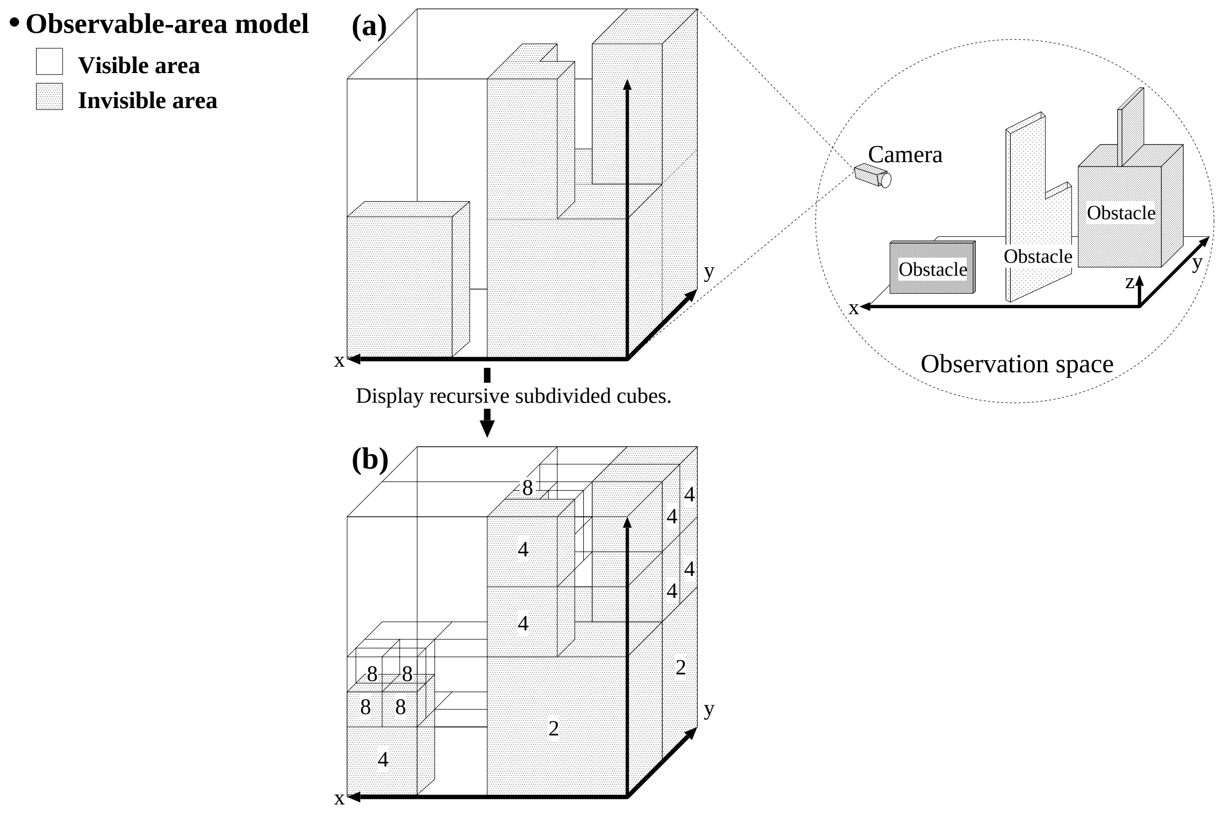


Figure 6.2: Data structure of the observable-area model.

- The octree representation allows us to reduce the amount of data, since the visible/invisible area usually masses in the scene: while a large octant is generated for widely spread visible/invisible area, many small octants are generated if the space are tangled with visible and invisible areas.

- Easiness of resizing cubes in the octree allows us to localize the resolution of the observable-area model. That is, we can adjust the resolution depending on the distance from the camera, the edge of each obstacle, and so on.

Figure 6.2 illustrates an example of the observable-area model. Depending on the geometric configuration between the camera and obstacles, visible and invisible areas of the camera are determined. In Figure 6.2 (a), white and gray regions indicate the visible and invisible areas of the camera, respectively. In Figure 6.2 (b), each octant in the visible/invisible area is shown. It is confirmed that each cube is divided into octants. Let a cubical region corresponding to the entire scene be a square 1 on a side. The number in each octant denote the reciprocal of its side's length.

In the proposed model, the above observable-area information is generated for each camera, and all the information is managed as the observable-are model in the scene: Each cube in the octree model includes visible/invisible information for each camera. This is a difference between the proposed model and common scene models that directly include the information of a scene.

In each cube in the octree model, the following three kinds of the visible/invisible labels are attached to each AVA:

UNDEFINE The system has not identified whether or not the AVA can observe the area.

VISIBLE The AVA can observe the area.

INVISIBLE The AVA cannot observe the area.

The visible/invisible labels in octants of each cube are examined when the system determines whether or not each cube is decomposed into octants.

6.2.3 Generating Visible/Invisible Information

In the cooperative tracking system proposed in the last chapter, object identification and 3D position reconstruction are realized by incorporating 3D view lines detected by each AVA. In the system with the observable-area model, however, the agency computes the intersection of the visual cones, each of which is determined by the projection center of a camera and the detected region in the observed image. If the intersection exists among the visual cones, all the detected regions corresponding to the visual cones are considered as the same object. Moreover, the 3D position of the object can be obtained, since the computed intersection corresponds to the volume of the object. We call this volume reconstruction method a *Volume Intersection Method*.

Since the volume intersection method needs the visual cones observed by AVAs, each AVA have to inform its agency manager of the observed visual cones. Each AVA sends an agency the detected object region in the observed image instead of the information about the visual cones themselves. The agency can estimate the visual cones from the received information of the detected object regions and the external parameters of the camera[1] .

In object identification and 3D reconstruction with the volume intersection method, if the image observed by AVA$_w$ is actually used for the volume reconstruction, the system can then identify the area where the detected object exists to be visible from AVA$_w$. Otherwise, the area is identified to be invisible from AVA$_w$. In an example illustrated in Figure 6.3, while the volume of the object is reconstructed from the detected results of AVA$_1$, AVA$_2$ and AVA$_3$, AVA$_4$ does not detect the same object in its observed image. In this case, the reconstructed object region is considered to be visible from AVA$_1$, AVA$_2$ and AVA$_3$.

[1] The external camera parameters are included in a message from an AVA to an agency (see Table 5.1).

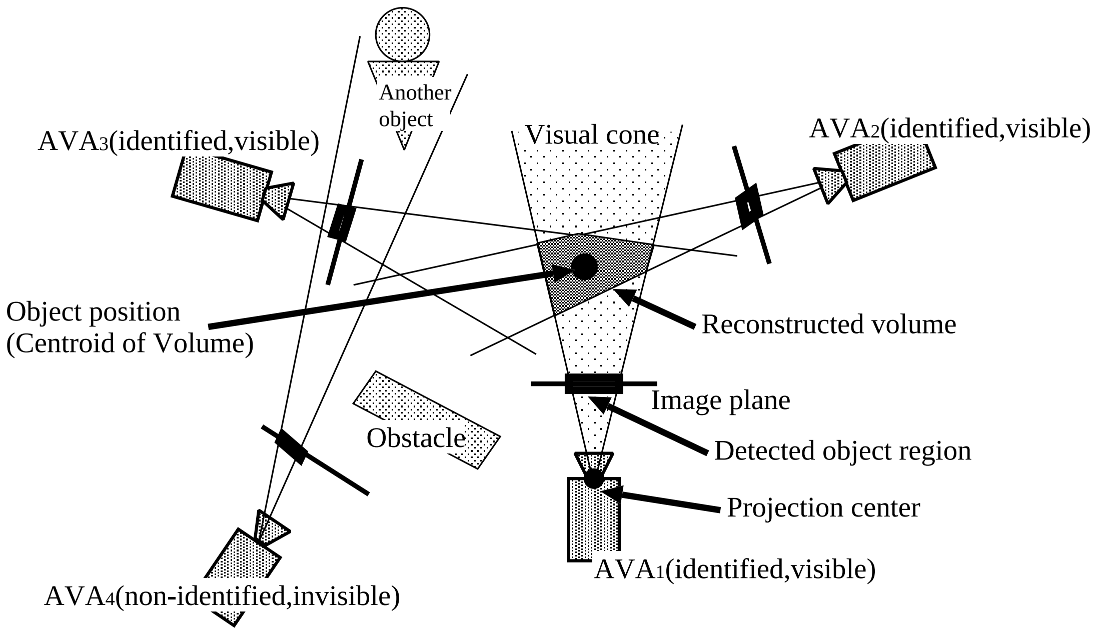


Figure 6.3: Volume and position reconstruction of the object and generating the visible/invisible area information by the volume intersection method.

The agency can incrementally generate the visible/invisible information while the object is being tracked by member-AVAs. The proposed system, therefore, changes its behavior during tracking as follows:

1. The system first keeps tracking target objects without any information about the scene.
2. During tracking, the system updates the observable-area model.
3. By employing the acquired observable-area model, the system assigns an appropriate role to each AVA.

With this scheme, the system can increase its tracking efficiency as the observable-area model progresses.

