Doctoral Dissertation, 2016

Stream Data Evaluation for a Lifelog Analysis System using a Data Quality Evaluation Framework

OCHANOMIZU UNIVERSITY
Humanities and Sciences (Doctoral Program)
Advanced Sciences, Computer Science

Akika Yamashita

March, 2016

Abstract

In recent years, due to the rapid development of sensor devices, many types of multimedia sensor data can be collected and analyzed to develop useful multimedia applications such as human activity recognition system. Realizing Lifelog which is to memorize life of people as digital data became much easier compared to before. As a result, various Lifelog Analysis Applications using collected sensor data have developed. However, for these Lifelog analysis applications, the quality of input data has not been considered in detail.

So, in this study, "Lifelog analysis application that verbalize human action" was actually assumed to be an example, and evaluated how the difference of quality of input data, for example, the frame rate of collected video data or interval of collecting acceleration sensor data or image quality of video data would influence the analysis of result of application. In my experiment, I have dealt with following three kinds of data quality of input data, that is video data and acceleration data, through actual experiment.

Data quality A Image quality of each frame of video data

Data quality B Obtained number of video data and acceleration data/data quantity per second

Data quality C Packet loss rate of video data when communicating through WLAN

I have implemented Lifelog Analysis Application and experimented the impact of input data quality on the result of human activity recognition system at real environment — Ocha House. Ocha House is a Japanese home setup to conduct Cyber Physical System experiments using several multimedia camera sensors to monitor and capture human motions and activity data in an end-to-end wireless multimedia network environment. That is, the collected human motions and activity data in Ocha House are transmitted real time over the WLAN to a server for processing and analysis in human activity recognition application. I have evaluated the correlation between input data quality and the result of human activity recognition system Quantitatively.

In addition, I have concerned to introduce the Lifelog Analysis Application to each family as abnormal activity detection system inside a smart house. When considering setting up the system in each house, it is effective to collect and analyze the sensor data on Cloud environment. I have also proposed how to implement the system in many houses, and evaluated how to migrate the data quickly and securely.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, with the rapid development of small and high technological devices such as network cameras, smart phones, and a log of kinds of sensor devices, many types of physical data such as multimedia sensor data can be collected from real world, and many types of cyber data such as SNS (Social Networking Service) data can be collected on cyber space like the Internet. These variety and unstructured big data will be stored on a massive scale almost without having to worry about the capacity of the storage. If I provide some processing methods toward these kinds of data, the data will be converted to useful information for people. The representative system to do data processing is called Lifelog Analysis Application which offer the user profitable information such as person's activity history, person's health condition with an analysis of network connected terminals like sensor terminals, smart phones and so on. Lifelog means the record of human life as digital data.

As shown in Tab. 1.1, there are many kinds of Lifelog Analysis Application, for example, GPS data will be converted to User's access history and favorite place then the Application can push coupon information which can used near the user's place. Call history, Email history, and SNS commented history will analyzed to know user's friend ship, and the system or application can recommend the friend who the user may know. If meal menu data is collected as picture data or text

data, that information can help user maintain a nutritional balance. Shopping history collected on online shopping store will be analyzed to know user's favorite thing, and the store can recommend what user will want next.

Table 1.1: Examples of Lifelog Analysis Application

Collected Data	Converted Information	Representative Service
GPS, Geological data	User's access history, favorite place	Push coupon information near user
Call history Email history SNS comment history	User's friendship	friend recommendation
meal menu	User's calorie balance	Health care system
Shopping history	User's favorite thing	Recommend what user will want

In each application, many kinds of sensor data is converted to useful information for people. This whole system is called CSP (Cyber Physical System) or IoT (Internet of Things) (Fig. 1.1).

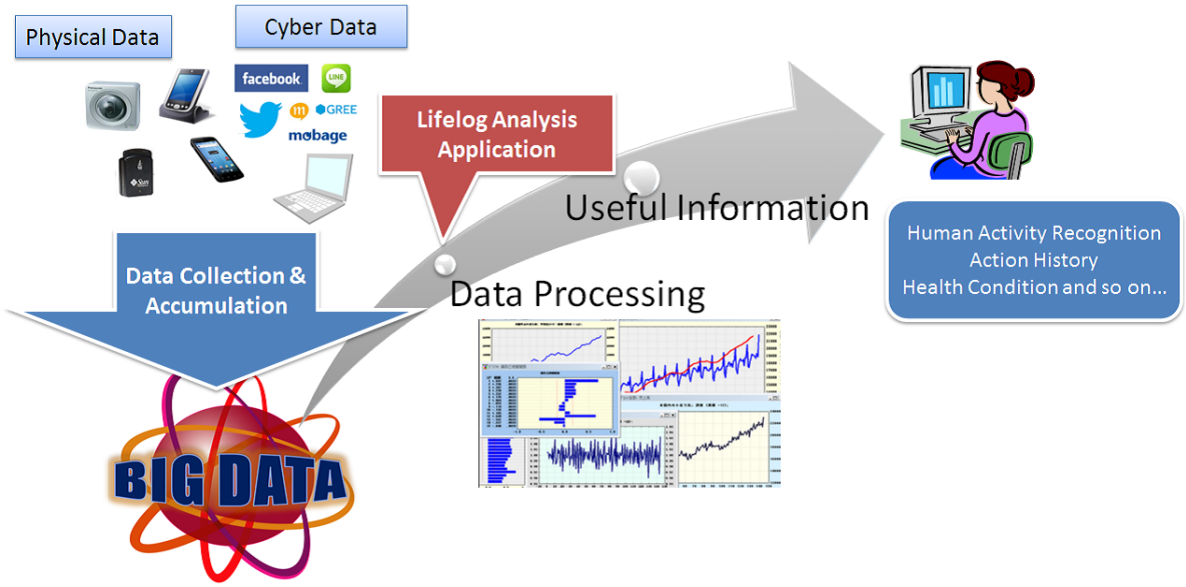

Figure 1.1: The System called CPS and IoT

Nowadays, "CPS"and "IoT"are very popular words. The meaning of these words is that everything exists in the world have the ability to connect to the network or communicate each other to do automatic recognition and automatic

control. It is said that the number of devices connected to the Internet will be about 26 billion by 2020. This technology will penetrate the real world with wider use of method and anyone can receive benefits of the technology Anytime, anywhere.

IoT and CPS also have relationship with a lot of research area, for example, information security, medical health care, smart city or smart house, and wireless sensor network. The detail of these research is introduced in chapter 2. I suggest there is a issue cannot be ignored in IoT and CPS area, which is the "data quality". When considering collecting data from sensor space and transmitting the data with wireless network to some analysis system or application, to take account of "data quality"is really important. If the data quality deterioration caused by sensor device failure when collecting data, the validity of the analysis system or application cannot be guaranteed. If I just collect the highest quality data, the storage will be filled up and it is not efficient.

There are many past and existing Lifelog Analysis Application research work focused solely on processing speed, algorithms, or how to collect high quality data in terms of their human activity recognition accuracy. However, these research work lack the understanding of end-to-end system characteristics affecting the captured multimedia sensor data quality that will impact the the accuracy of Lifelog Analysis Application. Thus, my objective of this research work is to study and analyze the impact of the quality on the Lifelog Analysis Application accuracy in the end-to-end multimedia system environment — OchaHouse. OchaHouse is a Japanese home setup to conduct cyber-physical system experiments using several multimedia camera sensors to monitor and capture human motions and activity data in an end-to-end wireless multimedia network environment. That is, the collected human motions and activity data in OchaHouse are transmitted real time over the WLAN to a server for processing and analysis in Lifelog Analysis

application.

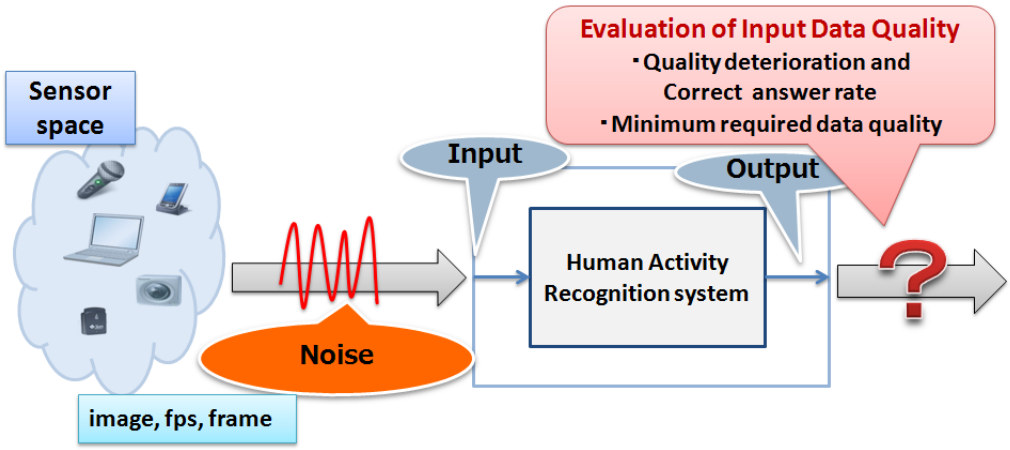

Figure 1.2: The purpose of this research

The goal of this study is to evaluate the impact caused by the quality of input data to Lifelog analysis application with quantitative indicators. In other words, as shown in the Fig. 1.2, I suppose a typical human activity recognition system of which the input data is video image, acceleration data, and sound data collected in a sensor space, and it provides users the result of analysis by applying some theoretical data processing to the input data. The data quality evaluation experiments have been executed to know how the difference of input data quality results in the output of the application. For example, while only a small number of frame drops or little noise will hardly influence the result of the human activity recognition system, if there are a lot of dropped frames or noise, the application would not be able to output the correct results. Therefore, it is important to clarify the quantitative indicators to show which level of data quality is required for the application to work correctly.

I have implemented a Human Activity Recognition System as a typical Lifelog Analysis Application. The input data of the system is the video data and acceleration data collected with sensor devices, and output of the system is the

verbalization of human's activity. I have utilized this Human Activity Recognition System to evaluate the impact of input data quality on the system output.

I assumed two kinds of data quality deterioration. One is the data quality deterioration when collecting data with sensor devices. That can occur by the failure of the sensor devices. The other is the data quality deterioration when transmitting the data trough network. That can occur by radio wave interference. The data deterioration type will differ in this two cases. I have done following three data quality evaluation experiment to reproduce several types of data quality deterioration.

Data quality A Image quality of each frame of video data

Data quality B Obtained number of video data and acceleration data / data quantity per second

Data quality C Packet loss rate of video data when communicating through Wireless LAN (Verbalization Application)

Data quality D Packet loss rate of video data when communicating through Wireless LAN (GTW algorithm)

In the data quality evaluation experiment A, I have utilized following four kinds of image data : "blurred image", "vertical to the blurred image", "next to blurred image", and "resolution degraded image". The quality of each image is changed artificially with applying four kinds of filtering function, and I have shown the correlation between input image quality and the accuracy of the system.

In the data quality evaluation experiment B, the image data and acceleration data is the input data, and the frames rate is artificially varied from 10 frame / sec to one frame / sec. I have shown the correlation between input data frame quality and the accuracy of the system.

In the data quality evaluation experiment C, and D, I have experimented the impact of the WLAN communication quality on the Human Activity Recognition System accuracy in the end-to-end networked multimedia system environment — OchaHouse (Fig. 1.3). OchaHouse is a Japanese home setup to conduct cyber-physical system experiments using several multimedia camera sensors to monitor and capture human motions and activity data in an end-to-end wireless multimedia network environment. That is, the collected human motions and activity data in OchaHouse are transmitted real time over the WLAN to a server for processing and analysis in Human Activity Recognition System. In this experiment, I have utilized two types of Human Activity Recognition System, one is the system to verbalize human's activity, and the other is the system to align multi-modal sequences from multiple human subjects performing similar human motion activities.

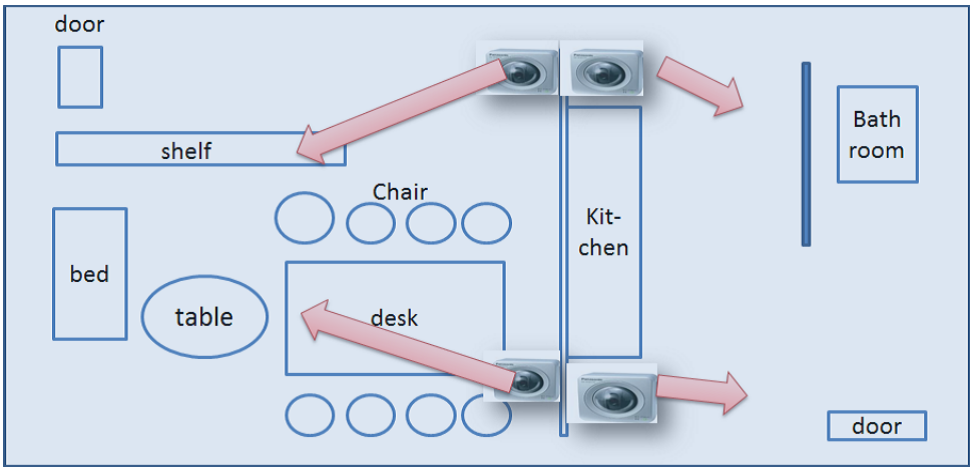

Figure 1.3: Data collection inside OchaHouse

In addition, I have concerned to introduce the Lifelog Analysis Application to each family as abnormal activity detection system inside a smart house. When considering setting up the system in each house, it is effective to collect and analyze the sensor data on Cloud environment. As shown in Fig. 1.4, I have also proposed how to implement the system in many houses, and evaluated how to migrate the

system quickly and securely.

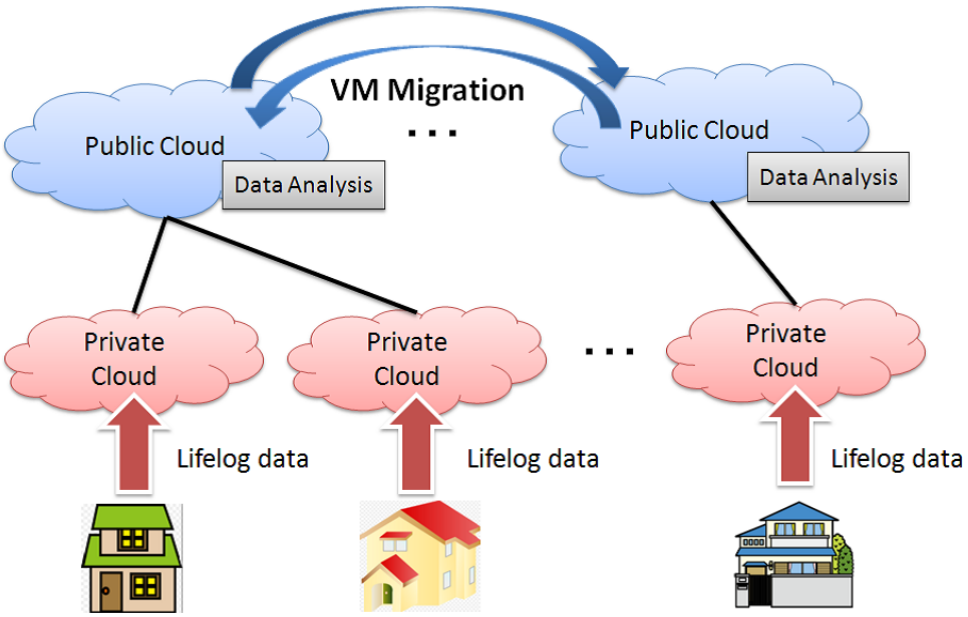

Figure 1.4: Lifelog Analysis Application on inter-Cloud

Outline. I discuss the related work of this research in chapter 2. In chapter 3, I describe the behavior of my Human Activity Recognition System and development environment including OchaHouse and sensor space. In chapter 4, I propose data quality evaluation framework for evaluating the data quantitatively. In chapter 5, I show evaluation experiment environment of the Human Activity Recognition sSstem and experiment result of data quality evaluation experiment A, B, and C. In chapter 6, I propose how to expand the system on inter-Cloud. Finally I conclude this thesis in chapter 7.

Chapter 2

Related Work

2.1 IoT and CPS

In IoT and CPS research area, many kinds of research work has done before. Many researchers engaged in IoT and CPS technologies have focused on its architecture or modeling, security, devices, and network. Many researchers has focused on security as they have to collect many types of data including sensitive data. In the reference [1][2][3], the authors are applying the technology to medical architecture and considering security. The network is also an important topic to consider IoT and CPS technology. In the reference [4][5][6], some architecture are proposed and how to control network model with middle ware is concerned. Some services or systems utilizing the IoT or CPS technology are proposed in [7][8][9][10][11] including IoT for smart cities [12].

My work is different with these IoT or CPS work as I have concerned the collected data quality and network quality to collect or store the sensor data efficiently.

2.2 Lifelog Analysis Application

A lot of work to recognize human activity with analysis of Lifelog data collected with sensor terminals have been performed and some of them are introduced

[13][14]. In the reference [15][16], a large amount of acceleration data collected with sensor terminals attached to people was studied and used to recognize the human activity. As a result, increasing the volume of collected data would improve the recognition rate. In the reference [17][18], activity information sharing system with which people can see others' activity and compare with himself has been provided by collecting acceleration and video data of human activities. The collected lifelog has been intensively used for location-based systems such as [19], in which a method for predicting the future location of the human based on the lifelog was discussed. In [20], a method was discussed to build context modeling for recognition of a human behavior. They proposed a hierarchical spatio-temporal context modeling.

2.3 Wireless Communication Quality

As one of the data quality evaluation experiments, I have focused on WLAN communication quality when collecting input data. I will discuss the research about WLAN communication quality. A WLAN defines multiple transmission rate to keep the bit and/or frame error rates to be proper values, and appropriate transmission rates are selected depending on the bit error state. The multi-rate control works as follows: when a fine radio environment can be obtained, a higher transmission rate is selected, if a higher transmission rate cannot be kept due to a poor radio environment, a lower transmission rate is selected. With such a mechanism, the multi-rate control can accommodates multiple terminals in different conditions. 11, 5.5, 2 and 1 Mbps are defined for IEEE802.11b and 54, 48, 36, 24, 18, 12, 9 and 6 Mbps are defined for IEEE802.11a/g as standard values in multi-rate control. Like [21] and [22], efficient rate adoption method and technique have been proposed, and [23] have evaluated the performance of multi-rate throughput with real-terminal.

For any Human Activity Recognition System, it requires the integration of the sensor devices to the network such as WLAN, for the transmission of captured data for analysis. Some researchers attach sensor devices such as gesture-based movement acceleration and medical measurement sensor devices directly to users and send the captured data via wireless network to the server for ANALYSIS (computer vision and image processing). In this work [24], Francesca De Simone et al. evaluated the video quality at the premises of two academic institutions. The video data include the packet loss data when transmitting via IP networks and groups of people are asked to rate the quality. In my research, I evaluated the video quality with GTW algorithm and its variants, which is the reliable quantitative evaluation. The research works that are related to Human Recognition System in the smart house are also performed. Simon Moncrieff et al. in [25] have developed "anxiety framework"to determine hazards such as abnormal activity inside the smart house. Lian Wang et al. in [26] have used wearable sensors and wireless networks to recognize multi-user activities inside the smart house. In these research, it exists the same problems of WLAN packet loss due to the network transmission but such research neglected the persistent network impact on the system quality. As such, my research work is the first that presents new findings and important observations that cannot be ignored in a total system view for end-to-end system quality.

2.4 Inter-Cloud

I have proposed how to utilize this human activity recognition system on inter-Cloud environment. If a lot of family are using this system, setting up the storage and servers inside a personal house if not efficient. The storage and the servers to analyze the collected data should be on Cloud side, and Virtual Machine (VM) should analyze the data. I have considered a method to transmit VMs faster and

securely.

The research in the area of VM migration mainly focused on optimizing migration performance through live migration. The mechanism of live migration which is a migration that not to stop VM before and after the migration is explained in [27] and mechanism is explained in [28]. In [29], the scalability of application services on Inter-Cloud is discussed. In [30], the technology that enables live mobility of VM on Inter-cloud is discussed but privacy and security is required. [31] and [32] is also discussing migration performance improvement.

As for the security consideration, J. Rexford et. al have come out with no hypervisor. Securing hypervisor architecture is one of the good methods to achieve secure migration, but the method proposed is not sufficient as the vulnerabilities in current migration mechanisms. While the semantics and performance of live VM migration are well explored, the security aspects have received very little attention in an optimistic point of view, I are planning to develop a method that satisfies both characteristic - security and performance - paying attention to the trade-off between them.

My research is different from others because I have considered end-to-end system. Then the captured and collected data in the storage should be sent to the terminal for data processing, and there should be data quality deterioration depending on the quality of communication environment. I have shown the correlation between data quality deterioration rate and the accuracy of the system result.

Chapter 3

Human Activity Recognition System

In this chapter, the behavior of my Lifelog Analysis Application is explained. The goal of this work is to evaluate the influence of input data quality change on Lifelog Analysis Application quantitatively. When considering utilizing the Lifelog Analysis Application efficiently, various type of data quality deterioration will occur. This quality deterioration will be caused by the data collection method, the situation of data collection, or data transmitting method. There are a lot of kind of Lifelog Analysis Applications and also many types of input data are used by these application. I have tried to evaluate how the input data quality will have impact on the output of Lifelog Analysis Application, and show the guidelines of countermeasure. In short, since the influence of data quality deterioration appears to come to the application level, I have shown the guideline of how to handle the data with the result of strict evaluation.

I have utilized a Lifelog Analysis Application which recognize human action with verbalization result as an example of Lifelog Application System. The input data of this system are video data and acceleration data. The output is verbalization result of human action. There was some work in which just one element of data quality deterioration has been taken up and evaluated. But in this study, the impact of the combination of plural data quality deterioration on the application

level has been evaluated.

Outline. Section 3.1 of this chapter explained how the human activity recognition system behaves when the image data and acceleration data has inputed into the system. I also show how to collect and synchronize plural input data with camera and acceleration sensor devices, and the modeling method output processing when the condition is met. In section 3.2, I will introduce the development environment and sensor devices used in the system. I also introduce OchaHouse which is an experimental smart house [37].

3.1 Behavior of Human Activity Recognition System

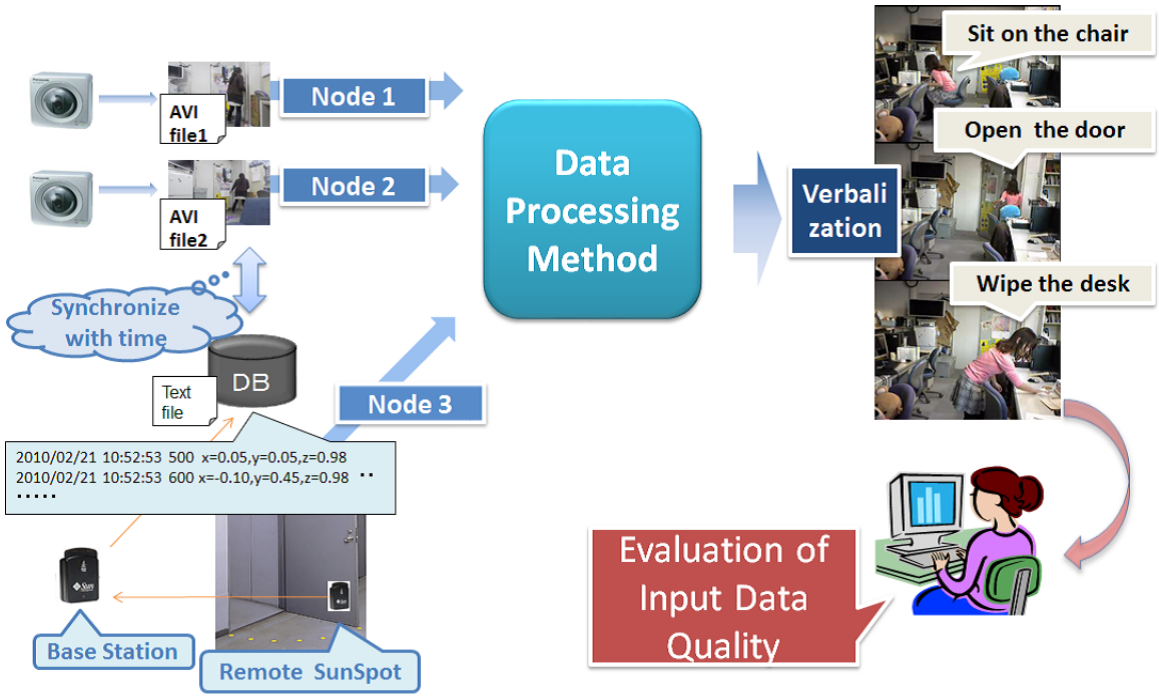

Figure 3.1: Execution environment of Human Activity Recognition System

In the evaluation experiments of this study, a human recognition system that verbalizes human activity [33] has been used. This application is called 'Lifelog

Analysis Application' in the rest of this paper. I have improved the application introduced in [33], and it works as follows:

This Lifelog Analysis Application describes human activity performed in the recorded data in natural language. As shown in the Fig. 3.1, the input data of the verbalization application is video image data taken by two cameras set up in two different angles in the room, as well as acceleration data taken by the SunSPOT [34] attached to moving objects like door or chair in the room. When the verbalization application receives the input data, video image data from two cameras and the acceleration data, they are synchronized based on time and called as Node1 (video data from camera1), Node2 (video data from camera2), and Node3 (acceleration data), respectively. These nodes are processed with some data processing method, and the verbalization application outputs verbalized expression and provides the information for users only when the predefined conditions are met.

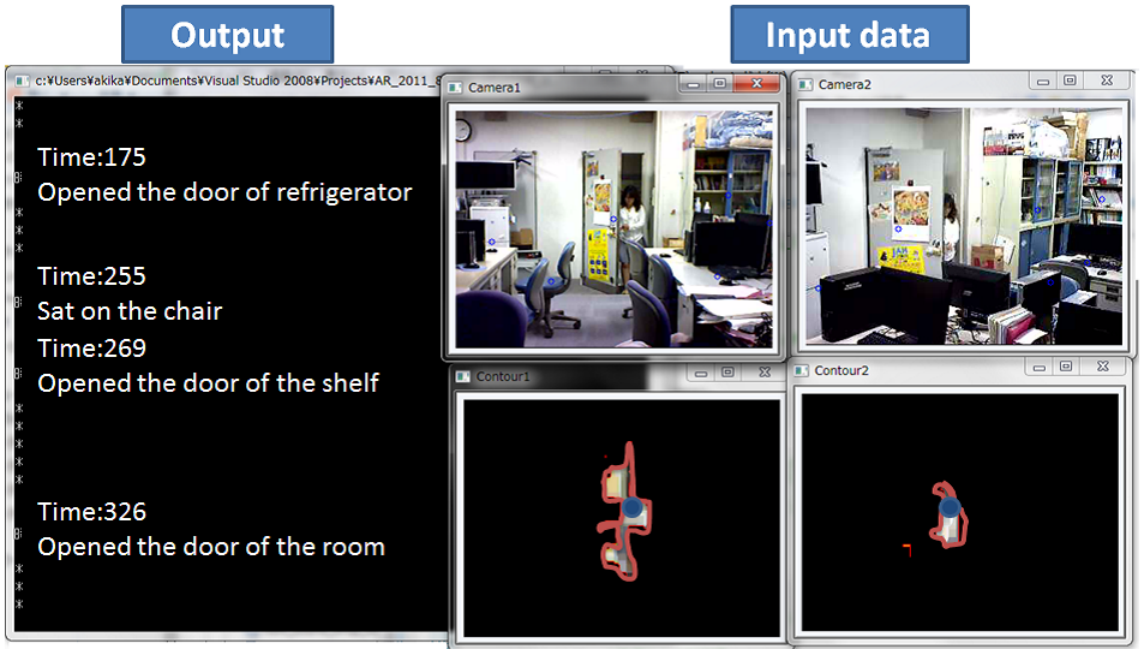

Figure 3.2: Behavior of Lifelog Analysis application

The output of the Lifelog Analysis Application is the verbalized expression of human activity. For example, as shown in the Fig. 3.2, if video images and

acceleration data of human activity 'open the door of the room' are put into the application, a verbalized expression 'Opened the door of the room' is kept to be outputted as a result of the analysis while the human is opening the door in the recorded video.

3.2 Development Environment

In my experiments, verbalization application has been developed with Microsoft Visual C++ 2008 Express Edition, and image processing of video frame has been executed with OpenCV library [35]. Network camera for taking video is Panasonic BB-HCM715 (130 megapixels, wired or wireless LAN) [36], acceleration data has been collected with SunSPOT [34]. As for the sound data, I have utilized VFW (Video For Windows) API [38].