6.2.4 Updating Observable-area Model based on Visible/Invisible Information

After new visible/invisible information is obtained as mentioned in Section 6.2.3, all the visible/invisible labels in the new information are respectively compared with those in the observable-area model to update the observable-area model. If the label in the new

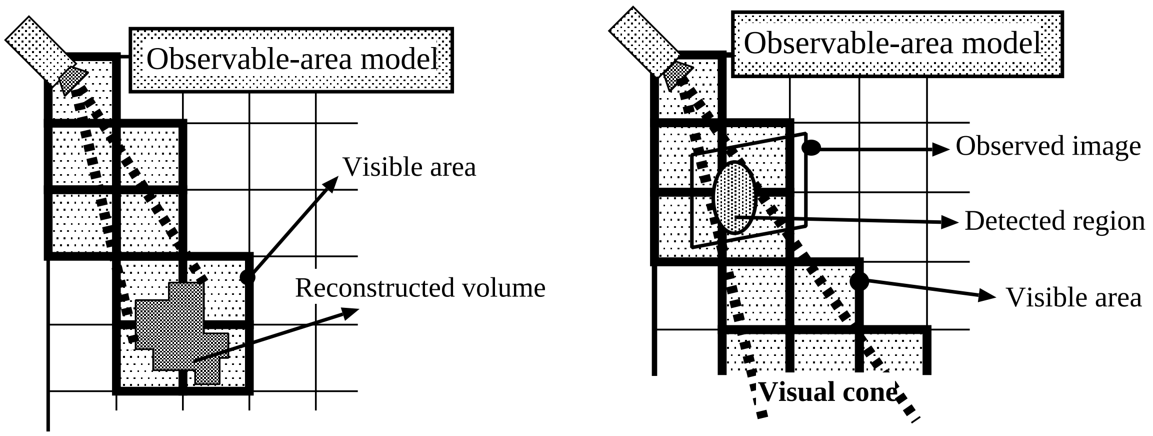


Figure 6.4: Visible area propagation (Left: Volume is reconstructed, Right: Volume is not reconstructed).

information is different from that in the cube whose position corresponds to the new information in the observable-area model, this cube is decomposed into octants. This decomposition is executed as long as the following two conditions are both satisfied.

1. The label in the new information is different from that in the observable-area model.
2. The following inequality is true:

$$\frac{distance}{focallength} < \frac{\mathrm{CONST}_r}{2^{depth}},$$

   where *distance* is the length from the camera to the area, *focallength* is the focal length of the camera and *depth* is the depth of the octree. CONST_r denotes a constant that determines the minimum size of the cube in the octree.

Since the resolution of the reconstructed volume depends on *focallength* and *distance*, the number of decomposition is defined by the above inequality.

After the decomposition, each cube is given the visible/invisible label. If all the labels of the octants are the same, these octants are unified to decrease the amount of the cubes.

Furthermore, the visible information is propagated to facilitate generating the observable-area model. Two cases exist for the propagation of the visible area (Figure 6.4).

Case A (Figure 6.4, left): The volume of the object has been reconstructed. In this case, each cube in the observable-area model, which corresponds to the area where the reconstructed volume exists, is identified to be visible. We can then identify the area between the object and the camera to be also visible from this camera if it can observe the object. The cubes between the object and the camera are, therefore, updated as the visible area.

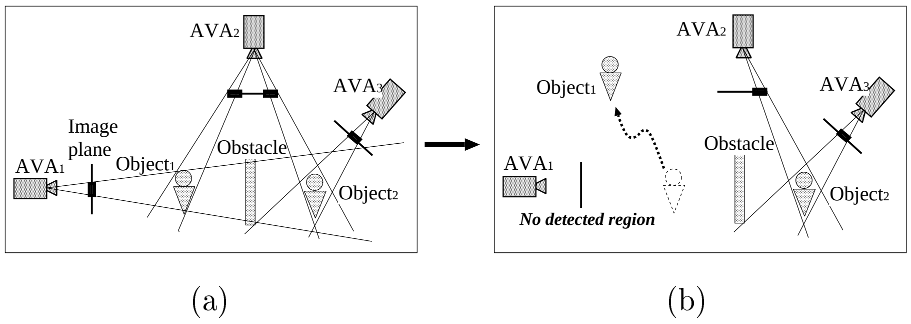


Figure 6.5: Updating the visible/invisible area information.

Case B (Figure 6.4, right): The volume of the object has not been reconstructed. In this case, the whole area included in the visual cone is updated as the temporary visible area except the area that has been already identified to be visible or invisible. Due to this function, there is a possibility of falsely attaching the temporary visible label to the area that is not yet estimated but actually invisible. To correct the observable-area model by the subsequent observation, the temporary visible label is updated when the area is identified as visible or invisible.

In the above case B, wrong VISIBLE labels might be possibly attached to the areas that are (1) invisible from the camera and (2) beyond the object position. This results in regarding the invisible area as the visible area by mistake. To correct this misunderstanding by the subsequent observations, the system has to rewrite the label in the observable-area model. We give INVISIBLE higher priority than VISIBLE when the observable-area model is updated. Therefore, if the labels in the observable-area model and the newest visible/invisible information are VISIBLE and INVISIBLE, respectively, VISIBLE in the observable-area model is updated in INVISIBLE.

Updating VISIBLE in INVISIBLE is required when the system corrects the visible/invisible information obtained by the volume intersection method, too. In an example illustrated in Figure 6.5 (a), AVA$_1$ cannot observe object$_2$ because of being disturbed by the obstacle. However, the system considers the volume area of object$_2$ to be visible from AVA$_1$ by mistake, since (1) the visual cone of AVA$_1$ includes the volume area of object$_2$ and (2) the volume area of object$_2$ is reconstructed by intersecting the visual cones of AVA$_2$ and AVA$_3$. This misunderstanding is corrected by the subsequent observation illustrated in Figure 6.5 (b). When object$_1$ moves away, AVA$_1$ cannot detect any object in the observed image. The system then considers the volume area of object$_2$ to be INVISIBLE from AVA$_1$ and update the observable-area model based on the newest visible/invisible information. Thus, updating the visible/invisible label allows the system to compensate for limitations of obtaining visible/invisible information with the volume intersection method.

Based on the above discussion, to the three kinds of labels, the following order of priority is applied in case of substituting the label for the cube.

$$\text{INVISIBLE} > \text{VISIBLE} > \text{UNDEFINE}$$

For example, VISIBLE and UNDEFINE labels are not recorded in the cube that has been identified as INVISIBLE. In the proposed model, the observable-area model starts from a single cube whose label is UNDEFINE, and the system divides it into octants and attaches visible/invisible labels in accordance with this priority.

6.2.5 Managing the Observable-area Model

The observable-area model should be referred by each agency to assign appropriate roles to member-AVAs. Since the observable-area model is made from the visible/invisible information generated by the agency, the agency manager can obtain the model by itself. Accordingly, the agency manager can refer the model immediately without increasing the network load.

However, the following problems result in difficulty in severally managing the observable-area model obtained by each agency:

- Since the agency manager is dynamically generated and eliminated, it cannot continue to manage the observable-area model. When the agency disappear, it should entrust the observable-area model obtained by itself to other agencies.
- To determine a role of each AVA by considering the visible/invisible information of all AVAs, all agencies have to exchange the observable-area models of their member-AVAs with each other.

Thus, these problems cause repeated network congestion. In the proposed system, therefore, the agency delegates the following tasks to a model-management module, *Observable-area Model Manager*.

- Keeping and updating the observable-area model.
- Planning the appropriate roles of AVAs by referring the observable-area model.

The system has a single observable-area model manager, and the manager receives all the information from every agency at each frame to have the observable-area model as illustrated in Figure 6.6.

By leaving the management of the observable-area model in the charge of the observable-area model manager,

- increasing a network-load can be avoided, and
- each agency can reactively cope with object motions for tracking because it is released from managing the observable-area model.

The performance of the system is, therefore, improved.

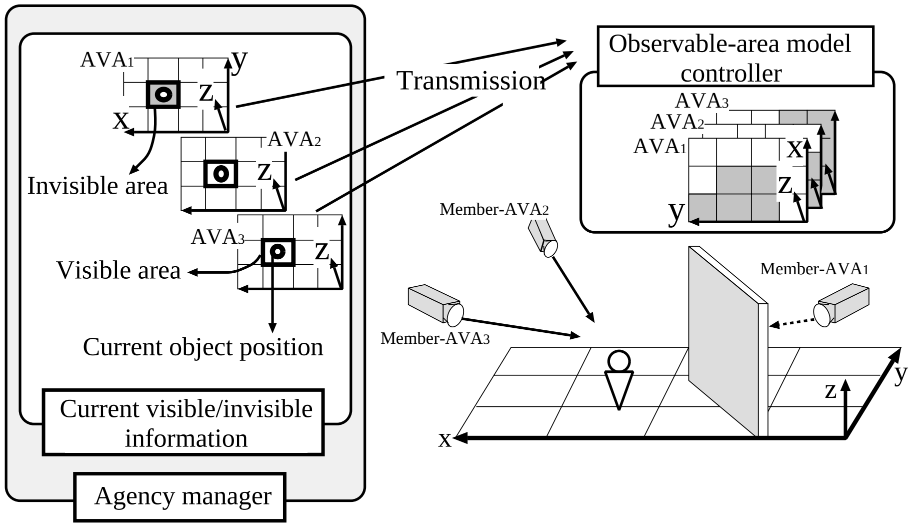


Figure 6.6: Managing the observable-area model.

6.3 Cooperative Tracking with the Observable-area Model

This section addresses communication protocols for (1) updating the observable-area model and (2) the role assignment to each AVA. Figure 6.7 illustrates the information exchange between the agency manager and the observable-area model manager.

6.3.1 Information flow from the AVA/Agency to the Observable-area Model Manager

At each frame, the agency manager transmits the following three messages to the observable-area model manager:

- **CURRENT ROLE :** For the observable-area model manager to plan the appropriate role for each AVA, the information about the current role of each AVA is required. The agency manager, therefore, reports its member-AVAs to the observable-area model manager.
- **OBJECT POSITION :** The transmitted 3D position of the object allows the observable-area model manager to keep the object's motion trajectory.
- **VISIBLE/INVISIBLE MAP :** The transmitted visible/invisible information for each AVA allows the observable-area model manager to update the observable-area model.

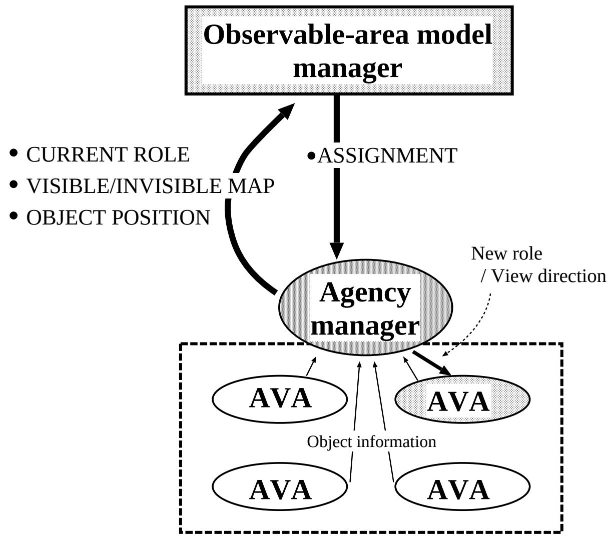


Figure 6.7: Message flows between an agency manager and the observable-area model manager. Each arrow indicates a message flow.

Similarly, the freelancer-AVA sends the observable-area model manager the CURRENT ROLE message, which informs that this AVA is working as the freelancer-AVA.

Note that, in our cooperative tracking system, both the agency and the freelancer-AVA broadcast messages including the CURRENT ROLE and OBJECT POSITION information as mentioned in Section 5.3.5. The observable-area model manager can obtain these information by receiving the broadcasted messages. All of the above three messages can be transmitted to the observable-area model by sending the VISIBLE/INVISIBLE MAP message in surplus.

6.3.2 Information flow from the Observable-area Model Manager to the AVA/Agency

The observable-area model manager decides the role assignment to each AVA every after updating of the observable-area model based on the information sent from the agency and the freelancer-AVA. If the observable-area model manager finds a new appropriate role assignment, the following message is transmitted to the agency manager or the freelancer-AVA:

- **ASSIGNMENT :** (1) The ID of the AVA that is assigned a new role[2] and (2) the details of the new role, are included in the message. Some actual examples of the new roles are shown in Figure 6.1).

[2] This information is required only if the ASSIGNMENT message is sent to the agency manager.

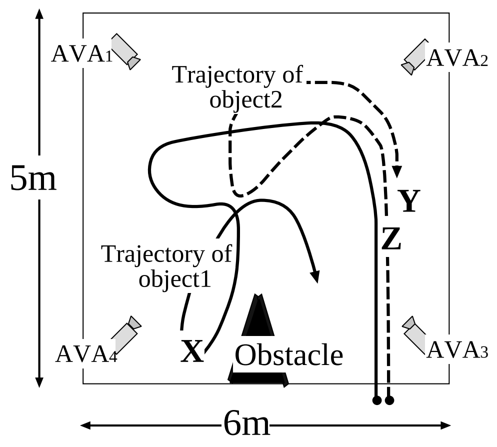


Figure 6.8: Experimental environment.

If the agency receives the ASSIGNMENT message and approve the effectiveness of the role assignment, the agency assigns the new role to the AVA. Since only the agency is allowed to order its member-AVAs, the system can avoid sending the different roles to a member-AVA simultaneously.

If the freelancer-AVA receives the ASSIGNMENT message, on the other hand, it judges the effectiveness of the role assignment by itself and decides whether or not it starts the assigned role.

Note that the observable-area model manager never *order* but *propose* a new role. Following are the reasons:

- Since each AVA and agency are an autonomous agent and group, respectively, they should determine their behaviors by themselves.
- While each AVA and agency determine their roles depending on the information managed by themselves, the observable-area model manager receives the required information from them through the network and assign a new role with referring the received information. This results in difficulty in reactive planning by the observable-area model manager.

6.4 Experiments

We experimented to verify the effectiveness of the proposed observable-area model for cooperative tracking. Our experimental results demonstrated the improvement in cooperation of AVAs while tracking.

We conducted our experiments in the environment shown in Figure 6.8. System organization is the same with that of the cooperative tracking system shown in chapter 5 except the number of AVAs; here we employed four AVAs.

In this environment, object 1 came into the observation space, and stayed for a while at the location X after moving along the trajectory. Note that AVA_3 could not observe object 1 when it was at the location X. Next, object 2 moved along the trajectory and stopped at the location Y. After that, object 1 started moving again.

The following three tracking systems were employed for comparative study:

System 1: Tracking system without cooperative action, namely each AVA track a target object independently without forming an agency.

System 2: Cooperative tracking system without the observable-area model.

System 3: Cooperative tracking system with the observable-area model.

Both objects moved along the almost same trajectories severally when each system worked. For cooperative tracking systems 2 and 3, we gave the following parameters to the systems as the task specification:

Task specification:

Task-constraint: The tracking-level and search-level were 1.0 and 0.0, respectively.

Object-importance: Values for object 1 and object 2 were 1.0 and 0.1, respectively.

Interval for releasing a member-AVA: If a member-AVA cannot detect its target object for 10[sec], the agency releases it based on the agency maintenance protocol.

6.4.1 Tracking Results

Figure 6.9, 6.10 and 6.11 show examples of image sequences observed by AVA_2 and AVA_3. Figure 6.9, 6.10 and 6.11 are captured by systems 1, 2 and 3, respectively. The size of each image is 320×240 [pixel]. The enclosed regions with the white and black lines in the images indicate the detected regions of object 1 and object 2, respectively.

Without cooperative actions (system 1), each AVA first searched for an object independently (a1, a2 and b1, b2). When the AVAs found object 1, they regarded it as the target object and started tracking it independently (a3, a4 and b3, b4). When object 1 was obscured by the obstacle (b5, b6), AVA_3 started searching for an object immediately (b7, b8). Then, AVA_3 detected object 2 and kept tracking it (b9 $\sim$ b14). Since AVA_3 could not notice that object 1 came into the observable are of AVA_3 again, it continued to gaze at object 2. Moreover, AVA_2 also changed its target object to object 2 when object 1 and object 2 became close (a11, a12), because their projected regions in the observed image overlapped each other and then AVA_2 considered object 2 to be its target object.

Without the observable-area model (system 2), after searching (c1, c2 and d1, d2), all the AVAs detected object 1 and regarded it as the target object, and then began to cooperatively track object 1 (c3, c4 and d3, d4). However, AVA_3 kept gazing at the direction of the 3D position of object 1 transmitted from the agency manager, though AVA_3 could not observe it due to the obstacle (d5 $\sim$ d13). This is because the agency did