3.2.1 SunSPOT : Acceleration Sensor Terminal

As shown in SunSPOT project page [34], the project SunSPOT was created to encourage the development of new applications and devices. It is designed from the ground up to allow programmers who never before worked with embedded devices to think beyond the keyboard, mouse and screen and write programs that interact with each other, the environment and their users in completely new ways. A Java programmer can use standard Java development tools such as NetBeans to write code.

I will explain how to collect acceleration data with sunSPOT. sunSPOT is a wireless sensor network device developed at Sun Lab as shown in Fig. 3.3. The device has a feature to collect acceleration data, temperature data, illumination data, and has eight LEDs. I have developed the processing program with Java, and the Java develop environment is Net Beans IDE 7.0.1.

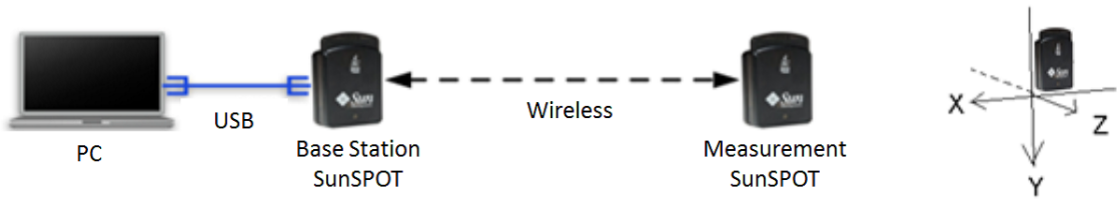

Figure 3.3: Acceleration sensor terminal SunSPOT

As shown in the Fig. 3.3, sunSPOT sensor terminal is composed of base station terminal and remote terminal for measurement. The base station terminal is used with USB connection to the PC, and the remote terminal is used attached with the objects. The two kinds of the terminals communicate with wireless connection. One base station terminal can communicate with several remote terminals, thus multi communication is possible.

3.2.2 Access Point

I will explain the terminal with which I did data quality experiment C mentioned in chapter 1. I have utilized wireless LAN multi pocket router MZK-MF300N [39] developed by planex company as shown in Fig. 3.4. There are AP mode and EC mode, IEEE802.11n, g, b is supported, and transmission rate can be fixed.

Figure 3.4: Access point

3.2.3 OchaHouse — Human Activity Recognition System

To evaluate the performance of human activity recognition system, I collected the real data in OchaHouse (Fig. 3.5). OchaHouse is an experimental smart house constructed in March 2009. The purpose of the Ochahouse is to implement and evaluate ubiquitous computing applications and cyber-physical systems research. The objectives of the OchaHouse project include, developing everyday computing applications, supporting health-care at home, and designing house of the future. I have setup cameras and other sensors like motion acceleration sensor and sound sensor to capture and collect "real"human actions inside the House. That are a natural data of people acting spontaneously. Particularly, I discuss the image data captured with camera in this paper. The camera is monitoring inside the house and start recording automatically if there is some actions and moving object. I collected a lot of kinds of human actions inside the house, for example, people open the door, sit on the chair, wipe the desk, open the door of shelf, etc. The human activity analytic system can detect what action people are doing inside the

house with image processing.

Figure 3.5: OchaHouse

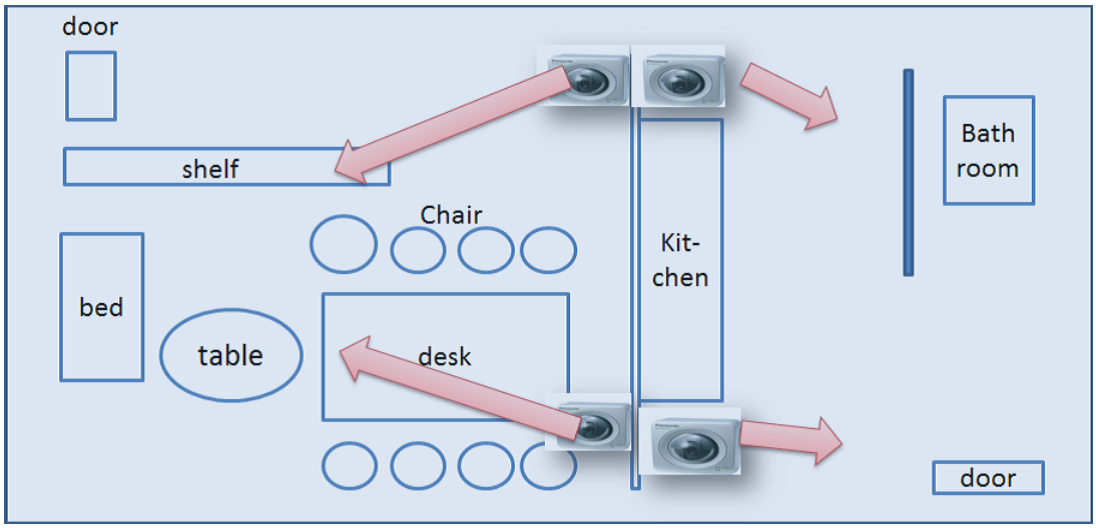

Figure 3.6: Inside OchaHouse

As shown in the Fig. 3.6, I have installed four cameras to record the video data from four different angles. The data captured with the cameras will be sent

to storage immediately through WLAN. The reason why I use WLAN is that I separated the system into three parts, data capturing part of cameras, data storage part, and data analytic part. Camera and storage are tied with Access Point (AP). Data sending protocol is UDP (User Datagram Protocol) as I put emphasis on higher data rate than image data quality even if there will be packet loss. In UDP, due to the influence of interference and noise around, packet loss will happen and throughput will decrease as a result. I assume the following situation when packet loss happens. The bandwidth of each terminal decreases and the quality of communication will deteriorate in the following situations: (1) several terminals are sharing the same AP and (2) several APs are near using closer channel. I used the wireless network packet loss data assuming situation (1) in my research.

Chapter 4

Data Quality Evaluation Framework

In this chapter, I will explain Data Quality Evaluation Framework which my proposed concept to evaluate input data quality deterioration impact on Lifelog Analysis Application output result strictly and quantitatively. To do evaluation of data quality deterioration impact on Lifelog Analysis Application, I have utilized an human activity recognition system whose input data is video data and acceleration data, and the input data is collected in the sensor space and transmitted trough wireless network toward a server to do activity recognition.

A framework to clarify the data quality evaluation experiment are proposed in this research. The framework is shown in Fig. 4.1. This framework is called "Data Quality Evaluation Framework". The Data Quality Evaluation Framework is divided into three layers, data collection layer, data processing layer, and information analysis layer. In data collection layer, what is the input data of the application and how to collect input data are defined. In data processing layer, how the input data is processed and what is the processing model are defined. In information analysis layer, what is the output of the application in other words, what is the analysis result are defined.

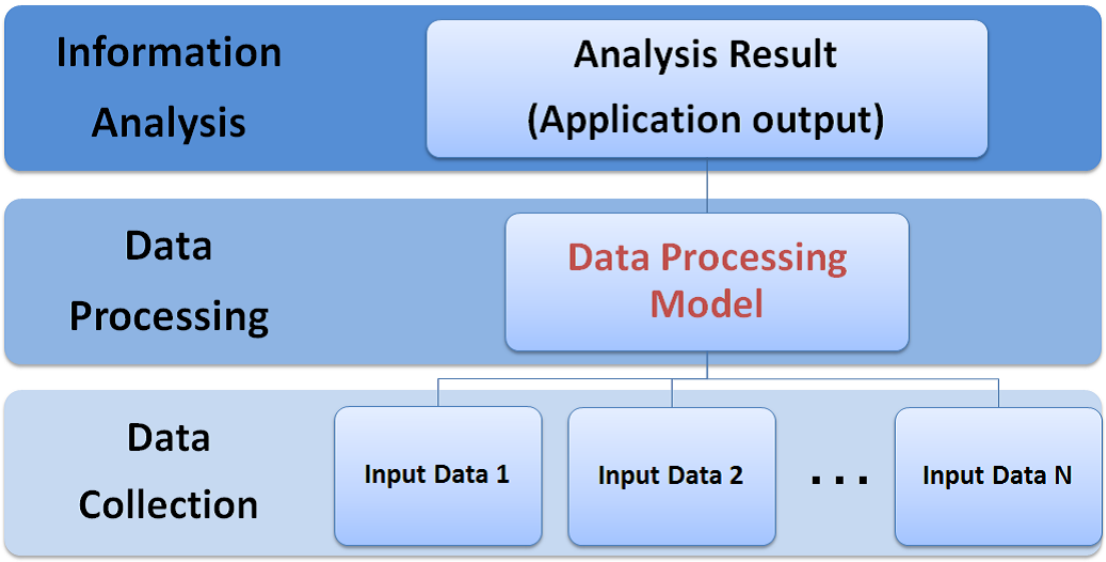

Figure 4.1: Concept of Data Quality Evaluation Framework

I suggest that the input data quality of Lifelog Analysis Application should be evaluated with this data quality evaluation framework. This framework is well defined because each of the layers are independent. Thus if Lifelog Analysis Application has some data quality problem, the problem can be absorbed in other layers. For example, if the input data quality of some Lifelog Analysis Application changes and it affects the output of the application badly, the impact of data quality on the application output can be measured with this framework. In addition, the countermeasure can be proposed by changing or improve data processing method defined in data processing layer. This data quality evaluation framework can be applied to other Lifelog Analysis Application as almost all Lifelog Analysis Application can be divided to three layer such as data input layer, data processing layer, and information analysis layer.

Before explaining how to apply data quality evaluation framework on the system in this research, typical lifelog analysis application will be discussed. When analyzing lifelog, human activity recognition is an important element. Typical

human action recognition system is shown in Fig. 4.2.

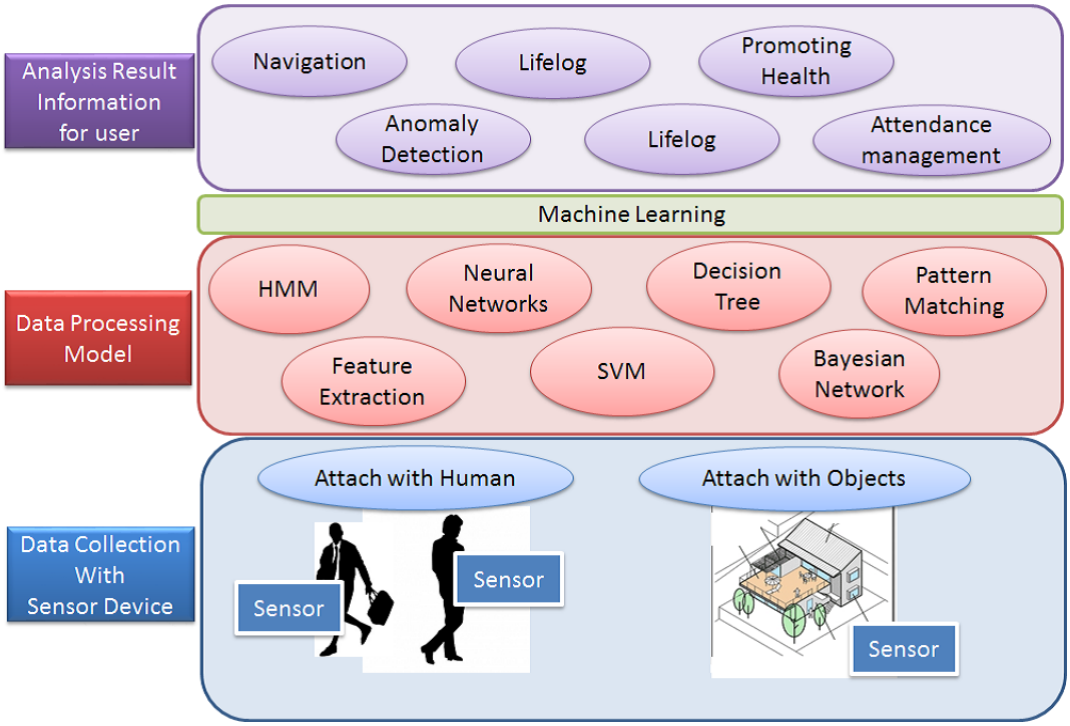

Figure 4.2: Typical human activity recognition system

Typically, the data is collected from some sensor devices in the human activity recognition system, and processed with many kinds of data processing model. To improve the processing accuracy, the elements in each model will go through the machine learning process, and the system output some useful result for the users. When collecting the data with sensor devices, there are two types of sensor devices attachment, one way is attaching to human, the other way is attaching to objects. In the experiment of this research, the sensor devices are attached to moving objects in the house. When processing the data with some model, there are also many types of models to apply with as shown in the Fig. 4.2. Even though the characteristics of these models are different, they can be divided to two types, the model which process the data per each frame, and the model which process the data per group of fames. Thus, two representative methods are applied to the verbalization application in the experiment of this research, Bayesian Classifier

which process the data per each frame, and HMM which process the data per group of frames. Each model has a track record in a variety of research fields, and the parameter setting is relatively simple compared with another models. After the data is processed with some models, machine learning procedures are necessary sometimes, but in this work, the machine learning process is omitted for reproducibility of the quality evaluation. Finally, the analysis result of the system is output for the users.

The proposed data quality evaluation framework has been applied to a Lifelog Analysis Application which are verbalization application and human motion alignment algorithm, three patterns of method have been applied.

In the first and second pattern, the process of Verbalization Application (one of the human activity recognition system) is divided into three layers, which consists of 'data collection layer', 'data processing layer' and 'information analysis layer' as shown in Fig. 4.3 and 4.4. Data collection layer is the input part of sensor data used for analysis, data processing layer is the part of theoretical analysis process with data per nodes given from data collection layer, and information analysis layer is the output part of the analyzed result given from data processing layer. In these two patterns, two kinds of different processing methods are applied to data processing layer, that is Bayesian Classifier model and HMM. The Bayesian Classifier method can cover the models which process the data per each frame, and HMM can cover the models which process the data per group of frames. The details of characteristics of these two methods are discussed in section 4.1.3 and 4.2.3. The influence of input data quality deterioration to the behavior of the verbalization application when verbalization has been performed through two different logical processes have been compared and evaluated. Which approach is suitable for absorb the variable type of data quality degradation are shown finally.

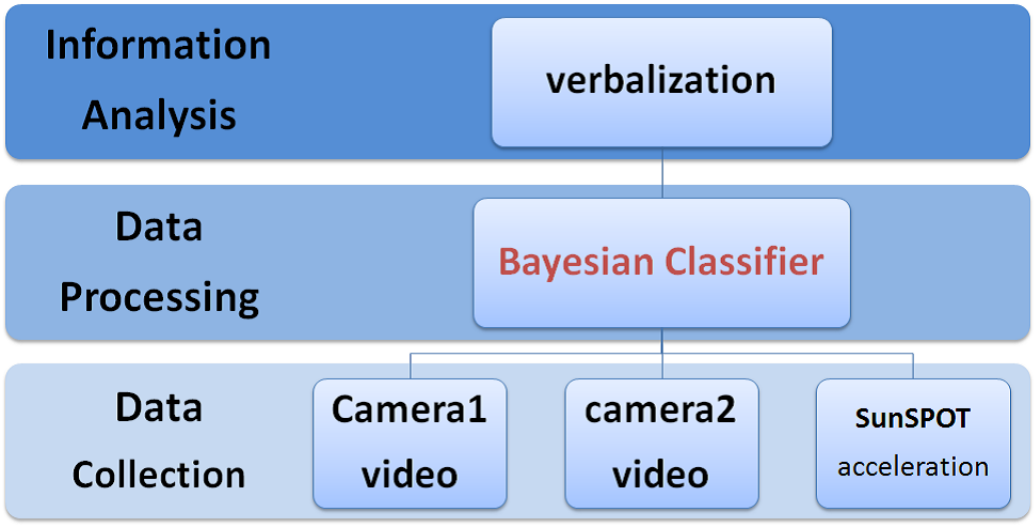

Figure 4.3: First pattern of data quality evaluation framework for verbalization application (Bayesian Classifier model)

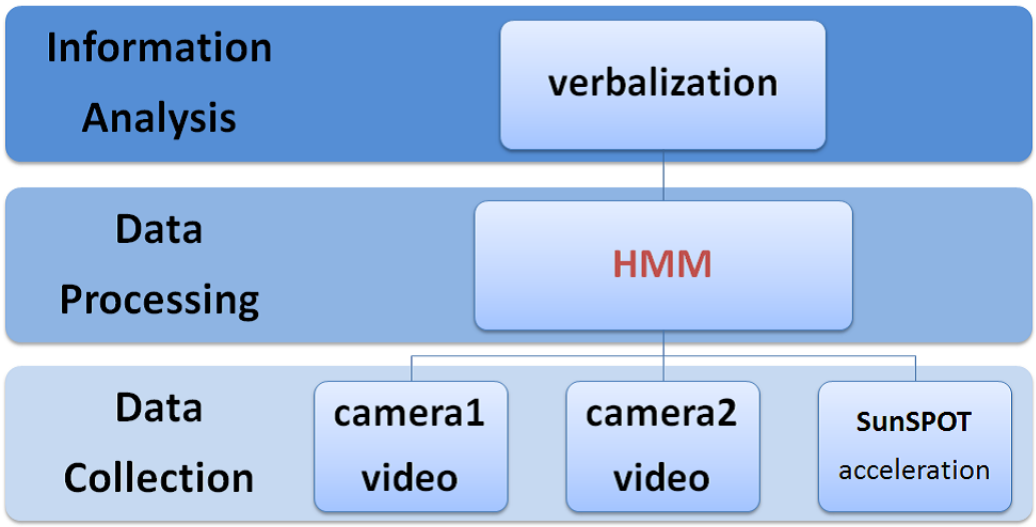

Figure 4.4: Second pattern of data quality evaluation framework for verbalization application (HMM)

In the third pattern, I have utilized GTW algorithm as one of the human activity recognition system and applied GTW algorithm to data processing layer of my data quality evaluation framework.

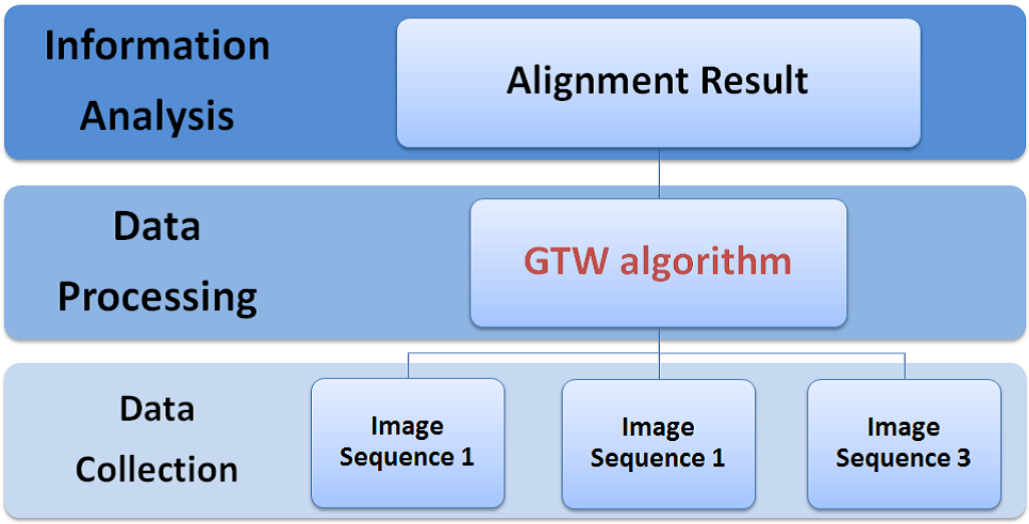

Figure 4.5: Third pattern of data quality evaluation framework for human activity recognition system (GTW)

GTW (The Generalized Time Warping) algorithm is a method of temporally alignment of multi-modal sequences from multiple human subjects performing similar human motion activities, which was proposed in Feng et al. [43]. The process of GTW alignment is also divided into three layers, which consists of 'data collection layer', 'data processing layer' and 'information analysis layer' as shown in Fig. 4.5. Data collection layer is the input part of image sequence data used for analysis. In each image sequence, human are playing almost the same action but the timing is different. Data processing layer is the part of GTW analysis algorithm process with image per frames. Information analysis layer is the output part of the GTW algorithm alignment result. In data quality data evaluation of GTW algorithm, I have utilized alignment error as output which value is defined by Feng et al.

In order to simplify the description, I define three patterns of data quality evaluation framework as follows:

Pattern (1) Data Quality Evaluation Framework applied to Verbalization Ap-

plication with Bayesian Classifier

Pattern (2) Data Quality Evaluation Framework applied to Verbalization Application with HMM

Pattern (3) Data Quality Evaluation Framework applied to Human Activity Recognition System with GTW algorithm

Outline. In section 4.1 of this chapter, I explain the process of each layer of the Data Quality Evaluation Framework applied to Verbalization Application with Bayesian Classifier, in section 4.2, the layers of the Data Quality Evaluation Framework applied to Verbalization Application with HMM, and in section 4.3, each layer of Data Quality Evaluation Framework applied to Human Activity Recognition System with GTW algorithm.

4.1 Pattern (1) : Data Quality Evaluation Framework applied to Verbalization Application with Bayesian Classifier

4.1.1 Data Collection Layer

To begin with, data collection layer is the input part of video image data recorded by the network cameras and acceleration data collected by SunSPOT.

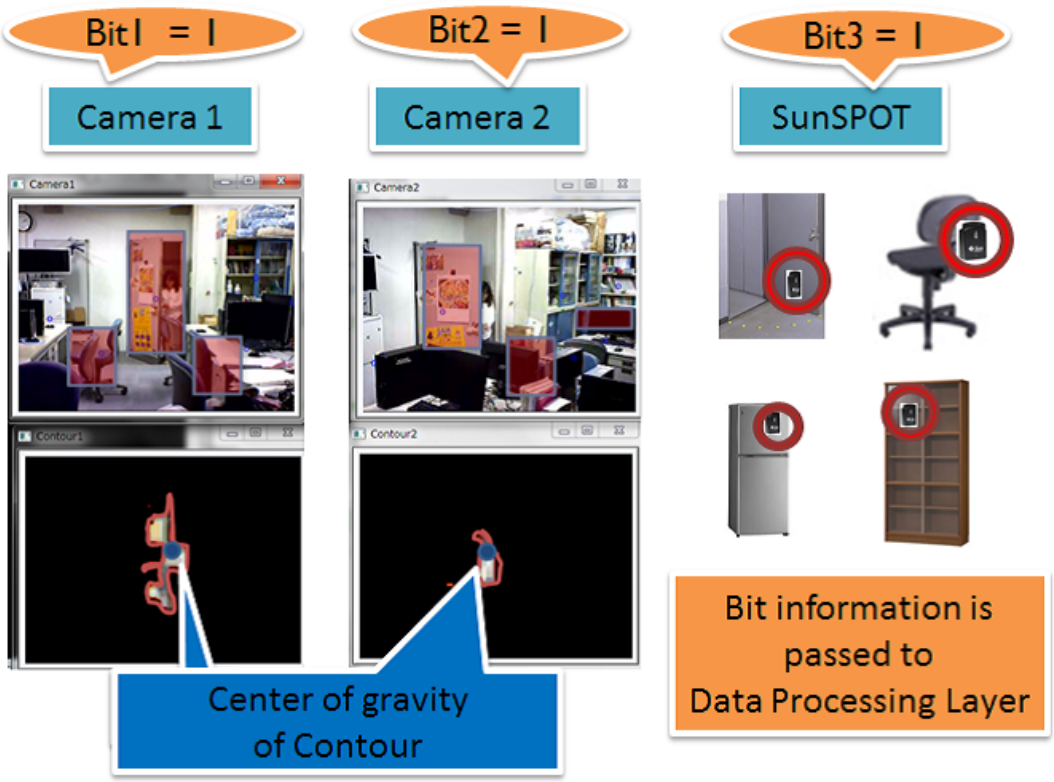

Figure 4.6: Data collection layer of Pattern (1)

In the video data processing, the contours of the difference between the current image frame and the previous image frame for each frame are extracted (the red line shown in Fig. 4.6), and a center of gravity of the portion surrounded by the contours (the blue point shown in Fig. 4.6) is sought. The contours are supposed to be moving object that is a human, and the center of gravity is center of the human. The number of overlapping of the center of the human and an object (door, chair, and so on) is counted, and if the count exceeds a pre-defined threshold, Bit1 and Bit2 are marked which are bits for two network cameras.

For the acceleration data processing, Bit3 is marked when the variation of x, y, z-axis acceleration collected each time exceeds a pre-defined threshold. As shown in the Fig. 4.7, the input file is the raw data of x, y, z-axis acceleration and the difference between the previous value is calculated and written to the the

rightmost column of output file as xyz pattern. The pattern definition is shown in Tab. 4.1. To clarify which axis of x, y, z reacted, I have defined eight patterns of x, y, z reaction as one digit number.

"Threshold 1"in the Fig. 4.7 stands for whether each x, y, z-axis reacted or not. "Threshold 2"stands for how many times the reacted node continue. If the xyz pattern continue to be bigger than zero for "Threshold 2"times, the acceleration sensor is treated as "reacted".

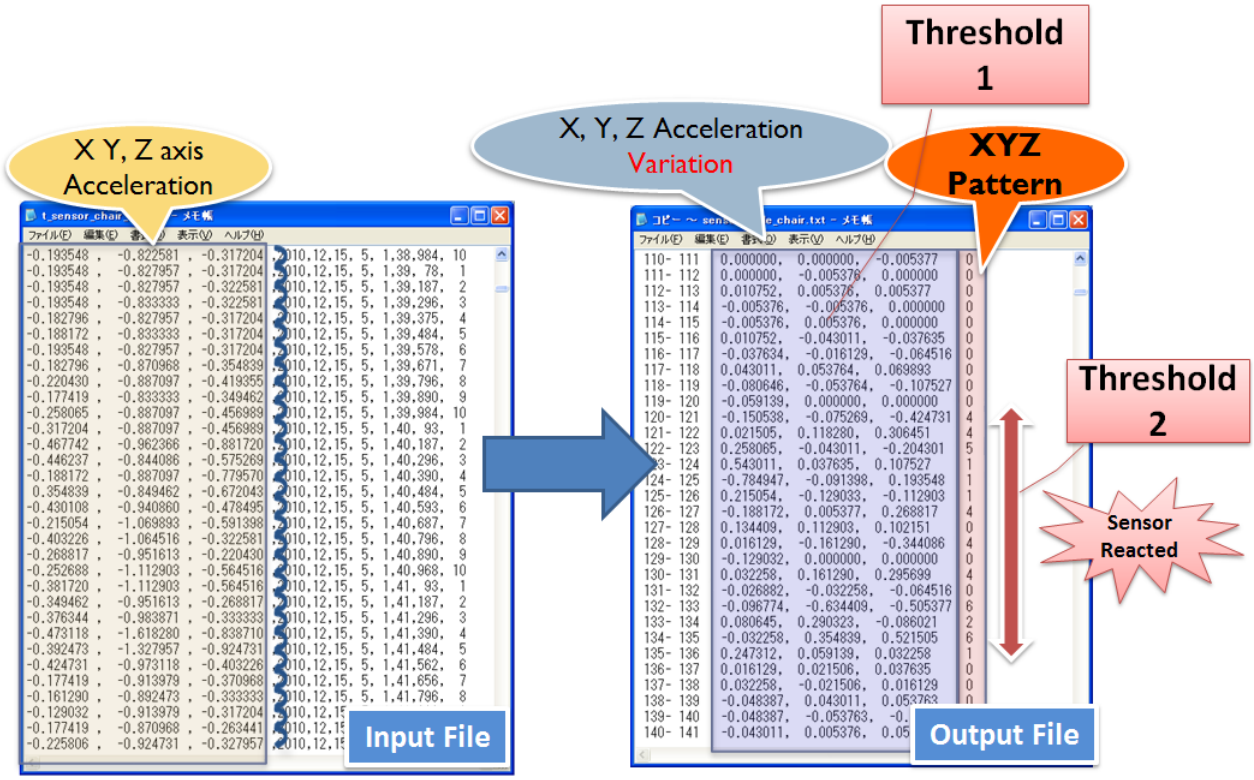

Figure 4.7: Definition of Acceleration Reaction

Table 4.1: XYZ pattern definition

XYZ pattern	Node definition
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

These three bits, Bit1 of camera1, Bit2 of camera2, and Bit3 of acceleration sensor are given to data processing layer.

4.1.2 Data Processing Layer

For data processing layer of Pattern (1), Bayesian Classifier model have been applied (Fig. 4.8).