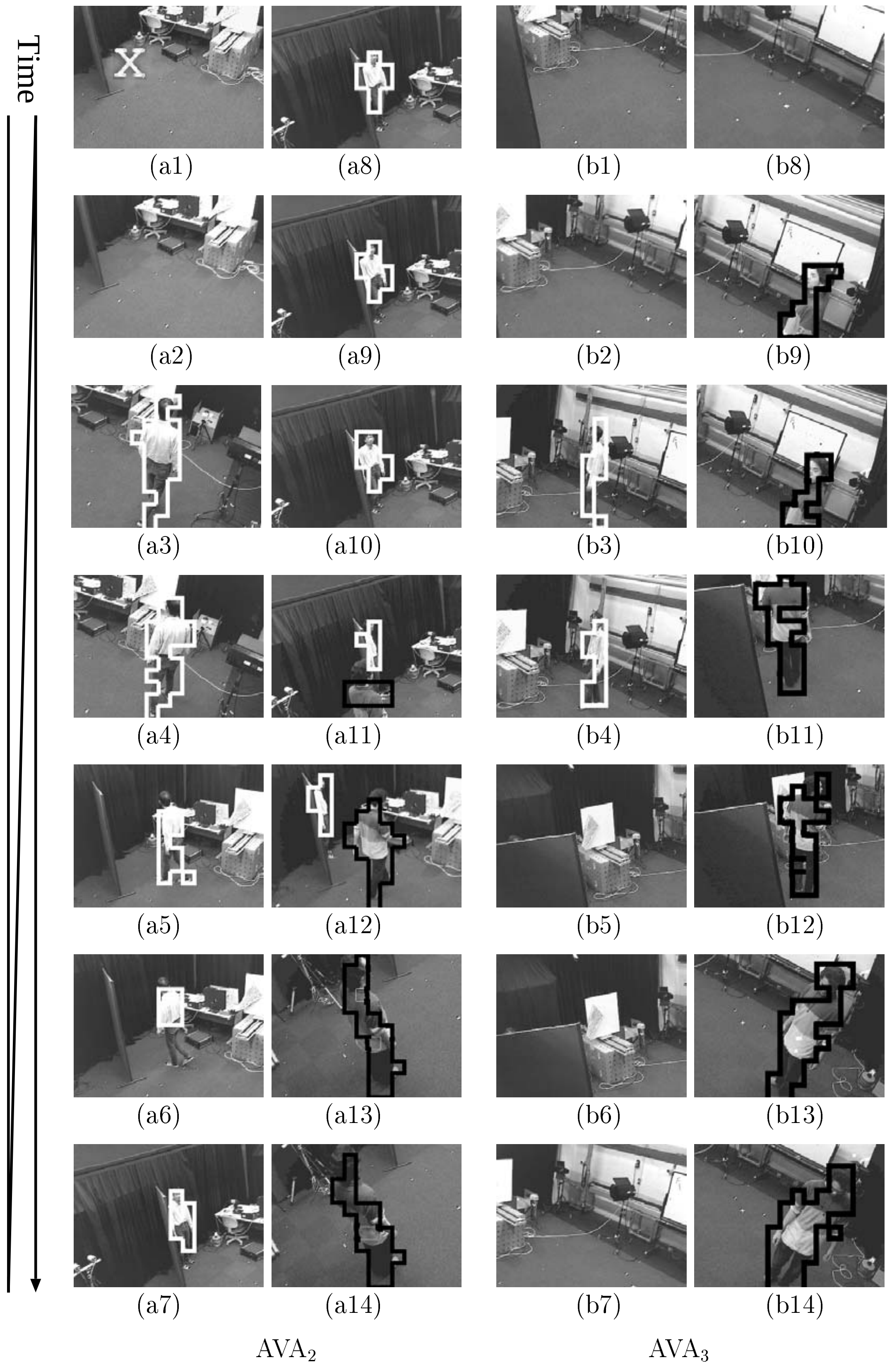


Figure 6.9: Partial images observed by system 1.

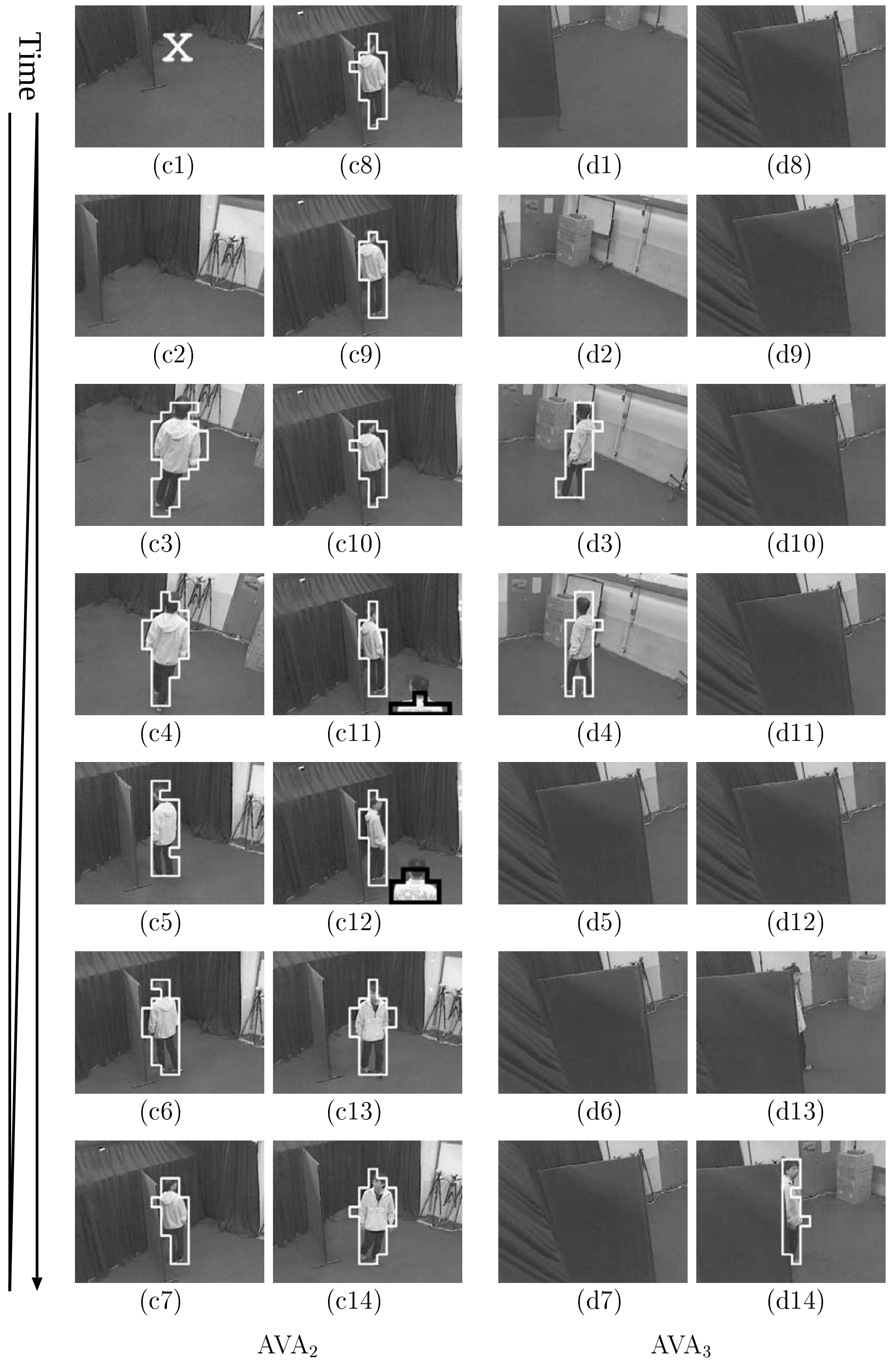


Figure 6.10: Partial images observed by system 2.

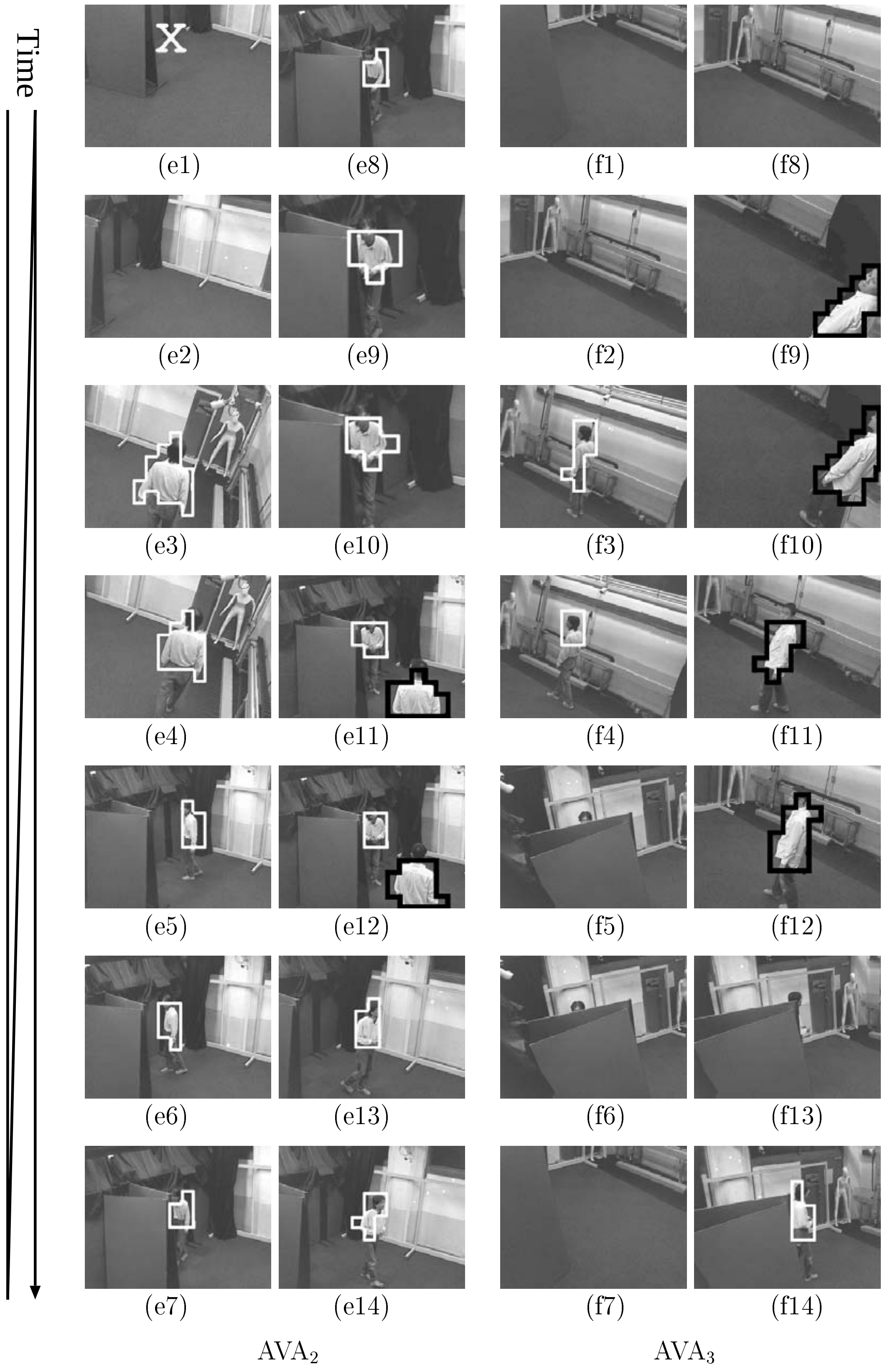
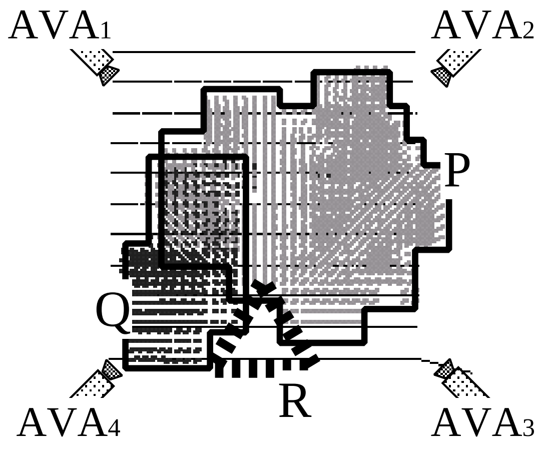
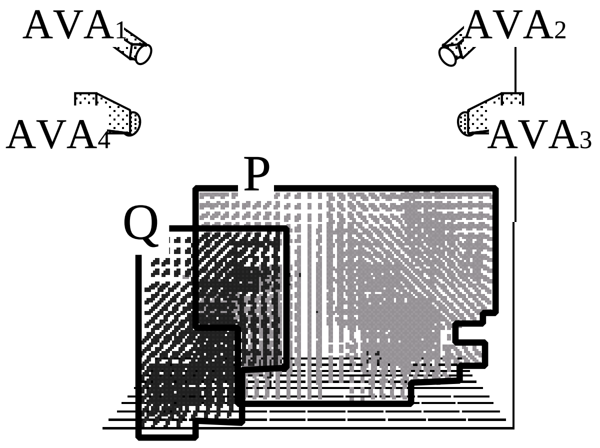


Figure 6.11: Partial images observed by system 3.



(a) Top View



(b) Side View

Figure 6.12: Acquired observable-area model: this data shows only the visible/invisible area information of AVA_3. Black and gray regions indicate the visible and invisible areas, respectively.

not release AVA_3 for 10[sec] (i.e., the interval for releasing a member-AVA based on the agency maintenance protocol). Therefore, AVA_3 could not obtain the efficient information about the target objects from the observed images. The other AVAs, on the other hand, kept tracking object 1 (c5 $\sim$ c14). Note that AVA_2 kept gazing at object 1 that was regarded as the target object even if object 2 was also detected in the observed image (e11, e12). This is because

- temporal object identification established by the agency assists each member-AVA to identify the target object, and
- since the object-importance of object 1 is much larger than that of object 2, the agency tracking object 1 did not release its member-AVAs, and the new agency was not generated.

With the observable-area model (system 3), each AVA first searched for an object independently (e1, e2 and f1, f2) and then cooperatively tracked object 1 while belonging the agency_1 (e3, e4 and f3, f4) (similar to the cooperative tracking without the observable-area model). AVA_3 started, however, searching for another object (f7, f8) immediately after AVA_3 could not detect object 1 (f5, f6). This reactive role assignment could be realized by the assistance of the observable-area model manager. AVA_3 then detected object 2 at the location Z and generated agency_2 for tracking object 2. AVA_3 then started tracking object 2 independently (f9 $\sim$ f12). After that, when object 1 started walking and became close to the area where AVA_3 could observe, the observable-area model manager

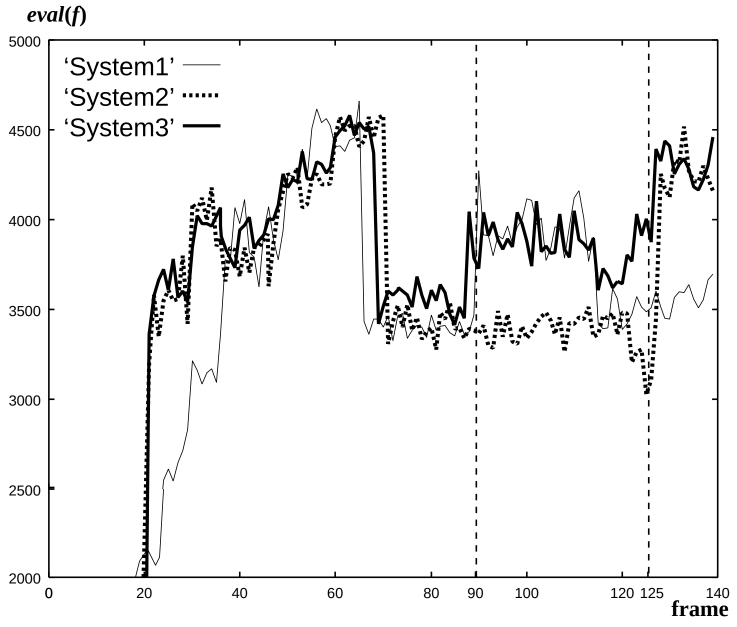


Figure 6.13: Histories of the evaluation functions.

instructed AVA$_3$ that it could observe object 1. Since the object-importance of object 1 is much larger than that of object 2, AVA$_3$ transferred from agency$_2$ to agency$_1$ and started again tracking object 1 (f14, f14). Then, agency$_2$ disappeared. The visible/invisible area information for AVA$_3$ is shown in Figure 6.12 where the two different views are illustrated. P and Q respectively indicate the visible and invisible area. Note that we see the obstacle area at R[3] .

6.4.2 Performance Evaluation

We quantitatively evaluated how effectively the system accomplishes the given task. The following criterion $eval(f)$ is employed for the evaluation:

$$eval(f) = \sum_{n=1}^{N_m} obj(n) \times area(n, f), \tag{6.1}$$

where

- N_m denotes the number of the member-AVAs,
- $obj(n)$ denotes the object-importance of AVA$_n$'s target object, and

[3] We can also estimate the 3D geometric information of the scene by integrating the observable-area information of each AVA.

- $area(n, f)$ denotes the area size (the number of detected pixels) of AVA_n's target object at f-th frame.

Figure 6.13 illustrates the histories of $eval(f)$ while system1, 2 and 3 were working. We can see that the evaluated result of system 3 surpasses those of the other systems at almost all frames. The average values of $eval(f)$ for system 1, 2 and 3 were 3026, 3143 and 3403 respectively.

We can see the following observations from Figure 6.13 (the histories of $eval(f)$ obtained by system 1, 2 and 3 are called graph 1, 2 and 3, respectively):

- Comparing graph 1, graph 2 and 3 rapidly rises after each system first detected object 1. This is because the gaze navigation by the agency is executed in systems 2 and 3.

- Between frames 90 and 125, graph 1 and graph 3 are larger than graph 2. This is because AVA_3 gazed at the obstacle and could not detect any target object in system 2.

- Between frames 125 and 140, graph 2 and graph 3 are larger than graph 1. This is because AVA_2 kept tracking object 2 whose object-importance was lower than that of object 1.

From the comparative experimental results shown in Section 6.4.1 and Section 6.4.2, we may conclude that the proposed model improves the effectiveness of cooperation in tracking.

6.5 Concluding Remarks

We proposed the incremental observable-area model for cooperative tracking. This model allows a tracking system to dynamically assign the appropriate role to each AVA.

In the proposed system, the visible/invisible area information of each AVA is acquired based on the reconstructed 3D information of the target object. The information, therefore, incrementally increases while the system keep tracking the target object and the accuracy of the observable-area model is augmented. As the information is accumulated, the system can utilize the required information for adaptive role assignment. As a result, the system as a whole can cooperatively work.

We should note that the proposed model is evaluated by multi-agent systems for real-time object tracking, however, the basic idea (i.e., the knowledge of partners' abilities is necessary for cooperation) is applicable to various types of multi-agent systems.

Chapter 7

Concluding Remarks

7.1 Thesis Summary

In this thesis, we have presented real-time cooperative multi-target tracking by communicating active vision agents. To realize real-time flexible tracking in a wide-spread area, we employ the idea of CDV. Based on the CDV system, our system consists of communicating AVAs. Multiple autonomous AVAs communicate with each other and cooperatively behave. Furthermore, the states and their transitions of the system increase enormously by combining with each other. This property allows the system to cope with various and complicated situations in the real world. This is a great advantage of the distributed processing system in contrast to the centralized processing system. We believe that this property is indispensable to realize the real-world system.