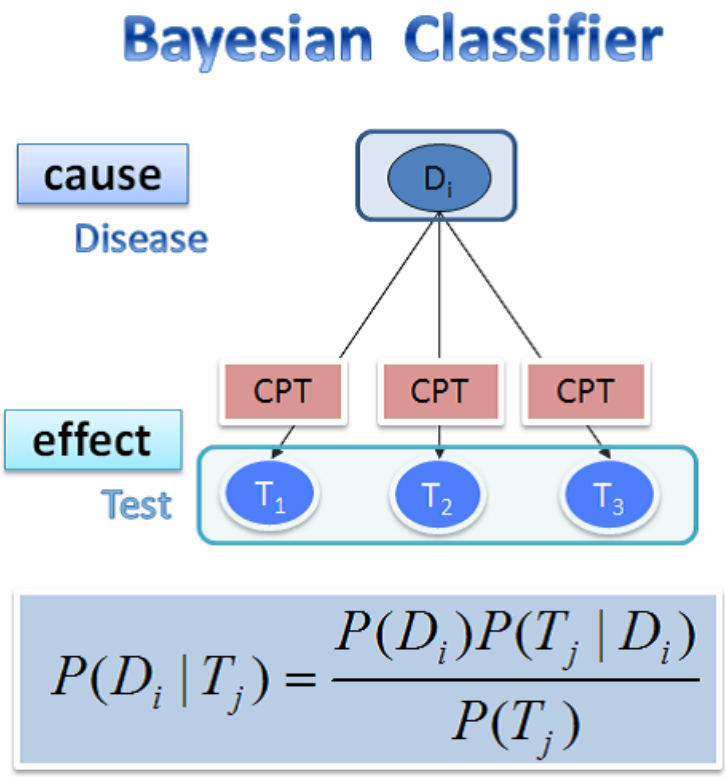

Figure 4.8: Data processing layer with Bayesian Classifier model

Bayesian Classifier is a probabilistic inference model which describes causal relationship with conditional probability table (CPT) and predicts the cause from results. In the modeling shown in the Fig. 4.8, CPT of a chance of testing positive (T_j) for the disease (D_i) is known and given. When I have the testing result (T_j) and want to know which disease (D_i) caused the result, I can predict the most plausible cause, that is, disease by getting D_i which maximizes the formula. Like this medical test, Bayes model is strong in the case with many uncertainty event.

Stochastic inference is a method for calculating the probability of event B occurrence when the event A occurred. On the other hand, people want to find the probability even if the condition is uncertain, for example, the cause estimation from symptoms such as there is a fever or throat hurts to explore the pathogenesis, or to predict the probability of bankruptcy from sales, prohibits, assets and borrowing situation. In these situation, Bayes model can be applied.

Bayesian Classifier has been utilized not only in medical or economic field, but also applied to many types system in information science field. I will list up some examples of past research using Bayesian Classifier model. Example 1 : Email support system : Automatically open the calender application for user after specific email was opened when needed. Example 2 : Human monitoring : Elderly people care support, or children accident prevention

There ware also many human activity recognition system with Bayes model mentioned in [51][52][53][54]. I have made use of Bayes model as representative method to modeling human activity recognition.

At modeling of verbalization application in this research, 'human activities' are the cause nodes and 'reaction of two angled network cameras and SunSPOT reaction' are the result nodes. Details will be discussed in section 4.1.3.

4.1.3 Information Analysis Layer : Information analysis based on Bayesian Classifier

I will discuss how to apply the Bayesian Classifier model to data processing layer of the verbalization application. The modeling of verbalization application in the information analysis layer with Bayesian Classifier is described, as shown in Fig. 4.9.

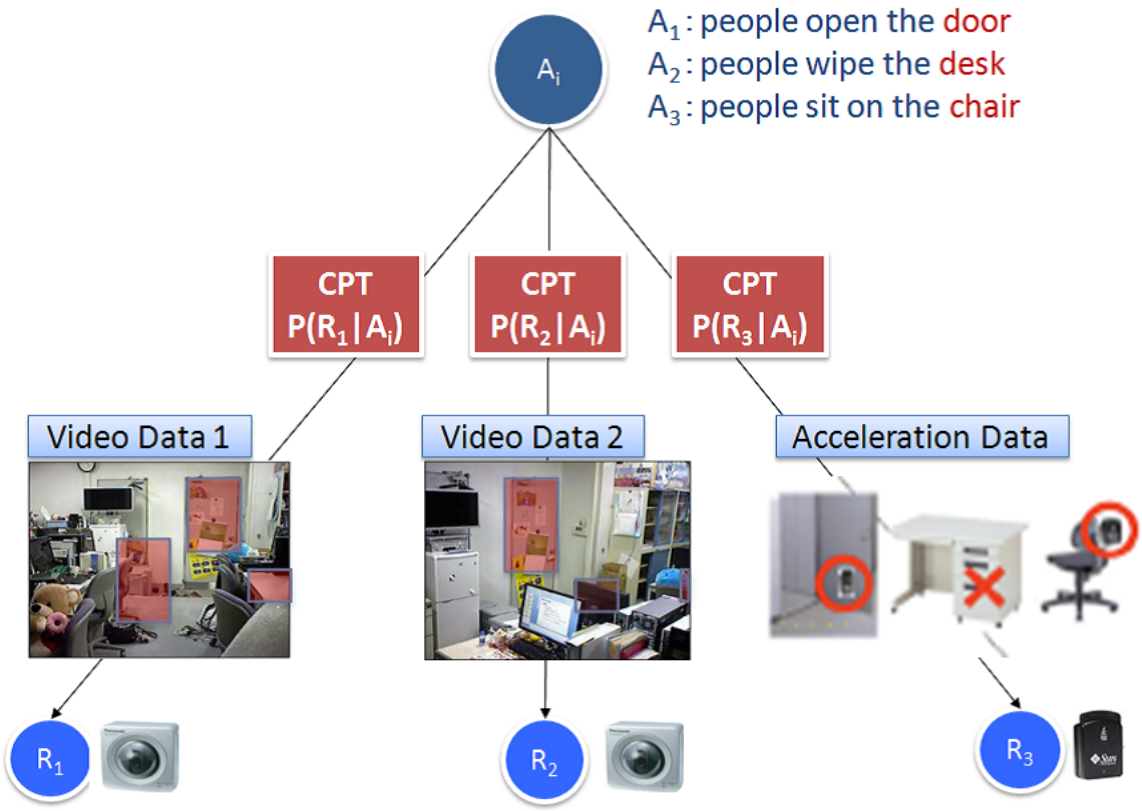

Figure 4.9: Modeling of verbalization application with Bayesian Classifier

Three bits information of two network cameras and SunSPOT given from data collection layer are R_1, R_2, R_3, respectively. In the modeling of verbalization application, R_1, R_2, R_3 are the result nodes and they are reaction of two network cameras and SunSPOT, respectively. A_i is the cause node and it means activities of people and more than one activities can be defined. Note that I have listed only three actions for convenience of explanation of modeling. The number of action

is unlimited as long as the process time meets commensurate. In data quality evaluation experiments, I have defined five activities of human.

For example, when the following three actions are dealt with;

- A_1 : people open the door
- A_2 : people wipe the desk
- A_3 : people sit on the chair

With given CPT, the human activity recognition system output one action (A_i) which maximize the following formula:

$$P(A_i|R_1, R_2, R_3) = \frac{P(A_i)P(R_1, R_2, R_3|A_i)}{P(R_1, R_2, R_3)}$$

Note that the detailed value of $P(A_i|R_1, R_2, R_3)$ is not required as only the value i which maximize the formula is important. So I have converted the formula as follows:

$$\begin{aligned} & \underset{i}{argmax}\, P(A_i|R_1, R_2, R_3) \\ = \; & \underset{i}{argmax}\, \frac{P(A_i)P(R_1, R_2, R_3|A_i)}{P(R_1, R_2, R_3)} \cdots (1) \\ = \; & \underset{i}{argmax}\, P(A_i)P(R_1, R_2, R_3|A_i) \end{aligned}$$

(1) Denominator is common

Table 4.2: Given CPT to Bayesian Classifier Model

$R_1 \backslash A$	A_1	A_2	A_3
0	0.1	0.4	0.5
1	0.9	0.6	0.5

$R_2 \backslash A$	A_1	A_2	A_3
0	0.1	0.9	0.8
1	0.9	0.1	0.2

$R_3 \backslash A$	A_1	A_2	A_3
0	0.01	0.9	0.9
1	0.99	0.1	0.1

$R_1 \backslash A$	A_1	A_2	A_3
0	0.9	0.1	0.9
1	0.1	0.4	0.1

$R_2 \backslash A$	A_1	A_2	A_3
0	0.9	0.9	0.9
1	0.1	0.1	0.1

$R_3 \backslash A$	A_1	A_2	A_3
0	1.0	1.0	1.0
1	0.0	0.0	0.0

$R_1 \backslash A$	A_1	A_2	A_3
0	0.9	0.9	0.1
1	0.1	0.1	0.9

$R_2 \backslash A$	A_1	A_2	A_3
0	0.5	0.9	0.6
1	0.5	0.1	0.4

$R_3 \backslash A$	A_1	A_2	A_3
0	0.9	0.9	0.01
1	0.1	0.1	0.99

CPT is the Conditional Probability table. As shown in the Fig. 4.9, I have applied each object as the node R_1, R_2, R_3. The decimal in the Tab. 4.2 is the conditional Probability $P(R_j|A_i)$, which is the probability that the event R_j occurs when A_i occurs.

As for the nine CPT shown in the Tab. 4.2, the upper three CPT surrounded by blue square are used for the action A_1, the middle three CPT are used for Action A_2, and the bottom three CPT are used for A_3. The three CPT in the left line are R_1, which is the probability of the camera1 sensor reacts, the middle line CPT are R_2, which is the probability of the camera2 sensor reacts, and the CPT in the right line are R_3, which is the probability the sunSPOT reacts.

The red number in the CPT is set smaller than other number as I emphasized the sunSPOT reaction to make it easy to output the result when the sunSPOT reacts. The value "0.0"is the product on which the sunSPOT is not attached.

I will show how to apply the CPT on my human activity recognition system with blue line surrounded CPT. The leftmost CPT of blue line surrounded is $P(R_1|A_1)$, which means "the probability of the door area defined in camera1 reacts

and the probability of that doesn't react".

- $P(R_1 = 0|A_1 = 1) = 0.1$

  :The probability that camera 1 sensor does not react when "the door is opened"is 0.1.

- $P(R_1 = 1|A_1 = 1) = 0.9$

  :The probability that camera 1 sensor reacts when "the door is opened"is 0.9.

- $P(R_1 = 0|A_2 = 1) = 0.4$

  :The probability that camera 1 sensor does not react when "the desk is wiped"is 0.4.

- $P(R_1 = 1|A_2 = 1) = 0.6$

  :The probability that camera 1 sensor reacts when "the desk is wiped"is 0.6.

- $P(R_1 = 0|A_3 = 1) = 0.5$

  :The probability that camera 1 sensor does not react when "people sit on the chair"is 0.5.

- $P(R_1 = 1|A_3 = 1) = 0.5$

  :The probability that camera 1 sensor reacts when "people sit on the chair"is 0.5.

I will explain how to determine whether the system output the human's action or not. I will use the case "people open the door"as a example. As it is the output of A_1, the blue line surrounded CPT of the Tab. 4.2 is referred. I have divided R_1, R_2, R_3 to eight cases of bit from 000 to 111. The system output the result "the door is opened"when A_1 maximize the formula and this is processed on each frame of the movie data collected with camera sensor.

1. $(R_1,\ R_2,\ R_3) = (0,\ 0,\ 0)$

   $P(R_1, R_2, R_3|A_1) = 0.1 * 0.1 * 0.01 = 0.0001$

   $P(R_1, R_2, R_3|A_2) = 0.4 * 0.9 * 0.9 = 0.324$

   $P(R_1, R_2, R_3|A_3) = 0.5 * 0.8 * 0.9 = 0.36$

   No output because A_1 dose not maximize the formula.

2. $(R_1,\ R_2,\ R_3) = (0,\ 0,\ 1)$

   $P(R_1, R_2, R_3|A_1) = 0.1 * 0.1 * 0.99 = 0.0099$

   $P(R_1, R_2, R_3|A_2) = 0.4 * 0.9 * 0.1 = 0.036$

   $P(R_1, R_2, R_3|A_3) = 0.5 * 0.8 * 0.1 = 0.04$

   No output because A_1 dose not maximize the formula.

3. $(R_1,\ R_2,\ R_3) = (0,\ 1,\ 0)$

   $P(R_1, R_2, R_3|A_1) = 0.1 * 0.9 * 0.01 = 0.0009$

   $P(R_1, R_2, R_3|A_2) = 0.4 * 0.1 * 0.9 = 0.036$

   $P(R_1, R_2, R_3|A_3) = 0.5 * 0.2 * 0.9 = 0.09$

   No output because A_1 dose not maximize the formula.

4. $(R_1,\ R_2,\ R_3) = (0,\ 1,\ 1)$

   $P(R_1, R_2, R_3|A_1) = 0.1 * 0.9 * 0.99 = 0.089$

   $P(R_1, R_2, R_3|A_2) = 0.4 * 0.1 * 0.1 = 0.004$

   $P(R_1, R_2, R_3|A_3) = 0.5 * 0.2 * 0.1 = 0.01$

   Output the result because A_1 maximize the formula.

5. $(R_1,\ R_2,\ R_3) = (1,\ 0,\ 0)$

   $P(R_1, R_2, R_3|A_1) = 0.9 * 0.1 * 0.01 = 0.0009$

   $P(R_1, R_2, R_3|A_2) = 0.6 * 0.9 * 0.9 = 0.486$

   $P(R_1, R_2, R_3|A_3) = 0.5 * 0.8 * 0.9 = 0.36$

   No output because A_1 dose not maximize the formula.

6. $(R_1,\ R_2,\ R_3) = (1,\ 0,\ 1)$

$P(R_1, R_2, R_3|A_1) = 0.9 * 0.1 * 0.99 = 0.0891$

$P(R_1, R_2, R_3|A_2) = 0.6 * 0.9 * 0.1 = 0.054$

$P(R_1, R_2, R_3|A_3) = 0.5 * 0.8 * 0.1 = 0.04$

Output the result because A_1 maximize the formula.

7. $(R_1,\ R_2,\ R_3) = (1,\ 1,\ 0)$

$P(R_1, R_2, R_3|A_1) = 0.9 * 0.9 * 0.01 = 0.0081$

$P(R_1, R_2, R_3|A_2) = 0.6 * 0.1 * 0.9 = 0.054$

$P(R_1, R_2, R_3|A_3) = 0.5 * 0.2 * 0.9 = 0.09$

No output because A_1 dose not maximize the formula.

8. $(R_1,\ R_2,\ R_3) = (1,\ 1,\ 1)$

$P(R_1, R_2, R_3|A_1) = 0.9 * 0.9 * 0.99 = 0.8019$

$P(R_1, R_2, R_3|A_2) = 0.6 * 0.1 * 0.1 = 0.006$

$P(R_1, R_2, R_3|A_3) = 0.5 * 0.2 * 0.1 = 0.01$

Output the result because A_1 maximize the formula.

The output of A_2, A_3 is the same.

4.2 Pattern (2) : Data Quality Evaluation Framework Applied to Verbalization Application with HMM

4.2.1 Data Collection Layer

Next, data collection layer is the input part of video image data recorded by the network cameras and acceleration data collected by SunSPOT. In pattern (2), the process of Data Collection Layer is as same as that of pattern (1). Thus, the description of Data Collection Layer in pattern (2) is omitted.

The three bits, Bit1 of camera1, Bit2 of camera2, and Bit3 of acceleration sensor are given to data processing layer as well as pattern (1).

4.2.2 Data Processing Layer

For data processing layer of Pattern (2), HMM have been applied (Fig. 4.10).

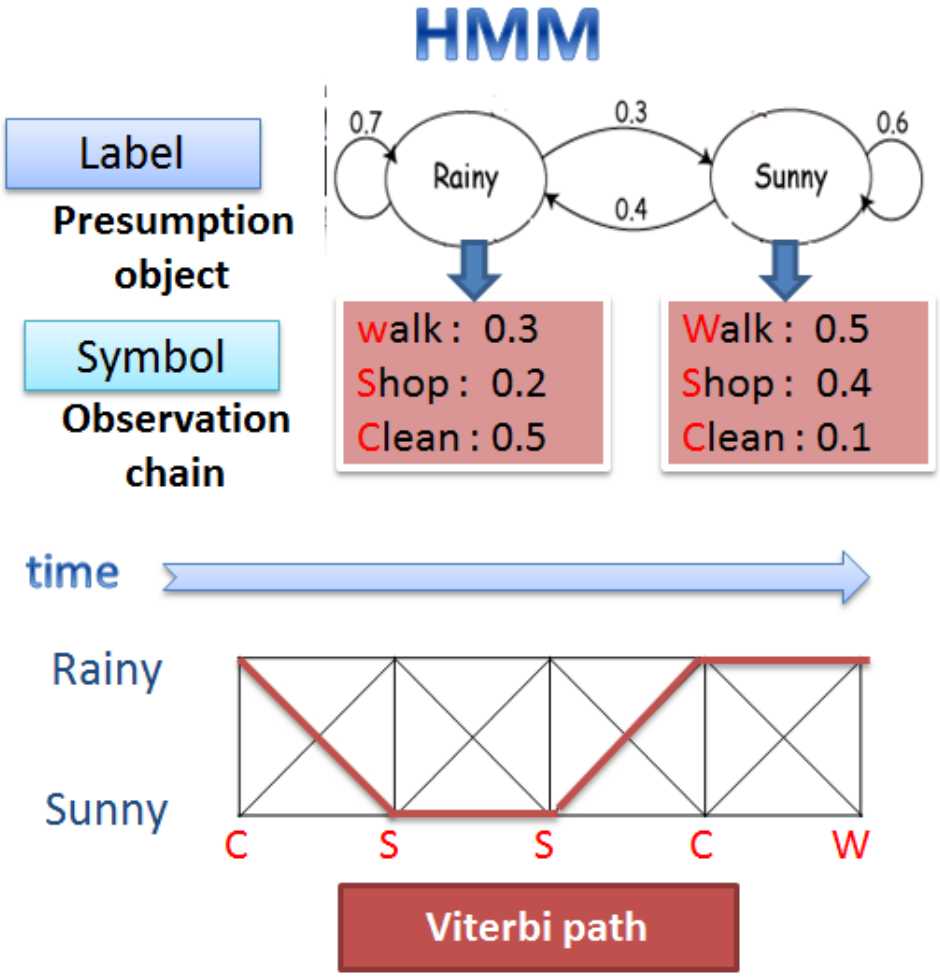

Figure 4.10: Data processing layer with HMM

HMM (Hidden Marcov Model) is a probabilistic model to estimate the system's internal state transitions based on Markov process from probability distribution of symbols according to each state. In the example of the right of Fig. 4.10, only the transition probability of weather (rainy, sunny) is known and the occurrence probability of raining or sunny is unknown (that is Hidden). In addition, the

probability distribution of three kinds of activities (walk, shop, clean) of a human during each weather is also given. In this case, I can predict the transition path of change of the weather from observed activity of people. This optimal path is called 'Viterbi path'.

HMM is an effective statistical method for modeling uncertain time series data. It is an automaton with a stochastic state transition and stochastic symbol output. Viterbi algorithm is a method to calculate maximum likelihood state sequence which output the symbol sequence (observed output sequence). This algorithm is not necessarily return the correct maximum likelihood state transition sequence. The maximum likelihood state transition sequence at a certain time t only depends on output symbol sequence observed until time t and the most probable maximum likelihood state transition sequence observed till time $t-1$.

Generally, HMM is defined by following five terms [59].

$$
\begin{aligned}
S &= \{s_1, s_2, ..., s_N\} \cdots Finite\ set\ of\ State \\
\Sigma &= \{o_1, o_2, ..., o_M\} \cdots Finite\ set\ of\ output\ symbols \\
A &= \{a_i j\} \cdots State\ transition\ probability\ distribution \\
&\quad\ Transition\ probability\ from\ the\ state\ q_i\ to\ state\ q_j \\
B &= \{b_i(o_t)\} \cdots Symbol\ output\ probability\ distribution \\
\pi &= \{\pi_i\} \cdots Initial\ state\ probability\ distribution
\end{aligned}
$$

Viterbi algorithm is particularly based on HMM and it is a kind of dynamic programing algorithm to find the most plausible sequence called viterbi path which is the cause of observed event sequence.

When the model M and the sequence X are given, the likelihood $P(X, M)$ is calculated with following formula :

$$
\begin{aligned}
P(X, M) &= \max_{1 \le i \le k} p_i(n) \\
p_i(t) &= \begin{cases} \pi_i b_i(x_1) & (t = 1) \\ max_{1 \le j \le k}\, p_j(t-1) a_j i b_i(x_i) & (2 \le t \le n) \end{cases}
\end{aligned}
$$

HMM was applied to a lot kind of research such as speech synthesis, speech analysis ([55][56]), image-based sign language work ([57][58]). Like these research, HMM was applied to time sequence data, including image.

While the observed data is image data sequence in verbalization application, I have defined verbalization threshold from the pattern of the optimal state transition sequence which is pattern of viterbi path to judge in the verbalization application. The detail is discussed in section 4.2.3.

4.2.3 Information Analysis based on HMM

I will explain how to apply HMM to my system. I have collected two angled movie data with two network cameras as shown in Fig. 3.1, and calculated the distance between the people doing activity and the center of gravity of defined object. The result of analysis of the distance is shown in Fig. 4.11. The vertical axis of the graph shows the time of the movie data and the horizontal axis of the graph shows the distance between people and the defined objects.

The color background part is when people doing actions. As the graph shows, the distance between people and the defined objects becomes under 50 when people doing the activity related each object, and the distance becomes 50 - 150 when people begins to do the activity. I have defined D_1, D_2, D_3 as distance 0 - 50, 50 - 150, 150 - 200. The average of two camera's distance were used in the HMM method.

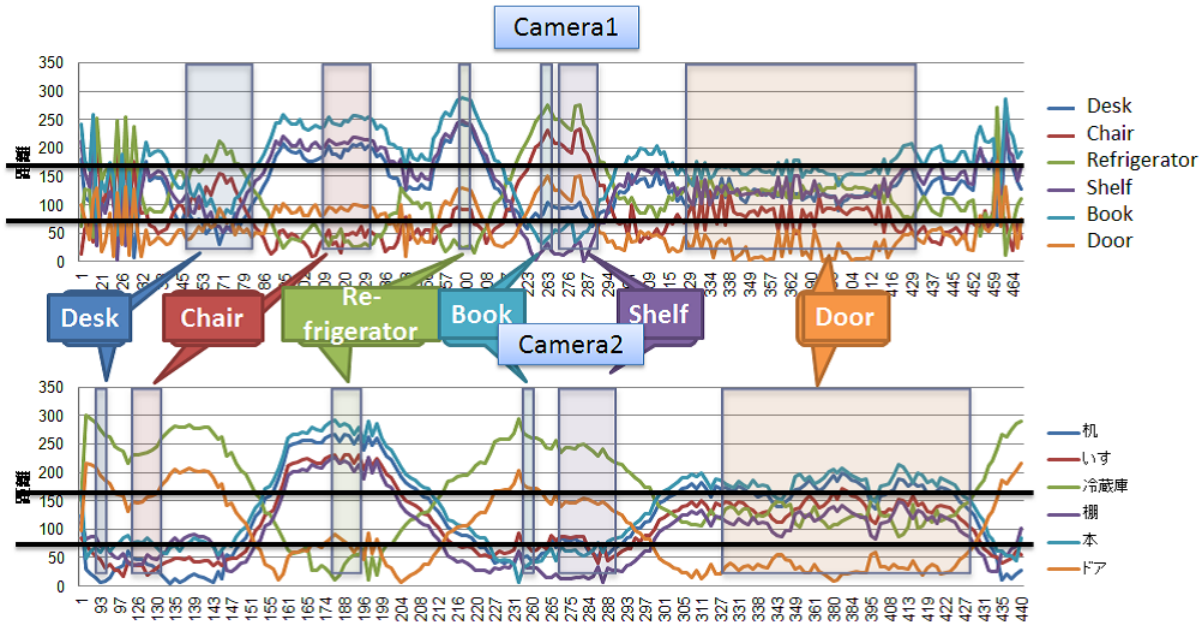

Figure 4.11: The distance of center of gravity between people and the object.

I have defined HMM state and probabilities as follows:

$$
\begin{aligned}
S &= \{d_1, d_2, d_3, d_4\} \cdots (1) \\
\Sigma &= \{o_1, o_2, ..., o_M\} \cdots Image\ sequence\ with\ camera\ Bits \\
A &= \{a_{ij}\} \cdots State\ transition\ probability\ of\ four\ states\ S \\
&\quad\ The\ probability\ is\ defined\ as\ (2) \\
B &= \{b_i(o_t)\} \cdots Probability\ of\ camera\ bits\ reaction \\
\pi &= \{\pi_i\} \cdots (3)
\end{aligned}
$$

(1) d_1 stands for D_1, the state that distance between people and the object is 0 - 50. d_2 stands for D_2, distance is 50 - 150. d_3 stands for D_3, distance is 150 - 200. d_4 stands for Ac, the state that acceleration attached with the object reacts, in other words, acceleration Bit in Data Collection Layer is 1.

- Ac : Acceleration terminal SunSPOT reacts

- D_i : Distance between the center of a human and areas of pre-defined objects ($D_1 < D_2 < D_3$)

(2) The state transition probability is as follows:

There are five matrix A_{door}, A_{Chair}, A_{Desk}, $A_{Refrigerator}$, A_{Shelf} but I just write down the representative two objects(activities), one is with acceleration sensor attached, the other is with no acceleration sensor. The transition probability in A_{door} matrix was calculated by recoded video data in which people are opening the door. The denominator number is based on the number of image frames and how many times people stayed at the distance.

$$A_{door} = \begin{pmatrix} 0.4 & 0.1 & 0 & 0.5 \\ 0.9 & 0.06 & 0.04 & 0 \\ 0 & 0.85 & 0.15 & 0 \\ 0.1 & 0 & 0 & 0.9 \end{pmatrix}$$

$$A_{desk} == \begin{pmatrix} 0.7 & 0.3 & 0 \\ 0.8 & 0.08 & 0.12 \\ 0 & 0.8 & 0.2 \end{pmatrix}$$

Next, B stands for the probability of two camera bits reaction when the people exist in each state of distance. The detail is described in first half part of section 4.1.1. The value of Bit1 and Bit2 (0 or 1) means whether the human is overlapping with the defined object (door, chair, desk, and so on...). Note that the matrix B is defined for each object. I just write down representative two matrix, B_{door} and

B_{desk}. The observed sequence is consisted of following four values :

$$\begin{aligned} \Sigma &= \{o_1, o_2, ..., o_M\} \\ o_t &= \{00, 01, 10, 11\} \end{aligned}$$

The matrix B is defined as follows. Note that the size of B_{door} and B_{desk} is different as B_{desk} does not have acceleration attached.

$$B_{door} = \begin{pmatrix} 0.02 & 0.09 & 0.09 & 0.8 \\ 0.04 & 0.08 & 0.8 & 0.08 \\ 0.7 & 0.02 & 0.2 & 0.08 \\ 0.01 & 0.01 & 0.08 & 0.9 \end{pmatrix}$$

$$B_{desk} = \begin{pmatrix} 0.04 & 0.1 & 0.2 & 0.66 \\ 0.1 & 0.3 & 0.2 & 0.4 \\ 0.88 & 0.05 & 0.05 & 0.02 \end{pmatrix}$$

Then, I have defined verbalization threshold from the pattern of Viterbi path. When the Viterbi path likelihood is over the threshold, the verbalization is output and the Viterbi path keep staying State D_1 or Ac, verbalization continues to be output. Note that there are as many Viterbi path as the number of defined object (action). The modeling of information analysis layer based on HMM is shown in Fig. 4.12 and Fig. 4.14. Fig. 4.12 shows the model of objects with acceleration

sensor terminal attached, and Figure Fig. 4.14 shows the model of objects without acceleration sensor terminal attached. Two Viterbi path pattern are shown in the Fig. 4.13 and Fig. 4.15.