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

1. How to design an active camera for dynamic object detection and tracking.

   For wide-area active imaging, we developed an FV-PTZ camera. This camera is designed so that the projection center is always placed at the rotational center irrespectively of pan, tilt and zoom controls. This property allows the system (1) to synthesize a wide panoramic image by mosaicing multiple images observed by changing pan-tilt-zoom parameters and (2) to generate an image taken with any pan-tilt-zoom parameters from the wide panoramic image. With the FV-PTZ camera, we can realize an active camera system that detects anomalous regions in the observed image by the background subtraction method.

2. How to realize real-time object tracking with an active camera.

   For real-time object detection and tracking, we designed an active background subtraction method with the FV-PTZ camera. To successfully gaze at the target during tracking, the system incorporates a flexible control system named the dynamic memory architecture to dynamically control visual perception and camera action modules. The dynamic memory enables parallel modules to asynchronously obtain the information of another process without disturbing their own intrinsic dynamics.

3. How to realize cooperation among AVAs for real-time multi-target object tracking.

   To implement a real-time cooperation among AVAs, we designed a three-layered interaction architecture:

   - 1st layer (Intra-AVA layer): Visual perception, action and communication modules work together as a single AVA by dynamically interacting with each other. Each module exchange its information through the dynamic memory.
   - 2nd layer (Intra-Agency layer): AVAs that track the same target form an agency. AVAs in the same agency exchange object information to cooperatively track the target. Each agency has its own dynamic memory, and all the AVAs exchange their information of the detected objects through the dynamic memory. The dynamic memory allows the agency to obtain the reliable result of object identification from asynchronous object information observed by the AVAs.
   - 3rd layer (Inter-Agency layer): In order to adaptively restructure agencies taking into account targets' motions, agencies exchange the target and agency information with each other.

   The dynamic interaction in each layer allows the whole system to track multiple moving objects under complicated dynamic situations in the real world.

   In addition, we devised a scene model for efficient cooperation among AVAs. The visible/invisible area information is included in the scene model that is called an observable-area model. The model is incrementally updated while the system works. With the help of the observable-area information in the model, all the AVAs can cooperatively and adaptively assign their works to each other.

7.2 Future Works

As mentioned above, our real-time tracking system can work under complicated dynamic situations in the real world. We believe that this system can be a basic technology to realize various real-world vision systems. To practically apply our system to real-world vision systems, other issues in Computer Vision should be discussed. In what follows, we briefly summarize several aspects untaken in the thesis and directions of future works.

1. More robust object detection

- **Robust background subtraction:** To detect object regions in the observed image taken with arbitrary combinations of pan-tilt-zoom parameters, we employ the background subtraction method with the following properties:
  - Wide panoramic background image (generating a background image taken with arbitrary combinations of pan-tilt-zoom parameters).
  - Variable threshold (coping with camera calibration errors).

  - Subtracting the background image from several shifted versions of the observed image (measures to stably detect object regions in the observed image taken with smooth camera motion).

  These properties allows the system to acquire reliable detection results in a stationary scene. The effectiveness of the proposed background subtraction method, however, is limited because the stationary background scene assumption does not always hold in the real world, especially in the outdoor scene. To cope with the following types of variations in scenes, many works have been reported:

  Continuous small variations in objects: In [SG99] and [HHD00b], probability distributions are employed to model the intensity variations at each pixel.

  Intermittent large variations in objects: In [SMKU00], the background scene image is adaptively renewed by employing M-Estimation.

  Variations in the illumination: In [MOH00], variations in the scene are modeled by (1) variations in the overall lighting conditions, and (2) local image pattern fluctuations, and so on.

  By applying these methods to our tracking system, the system can work under dynamic variations in the real world.

- **Detection using knowledge about the target object:** Although the background subtraction method is effective, it is difficult to detect an object in all situations and environments. To solve this problem, the knowledge about the target object should be utilized:
  - In [TM00], the object region in the observed image is roughly estimated based on the past object trajectory. Then, the threshold for subtraction is dynamically determined depending on the gray level distribution in the estimated region.
  - In [KMS00], the system detects a human face using the given knowledge: eigen images[MN95] [AIS95] of the human face are generated in advance and used for face detection.

- **How to look problem:** In the proposed system, each camera captures an image while varying the view direction. In general, such image capturing incurs motion blurs and results in difficulty in object detection. We restricted the upper speed of the camera to suppress motion blurs. To solve this problem radically, we have to consider 'how to look problem'. That is, the observation method should be changed depending on the states of the system and camera. For example:
  - If object motion is slow enough to track it by a slow camera action, the active background subtraction method is effective.
  - In [MWM98b], object regions are detected while a camera is rotated at high speed based on the optical flow analysis. The optical flow, however, cannot detect stationary objects.

Since these two methods compensate the disadvantage of the other, the system can continue to detect objects by selecting the method according to camera motion.

2. **More reliable object identification**

   - **Error detection and correction for object identification:** In the proposed system, an error in object identification is corrected in the higher layer and the subsequent observation:

     – If a member-AVA makes unsuccessful object identification, the agency detected it, and the corrected object information is sent from the agency to the member-AVA (i.e., the agency maintenance).

     – If an agency makes unsuccessful object identification, it is detected and corrected by the inter-agency interaction in the subsequent observation (e.g., the agency unification).

     While these methods work well, each AVA and agency need the interval to adapt themselves to actual situations.

     Such identification error detection and correction methods are indispensable for multi-target tracking, and many works have been reported (see [BS78] [Rei79], for example). These methods can be utilized to improve (1) the reliability of object identification established by each AVA and agency and (2) the reactiveness of the error correction.

   - **Object identification based on multimodal information:** In the thesis, we established object identification among multiple AVAs by integrating the detected results of AVAs. To implement the integration, we put our focus upon a real-time cooperation among AVAs.

     On the other hand, some researchers addressed this problem by developing high-performance image recognition methods (see [IB98] and [HHD00a], for example). In these methods, the appearance information is employed for object identification. These methods can be applied to our system. In particular, the appearance information is useful when it is hard to identify the target object by employing only the 3D trajectory data. Followings are actual examples:

     – When the object is observed again after it has got out of the scene.

     – When multiple objects become close together.

     To employ the appearance information for the proposed system, we have to note that it should be represented as follows:

     – For each AVA and agency to effectively manage the appearance information in the dynamic memory, it should be represented as time-series data.

     – For object identification among multiple cameras, the omnidirectional appearance information is required.

Besides the visual information, employing multimodal information is effective not only for object identification but also for object detection. While only the visual information is employed for object detection and tracking in the proposed system, the auditory and visual information is observed and analyzed for real-time multi-object tracking in [NHM$^+$01]. In this system,

- auditory streams with sound source direction are extracted, and
- visual streams with face ID and 3D-position are extracted by combining skin-color extraction, correlation-based matching, and multiple-scale image generation from a single camera.

These auditory and visual streams, each of which is obtained asynchronously on different PCs connected via network, are associated by comparing the spatial location.

3. The number of the trackable target objects:

The completeness for persistent tracking in our system (i.e., the relations between the numbers of cameras and target objects) is mentioned in Section 5.4.1. Here, we discuss how to increase the number of the trackable objects.

To increase the number of the trackable objects, we can modify the system as follows:

- **Vacuous agency without member-AVAs:** In general, since an agency manager is a software agent, the number of agencies can increase regardless of the number of AVAs. In the proposed system, however, the number of agencies cannot increase more than the number of AVAs. This is because an agency has to be attended by at least one member-AVA. Although this restriction assists each agency in tracking its target object persistently, the flexibility of the system declines.

  To solve this problem, we can modify the system so that an agency without any member-AVAs can be generated. This definition allows the system to track target objects more than the number of AVAs (i.e., cameras). Instead of this advantage, we have to consider the following problems:

  When is an agency generated by whom? As well as the proposed system, an agency should be generated by an AVA that detects a target object when it is confirmed that no agency tracks the newly detected object. The difference from the proposed system is that the AVA necessarily belongs to the newly generated agency as a member-AVA.

  How does a vacuous agency obtain the information of the target object? A vacuous agency has to receive the object information detected by non-member-AVAs (i.e., freelancer-AVAs and member-AVAs of other agencies). While a freelancer-AVA broadcasts the information of the detected object, a member-AVA sends it only to its agency manager. The following two methods can be employed for the member-AVA to report the detected information to vacuous agencies:

Broadcast: By broadcasting the message, each member-AVA can easily report the information of its detected objects to all agencies. This method, however, increases a network-load enormously.

Multicast: To suppress a network-load, each member-AVA should send the information only to agencies. For this message transmission, each member-AVA has to grasp the existence of all the agencies in the system. An agency should distinguish its member-AVAs from member-AVAs in other agencies, which send the information of their detected objects to the agency. This is because the agency cannot control their cameras to keep tracking the target object. We call such an AVA a *Supporter-AVA* (mentioned later).

This information transmission to multiple agencies allows not only vacuous agencies but also common agencies with their member-AVAs to obtain more information of their target objects.

- **Supporter-AVA:** While a supporter-AVA works as a member-AVA in an agency, it sends the information of the detected objects to other agencies. A member-AVA can be supporter-AVAs of multiple other agencies simultaneously. Note that each member-AVA is controlled only by the agency that it belongs to, and the agency cannot interfere in the behaviors of its supporter-AVAs.

4. Camera configuration planning:

While we do not consider how to arrange the cameras to effectively gaze at the target objects, there are many researches about the effective camera configuration for realizing the given task.

In [CK88], an automatic camera placement method for object feature detection is proposed. In this method, each camera is placed so that all surface points be in focus, all surfaces lie within the visual field of the camera, and no surface points be occluded. In [TTA95], the MVP sensor planning system is proposed. This system determines the optimal settings of the camera and illumination by virtually synthesizing desirable camera views based on geometric models of the environment, optical models of the cameras, and models of the task.

Similarly, the effective camera configuration for object tracking should be planned depending on the given task. For example:

- The wider the observation scene becomes, the larger the number of cameras becomes. Then, a network-load increases. To avoid this problem, an efficient camera configuration is required.

- For the system to keep tracking a target object in a wide area, all areas have to be observable from a camera. In addition, to reconstruct 3D information of an object, visual fields of cameras have to be overlapped and all areas have to be observable from multiple cameras.

- If *search* is important, cameras should be embedded in the scene so that each camera is suitable for observing a wide area.

- If there exists a place that should be monitored selectively, many cameras should be embedded around there.

5. Tracking with isolated camera configuration:

In all the experiments conducted in this thesis, visual fields of all AVAs are overlapping with each other. This situation does not always hold depending on the task. To keep tracking the target object even if cameras are embedded sparsely in the scene, the system has to employ not only the 3D trajectory of the target object but also other information for object identification:

- The above appearance-based object identification is useful.
- In general, for object identification among widely distributed cameras, the system searches enormous candidates for an optimal solution. Since such a method makes real-time processing difficult, we should reduce the candidates by applying some constraints. In [KZ99] [WTM96], several constraints on the route and lapse assist object identification in addition to the appearance information.

6. Capturing selective object image depending on the task:

Depending on the task, the required information of the target object varies; whole body, face, hands and so on. For example:

- In order to acquire not only the target trajectory but also the precise volumetric and appearance information (e.g., [WWTM00] and [BD00]), the system should control cameras to capture the high-resolution untaken and meaningful object image.
- For individual identification, information on human face is significant. In [KMS00] and [YWM00], the human head is detected based on the appearance and feature models.

As mentioned above, the proposed system has to be expanded for actual real-world systems. We, however, believe that the fundamental and essential problems have been solved in this thesis, and our proposed system can be applied to various real-world systems: visual surveillance and monitoring systems, ITS (Intelligent Transport System), navigation of mobile robots and disabled people, and so on.

We hope that all the fruits of this thesis are utilized for many researches in future.

Bibliography

[AAB97] A. Amano, N. Asada, and M. Baba. Photometric calibration of zoom lens systems: Analysis and correction of vignetting distortion by the variable cylinder model. *Transaction of The Institute of Electronics, Information and Communication Engineers D-II*, J80-D-II(6):1458–1465, 1997. (written in Japanese).

[AB99] A. A. Argyros and F. Bergholm. Combining central and peripheral vision for reactive robot navigation. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition '99*, volume 3, pages 646–651, 1999.

[AIS95] Y. Ariki, N. Ishikawa, and Y. Sugiyama. Extraction and recognition of facial recognitions by subspace method. In *Proceedings of Asian Conference on Computer Vision '95*, volume 3, pages 738–742, 1995.

[AV89] N. Ahuja and J. Veenstra. Generating octrees from object silhouettes in orthographic views. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2(2):137–149, 1989.

[AWB88] Y. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, 1988.

[Bal89] D. H. Ballard. Reference frames for animate vision. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1635–1641, 1989.

[BD00] E. Borovikov and L. Davis. A distributed system for real-time volume reconstruction. In *Proceedings of Fifth IEEE International Workshop on Computer Architectures for Machine Perception*, pages 183–189, 2000.

[Bro90] C. M. Brown. Gaze controls with interactions and delays. *IEEE Transaction on Systems, Man and Cybernetics*, 20(1):518–527, 1990.

[BS78] Y. Bar-Shalom. Tracking methods in a multitarget environment. *IEEE Transaction on Automation and Control*, AC-23(4):618–626, 1978.

[BT93] M. Barth and S. Tsuji. Egomotion determination through an intelligent gaze control strategy. *IEEE Transaction on Systems, Man and Cybernetics*, 23(5):1424–1432, 1993.

[CA99] 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):1241–1247, 1999.

[Che95] S. E. Chen. Quicktime vr – an image-based approach to virtual environment navigation. In *Proceedings of SIGGRAPH '95*, pages 29–38, 1995.

[CK88] C. K. Cowan and P. D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 10(3):407–416, 1988.

[CLK$^+$00] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Bert, and L. Wixson. A system for video surveillance and monitoring. Technical report, The Robotics Institute, Carnegie Melon University, 2000.

[Fau93] O. D. Faugeras. *Three-Dimensional Computer Vision*. The MIT Press, 1993.

[Gre86] N. Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, 1986.

[HB96] G. D. Hager and P. N. Berhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition 96*, pages 403–410, 1996.

[HHD00a] I. Haritaoglu, D. Harwood, and L. S. Davis. An appearance-based body model for multiple people tracking. In *Proceedings of 15th International Conference on Pattern Recognition*, volume 4, pages 184–187, 2000.

[HHD00b] I. Haritaoglu, D. Harwood, and L. S. Davis. A fast background scene modeling and maintenance for outdoor surveillance. In *Proceedings of International Conference on Pattern Recognition 2000*, volume 4, pages 179–183, 2000.

[HIZ00] T. Hasegawa, K. Imamura, and H. Zen. Observation of moving vehicles by the plural cameras established freely. In *Proceedings of 3rd IEEE International Conference on Intelligent Transportation Systems*, pages 328–333, 2000.

[IB98] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[Ish97] H. Ishiguro. Distributed vision system: A perceptual information infrastructure for robot navigation. In *Proceedings of 15th International Joint Conference on Artificial Intelligence*, pages 36–41, 1997.

[IYT90] H. Ishiguro, M. Yamamoto, and S. Tsuji. Omnidirectional stereo for making global map. In *Proceedings of IEEE International Conference on Computer Vision*, pages 540–547, 1990.

[Kan97] T. Kanade. Cooperative multisensor video surveillance. In *Proceedings of DARPA Image Understanding Workshop 1997*, pages 3–10, 1997.

[KIM00] Y. Kameda, K. Ishizuka, and M. Minoh. A study for distance learning service - tide project -. In *Proceedings of IEEE International Conference on Multimedia and Expo*, volume 3, pages 1237–1240, 2000.

[KKD$^+$98] Nobuhiro Kataoka, Hisao Koizumi, Hideru Doi, Kenichi Kitagawa, and Norio Shiratori. A proposal of a method of total quality evaluation in remote conference systems based on atm networks. *Transaction of The Institute of Electronics, Information and Communication Engineers on Communications*, E81-B(9):1709–1717, 1998.

[KMS00] T. Kato, Y. Mukaigawa, and T. Shakunaga. Cooperative distributed tracking for effective face recognition. In *Proceedings of IAPR Workshop on Machine Vision Applications 2000*, pages 353–358, 2000.

[KOK00] I. Kitahara, Y. Ohta, and T. Kanade. 3d video display of sports scene using multiple video cameras. In *Proceedings of Meeting on Image recognition and Understanding 2000*, volume 1, pages 3–8, 2000. (written in Japanese).

[KZ99] V. Kettnaker and R. Zabih. Bayesian multi-camera surveillance. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition 99*, pages 253–259, 1999.

[LC83] V. R. Lesser and D. D. Corkill. The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks. *AI Magazine*, 4(3):15–33, 1983.

[LDPD97] J. M. Lavest, C. Delherm, B. Peuchot, and N. Daucher. Implicit reconstruction by zooming. *Computer Vision and Image Understanding*, 66(3):301–315, 1997.

[LK93] J.J. Little and J. Kam. A smart buffer for tracking using motion data. In *Proceedings of IEEE International Workshop on Computer Architecture for Machine Perception*, pages 257–266, 1993.

[Mat98] T. Matsuyama. Cooperative distributed vision - dynamic integration of visual perception, action and communication -. In *Proceedings of DARPA Image Understanding Workshop 1998*, pages 365–384, 1998.

[MB94] D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(5):449–459, 1994.

[MHW$^+$00] T. Matsuyama, S. Hiura, T. Wada, K. Murase, and A. Yoshioka. Dynamic memory: Architecture for real time integration of visual perception, camera action, and network communication. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2000*, pages 728–735, 2000.

[Mil91] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Transaction of Communications*, 39(10):1482–1493, 1991.

[MN95] H. Murase and S. K. Nayar. Virtual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[MOH00] T. Matsuyama, T. Ohya, and H. Habe. Background subtraction for non-stationary scenes. In *Proceedings of Asian Conference on Computer Vision 2000*, pages 662–667, 2000.