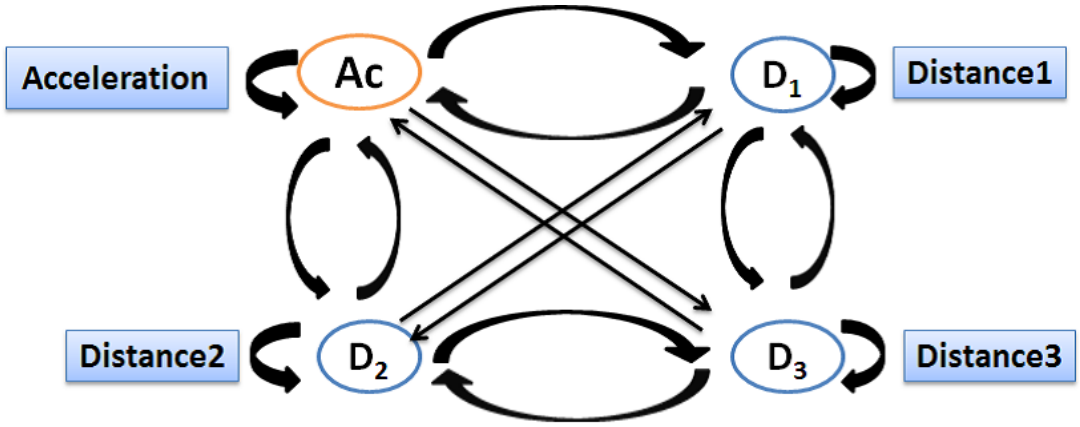

Figure 4.12: Modeling of verbalization application with acceleration terminal attached

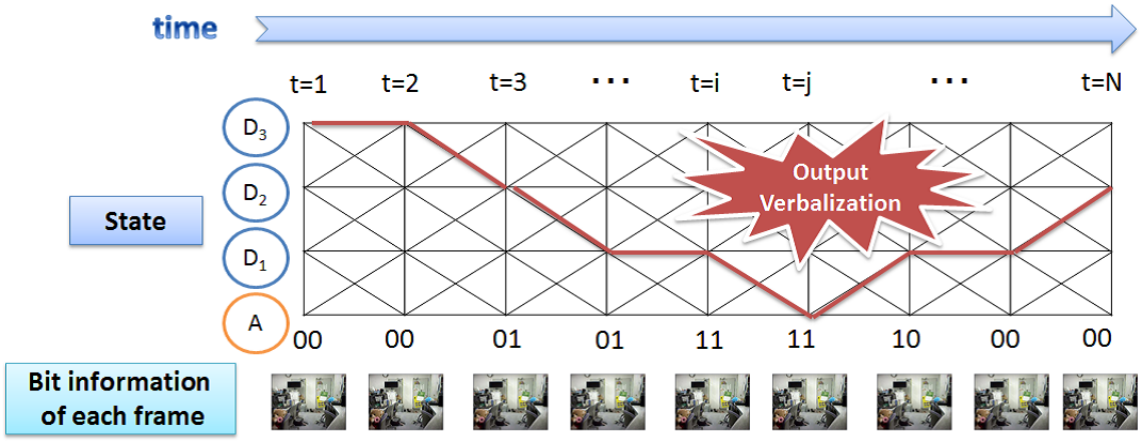

Figure 4.13: Viterbi path pattern with acceleration terminal attached

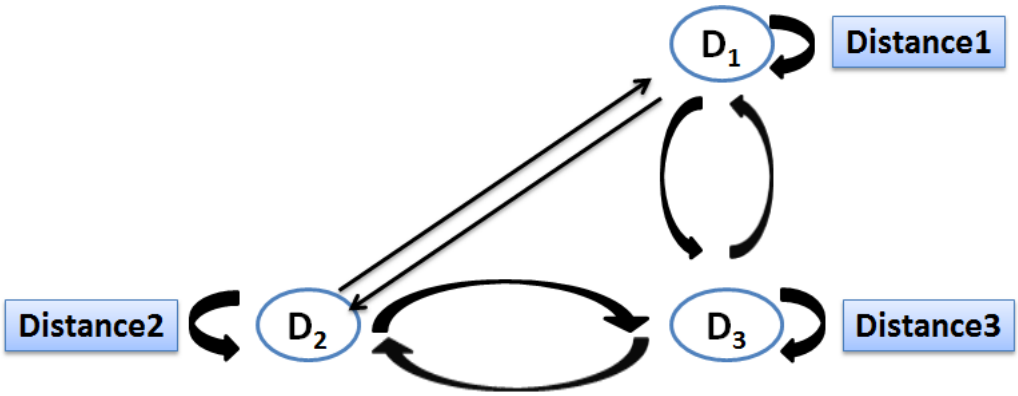

Figure 4.14: Modeling of verbalization application without acceleration terminal attached

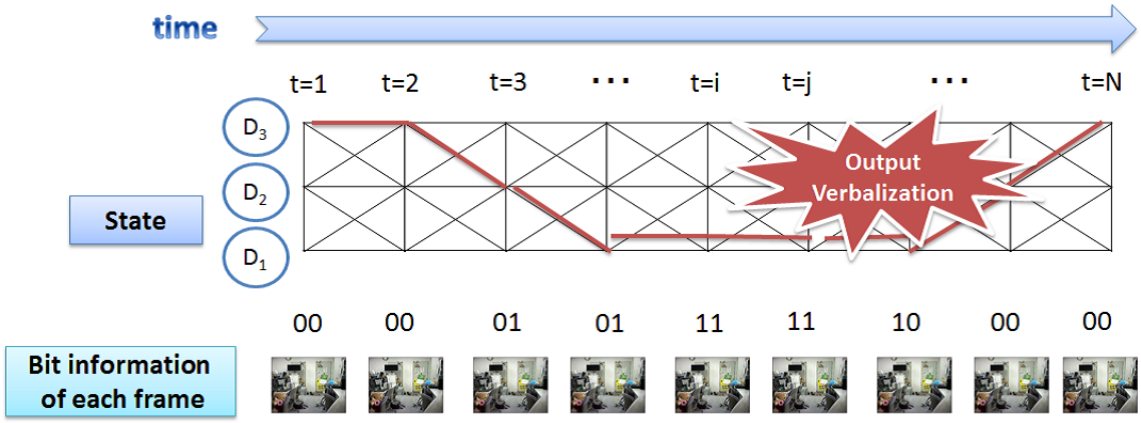

Figure 4.15: Viterbi path pattern without acceleration terminal attached

For example, the verbalization application output occur if following pattern of Viterbi path is observed. Note that the initial state is D_3.

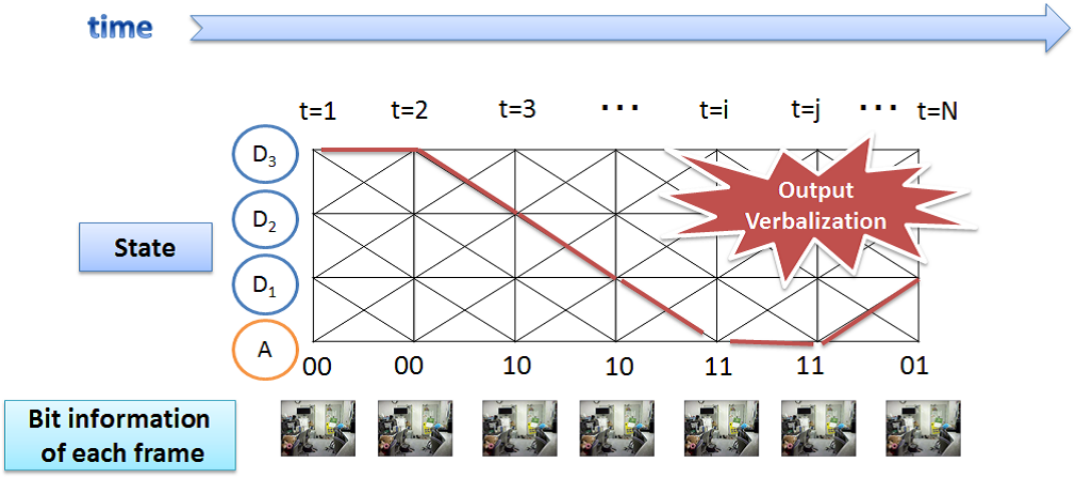

Figure 4.16: Viterbi path pattern without acceleration terminal attached

As the Fig. 4.16 shows, when the path pass through the state Ac, the verbalization is output, and while the path keep pass through Ac the verbalization continue to be output. In the model of objects that acceleration sensor device is not attached with, the is no Ac state. In this case, the verbalization is output when the Viterbi path pass through the state D_1 and keep staying at D_1.

4.3 Pattern (3) : Data Quality Evaluation Framework Applied to Motion Alignment Method GTW

4.3.1 Data Collection Layer

The third pattern of Data Quality Evaluation Framework is applied to a human motion alignment method GTW. At first, we will discuss the Data Collection Layer of Pattern (3), which is different from Pattern (1) and Pattern (2).

In Data Collection Layer of GTW algorithm, I used image data sequence as input data of GTW. GTW is a method to align multi-modal sequences from multiple human performing almost the same motion. The detail of GTW will be explained in section 4.3.3, input data sequence should be matrix.

At fist, I have collected many kinds of image sequence with recording unspeci-

fied number of person who are performing various actions such as open the door, sit on the chair, wipe the desk, open the door of refrigerator, open the door of book shelf, and so on. Next, the image sequence is converted to one matrix. I will explain how to do this conversion with two Figures.4.17 and Fig. 4.18.

As Fig. 4.17 shows, the first step is binarization. I have converted the color image to binary image for reduction of calculation order. The other reason to use binary image is that GTW does not process complex matrix exactly like image with color pixels. The value of each pixel of color image is 0 - 255. I have set threshold as 180. When the value of pixel is under 180, that pixel will 0, and when value of pixel is over 180, that pixel will be 1. I have done this process to all pixels with each image in the sequence.

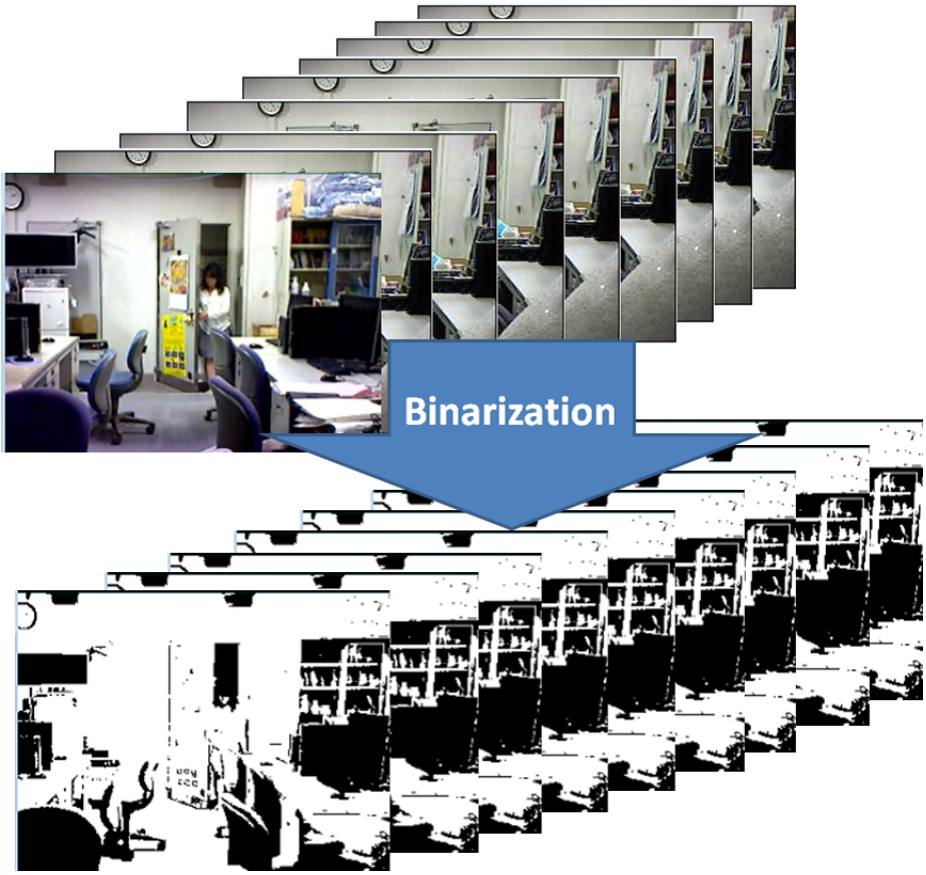

Figure 4.17: Image sequence binarization

The next step is making matrix. As shown in the Fig. 4.18, first image in the

sequence is converted to the leftmost column in the matrix, the second image is converted to second column. The scanning line of every image starts from upper left and ends in the lower right. One image sequence is converted to one matrix.

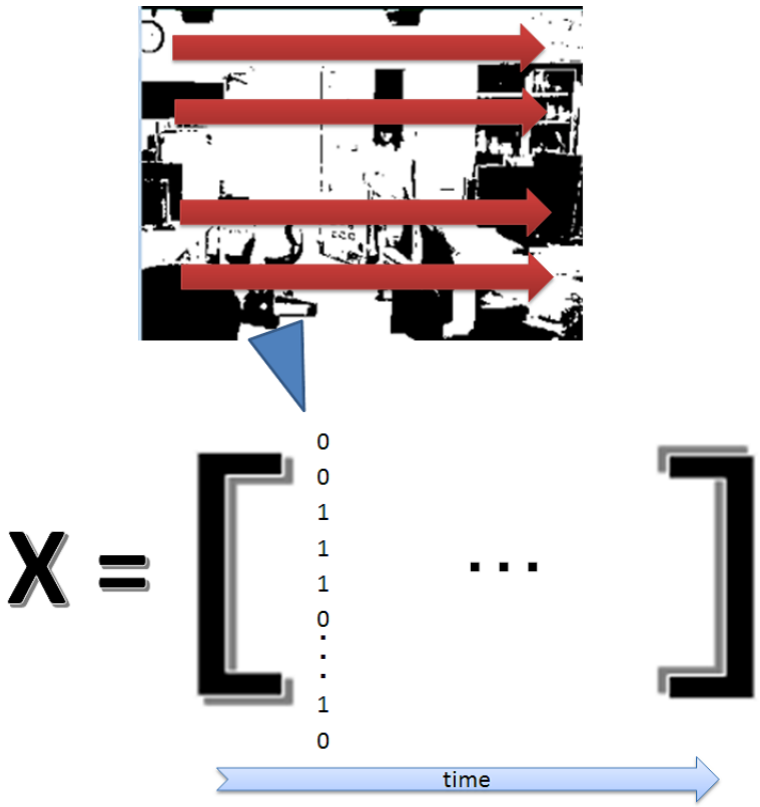

Figure 4.18: Conversion of image data sequence to matrix

I have collected three image sequences per one motion. In the three sequences, different people are doing all most the same motion, but the timing is not the same. The three matrices are given to Data Processing Layer.

4.3.2 Data Processing Layer

For data processing layer of Pattern (3), GTW algorithm have been applied. The formula of GTW is as shown in the Fig. 4.19).

The Generalized Time Warping (GTW) algorithm and its variants for temporally alignment of multi-modal sequences from multiple human subjects performing similar human motion activities are proposed in Feng et al. [43]. I will explain in

detail in this section.

$$J_{gtw}(\{\mathbf{W}_i, \mathbf{V}_i\}) = \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{1}{2}\|\mathbf{V}_i^T\mathbf{X}_i\mathbf{W}_i - \mathbf{V}_j^T\mathbf{X}_j\mathbf{W}_j\|_F^2 + \Big(\sum_{i=1}^{m}\psi(\mathbf{W}_i) + \phi(\mathbf{V}_i)\Big),$$
$$\text{s.t.} \quad \mathbf{W}_i \in \Psi \text{ and } \mathbf{V}_i \in \Phi, \quad \forall i \in \{1:m\},$$

Figure 4.19: GTW formula

Objective function is as follow:

$X_1, X_2, ..., X_m$ is collection of m time series where $X_i = [x_1^i, ..., x_{n_i}^i]$. W_i is a non-linear temporal transformation which is consist of 0, 1. V_i is a lo-dimensional spatial embedding. Then $V_i^T X_i W_i$ is well aligned with the others in the latest-squares sense. $J_{gtw}(W_i, V_i)$ stands for minimized sum of pairwise distances. Please refer to [43] for more details.

I will talking about why GTW algorithm is proposed, and its variants. Recently, accurate alignment of human motions of different style or speed have been challenged in image processing research area. In the paper [43], Feng et al. have challenged to align multi-modal alignment of time series of people performing almost the same activities from different sensors. They proposed GTW which is the extension of DTW (Dynamic Time Warping) which is an old method to align human behavior [43] among two subjects only. Before GTW, many kinds of time warping algorithms have been proposed, for example, CTW (Canonical Time warping) [44] and DMW (Dynamic Manifold Warping) [45]. CTW can temporally

align data of different dimensionality, for example, motion capture, video, and CTW combines DTW with canonical correlation analysis. DMW is extension of CTW, which can incorporates more complex spatial transformations using manifold learning. However, these kinds of time warping algorithm have limitations because they rely on DTW. The limitations are as follows:

(1) Computational complexity is high

(2) Only two sequences can be aligned, not multiple sequences

(3) They rely on dynamic programming to find the optimal path

To overcome these limitations, GTW has been proposed, which can efficiently and flexibly align two or more multi-dimensional time series of different modalities, and has linear complexity because it uses a Gauss-Newton algorithm to optimizes the time warping function efficiently. These approaches differ from existing approaches based on dynamic programming. See details about Objective function in [43].

In the experiments, Feng et al. utilized three types of input video data to evaluate the accuracy of GTW and compared GTW with other variant methods such as pDTW (Procrustes dynamic time warping), pDDTW (procrustes derivative dynamic time warping), and pIMW (procrustes iterative motion warping) for the temporally aligning of multi-modal sequences from multiple subjects performing similar human activities. pDTW [48] is used for shape for alignment and they extended pDTW to align multiple sequences. pDDTW is a technique to do invariant translation of DTW, using derivatives of the original features [47]. pIMW [49] is extension of IMW which time warping and spatial transformation are alternated to align two sequences. Input data in each experiment that were conducted by Feng et al. are as follows:

Experiment1: time series alignment with known ground truth

Experiment2: several video sequences alignment of different people doing similar actions

Experiment3: three different sensor alignment of people doing similar actions

In Experiment1, synthetically generated 3-D spatio-temporal signals is used as input data. As a result, GTW performed the best. pDTW fails since the sequences have been distorted in space. pDDTW also fails since the feature derivatives do not capture the structure of the sequence well. pIMW is more prone to over-fitting. In Experiment2, GTW and its variants have been applied to align video sequences, the silhouette with background subtraction was extracted. Each sequence is two cycles of human walking sequences. As a result, pDTW nor pDDTW could align the video sequences since they lack the ability to solve for correspondence among signals of different nature. pIMW was well for top three components in space. However, it overfits all the dimensions and a biased time warping path was obtained. Only GTW could warp the sequences accurately in both space and time. In Experiment3, even if input data are different types captured from different sensor, and people in one of three sequence performing a little different action from others, GTW could solve the temporal correspondence among three sequences.

There are many research in the area of aligning human motions from sensory data in the context of computer graphics and computer vision. In this Data Quality Evaluation Framework of Pattern (3), I particularly focused on the technique of Time warping which is the technique is to align several sequences. For example, it aligns the sequence of video or frames in which different people are doing the same actions but the timing is not the same.

4.3.3 Information Analysis with GTW

The output of the GTW algorithm is error rate of alignment error ([43] (p.5 Formula (9))). The formula to calculate the alignment error is as follow:

$$err_{alg} = \frac{dist(P_{alg}, P_{tru}) + dist(P_{tru}, P_{alg})}{l_{alg} + l_{tru}} where dist(P_1, P_2) = \sum_{i=1}^{l_1} min_{j=1}^{l_2} ||P_1^{(i)} - P_2^{(j)}||_2$$

The smaller err value means better GTW aliment.

Chapter 5

Data Quality Evaluation Experiment

In this chapter, the data quality evaluation experiment and the result is shown.

First of all, I will discuss what is the data quality mentioned in this research and how data quality change or when data quality change. As described before, to consider data quality of Lifelog Analysis Application is very important because the accuracy of the Lifelog Analysis Application is affected by the data quality. Typically, when talking about Lifelog Analysis Application which collect input data from sensor space, and transmit the collected data to some terminals through Wireless LAN (WLAN) for analysis, there are two factors to determine the data quality.

- Factor 1 : How data collection devices behave
- Factor 2 : How to send collected data to Analysis terminals

Suppose the input is stream data, the data quality change during data collection rely on Factor 1. For example, if some processing error of device happen, some frames of stream data may drop or the quality of each frame may drop. The data quality change in this case can be divided to following two types of data quality deterioration:

- Quality deterioration A : Individual data quality deterioration on the time series

- Quality deterioration B : Data quality degradation when arranging them in the series

The example of Quality deterioration A is image quality change. Considering the quality deterioration of the individual data on the time series, the data deterioration of each image is representative issues. The example Quality deterioration B is dropping frames. The main problem of the time series data may be dropping frames, not only the image frames but also falling out of a part of time series data. For each of the quality deterioration, Bayesian Classifier and HMM were utilized as a typical data processing technique, and clarified how much the two processing method can deal with the data quality deterioration, and finally I have showed the guidelines to handle the quality deterioration.

Suppose the transmission method is Wireless LAN, the data quality change during transmitting the data toward analysis terminals rely on Factor 2. If network packet lose happen because of radio wave interference or collision while stream data transmission, the quality of transmitted data may drop. The data quality change in this case can be divided to following two types of data quality deterioration:

- Quality deterioration C : Data quality deterioration due to the method of data transmission (transmission rate, packet loss rate)

- Quality deterioration D : Data quality deterioration due to the model of data processing at the time of motion analysis

I have utilized Bayesian Classifier and HMM as typical data processing technique and showed the guidelines of how to handle due to the degree of quality

deterioration. I have also utilized GTW model which is different from Bayesian Classifier and HMM.

For evaluate above four kinds of data quality deterioration impact on the system's output, I have utilized Verbalization Application and Human motion alignment algorithm as one of the instance of all the Lifelog Analysis Application. I have evaluated how the data quality deterioration impact appear to the Analysis output in application level. The possible problems, A, B, C, and D described above has been covered with the data quality evaluation experiments in this chapter.

Outline. In section 5.1 of this chapter, the data quality evaluation experiment A, B is discussed. Section 5.2 discuss data quality evaluation experiment C which is wireless communication quality evaluation. In section 5.3, data quality evaluation experiment of human motion alignment algorithm is discussed.

5.1 Quality Deterioration A and B

The data quality evaluation experiment to evaluate the impact of data quality deterioration A and B will be described in this section. In the case of input data quality evaluation experiments, the following two kinds of data qualities are evaluated:

- A : 'Image quality' of video data
  (described in section 5.1.1)
- B : 'Number of frames (the number of data per time)' of video data and acceleration data
  (described in section 5.1.2)

5.1.1 A : Quality Deterioration of Image Quality

Next, the 'image quality' focused on each frame of video data is described. In the evaluation experiments, the following four kinds of filtering for each image frame of video data artificially are executed and the quality is divided into 10 steps, as shown in Fig. 5.1.

- experimentA-1 : filtering process intend for blurry image (smoothing).
- experimentA-2 : filtering process intend for images blurred into length.
- experimentA-3 : filtering process intend for images blurred into side.
- experimentA-4 : filtering process intend for images the resolution deteriorated.

The wider the space (a square) of filter for each frame is, the lower the quality becomes. The image quality has been calculated with the following formula.

$$Q = \frac{1}{p^2} \times 100 \quad (\%)$$

Q:quality of image, p:range of the filtered space (the length of one side of the square)

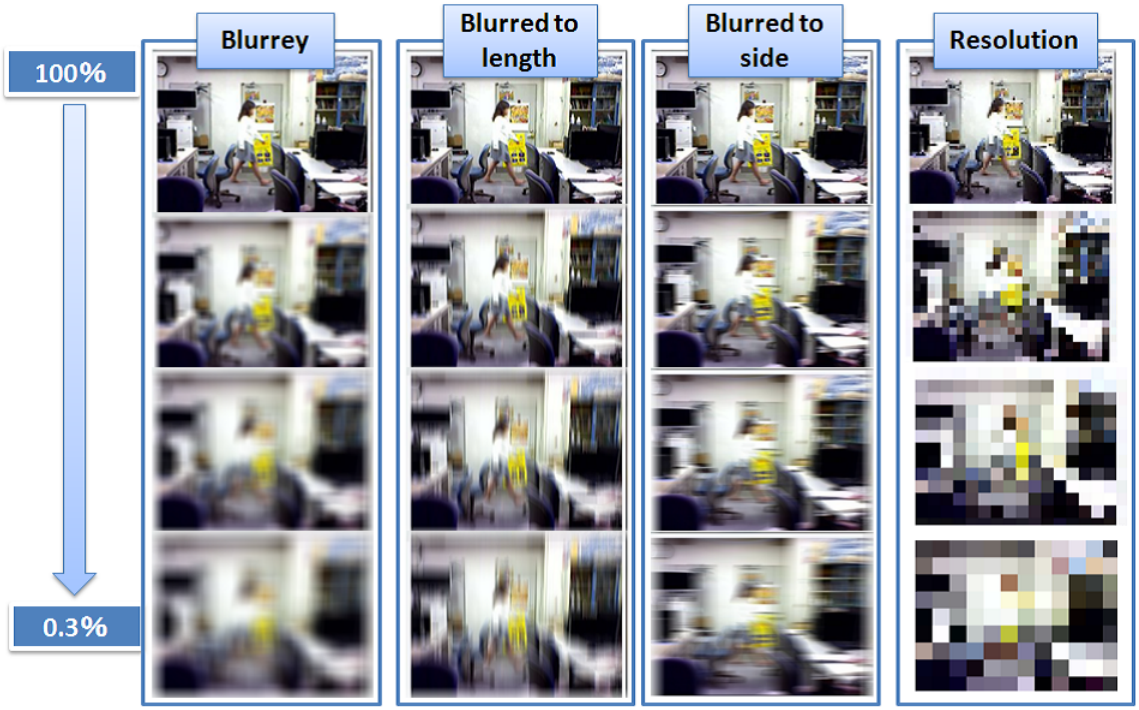

Figure 5.1: Image quality of each frame of video data

5.1.2 B : Quality Deterioration of Obtained Number of Frames

As the quality of 'obtained number of frames (the number of data per time)', the quality of video data and acceleration data, which is two kinds of input data of verbalization application, is divided into 10 steps respectively, as shown in Fig. 5.2.

For the video data, the quality is divided into 10 steps from 10fps (frame per second), the maximum quality (100%), to 1fps, the minimum quality (10%). For the acceleration data which is collected more than 100 times per second by SunSPOT, the quality is also divided into 10 steps from 10 times per second, the maximum quality (100%), as the maximum quality of video data is 10fps, to 1 time per second, the minimum quality (10%). As shown in the right graph of Fig. 5.2, the lower the quality of obtained number of frames of acceleration becomes, the rougher the graph becomes.

I have evaluated the correlation between input data quality difference and the correct answer rate of the verbalization application. The input to verbalization application of evaluation experiments is following three kinds.

- experimentB-1 : Only changing the quality of video data (acceleration data quality is fixed 100%)
- experimentB-2 : Only changing the quality of acceleration data (video data quality is fixed 100%)
- experimentB-3 : Changing the quality of both video data and acceleration data together

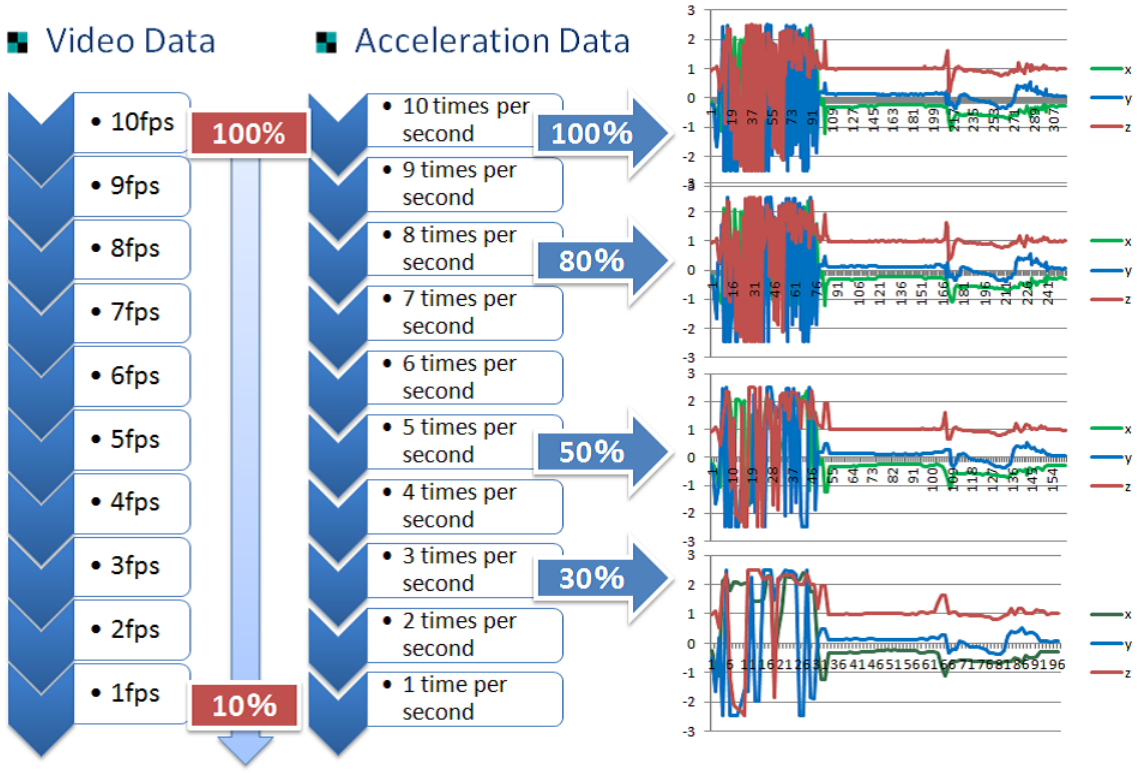

Figure 5.2: Obtained number of frames (the number of data per time) of video data and acceleration data

5.1.3 Calculation Method of Correct Answer Rate

How to calculate the correct answer rate is as follows:
When the quality of both video data and acceleration data is 100%, the highest quality, verbalization is kept to be outputted during the occurrence of human behavior. The correct answer rate of this condition is regarded as 100%, and V_{100} is defined as the number of output in this case. Possible wrong verbalizations caused by the quality deterioration of input data are following three patterns.