[MWM98a] T. Matsuyama, T. Wada, and M. Maruyama. Cooperative object tracking by multiple active vision agents. In *Proceedings of Meeting on Image recognition and Understanding '98*, volume 1, pages 365–370, 1998. (written in Japanese).

[MWM98b] K. Murase, T. Wada, and T. Matsuyama. Moving object detection by a rotating camera. In *Proceedings of Meeting on Image Understanding and Recognition '98*, volume 1, pages 425–430, 1998. (written in Japanese).

[MWM99] T. Matsuyama, T. Wada, and Y. Monobe. Real-time object detection and tracking with a fixed-viewpoint pan-tilt-zoom camera. *Information Processing Society of Japan Journal*, 40(8):3169–3178, 1999. (written in Japanese).

[Nak01] A. Nakazawa. *Human Tracking using Distributed Vision Systems.* PhD thesis, Osaka university, 2001.

[NHM+01] K. Nakadai, K. Hidai, H. Mizoguchi, H. G. Okuno, and H. Kitano. Real-time auditory and visual multiple-object tracking for robots. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, volume 2, pages 1425–1432, 2001.

[NKI98] A. Nakazawa, H. Kato, and S. Inokuchi. Human tracking using distributed vision systems. In *Proceedings of 14th International Conference on Pattern Recognition*, pages 593–596, 1998.

[NO92] S. Nishio and Y. Ohta. Tracking of vehicles at an intersection by integration of multiple image sensors. In *Proceedings of IAPR Workshop on Machine Vision Applications '92*, pages 321–324, 1992.

[OK93] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1993.

[PDBSW92] K. R. Pattipati, S. Deb, Y. Bar-Shalom, and R. B. Washbarn. A new relaxation algorithm and passive sensor data association. *IEEE Transaction Auto. Contr.*, AC-37(2):198–213, 1992.

[PN97] V. N. Peri and S. K. Nayar. Generation of perspective and panoramic video from omnidirectional video. In *Proceedings of DARPA Image Understanding Workshop*, pages 243–245, 1997.

[Rei79] D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transaction on Automation and Control*, AC-24(6):843–854, 1979.

[RH90] D. Raviv and M. Herman. Towards an understanding of camera fixation. In *Proceedings of International Conference on Robotics and Automation*, pages 28–33, 1990.

[SG99] C. Stauffer and E. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition '99*, volume 2, pages 246–252, 1999.

[SH97] T. M. Strat and S. J. Henessy. Video surveillance and monitoring. In *Proceedings of First International Workshop on Cooperative Distributed Vision*, pages 195–219, 1997.

[SIT00] T. Sogo, H. Ishiguro, and M. M. Trivedi. Real-time human tracking system with multiple omni-directional vision sensors. *Transaction of The Institute of Electronics, Information and Communication Engineers D-II*, J-83-DII(12):2567–2577, 2000. (written in Japanese).

[SMKU00] H. Shimai, T. Mishima, T. Kurita, and S. Umeyama. Adaptive background estimation from image sequence by on-line m-estimation and its application to detection of moving objects. In *Proceedings of Infotech Oulu Workshop on Real-Time Image Sequence Analysis*, pages 99–108, 2000.

[SO00] Y. Sugaya and Y. Ohta. A video-rate stereo system by integration of two algorithms with/without occlusion handling. In *Proceedings of IAPR Workshop on Machine Vision Applications 2000*, pages 244–247, 2000.

[SST86] S. A. Shafer, A. Stentz, and C. E. Thorpe. An architecture for sensor fusion in a mobile robot. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 2002–2011, 1986.

[Ste99] G. P. Stein. Tracking from multiple view points: Self-calibration of space and time. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition 99*, volume 1, pages 521–527, 1999.

[TAT95] 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):86–104, 1995.

[TM00] T. Tomiyama and T. Matsuyama. Real-time object tracking using the dynamic memory. In *IPSJ SIG Notes, 2000-CVIM-121*, pages 49–56, 2000. (written in Japanese).

[Tsa86] R. Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition 86*, pages 364–374, 1986.

[TTA95] K. A. Tarabanis, R. Y. Tsai, and P. K. Allen. The mvp sensor planning system for robotic vision tasks. *IEEE Transaction on Robotics and Automation*, 11(1):72–85, 1995.

[TWM01] S. Tsunetani, T. Wada, and T. Matsuyama. Object tracking with fixed-viewpoint pan-tilt stereo camera. In *IPSJ SIG Notes, 2001-CVIM-128*, pages 103–110, 2001. (written in Japanese).

[WADP97] 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):780–785, 1997.

[WGCM96] N. Nandhakumar Wei-Ge Chen and Worthy N. Martin. Image motion estimation from motion smear - a new computational model. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(4):412–425, 1996.

[WSN87] L. E. Weiss, A. C. Sanderson, and C. P. Neuman. Dynamic sensor-based control of robotics with visual feedback. *IEEE Transaction on Robotics and Automation*, RA-3(5):404–417, 1987.

[WTM96] T. Wada, M. Tamura, and T. Matsuyama. Cooperative distributed object identification for wide area surveillance systems. In *Proceedings of Meeting on Image recognition and Understanding '96*, volume 1, pages 103–108, 1996. (written in Japanese).

[WWTM00] T. Wada, X. Wu, S. Tokai, and T. Matsuyama. Homography based parallel volume intersection: Toward real-time volume reconstruction using active cameras. In *Proceedings of Fifth IEEE International Workshop on Computer Architectures for Machine Perception*, pages 331–339, 2000.

[YAT00] S. Yonemoto, D. Arita, and R. Taniguchi. Real-time human motion analysis and ik-based human figure control. In *Proceedings of IEEE Workshop on Human Motion*, pages 149–154, 2000.

[YK86] M. Yachida and Y. Kitamura. 3-d data acquisition by multiple views. In *Third International Symposium on Robotics Research*, pages 11–18. MIT Press, 1986.

[YM96] N. Yoshida and A. Mitani. Decentralized processing for multitarget motion analysis. In *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 297–303, 1996.

[YWM00] K. Yachi, T. Wada, and T. Matsuyama. Human head tracking using adaptive appearance models with a fixed-viewpoint pan-tilt-zoom camera. In *Proceedings of Fourth International Conference on Automatic Face and Gesture Recognition*, pages 150–155, 2000.

[YY91] Y. Yagi and M. Yachida. Real-time generation of environmental map and obstacle avoidance using omnidirectional image sensor with conic mirror. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition 91*, pages 160–165, 1991.

[YYY95] K. Yamazawa, Y. Yagi, and M. Yachida. Obstacle detection with omnidirectional image sensor hyperomni vision. In *Proceedings of International Conference on Robotics and Automation '95*, pages 1062–1067, 1995.

[ZM95] Y. Zhang and A. Mackworth. Constraint nets: A semantic model for a hybrid dynamic systems. *Theoretical Computer Science*, 138:211–239, 1995.

List of Publications

- **Journal Paper**

  1. T. Wada, N. Ukita and T. Matsuyama: "Fixed-Viewpoint Pan-Tilt-Zoom Camera and Its Applications", Transaction of The Institute of Electronics, Information and Communi cation Engineers D-II, Vol. J81-DII, No.6, pp.1182–1193, 1998.
  2. N. Ukita and T. Matsuyama: "Incremental Observable-Area Modeling for Cooperative Tracking", IPSJ Journal, Vol.42, No.7, pp.1902–1913, 2001.

- **International Conference**

  1. N. Ukita and T. Matsuyama: "Incremental Observable-Area Modeling for Cooperative Tracking", Proceedings of 15th International Conference on Pattern Recognition, Vol.4, pp.192–196, 2000
  2. N. Ukita, T. Nagao and T. Matsuyama: "Versatile Cooperative Multiple-Object Tracking by Active Vision Agents", Proceedings of IAPR Workshop on Machine Vision Applications 2000, pp.569–573, 2000

- **Presentation**

  1. N. Ukita and T. Shakunaga: "3D Reconstruction from Wide-Range Image Sequences Using Perspective Factorization Method", Technical Report of IEICE, PRMU, Vol.97, No.596, pp.81–88, 1998.
  2. N. Ukita, S. Tokai, T. Matsuyama and R. Taniguchi: "Database Development of Multi-Viewpoint Image Sequences for Evaluation of Image Understanding Systems", Technical Report of IEICE, PRMU, Vol.99, No.182, pp.65–72, 1999.
  3. N. Ukita and T. Matsuyama: "Incremental Observable-Area Modeling for Cooperative Tracking", Proceedings of Meeting on Image Recognition and Understanding 2000, Vol.1, pp.421–426, 2000.
  4. T. Wada, N. Ukita and T. Matsuyama: "Appearance Sphere – Background Model for Pan-Tilt-Zoom Camera–", Proceedings of Meeting on Image Recognition and Understanding '96, Vol.2, pp.103–108, 1996.
  5. T. Nagao, N. Ukita and T. Matsuyama: "Multi-target Tracking by Communicating Active Vision Agents", IPSJ SIG Notes, CVIM, Vol.2000, No.33, pp.57–64, 2000.