- Though verbalization is outputted while human behavior is occuring, extra wrong output is also issued in addition to V_{100}. (V_{extra} is defined as the number of output in this case.)
- Wrong verbalization that is different from the human behavior is outputted. (V_{error} is defined as the number of output in this case.)
- Verbalization is not outputted even though human behavior is occurring.

The number of verbalization output is V_q when the input data quality is $q(0 < q <= 100)$. I have calculated the correct answer rate C with the following formula.

$$C = \frac{V_q - V_{extra} - V_{error}}{V_{100} + V_{extra} + V_{error}} \times 100 \quad (\%)$$

In addition, the minimum data quality required for the verbalization application to output correct answer is called the minimum required quality. This means that verbalization is outputted for all performed activities at least once. This is a low limit quality, which is before the condition that 'Verbalization is not outputted even though human behavior is occurring'.

5.1.4 Experiment Result of Data Quality Deterioration A and B

Fig. 5.4-5.5 show the correlation between the change of the quality (the number of obtained frames/data, image quality) and the correct answer rate, and minimum required quality. The horizontal axis is the rate of the number of obtained frames/data per second and vertical axis is the correct answer rate. In the graph of minimum required data quality, the horizontal axis is the data quality that is changed in each case and vertical axis is the minimum required quality.

The three graphs (Video data, Acceleration data, Video data & Acceleration data) shown in Fig. 5.4 and 5.6 are the results of experiment B-1 − B-3 described in section 5.1.2, respectively. They are the correlation between the rate of the number of obtained frames/data and correct answer rate. The Four graphs (Blurry, Blurred to length, Blurred to side, Resolution) shown in Fig. 5.3 and 5.5 are the results of experiment A-1 − A-4 described in section 5.1.1, respectively. They are the correlation between image quality and correct answer rate.

The results of experiments with experimental data are shown in Fig. 5.4 and Fig. 5.3, and the results of experiments with real data are shown in Fig. 5.6 and Fig. 5.5. The difference between experimental data and real data is described in section 6.3.

The results of both Bayesian Classifier and HMM are shown in all the Fig. 5.4-5.5. In the graph of minimum required quality, when the bar graph is higher, it means that higher quality input data is required for verbalization application to output correct answer. On the other hand, when the bar graph is lower, it means that verbalization application can output correct answer even if lower quality data is inputted.

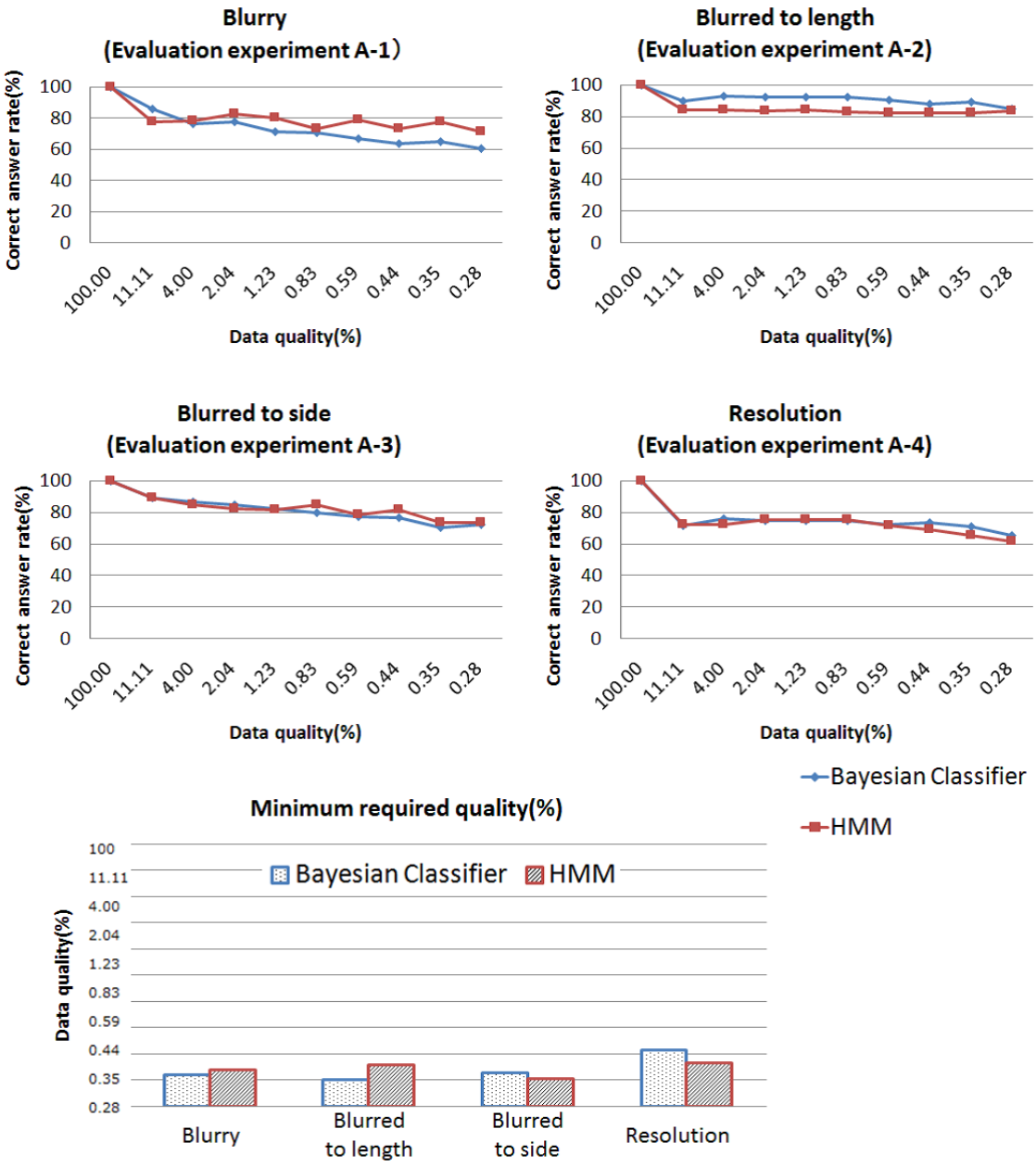

Figure 5.3: Correlation between the change of the image quality and the correct answer rate (experimental data)

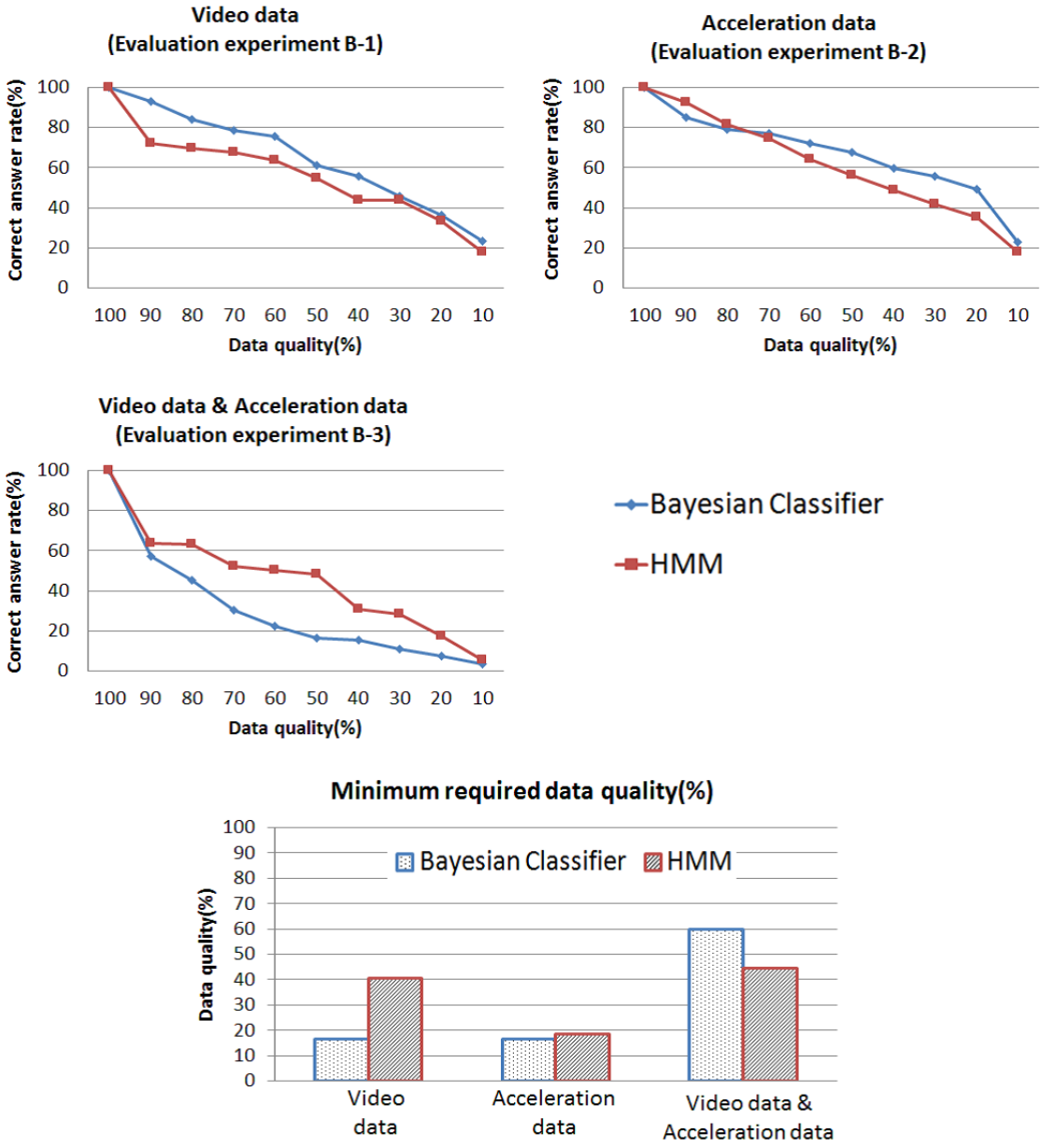

Figure 5.4: Correlation between the rate of the number of obtained frames/data per second and the correct answer rate (experimental data)

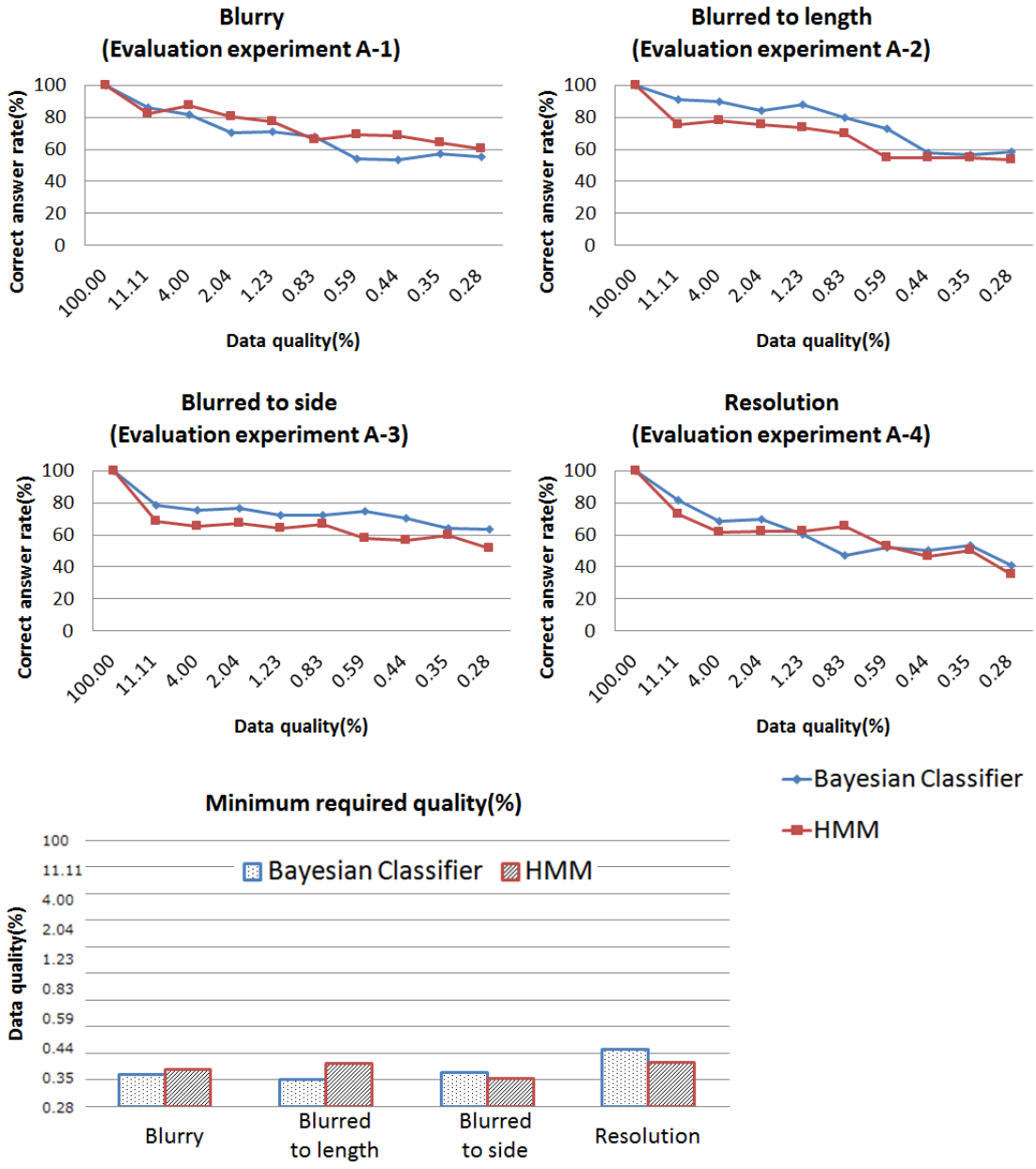

Figure 5.5: Correlation between the change of the image quality and the correct answer rate (real data)

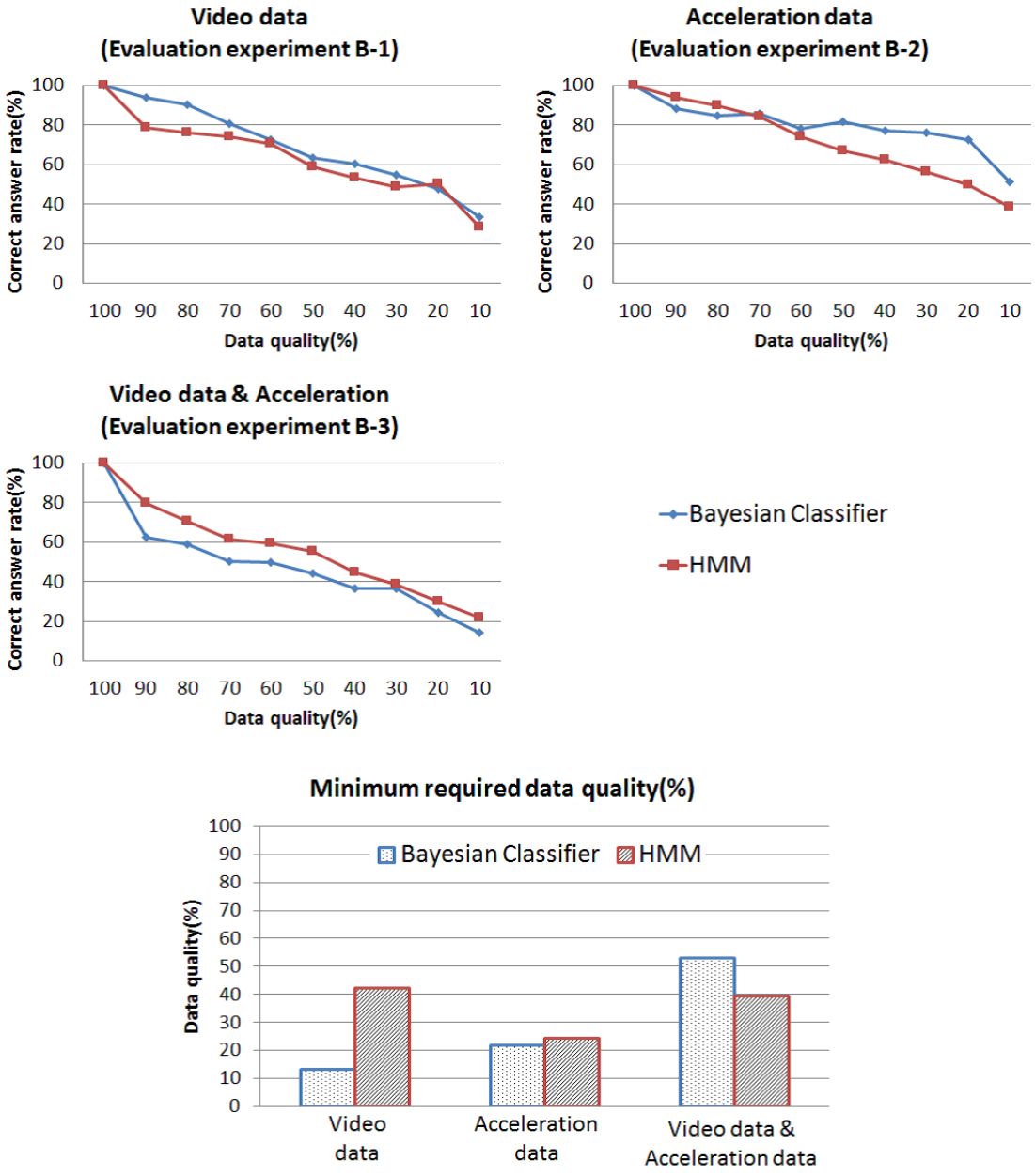

Figure 5.6: Correlation between the rate of the number of obtained frames/data per second and the correct answer rate (real data)

5.1.5 Discussions of Date Quality Evaluation Experiment A and B

As shown in Fig. 5.4 and Fig. 5.6, when the input data quality becomes lower, the correct answer rate decreases. This is an appropriate result. As shown in Fig. 5.4, the deterioration in either only video data or only acceleration data causes correct answer rate to drop to about 20%, while the deterioration in both video data and acceleration data decreases the correct answer rate to nearly 0%. In the case of real data shown in Fig. 5.6 also, though the numerical value differs a little, the feature of the graph is almost the same.

According to this result, the quality deterioration of multiple input data has a lot of influence on the lifelog analysis application. However, if the quality deterioration occurs only in a part of input data, the percentage of correct answers can be kept up to 60% even if the input data quality drops around half.

In terms of comparison between Bayesian Classifier and HMM in the graph of video data, the line of HMM is drawn in lower place at all input data qualities and the bar graph of HMM is higher in the graph of minimum required quality. Thus, to conclude, the process of HMM depends on image frames of video data and is more sensitive to the quality change of video data than that of acceleration data.

In the graph of acceleration data, judging from the height of the bar graph, verbalization application can output correct answer even if the quality of both Bayesian Classifier and HMM drop to about 20%. In the case of real data, about $40 - 50$ % of correct answer is kept all the time. This means that while the features of both cases are almost the same, the acceleration data is less important for the real data. However I should increase the quantity of evaluation data and see whether the graph will be changed or not.

On the other hand, in the graph of video data & acceleration data, the deterioration of input data quality decrease correct answer rate earlier in both Bayesian

Classifier and HMM. If the quality becomes lower in both cases, the case of Bayesian Classifier becomes lower than that of HMM. This result is also observed in both cases of using experimental data and real data.

The graph of all kinds of image quality that are Blurry and Blurred to length and Blurred to side and Resolution, the correct answer rate only decreases to 60 - 80% as shown in Fig. 5.3, and decreases to about 60% as shown in Fig. 5.5. Therefore, to conclude, the deterioration of the rate of the number of obtained frames/data have much influence on the correct answer rate of the verbalization application than deterioration of image quality. As shown above, since almost the same feature graph is acquired in both cases of experimental data and real data, the result should have credibility.

Because the results of experimental data and real data are almost the same, these results are reliable even though it is necessary to increase the quantity of evaluation data in the future.

As the graphs of correlation between quality change of input data and the correct answer rate are shown with quantitative measure, I can confirm the input data quality required for the verbalization application clearly. As shown in the bar graph of minimum required data quality, it is not necessary to use the highest 100% quality input data to output correct answer. In some data like acceleration data, even the very low quality, about 20%, can output correct answer.

5.2 Quality Deterioration C

In this section, Wireless communication quality impact on verbalization application is discussed.

The experimental environment to evaluate date quality deterioration caused by Wireless LAN network quality is shown in Fig. 5.7. In the sensor space, I have settled two network cameras and four SunSPOTs attached to moving objects

including chair, refrigerator, shelf, and door. Five kinds of human activities are verbalized as shown in the right of Fig. 5.7. In the evaluation experiments, a correlation between the correct answer rate of verbalization application and the quality of input data (video data and acceleration data) has been evaluated.

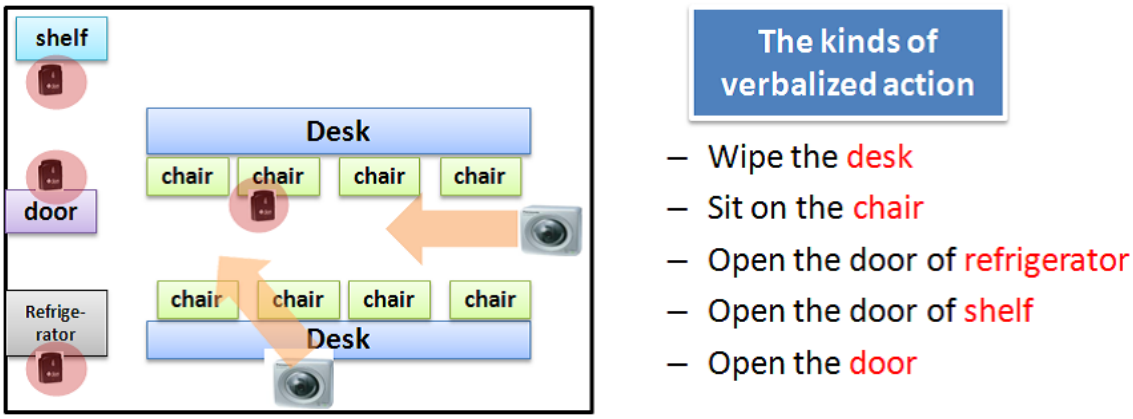

Figure 5.7: Experimental environment and actions to be verbalized

In the experiments, both "experimental data"and "real data"are used. In the case of experimental data, the result of the evaluation is the average of more than one video data in which the same specific activities are intentionally performed. On the other hand, real data has been recorded at the same space for two days, in which people have performed their own activities naturally.

5.2.1 Network Packet Loss Phenomenon

IEEE (Institute of Electrical and Electronics Engineer) 802.11 is a WLAN standard and IEEE 802.11g used in my experiments can communicate about 54Mbps in the 2.4GHz band. The target protocol of IEEE802.11 standard is Medium Access Control (MAC) as data link layer and physical layer. In wireless network, frame transmission is performed in form of broadcast and several terminals can share the bandwidth. Thus, if several terminals send the frames at the same time,

the signal overlaps, frames cannot be sent correctly. When the frames conflict, the frames will be lost and retransmission happens. As the result, the throughput will decrease. To avoid frame collision, CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) is used. This is autonomous distributed control to avoid frame collision. CSMA can avoid frame collision as far as possible by controlling the timing of each terminal sending frames and detecting the usage of Wireless channel before each terminal sends frame. As shown in Fig. 5.8, each terminal can only send frame when a certain idle time is detected. The interval of sending frame is called DIFS (DCF Inter-Frame Space). Frame cannot be sent immediately, terminals have to wait during DIFS time. If there is no signal sensed within DIFS time, terminals judge that there is no signal and start to send their frames. However, sometimes, several terminals detect the channel is idle and send the frames at the same time, and frame collision happens. To avoid this, Backoff control is defined. Backoff is random time based on Contention Window (CW). Terminals have to wait during Backoff time after DIFS idle time. As the backoff time is randomly set, frame collision will decrease than only sensing carriers. If frame collision happens, the range of CW will be double, and it can reduce the probability of re-collision. Thus, if many terminals are sharing one WLAN bandwidth, frame collision will happen. In my experiment, seven times frame collision leads to one time packet loss.

In my paper, "noise"means the noise of radio wave (electric wave) that is delivered from other electromagnetic devices, existing at the space of the experiment. When I use WLAN to transmit packets, not all packets reach to the receiver side by several causes. There are two main causes of the packet loss in WLAN: noise of radio wave and congestion on the transmitted route. I used UDP for the transmission of video data frames. Since UDP is an algorithm whose transmitted data is affected when packet loss happens, different from TCP, the throughput must be

affected by the packet loss because throughput is the number of received packets in a unit time when they are transmitted through WLAN.

As the lost packets are just abandoned and will not reach to the receiver, throughput should decrease in such a case. Flow control in WLAN is transmission rate control. Therefore, the number of video data frames received at the receiver side should be affected by the flow control in WLAN. In WLAN, congestion typically occurs when many senders deliver their packets almost simultaneously. As the noise is not strong in the condition of this experiment, the main reason of losses should be due to congestion.

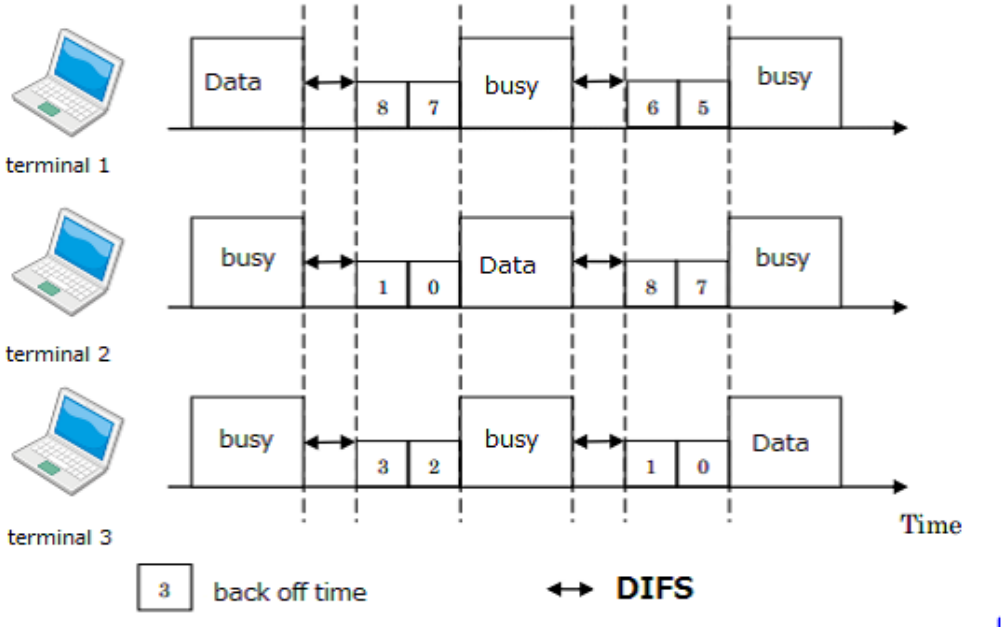

Figure 5.8: Backoff control in WLAN

5.2.2 Packet Loss Caused by Android Terminal

In this section, I will describe how I collected the various multimedia sensor data over the wireless network in Ocha House. For quantitative evaluation, I first carried out the experiments with real terminals. Fig. 5.9 shows the experiment environment of transferring captured data with camera. A certain size of image frame data will be sent from sender side with Ethernet Converter (EC). I also

fixed the transmission rate for quantitative evaluation. In IEEE802.11g, eight kinds of multi-rate values are defined (54, 48, 36, 24, 18, 12, 9, and 6Mbps). In my experiment, I used AP "MZK-MF300N [39]"as AP made by Planex company, which can fix the transmission rate. I used 54, 36, 18, and 6 Mbps, and auto transmission rate as the representative value.

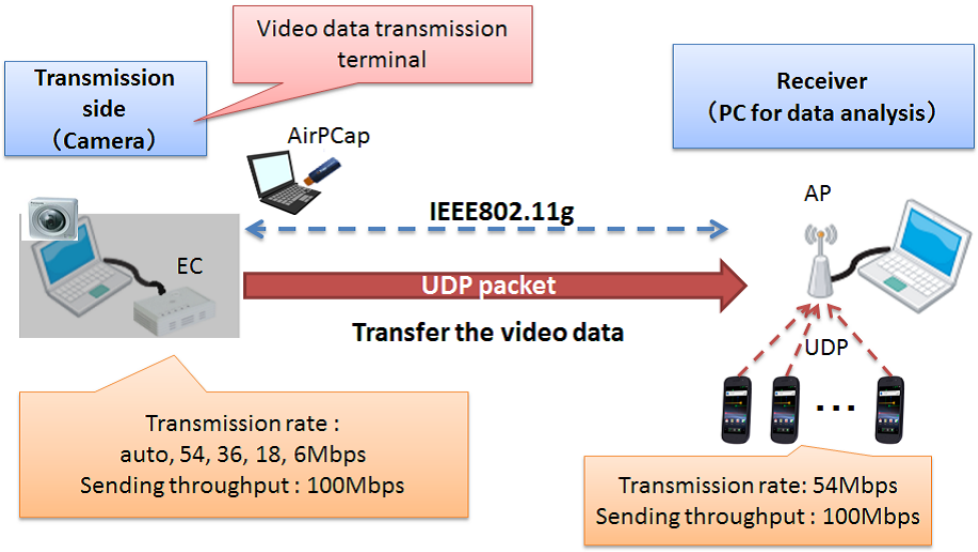

Figure 5.9: Experiment environment of data transfer

As for the receiver side, several Android terminals are communicating with AP as background terminals to interfere with the main stream, video data transmission. I used Android terminal Nexus S [41] (NS) as background terminals, the number of background terminal NS is 0 - 5. The transmission rate of NS is fixed to 54Mbps and they are communicating with AP using Iperf [40] which is a software to send UDP packets to another terminal. The main terminal transmitting video data and 5 NS compete for bandwidth. As there are several background terminals communicating with AP, the packet sent from main terminal (video stream) will conflict with the packet sent from background terminals, and more back ground terminals there are, less packets from main terminal can be reached to the receiver. When the transmission rate of main terminal is lower, the throughput should be

lower as main terminal has less chance to send packet. I have analyzed in detail what happens at main terminal when its transmitted packets conflict with those from background terminals.

First, I have estimated the ability of EC used in the experiment. The EC is connected to PC at the sender side with wired LAN, and the transmission throughput of PC at the sender side is 100 Mbps. Thus the packets that sent out from sender PC are converted to WLAN packets, and sent out from EC with five pattern of transmission rate, 54, 36, 18, 6 Mbps and auto. This means some packets might be lost at EC as the wired LAN speed is much higher than that of WLAN. The buffer overflow on EC with various transmission rate is shown in Fig. 5.10. This graph shows the packet loss rate at EC, which is calculated from the used sequence number captured with AirPCap [42]. In other words, the rate of lost sequence number after EC. When the transmission rate is 6 Mbps, about 6% of packets are lost at EC, and the loss rate decreases as the transmission rate increases. As shown in the graph, the packet loss rate at EC is not so significant.

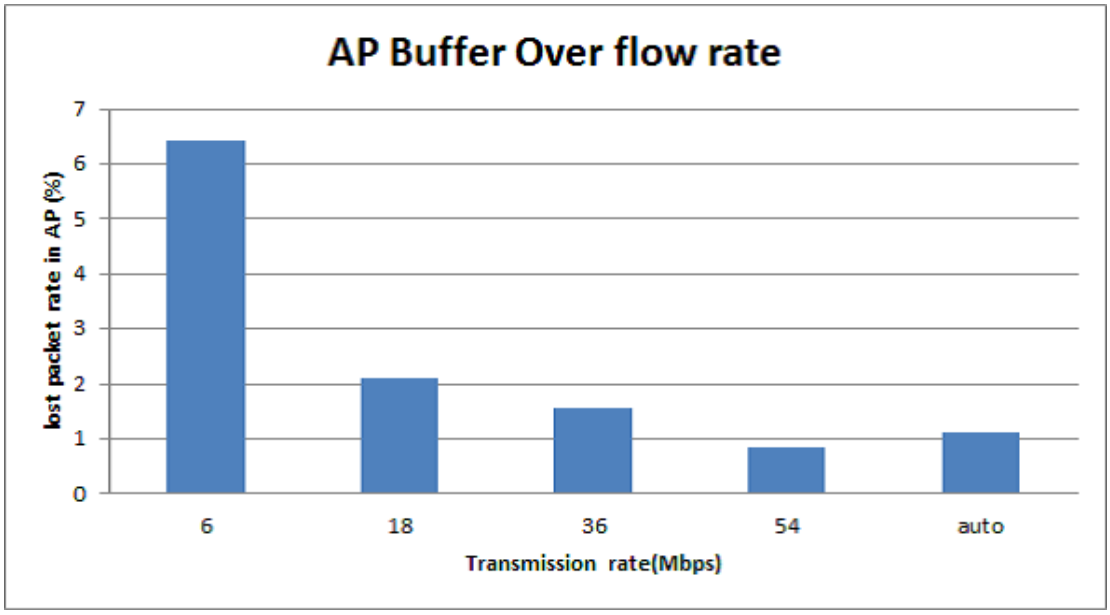

Figure 5.10: Buffer overflow on EC

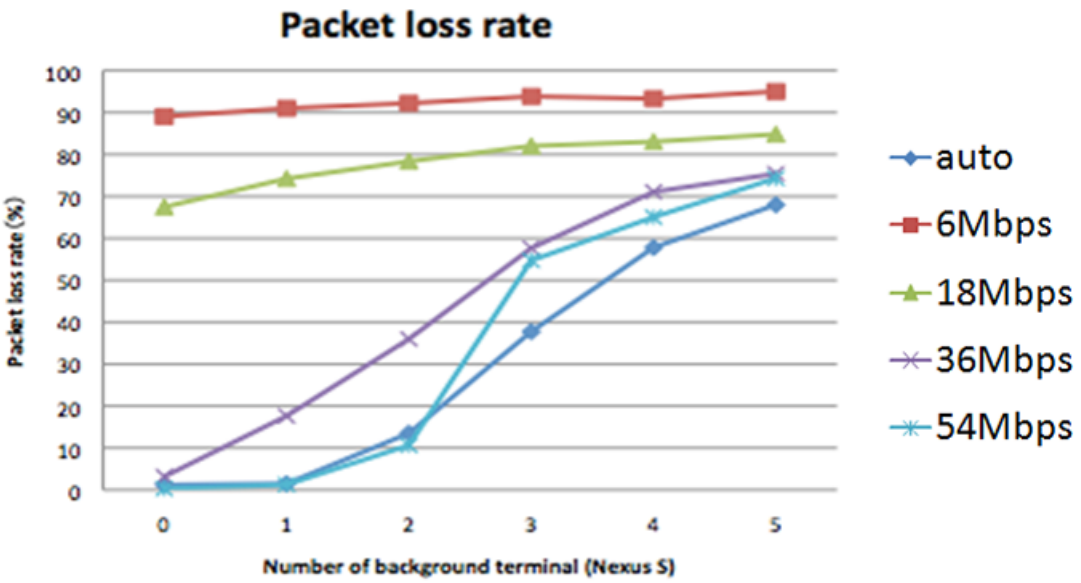

Figure 5.11: Result of data transmission interference experiment

I evaluated how much quality of video data can reach the receiver. The result is shown in Fig. 5.11. This graph shows the result of each transmission rate. Horizontal axis is the number of background terminals (NS) interfering with the main communication, vertical axis is the packet loss rate of main terminal which is transmitting the main video data. For example, when the transmission rate is 36 Mbps, the main video data will suffer 20% loss with one background terminal (NS), 40% loss with two NS, 60% loss with three NS, about 80% loss with five NS. According to the graph, when the transmission rate is lower than 18Mbps, not all packets reach the receiver even when no background terminal exists. The transmission rate is too low to carry all the packets yielded at the sender PC in this case. When the transmission rate is more than 36Mbps, almost all packets can reach the receiver if no background terminal exists. However, as the number of background terminal increases, the packet loss rate increases dramatically, which means a conflict occurs between the transmitted packets from main terminal and those from background terminals. The examples of packet loss frame data are shown in Fig. 5.14. In each frame, people are doing the action, “opening the

door"and three sequences are selected as the input data. As the packet loss rate increases, the pixels will drop because the packet could not reach to the receiver.

The characteristic of Wireless LAN throughput is shown in the Fig. 5.12. This graph shows the correlation between average throughput of background terminal Android terminals (NS : Nexus S), total throughput of that, the throughput of the main terminal transmitting the video data, and MAC frame retransmission ratio.

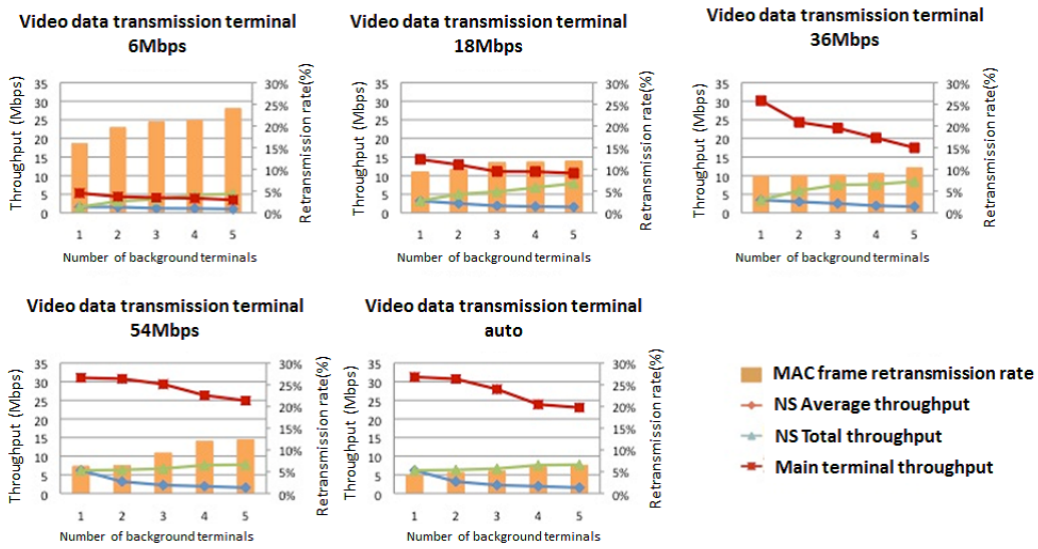

Figure 5.12: Result of throughput and packet loss rate affected by Android terminals

For the graph of the 6Mbps transmission rate of video data transmission terminal, since the original throughput can not exceed 6Mbps, both the throughput of main terminal and the throughput of the background terminals are lower. At the graphs of above 18 Mbps, a large difference in throughput was observed. Though the main video data transmission terminal obtained the throughput close to the transmission rate setting, the throughput of background terminals of which the transmission rate is fixed to 54 Mbps and communicating with the immediate vicinity of the AP Nevertheless was only 5 - 10 Mbps.

Wireless standards employed in this experiment is IEEE802.11g, and in IEEE802.11g, the transmission opportunity is allocated evenly for all terminals in communication

with the AP. In spite of that, the apparent difference between throughput of background terminals and video data transmission main terminals was observed. The cause of the obvious throughput difference between the main video data transmission terminal and the background terminals NS is a power saving of background terminal Android. That is, the main terminal transmit the packets aggressively when transmission opportunity is given, but the background terminals does not transmit the packet every time when transmission opportunity is given.

Next, As for the MAC frame retransmission rate, graph of 6Mbps was the highest even though the 6Mbps transmission rate is stable speed. The cause of this is also the power saving of the background terminals. When the transmission rate of main terminal is a 54 Mbps, the EC sends the packet at a faster rate and the transmission opportunity interval of the background terminals is narrow. In this case, since the background terminal it is not given sufficient rest time, often do not send packets even transmission opportunity of itself comes. As a result, collisions difficulty occurs between the main terminal and the background terminals near the AP, thus, MAC frame retransmission rate of the main video data transmission terminal is small.

On the other hand, when the transmission rate of the main video data transmission terminal is 6Mbps, the EC sends the packet at a sufficiently slow rate. Then the background terminals have a long interval of transmission opportunity, and it is possible to send packets almost every time when the order of its transmission opportunity has come. As a result, on the receiving side AP, the probability of collision occurrence of the packet of the main terminal and the background terminals increases, thus, MAC frame retransmission ratio is higher.

In addition, depending on the number increase of background terminals, the total throughput of background terminals is reduced because of packet collision of each other. In a situation where a plurality of background terminals are commu-

nicating with AP while competing for bandwidth, if the main terminal interrupts the data transmission at a constant transmission rate, the main terminal can not increase the bandwidth when the transmission rate of itself is low. On the other hand, if the transmission rate is high, the main terminal can increase the bandwidth because the interference between background terminals themselves occur.

5.2.3 Experiment Result and Discussion of Wireless LAN Impact on Verbalization Application

I have shown the correlation between the packet loss rate of Wireless LAN communication quality and correct answer rate of Verbalization application. I have evaluated whether the percentage of correct answers of verbalization application is subjected to impact caused by the wireless LAN quality deterioration quantitatively when the packet lost video data was inputted to the application.

In either graph, comparing Bayesian Classifier and HMM, the correct answer rate of HMM is lower. This is caused by difference in characteristics of the data processing as Bayesian Classifier process the stream data per each frames, on the other hand, HMM process the stream data with time series. When processing method is HMM, current frame is dependent on the previous and suspend frames, thus, it is sensitive to the quality changes due to packet loss of previous frames.

In addition, as a feature of correct answer rate transition caused by packet loss, Bayesian Classifier decrease smoothly, but taking about HMM, when background terminals are two to four, the correct answer rate begin to decrease and when the number of background terminals are four to five, the correct answer rate was almost 0 %.

As for the Bayesian Classifier, when the transmission rate is more than 36 Mbps, all the correct answer rate was over 60% even though the background terminals are five. I found that HMM is sensitively affected by packet loss compared

to Bayesian Classifier.

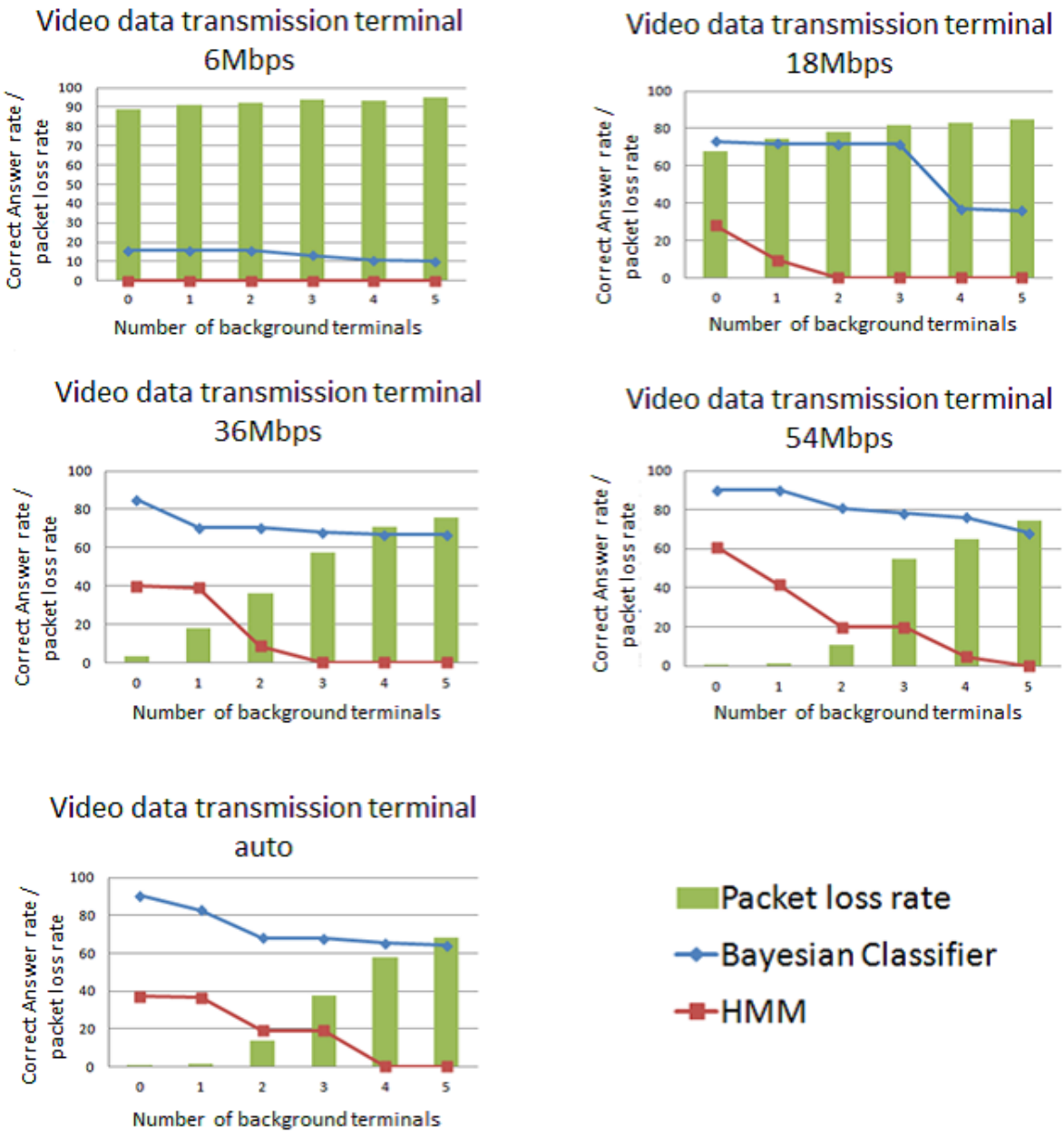

Figure 5.13: Wireless quality impact on Verbalization Application

5.3 Data Deterioration D

As shown in [44], CTW (Canonical Time Warping) combines DTW (Dynamic Time Warping) with canonical correlation analysis to align data of different dimensionality temporally. Existing methods using CTW and DTW have following

limitations: (1) computational complexity is quadratic in space and time, (2) only solved the problem of aligning two sequences, therefore it is unclear how to extend it to the alignment of multiple sequences, (3) rely on dynamic programming to find the optimal path, however it is unclear how to constrain the temporal warping adaptively. To overcome these limitations, GTW that allows an efficient and flexible alignment among two or more multi-dimensional time series of different modalities have been proposed [43]. The deterioration of the captured data quality through WLAN transmission affecting the accuracy of the GTW system has not been studied before. Hence, this is the first work to investigate the impact of the WLAN in GTW system.

In Experiment1, synthetically generated 3-D spatio-temporal signals is used as input data. As a result, GTW performed the best. pDTW fails since the sequences have been distorted in space. pDDTW also fails since the feature derivatives do not capture the structure of the sequence well. pIMW is more prone to over-fitting. In Experiment2, GTW and its variants have been applied to align video sequences, the silhouette with background subtraction was extracted. Each sequence is two cycles of human walking sequences. As a result, pDTW nor pDDTW could align the video sequences since they lack the ability to solve for correspondence among signals of different nature. pIMW was well for top three components in space. However, it overfits all the dimensions and a biased time warping path was obtained. Only GTW could warp the sequences accurately in both space and time. In Experiment3, even if input data are different types captured from different sensor, and people in one of three sequence performing a little different action from others, GTW could solve the temporal correspondence among three sequences.

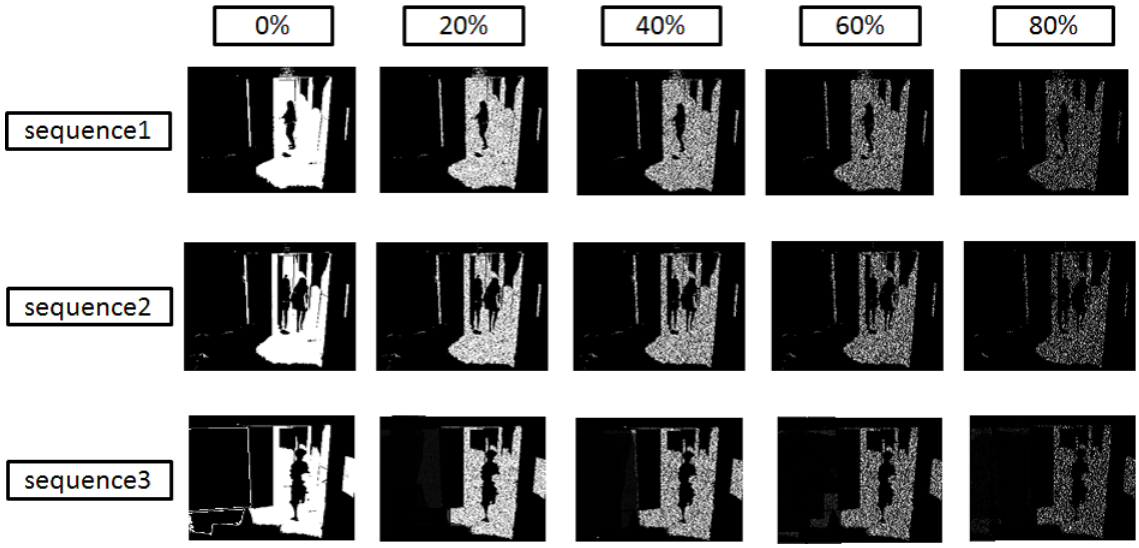

Figure 5.14: Examples of packet loss frames

5.3.1 Experimental System of Collecting Data

In this section, I will describe how I collected the various multimedia sensor data over the wireless network in Ocha House. For quantitative evaluation, I first carried out the experiments with real terminals. Fig. 5.9 shows the experiment environment of transferring captured data with camera. A certain size of image frame data will be sent from sender side with Ethernet Converter (EC). I also fixed the transmission rate for quantitative evaluation. In IEEE802.11g, eight kinds of multi-rate values are defined (54, 48, 36, 24, 18, 12, 9, and 6Mbps). In my experiment, I used AP "MZK-MF300N [39]"as AP made by Planex company, which can fix the transmission rate. I used 54, 36, 18, and 6 Mbps, and auto transmission rate as the representative value.

As for the receiver side, several Android terminals are communicating with AP as background terminals to interfere with the main stream, video data transmission. I used Android terminal Nexus S [41] (NS) as background terminals, the number of background terminal NS is 0 - 5. The transmission rate of NS is fixed to 54Mbps and they are communicating with AP using Iperf [40] which is a software

to send UDP packets to another terminal. The main terminal transmitting video data and 5 NS compete for bandwidth. As there are several background terminals communicating with AP, the packet sent from main terminal (video stream) will conflict with the packet sent from background terminals, and more back ground terminals there are, less packets from main terminal can be reached to the receiver. When the transmission rate of main terminal is lower, the throughput should be lower as main terminal has less chance to send packet. I have analyzed in detail what happens at main terminal when its transmitted packets conflict with those from background terminals.

First, I have estimated the ability of EC used in the experiment. The EC is connected to PC at the sender side with wired LAN, and the transmission throughput of PC at the sender side is 100 Mbps. Thus the packets that sent out from sender PC are converted to WLAN packets, and sent out from EC with five pattern of transmission rate, 54, 36, 18, 6 Mbps and auto. This means some packets might be lost at EC as the wired LAN speed is much higher than that of WLAN. The buffer overflow on EC with various transmission rate is shown in Fig. 5.10. This graph shows the packet loss rate at EC, which is calculated from the used sequence number captured with AirPCap [42]. In other words, the rate of lost sequence number after EC. When the transmission rate is 6 Mbps, about 6% of packets are lost at EC, and the loss rate decreases as the transmission rate increases. As shown in the graph, the packet loss rate at EC is not so significant.

I evaluated how much quality of video data can reach the receiver. The result is shown in Fig. 5.11. This graph shows the result of each transmission rate. Horizontal axis is the number of background terminals (NS) interfering with the main communication, vertical axis is the packet loss rate of main terminal which is transmitting the main video data. For example, when the transmission rate is 36 Mbps, the main video data will suffer 20% loss with one background terminal

(NS), 40% loss with two NS, 60% loss with three NS, about 80% loss with five NS. According to the graph, when the transmission rate is lower than 18Mbps, not all packets reach the receiver even when no background terminal exists. The transmission rate is too low to carry all the packets yielded at the sender PC in this case. When the transmission rate is more than 36Mbps, almost all packets can reach the receiver if no background terminal exists. However, as the number of background terminal increases, the packet loss rate increases dramatically, which means a conflict occurs between the transmitted packets from main terminal and those from background terminals. The examples of packet loss frame data are shown in Fig. 5.14. In each frame, people are doing the action, "opening the door"and three sequences are selected as the input data. As the packet loss rate increases, the pixels will drop because the packet could not reach to the receiver. I used these packet loss data to input to GTW and its variants and found that GTW is good for aligning multiple sequences, but sensitive to packet loss data.

5.3.2 Experimental Result Discussion

I discuss the result of my experiments in this section. As for the experiment method, I applied the project source codes used in [50] which were developed by Feng et al. Input data are captured frames in the Ocha House. I applied pDTW, pDDTW, pIMW, GTW algorithm respectively. For each experiment, I selected three sequences of different people doing the same action. The results are shown in Fig. 5.15, 5.16, and 5.17. For each result, five patterns of packet loss rate, 0%, 20%, 40%, 60%, and 80% are given. The graphs are error rate of alignment. In the experiments, I have applied three types of captured input video data to each algorithm: (1) extremely long and short sequences with large size of image matrix, (2) data are smoothed, moderately aligned to almost the same length of sequences with small size of image matrix, (3) all backgrounds of

small image matrix data are removed. I will compare the GTW accuracy with other algorithms, pDTW, pDDTW, pIMW. I found that GTW cannot solve the multi-modal temporal alignment problem efficiently in the situations whereby the transmitted data are distorted due to packet loss phenomenon caused by degraded quality in wireless network transmission. Sometimes, pDDTW and pIMW can align the sequences better than GTW.

Big Image Matrix - Extremely Long and Short Sequences

Big matrix means that many pixels are selected per each frame. Long sequence is composed of many frames that people take longer period of time to do one action, for example, about 50 frames. Short sequences is composed of a few frames that people do one action quickly, for example, about 5 frames. For this experiment, three sequences $X_1, X_2, and X_3$, which is three videos of people opening the door. The size and length of each matrix X_i is as follows. $X_1 : 400 \times 50, X_2 : 400 \times 18, X_3 : 400 \times 9$. Note that space 400 of matrix is big enough, and length of three sequences are varied as 50, 18, and 9. As the Fig. 5.15 shows, the alignment error rate is the lowest when the packet loss rate of input frame is 0%, (which is the perfect data frame without packet loss), but the alignment error rises dramatically as the packet loss rate increases and the error rate when the packet loss rate is 80% is about three times as that of perfect data. GTW alignment error rate is lower than other methods such as pDTW, pDDTW, and pIMW when packet loss rate is low, but when the packet loss rate is 80%, GTW alignment error rate is almost the same as other methods. Especially when the packet loss is 40%, pDDTW error rate is the lowest. Thus, GTW cannot tolerate packet loss data and even be the highest alignment error rate in this experiment.

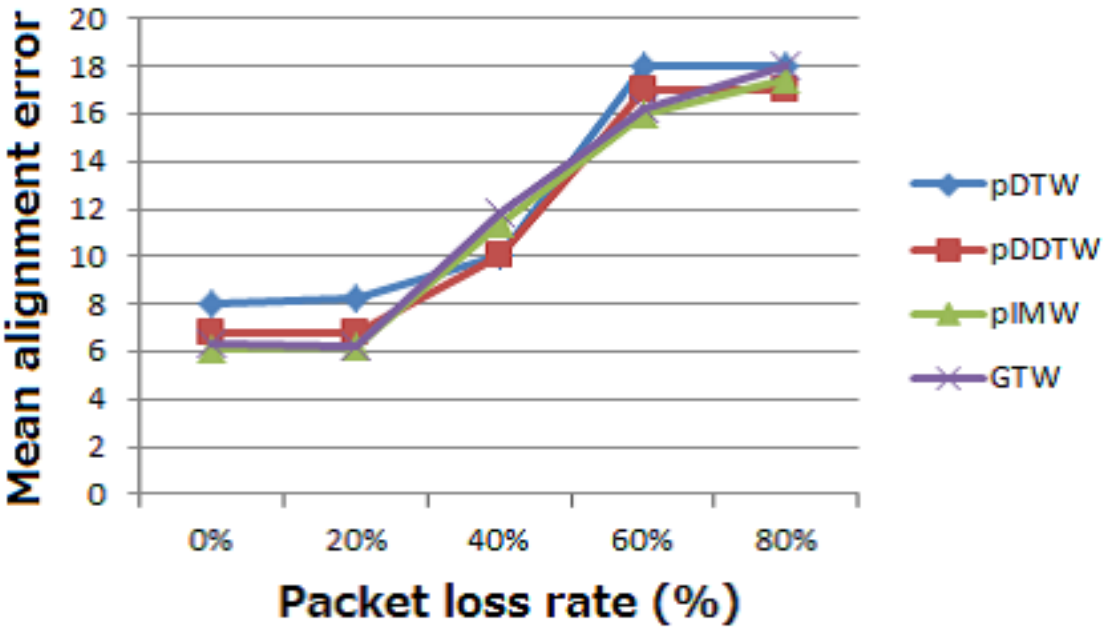

Figure 5.15: Big matrix, extremely long and short sequences

Small Image Matrix — Smoothed with Almost Same Length of Sequences

In response to the result of Fig. 5.15, I reduced the size of matrix from 400 to 12, because the matrix space was too large. A few pixels representative one frame in small matrix. I also aligned the length of sequences to be almost the same and smoothed each frame with a function defined in OpenCV (Open Computer Vision)library [35]. The size and length of each matrix X_i in this experiment is as follows. $X_1 : 12 \times 50, X_2 : 12 \times 48, X_3 : 12 \times 50$. Note that the space of matrix is small enough (12) and the length is long enough to align in this experiment. As shown in the Fig. 5.16, GTW and other time warping alignment error rate are lower when the input frame has no packet loss (0%), but when the packet loss rate is 80%, GTW error rate is the worst than any other time warping algorithm. pIMW could align the sequences the best when the packet loss rate is 80%.

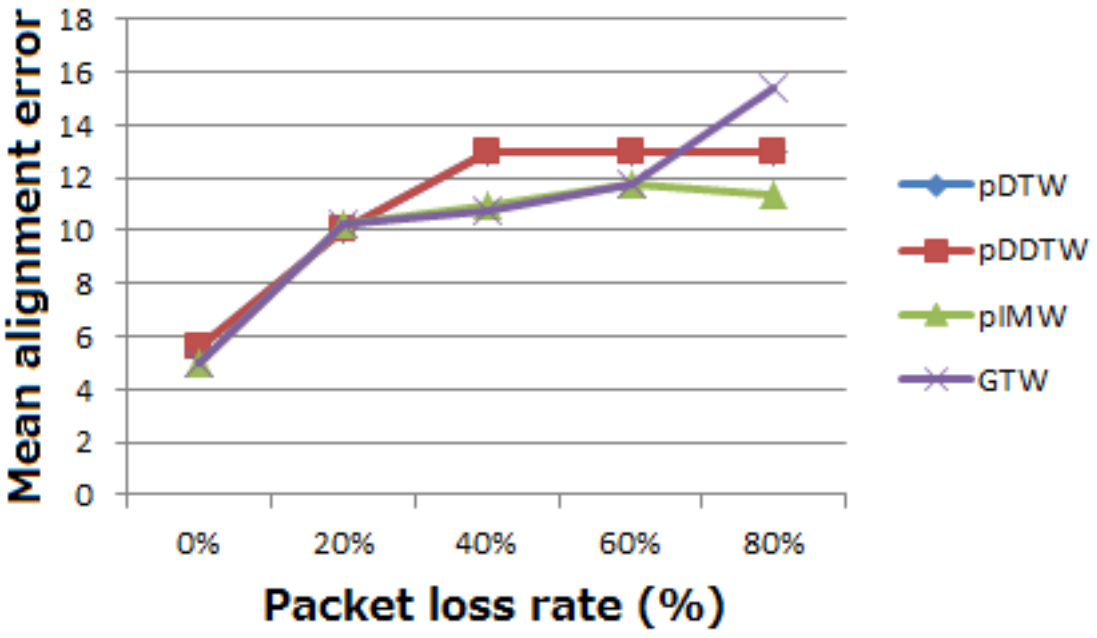

Figure 5.16: Small matrix, smooth frames, almost the same length of sequences

Small Image Matrix — Without Background

Lastly, I removed the background of all frames to align the sequences better. Each frame only has the information of people doing motion activity without background in this experiment. The size and length of each matrix are the same as the experiment in Fig. 5.16. The result is shown in Fig. 5.17. Even though the whole alignment error rate became lower and alignment error rate didn't rises so much as the packet loss rate increases, alignment error rate of GTW was the lowest only when the packet loss rate is 0% (which is the perfect data without packet loss). When the packet loss rate is higher than 20%, GTW alignment error rate become the same as its variants (pDTW, pDDTW, and pIMW).

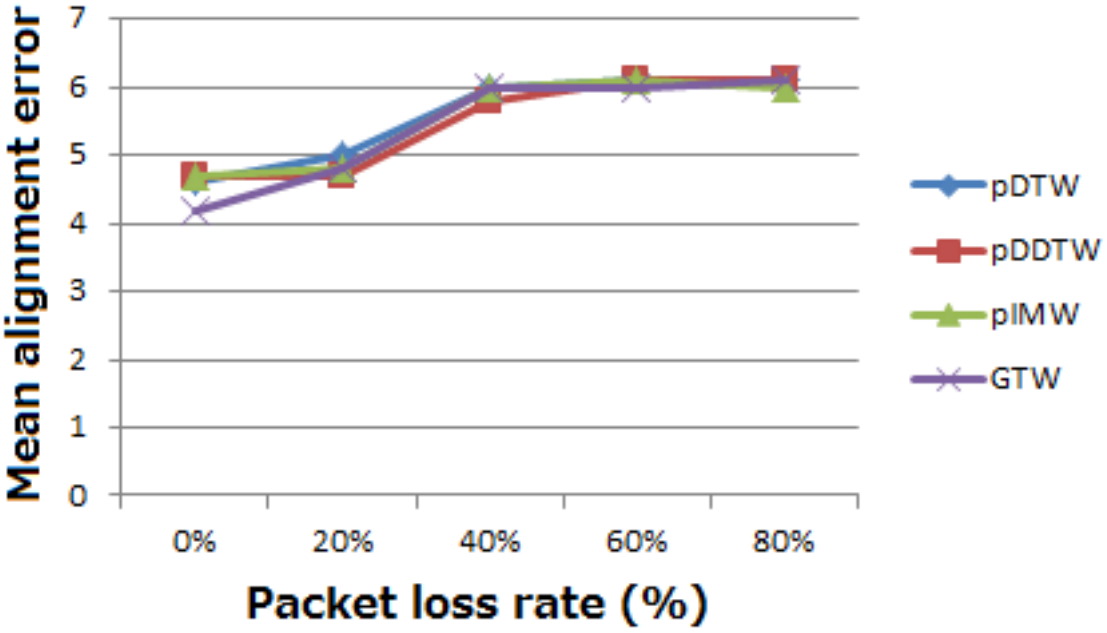

Figure 5.17: Without background

In my Ocha House, I have collected the video data frames of multiple human subjects with various similar human activities which are transmitted real time over the WLAN network to a processing and analysis server running GTW and its variants (pDTW, pDDTW, pIMW) for temporally aligning multi-modal sequences from multiple human subjects performing similar human activities. The captured video data frames are affected by the natural WLAN packet loss phenomenon due to the wireless network flow control and congestion protocol management in an end-to-end wireless networked environment. I discover that the alignment error rate will rise dramatically and the accuracy of human activity recognition deteriorated as the wireless network packet loss rate increases. When the wireless network packet loss rate increases to 80%, the alignment error rate of GTW is almost the same as its other time warping variants — in some cases, pDDTW and pIMW can align the sequences better than GTW. Thus, GTW is very sensitive to the quality of the captured video data frames and cannot solve the multi-modal

temporal alignment problem efficiently in such situations.

I suggest looking into the problems and weaknesses of GTW characteristics in network packet loss environment as the intuition to help GTW tolerate network packet loss. The weaknesses of GTW in the network packet loss environment are as follows: (1) It cannot align small spaces and shorter length matrices. (2) When aligning bigger matrices, some pixels must be selected as representative values. So, GTW should be able to align longer and bigger matrix features better with appropriate selection representative value. I believe that adjustment of such kind of features to improve captured data quality can help GTW to tolerate network data loss environment.

5.4 Discussion

All the Data Quality Evaluation result is summarized in Fig. 5.18. Which type of processing method is suitable with the data quality deterioration type caused by various factor is evaluated. The "○ ', '× ', and '△"is comparison result of Data Processing Method type. When the Application answer rate kept over 80 % at any input data quality, the suitability is The "○'', when the Application answer rate kept under 20 % at any input data quality, the suitability is "×", if there is not so much difference between Data Processing Method, the suitability is The " △".

As mentioned before, Bayesian Classifier is a representative of some methods which process the stream data per each frame, and HMM is a representative of some methods which process the stream data per group of some frames.

The system output result is not affected in the situation that data collection device has some error and individual data quality deterioration occurs. When the data collection device has some error and some frames drop, the impact on output is different due to which method to use. If Video frames drop, the method process

per each frames can cover the deterioration better than the method process the data per time series. But if both the video frames and acceleration frames drop, the method to process the stream data per time series is better. It can be said that when some frame in the stream data loose, stream data should be processed per each frames if single kind of data frame loose, on the other hand, stream data should be processed per time series if multiple kind of data frame loose.

In the case of data transmission error, when packet loss happen on verbalization application, the method to process the data per time series is strong to over this type of data quality deterioration. But the application output is sensitively affected by the very slower speed of transmission rate. From this result, it can be said that when the wireless LAN quality is bad, data should be processed per each frame. The method to process the data per time series is affected by this type of data quality deterioration badly.

When packet loss happen on human motion alignment algorithm, the method which process the stream data per time series can align small matrix data and the length of processed data should be almost the same.

The detailed discussion of each data quality deterioration is written in section 5.4.1 - 5.4.4

5.4.1 Discussion A : Application Correct Answer Rate Caused by Image Quality Deterioration

The correct answer rate of application when each frame is deteriorated have been evaluated assuming a case where only the low quality of image with lower resolution can be obtained because of poor performance of camera sensors. In this case, quality deterioration of individual data had less impact on the output of the application. In the case of all of four image quality deterioration, the percentage of correct answers in the application kept more than 60% even if the

data input quality is the lowest. Application correct answer rate of vertical blurred and horizontal blurred image has kept more than 80% even if the input data is the lowest. This is because of that all the frame image quality deterioration type is the same, so the accuracy of contour extraction of each frame does not fall, and the center of gravity of the "people "can be correctly determined, as the result, the application is capable of maintaining high percentage of correct answers. Note that the contour extraction accuracy is kept only in the case where all the individual frames deteriorate with the same type, not in the case where the type of image quality deterioration is random.

The result difference between two types of data processing method was infinitesimal in both index, the correlation between input data quality and application correct answer rate, and the minimum required input data quality. As for the minimum required image quality, all of four types of image quality was as low as 10% - 20% .

It can be said that it is not necessary to set up wastefully high image quality in the system such as to extract contour difference of each frame, the application might be able to satisfy the requirements even if half image quality of data is utilized.

5.4.2 Discussion B : Application Correct Answer Rate Caused by Frame Drop

The percentage of correct answers of the application in the case of quality deterioration at the time of frame connection was evaluated in this experiment assuming frame drop caused by temporary error of sensor terminals for example buffer overflow. Compared with the case in section 5.4.1, the frame dropping has a larger impact on the percentage of correct answer rate of the application. The reason is that if exactly the moment of the frame when people are doing some kind

of action drops, the node information can not be passed to information analysis layer, and as the result, the number of times of verbalization output is reduced.

In the case in section 5.4.1, the changes in the percentage of correct answer was small because all the frame while people are doing some actions ware obtained without any frame drop. In the case of frame drop, the correct answer rate is clearly reduced according to the reduction of number of acquisition frames. Since the number of acquisition frames are changed, effects experienced by dropped frames are different between two approach of the data process (process per frame and process per frame group). Like the experiment B-3, in the case where both video data and acceleration data dropped frames, the correct answer rate degraded quickly when the data is processed per each frame, on the other hand, the correct answer rate could keep over 40% until the input data quality is as low as 50 % . The reason is that in the processing of each frame, the verbalization judgment would have been reset in the next frame after the moment of the frames while people's action is happening drop, on the contrary, in the processing of frame group, the distance of the center of gravity and the objects is held before the frame drop, and the judgment is resumed from the frame after frame dropping, and as the result, it is likely to lead to verbalization output.

When considering the effects of frame drop on two types of data processing method, as for the data processing method per each frame, even if an important frame is dropped, it can be covered by other frame before and after the dropped frame, but when the reduction rate of the frames necessary for data process is high, correct answer rate will be reduced. As for the data processing method per frame group, the correct answer rate is affected as soon as an important frame drops, but even with the fewer number of frames required for the processing gives less impact on the correct answer rate.

5.4.3 Discussion C : Application Correct Answer Rate Caused by Packet Loss

The wireless LAN communication quality change cased by multi-rate multiplexing has been evaluated in this experiment and the impact on the application correct answer rate affected by that data was experimented. The difference was obvious between the processing method per each frame and per frame group. When the data transmission rate is auto, the correct answer rate difference was as high as 60%.

When the quality of the input data is degraded, the information of an important frame while people's action is happening can not be obtained, and averagely less information from each frame can be obtained because the all the image quality is degraded evenly. As discussed in section 5.4.2, the data processing method per frame group is weak in this case. The data processing method per each frame is strong in this case because information of an important frame is reflected immediately to output of verbalization.

5.4.4 Discussion D : Application Correct Answer Rate Caused by Packet Loss

The effect on the GTW algorithm which is one of the Time Warping algorithms caused by wireless communication quality was evaluated in this experiment. A method to absorb quality deterioration was proposed when GTW algorithm is influenced by the input data quality degradation by the packet loss. An better selection of matrix pixels can absorb the data processing method weakness. In the case of Time Warping, small size of matrix and almost the same length of sequence is effective to improve the analysis results.

Factor to Determine Data Quality	Data Deterioration type	Data Quality Deterioration type Subdivision	Data Processing Method	Suitability
Data Collection Terminal Error	A : Image Quality (Individual data quality deterioration on the time series)	Blurry	Each Frame	○
			Frame Group	○
		Blurred to length	Each Frame	○
			Frame Group	○
		Blurred to side	Each Frame	○
			Frame Group	○
		Resolution	Each Frame	○
			Frame Group	○
	B: frame drop (Data quality degradation when arranging them in the series)	Video frames drop	Each Frame	○
			Frame Group	×
		Acceleration frames drop	Each Frame	△
			Frame Group	△
		Video and Acceleration frames drop	Each Frame	×
			Frame Group	○
Data Transmission Error	C : Packet Loss Verbalization Application	6Mbps	Each Frame	×
			Frame Group	×
		18Mbps	Each Frame	○
			Frame Group	×
		36Mbps	Each Frame	○
			Frame Group	×
		54Mbps	Each Frame	○
			Frame Group	×
		Auto	Each Frame	○
			Frame Group	×
	D: Packet Loss Human motion alignment algorithm	Big Matrix Long and short sequence	Time Warping	△
		Small Matrix Same Length sequence		△
		Small Matrix Without background		○

Figure 5.18: Summary of Data Quality Evaluation Experiment

This chapter discussed four possible patterns of quality deterioration of “input data in Lifelog Analysis Application ”. When considering quality deterioration of Lifelog Analysis Application, the “operational quality”of the Lifelog Analysis Application is also a essential topic. It is mentioned in chapter 7.

Chapter 6

Human Activity Recognition System Implement on Inter-Cloud

In this chapter, I will mention the operational quality of discussed Lifelog Analysis Application. I will discuss how I expand the Lifelog Analysis Application.

I have concerned to introduce the Lifelog Analysis Application to each family as abnormal activity detection system inside a smart house. When considering setting up the system in each house, it is effective to collect and analyze the sensor data on Cloud environment. As shown in Fig. 1.4. Only network camera and a small sensor terminal, such as acceleration sensor are settled in the house, and for a large amount of data processing, it is desirable to analysis the data on the cloud and returns the result for each user after data analysis. This can be realized by utilizing a technique called cloud computing.

A large amount of Lifelog data collected from each household is analyzed by the private cloud and public cloud (inter-Cloud), and VM on each server analyzing the Lifelog data is migrated to ensure the fault tolerance of server. In the evaluation experiments in this chapter, I have particularly focused on security and speed of VM migration. I propose an efficient migration method, in terms of the evaluation, and implementation in the cloud, aim to build a system.

Outline. In section 6.1 of this chapter, what is Inter-cloud is explained, and in

section 6.2, migration technology in inter-Cloud is discussed. In section 6.3, migration experiment with proposed method is discussed, and the result and discussion of the experiment is mentioned in section 6.4.

6.1 Inter-Cloud and Data Security

In recent years, the data on real-world and cyber-space have increased dramatically due to the recent development of data collection and analytic technologies as well as high diffusion of Social Networking Service (SNS). As a result, these big data are managed by Cloud instead of local environment such as users' own devices. The advantages of managing big data on Cloud are as follows: (1) users do not need to possess large storage capacity or dedicated data management software, (2) users can access required data through wired or wireless networks and by any devices such as personal computers (PC) or Smart phones or tablet devices, (3) the virtual machine (VM) on Cloud can be replicated and migrated to another server for disaster recovery seamlessly.

Cloud-computing technology has evolved from "single-Cloud"(composed of private cloud and public cloud separately) to "hybrid Cloud"(composed of private cloud and public cloud connected with network), and then to near future "inter-Cloud"(composed of several clouds connected with network). However, Cloud computing technology has demerits that cannot be ignored — Security. For example, user cannot access their own data or VM physically when the VM is optimally migrated to another server. In addition the operating system (OS) on the host server is managing all VM, thus, if the host server are compromised, the important data and information of users will be lost and stolen.

Migration must be executed securely especially on inter-Cloud because different data and VM will be migrated to another cloud not just another host. The VMs must be migrated through safe and secure network such as security architecture for

Internet Protocol (IPsec). However, there is trade-off between security and speed of migration. Stronger security mechanism makes the migration speed slower. Especially in IPsec, the whole data is divided to small fragments and encryption is applied to each fragment. As a result, big-sized VM will take much longer time to complete the migration. This is not efficient.

I have proposed a method to execute Cloud migration securely and quickly. My method is based on the intuition on optimization efficiency to execute encryption and decryption to increase the speed of migration. In my method, I have encrypted all VM on source Cloud (server) before migration, send the VM to another Cloud (server) through the network, and decrypt the VM at the destination Cloud (server). I compared my method with existing migration technique using IPsec tunnel. My realistic comparison results showed dramatic performance improvement. I are planning to use this promising research results to examine (1) speed of encryption (2) partial encryption, as my future research work.

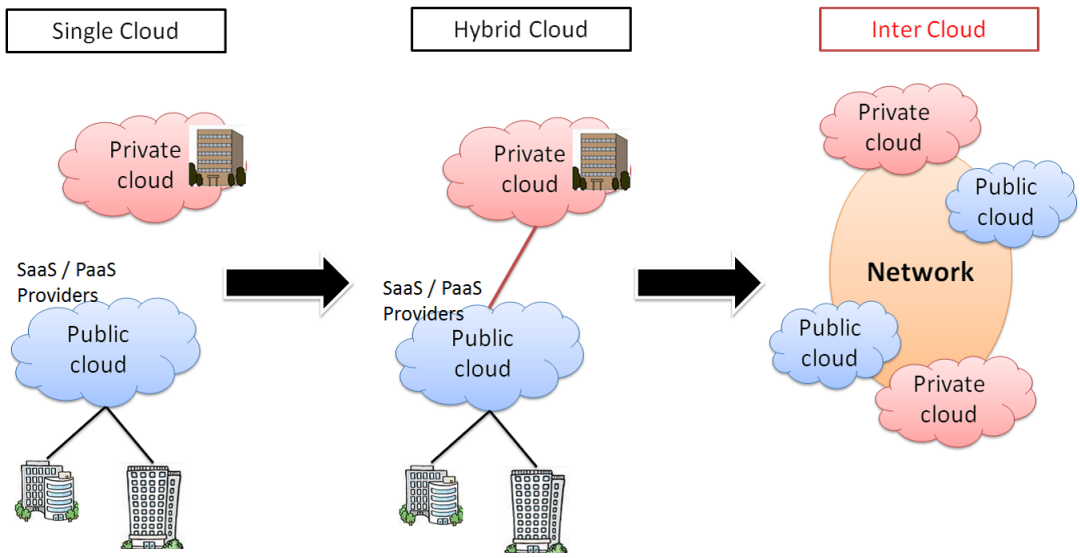

Figure 6.1: Evolution of cloud computing

As discussed above, the inter-cloud is an extension of single-cloud and hybrid-cloud. The evolution of cloud computing is shown in Fig. 6.1. In single cloud,

private cloud and public cloud exists separately. Private cloud is used by some companies or personal users with private internal network. Public cloud is managed by cloud providers such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) providers. User who want to use public cloud have to contract and pay for a cloud provider service. In hybrid cloud, private cloud and public is connected with network, and the data and information on private cloud can be seen or used by public cloud at least temporarily. Though the data and information of each contractor can not be seen from each other, the network connection must be secure. In inter-cloud, private cloud and public cloud of different cloud providers are connected with same network, thus, the security should be even more strong.

The architecture of Inter-Cloud is shown in Fig. 6.2. Inter-cloud is managed by cloud broker, cloud manager, and cloud directory. Cloud broker receives the request of resource reservation from users and selects proper cloud and send the request of resource reservation to the cloud and network. Cloud manager manages each cloud resources and executes accounting, allocation. Cloud directory collects information of cloud service and resources from cloud manager, and corresponds to the query of proper cloud service.

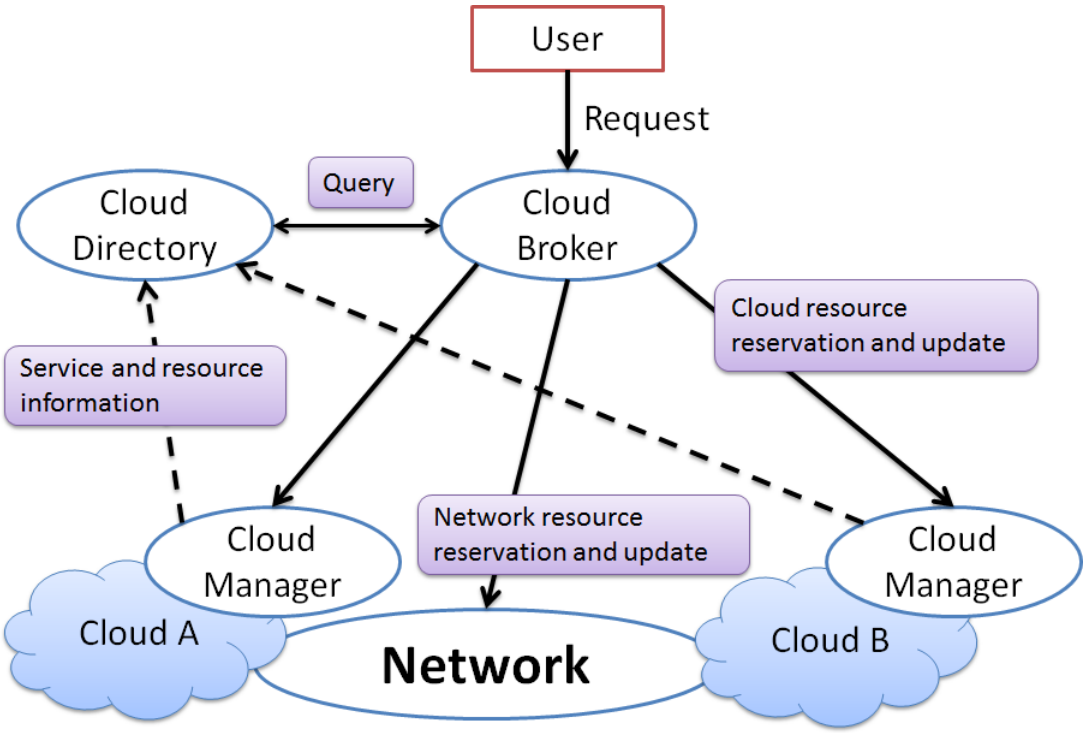

Figure 6.2: The architecture of Inter-cloud

6.2 Characteristic of Migration on Inter-cloud

Inter-cloud technology allows different types of cloud connected with network — inter-cloud. In addition, VM and data migration can be executed through different types of cloud which managed by different cloud providers. Thus, when a server host crashes or wide-scale disaster happens, inter-cloud technology can protect users data, information and lifeline safely with sharing the resources among different cloud. Additionally, with migrating VM optimally to the nearer server of users, users can use their resources without being aware of delay.

As for the network for migration on inter-cloud, the secure network such as IPsec is supposed to be used. As shown in the Fig. 6.3, the existing migration on inter-cloud are supposed to be as follows : (1)shutdown the VM of host server on a cloud, (2)migrate the VM another cloud through IPsec tunnel, (3)restart the VM on destination server on the other cloud. While migrating the VM through IPsec

tunnel, the data will be separated into small part on IP layer and each IP layer packet will be encrypted each time. Though the security is strong, IPsec is not so efficient. In addition, there is another way of migration — live migration in which shutdown and restart of VM are not necessary. In live migration using IPsec, the existing method is supposed to take much time as the process is complex. In IPsec, the packet of IP layer is encrypted and safely transmitted. The sender side and receiver side decide the key based on Security Association (SA), and Internet Key Exchange (IKE) is used on receiver side authorization. In this paper, I only executed the experiment of migration, live migration will be future work.

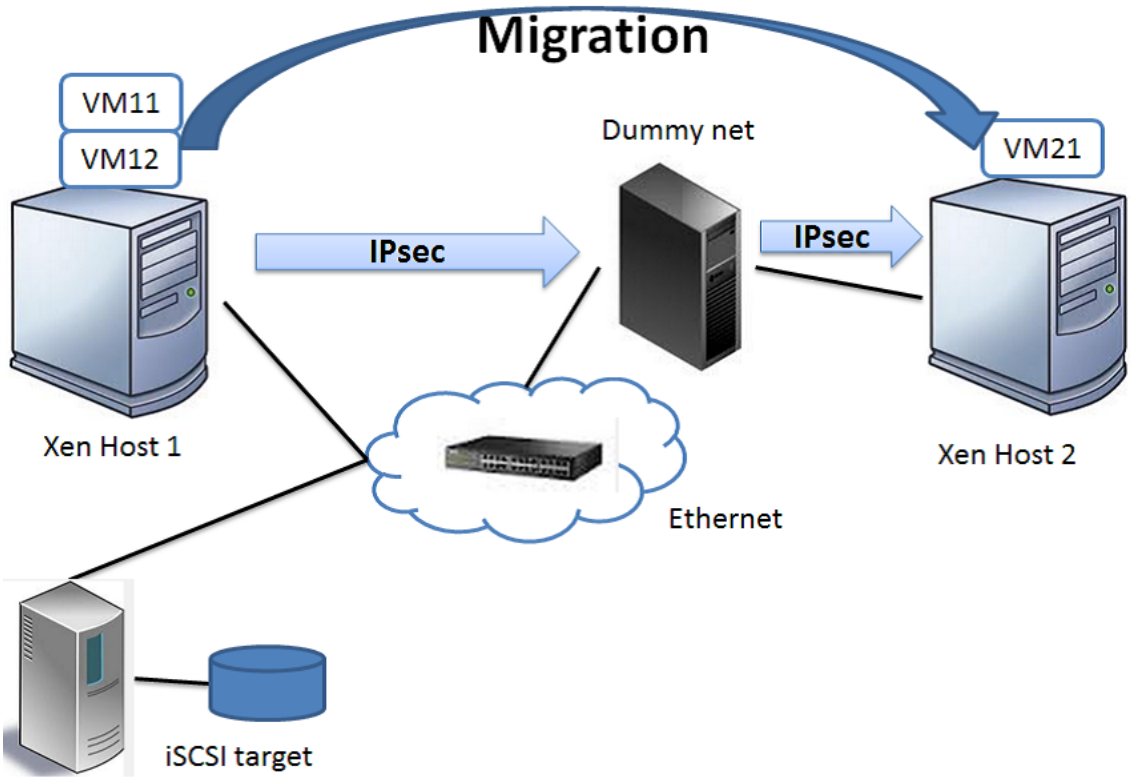

Figure 6.3: Supposed migration on inter-cloud

I suggest that encryption and decryption on each IP packet very inefficient. As shown in the Fig. 6.4, my proposed way is as follows:

1. Shutdown the VM on sender side

2. Encrypt all the VM

3. Send VM with Secure Copy protocol (SCP)

4. Decrypt all the VM on receiver side

5. Restart the VM on receiver side

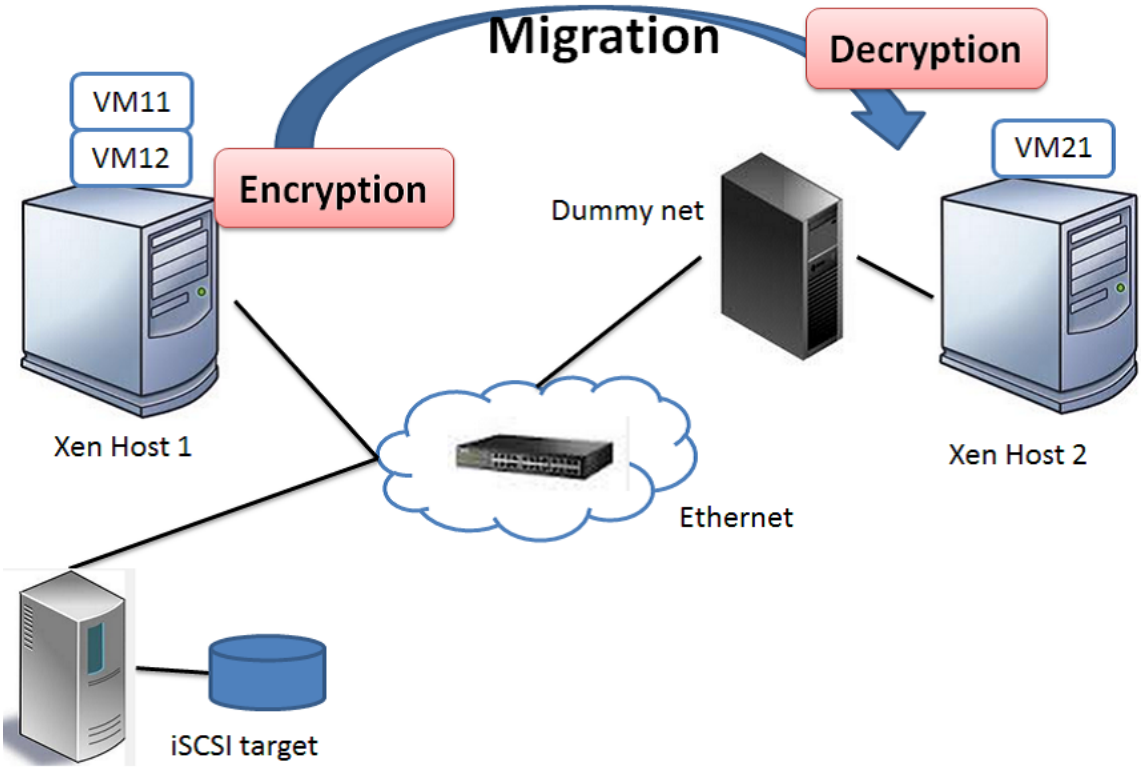

Figure 6.4: Proposed migration on inter-cloud

6.3 Experimental Environment

I have compared performance of following two methods: (1) migration using IPsec tunnel (Fig. 6.3), (2) my proposed migration method (Fig. 6.4). The real terminal experiment have been executed with two nodes supposed to belong to two different clouds.

6.3.1 Terminal in the Experiment

I have installed Xen [60] as virtual machine monitor on each host server. Xen provides a service of parallel processing and control of multi operating system (OS) with one hardware. In xen, each VM is called "domain". Xen hypervisor supports one or several OS and executes scheduling to physical Central Processing Unit (CPU). The host OS is called Domain-0 (dom0) and newly added OS (VMs) are called Domain-U (domU). On each host server, one dom0 manages several VM, domU. Dom0 starts automatically after hypervisor is booted and management authority is given. Every physical hardware can be accessed from dom0 directly. System administrator can login to all domU from dom0.

I have used iSCSI [61] as data storage for two xen host servers. iSCSI is a protocol to use SCSI protocol on TCT/IP network. It is standardized on transport layer. Recently, SAN storage is composed with Fiber Channel, however, the cost of establishing a network of Fiber channel is high. iSCSI is a technique to compose SAN, but iSCSI can be used with TCP/IP network instead of fiber channel. I have settled one iSCSI target which is storage server and iSCSI initiator which is client to access iSCSI target. In my experimental environment, iSCSI initiator is installed to dom0 on each xen host server.

I have sandwiched dummynet between Xen host 1 and Xen host 2, in other word, source server and destination server of migration. Dummynet is used to make high latency environment artificially. The spec of each terminal in the experiment is shown in Tab. 6.1, 6.2, 6.3, and 6.4.

When migrating the VM through IPsec tunnel, I have run IPsec with installing openswan package [63]. Xen host 1 (source terminal of migration) is IPsec client, and Xen host 2 (destination terminal of migration) is IPsec server. In my proposed method, I have executed encryption and decryption with openSSL [62].

Table 6.1: Xen host server 1 and 2 (same spec)

OS	Linux 2.6.32-5-xen-amd64 and xen-4.0-amd64
Distribution	Debian GNU / Linux 6.0.2
CPU	Intel (R) Xeon (R) CPU 3.60GHz
Memory	4 GByte
Disk	222 GByte

Table 6.2: VM and iSCSI initiator on Xen host server 1 and 2 (same spec)

OS	Linux 2.6.32-5-xen-amd64 and xen-4.0-amd64
Distribution	Debian GNU / Linux 6.0.2
CPU	Intel (R) Xeon (R) CPU 3.60GHz
VCPU	1 core
VCPU Memory	2 GByte
Storage	open-iscsi-2.0-873
Disk	5 GByte

Table 6.3: iSCSI target

OS	Linux 2.6.32-5-amd64
Distribution	Debian GNU / Linux 6.0.2
CPU	Intel (R) Xeon (R) CPU 3.60GHz
Storage	open-iscsi-2.0-873
Disk	130 GByte

Table 6.4: Dummynet

OS	FreeBSD 6.4-RELEASE
CPU	Intel (R) Xeon (R) CPU 3.60GHz
Disk	64 GByte

6.3.2 Command Line in the Experiment

Migration through IPsec tunnel

1. Start iSCSI target

2. Connect to iSCSI target from two iSCSI initiator
   (Xen host server 1 and 2)

3. Create and start VM (domU) on migration source terminal
   (Xen host server 1)

4. Start IPsec tunnel on both Xen host server
   (1024bit RSA key)

5. Migrate the VM (domU) from migration source terminal to destination terminal
   (Xen host server 1 to 2)

my Proposed Migration

1. Start iSCSI target

2. Connect to iSCSI target from two iSCSI initiator
   (Xen host server 1 and 2)

3. Create and start VM (domU) on migration source terminal
   (Xen host server 1)

4. Shutdown the VM (domU) on migration source terminal
   (Xen host server 1)

5. Encrypt all the VM (domU)
   (root directory of domU)
   (256bit AES key)

6. Send encrypted VM (domU) with Secure Copy protocol (SCP) from source terminal to destination terminal
   (Xen host server 1 to 2)

7. Decrypt all the VM (domU) on migration destination terminal
   (Xen host server 2)

8. Restart the VM (domU) on migration destination terminal
   (Xen host server 2)

6.4 Experimental Result and Discussion

The result is shown in Fig. 6.5. The graph is comparison of existing migration method (IPsec tunnel migration) and my proposed method. The vertical axis is time to complete VM migration and the unit is minute. The horizontal axis is round trip time (RTT) and the unit is millisecond (ms). I have measured RTT value from 0 (ms) to 200 (ms). When considering migration between two different located server, the distance is supposed to be longer and network latency is higher as the VM will be migrated among servers all over the world. For example, the distance from Tokyo, Japan to Osaka, Japan is about 550 km and RTT is about 20 ms. The RTT is about 120 ms from Tokyo, Japan to west part of United states, about 200 ms and more to Europe countries.

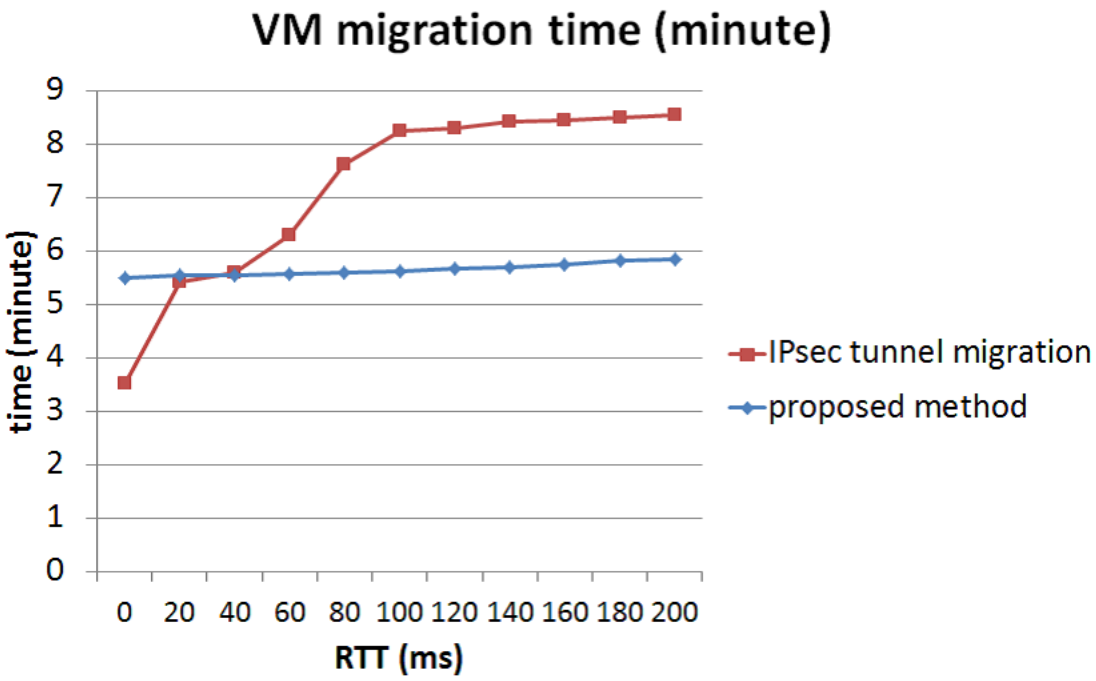

Figure 6.5: Comparison of existing migration method and my proposed method

As shown in the graph in Fig. 6.5, The migration of domU took about 3.5 minutes (min) even though in no-latency environment and as the RTT increases, the migration time grows dramatically and migration took over eight minutes when the RTT is over 100 ms. IPsec is bottleneck of migration because the packet is divided into small segments, in addition, CPU allocation also makes migration time longer.

On the other hand, my proposed method performs well and migration time does not increase so much as RTT increases. The migration time is kept between 5 min and 6 min. In my proposed method, the bottleneck of migration time is encryption and decryption time of VM which is shown in Fig. 6.6. The graph in Fig. 6.6 shows the constant time of my proposed method. Each bar graph shows the time of Encryption time before migration, decryption time after migration, the total time of encryption and decryption, VM shutdown time before migration, VM restart time after migration, and total time of shutdown and restart of VM.

As the graph shows, total time of encryption and decryption is about 5 min.

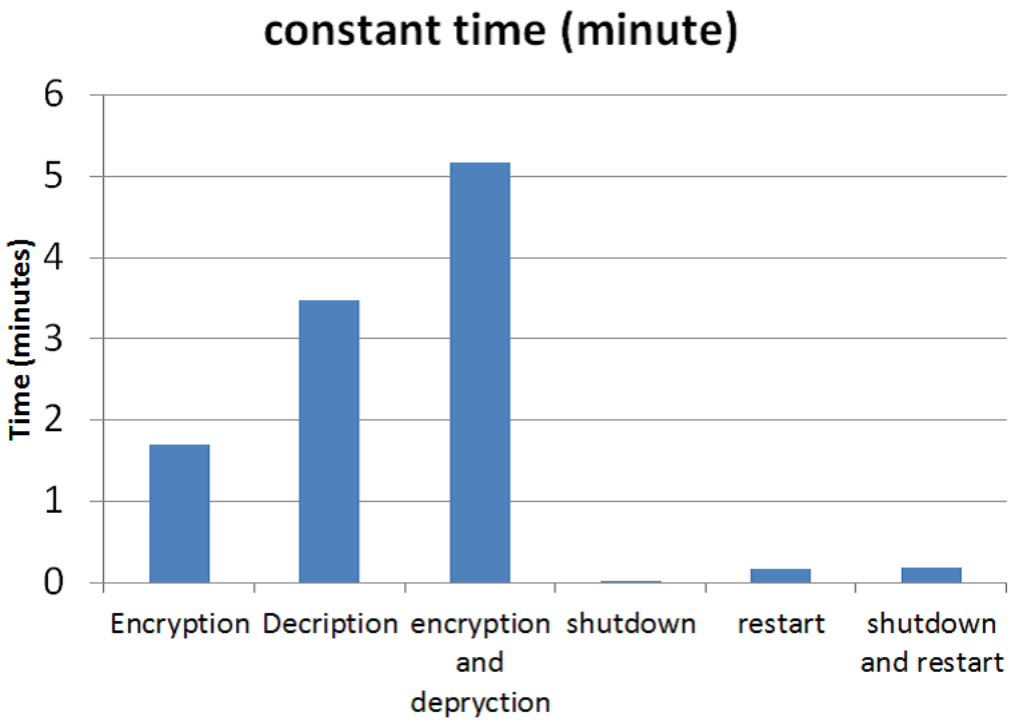

Figure 6.6: Constant value on my proposed method

As the graph in Fig. 6.5 shows, when the RTT is 40 ms, the performance of two methods be almost same, and when RTT is over 60 ms, my proposed method performs better than existing method. An shown in Fig. 6.7, when the RTT is over 100 ms, the time of proposed method is about 60% of existing method. My proposed method have achieved 40% performance improvement.

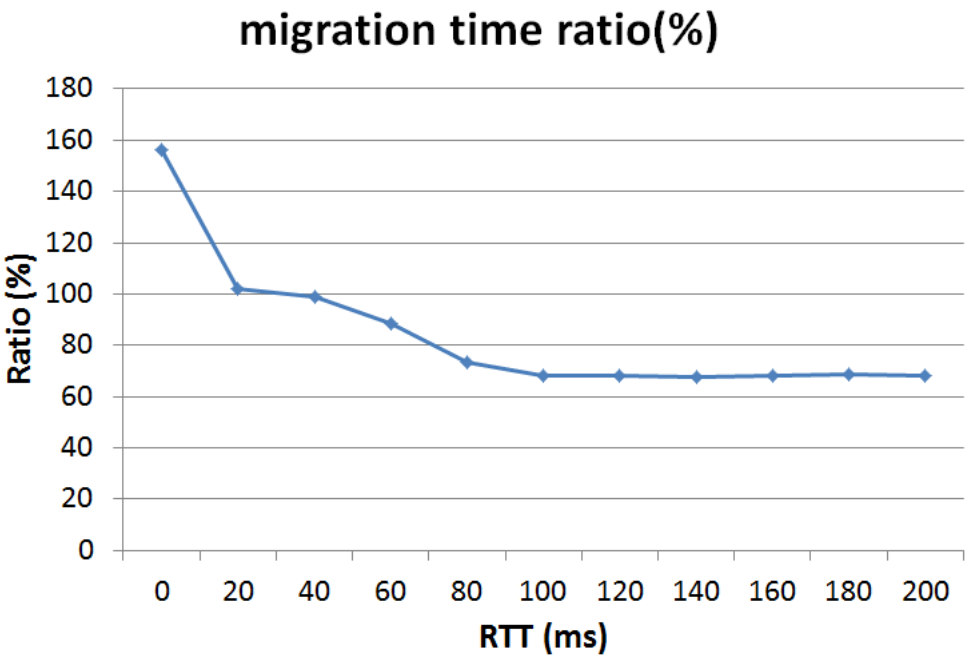

Figure 6.7: The time ratio of proposed method to existing method

In the experiment in this paper, the encryption key type of existing method with IPsec tunnel (1024bit RSA key) and proposed method (256bit AES key) is different. I will uniform the key length and key type in future work. If the key type is changed, the cross point of graph shown in Fig. 6.5 is supposed to move. Stronger and longer key makes the graph of proposed method move to up. With improving speed of encryption and partial encryption, my proposed method can be improved more.

Chapter 7

Conclusion

In this research, I have focused on correlation between input data quality and Lifelog Analysis Application analysis result. There are a lot of types of Lifelog Analysis Application developed before, but the input data quality of them has not been concerned. When developing Lifelog Analysis Application, to consider the impact of data quality on the result of the application is very important. If the best quality for application is known, the input data can be collected effectively, no need to collect very high quality of data to waste storage, analysis resources, and analysis time. Also in the situation where the data quality deterioration can not be voided, the type of data processing model can cover the problem and help the application analysis for better output result. To cover this problem, I have performed data quality evaluation experiment to clarify the input data quality deterioration impact on the analysis result at application level, and showed a guideline to handle the input data quality deterioration.

In the data quality evaluation experiment, I have utilized verbalization application and human motion alignment algorithm GTW as the instance of Lifelog Analysis Application to evaluate data quality influence. The input data of these system is image data series (video data) and acceleration data.

The data quality valuation experiment is performed with proposed a Data Quality Evaluation Framework, which is consist of typical three layers (data col-

lection layer, data processing layer, and information analysis layer), which can also applied to another Lifelog Analysis Application. It is a method to evaluate the influence of the Lifelog Analysis Application caused by the deterioration of input data quality.

In the data quality evaluation experiment, I have dealt with the factor to determine input data quality. One is the data quality change during data collection, for example, if some processing error of device happen, some frames of stream data may drop or the quality of each frame may drop. The other is data quality change during transmitting the data toward analysis terminals. Wireless LAN communication quality with which the input data is transmitted in real time was considered. If network packet lose happen while stream data transmission, the quality of transmitted data may drop.

Data quality evaluation experiments, I have implemented three different kinds of data processing method on the human activity recognition system. Bayesian Classifier model, HMM, and GTW. I have selected these methods as they ware widely utilized for human activity recognition system. The difference of the model is that Bayesian Classifier model process the data per each frame of the time series, and HMM and GTW process the data per series of frames. I have dealt with plural model to show which model is strong to handle each data quality deterioration. I have shown the correlation between data quality deterioration quantitatively and the correct answer rate and the minimum required quality for the human activity recognition system.

From the data quality evaluation experiment result, we found that individual data quality deterioration caused by data collection device error does not affect the Application Analysis result so much, but the frame dropping while arranging them to series, the processing method should be used properly depend on the number of objects to drop data frame. The method to process the data per each frame can

cover single object of data frame drop, but if multiple objects to drop data frame, the method to process the data per time series is better.

Data data transmission error affected the method to process the data per time series badly, thus, the method to process the data per each frame should be used in case of this type of quality deterioration. If only the method to process the stream data per time series is implemented on the application such as human motion alignment algorithm, input time series can be adjusted to cover this problem. In GTW algorithm, the size of each frame in the time series should be small, and the length of the time series to be processed should be almost the same.

Acknowledgments

First of all, special thanks to my supervisor Professor Masato Oguchi for his invaluable support and guidance through the hard moments of my school. Thanks to Professor Oguchi's 24-hour support, I could attend a lot of kinds of conference held at many countries and cities to present our work, and discussed the research. I am grateful for Prof. Oguchi's valuable feedback, and insightful ideas to this research. I also appreciate Prof. Oguchi's generosity letting me become an adult graduate student for doctor's course. I could spend great school life owing to his technical assistance.

Professor Ichiro Kobayashi is my deputy supervisor. I have learned how to apply Bayesian Classifier and HMM model to our human recognition system from him. His guidance was really indispensable for our work as natural language processing was out of our line. Thanks to him, I could also spread my knowledge.

Professor Eng Keong Lua was my supervisor when I was studying abroad at Monash University in Malaysia. I have worked on image processing with GTW model. He also helped me brush up my English and advised us a lot even after I come back to Japan.

Professor Saneyasu Yamaguchi helped us implement secure and fast VM migration method with kernel monitor. I feel very sorry that I could not complete the implementation of kernel monitor because the work at my company was so hard. His skill on implementing kernel monitor inspired me a lot, and what I have

learned from Prof. Yamaguchi will be the future foundation.

This paper was completed, supported by a lot of people I can not write down enough. Special thanks to teachers at Ochanomizu University who helped me study information technology, the students at Oguchi Laboratory who inspired each other, and all the people I met in the study life.

References

[1] A.J. Jara, M.A Zamora, and A.F.G Skarmeta
An Architecture Based on Internet of Things to Support Mobility and Security in Medical Environments
IEEE Consumer Communications and Networking Conference (CCNC), pp.1-5, 2010

[2] Y. Liu, B. Dong, B. Guo, J. Yang, and W. Peng
Combination of Cloud Computing and Internet of Things (IoT) in Medical Monitoring Systems
International Journal of Hybrid Information Technoloty, Vol.8, No.12, pp.367-376, 2015

[3] S. Sridhar, A. Hahn, and M. Govindarasu
Cyber Physical System Security for the Electric Power Grid
Proceedings of the IEEE Vol.100, pp.210-224, 2011

[4] E. Asimakopoulou, N. Bessis, S. Sotiriadis, F.Xhafa, and L. Barolli
A Collective Intelligence Resource Management Dynamic Approach for Disaster Management: A Density Survey of Disasters Occurrence
Third International Conference on Intelligent Networking and Collaborative Systems, pp.735-740, 2011

[5] S. Mohammadreza
A Network Mobility Solution Based on 6LoWPAN Hospital Wireless Sensor Network (NEMO-HWSN)
2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp.433-438, 2013

[6] S. Graham, G. Baliga, and P.R. Kumar
Abstractions, Architecture, Mechanism, and Middleware for Networked Control
IEEE Transactions on Automatic Control, vol. 54, no. 7, pp.1490-1503, 2009

[7] D. Singh
A survey of Internet-of-Things:Future Vision, Architecture, Challenges and Services
IEEE World Forum on Internet of Things(WF-IoT), pp,287-292, 2014

[8] Z. SHAO, and C. LIU
Intelligent management and service for Wisdom Scenic Based on Internet of Things
IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, pp.238-287, 2011

[9] W. He, and L. D. Xu
Integration of Distributed Enterprise Applications: A Survey
IEEE Trans. on Industrial Informatics ' , Vol.10, No.1, pp.35-42, 2014

[10] M. D. Ilic, L. Xie, U. A. Khan, and J. M. F. Moura
Modeling of Future Cyber Physical Energy Systems for Distributed Sensing and Control
IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, pp.825-838, 2010

[11] A. Rajhans, S.W. Cheng, B. Schmerl, B.H. Krogh, C. Aghi, and A. Bhave
An Architectural Approach to the Design and Analysis of Cyber-Physical Systems
Third International Workshop on Multi-Paradigm Modeling, 2009.

[12] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi
Internet of Things for Smart Cities
IEEE Internet of Things Journal Vol.1 No.1, pp.22-32, 2014

[13] P. C. Ribeiro, and J. S. Victor
Human Activity Recognition from Video: modeling, feature selection and classification architecture.
HAREM2005, ACTIVITY RECOGNITION (I), 2005.

[14] T. T. Truyen, H. H. Bui, and S. Venkatesh.
Human Activity Learning and Segmentation using Partially Hidden Discriminative Models.
HAREM2005, ACTIVITY RECOGNITION (II), 2005.

[15] N. Kawaguchi, N. Ogawa, Y. Iwasaki, K. Kaji, T. Terada, K. Murao, S. Inoue, K. Kawahara, Y. Sumi, and N. Nishio.
HASC Challenge2010:Construction of Wearable Accelerometer Sensor Corpus for Activity Recognition.
DICOMO2011, 1E-1, 2011.

[16] N. Ogawa, K. Kaji, and N. Kawaguchi.
Effects of Number of Subjects on Activity Recognization - HASC2010corpus -.
DICOMO2011, 1E-2, 2011.

[17] Y. Hattori, S. Inoue, and G. Hirakawa.
Activity Recognization and Visualization for Activity Information Sharing System.
DICOMO2011, 1E-4, 2011.

[18] M. Nakamura, Y. Hattori, S. Inoue, and G. Hirakawa.
Evaluatoin of Large Scale Activity Information Sharing System using Acceleration Data and Video Data.
DICOMO2011, 2E-1, 2011.

[19] M. Nishino, Y. Nakamura, T. Yagi, S. Muto, and M. Abe.
A location predictor based on dependencies between multiple lifelog data.
ACM LBSN2010, pp.11-17, 2010.

[20] J. Sun, X Wu, S. Yan, L. F. Cheong, T. S. Chua, and J. Li.
Hierarchical spatio-temporal context modeling for action recognition.
IEEE CVPR2009, pp.2004-2011, 2009.

[21] P. A. K. Acharya and A. Sharma and E. M. Belding and K. C. Almeroh and K. Papagiannaki.
Rate Adaption in Congested Wireless Networks through Real-Time Measurements
IEEE Transactions on Mobile Computing, pp.1535-1550, 2010

[22] Z. Zheng
Constrained Energy-Aware AP Placement with Rate Adaption in WLAN Mesh Networks
IEEE Global Telecommunications Conference (GLOBECOM2011), pp.1-5, 2011

[23] S. Iwaki, T. Murase, and M. Oguchi
Throughput Analysis and Measurement on Real Terminal in Multi-rate Wireless LAN
ACM International Conference on Ubiquitous Information Management and Communication (ICUIMC2011), Article No.119, 2011

[24] F. D. Simone, M. Naccari, M. Tagliasacchi, F. Dufaux, S. Tubaro, and T. Ebrahimi
Subjective Quality Assessment of H.264/AVC Video Streaming with Packet Losses
EURASIP Journal on Image and Video Processing 2011, Article No.190431, 2011

[25] S. Moncrieff, S. Venkatesh, G. West, and S. Greenhill
Multi-modal emotive computing in a smart house environment
Pervasive and Mobile Computing, pp.74-94, 2007

[26] L. Wang, T. Gu, X. Tao, H. Chen, and J. Lu
Recognizing multi-user activities using wearable sensors in a smart home
Pervasive and Mobile Computing, pp.287-298, 2011

[27] D. Perez-Botero
A Brief Tutorial on Live Virtual Machine Migration From a Security Perspective
Princeton University, Princeton, 2011

[28] Jansen, and T. Gerardus
Mechanism for Inter-Cloud Live Migration of Virtualization Systems
Patent, ad Number. US9104460 B2, 2015

[29] R. Buyya, R. ranjan, and R. N. Calheiros
InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services
the 10th International Conference on Algorithms and Architecture for Parallel Processing (ICA3PP2010), pp.13-31, 2010

[30] K. Nagin, D. Hadas, Z. Dubitzky, A. Glikson, I. Loy, B. Rochwerger, and L. Schour
Inter-cloud mobility of virtual machines
the 4th Annual International Conference on Systems and Storage Article(SYSTOR2011), Article No. 3, 2011

[31] M. Tsugawa, P. Riteau, A. Matsunaga, and J. Fortes
User-level virtual networking mechanisms to support virtual machine migration over multiple clouds
IEEE GLOBECOM Workshops (GC Wkshps 2010), pp.568-572, 2010

[32] H. Xu, and B. Li
Egalitarian Stable Matching for VM Migration in Cloud Computing
IEEE INFOCOM Workshop on Cloud Computing(INFOCOM Workshop2011), pp.631-636, 2011

[33] E. Ochiai, and I. Kobayashi.
Efforts to verbalize human activity in given space based on the predict model.
JSAI2010, 2G1-OS3-2, 2010.

[34] Wireless sensor network devices:
http://www.jp.sun.com/products/software/sunspot,
SunSPOT memo wiki, http://www.klab.ai.kyutech.ac.jp/yatti/sunspot/
index.php?Sun\%20SPOT\% 20memo\% 20wik

[35] Open Computer Vision Library:
http://sourceforge.net/projects/opencvlibrary/

[36] PLANEX CS-WMV04N :
http://www.planex.co.jp/product/camera/cs-wmv04n

[37] OchaHouse PukiWiki,
Ubiquitous Computing Experiment House of Ochanomizu University:
http://ochahouse.com/

[38] Video For Windows :
http://www.ecoop.net/coop/vfw/avi.html

[39] Planex MZK-MF300N :
http://www.planex.co.jp/product/router/mzk-mf300n/spec.shtml

[40] Iperf :
http://sourceforge.net/projects/iperf/

[41] Nexus S :
http://ja.wikipedia.org/wiki/Nexus_S

[42] AirPCap :
http://www.cacetech.com/products/airpcap.html

[43] F. Zhou, Fernando, and D. la Torre.
Generalized Time Warping for Multi-modal Alignment of Human Motion.
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1282-1289, 2012.

[44] F. Zhou, Fernando, and D. la Torre.
Canonical Time Warping for alignment of human behavior.
Neural Information Processing Systems (NIPS), pp.2286-2294, 2009

[45] D. Gong, and G. Medioni
Dynamic Manifold Warping for view invariant action recognition
IEEE International Conference on Computer Vision (ICCV), pp.571-578, 2011

[46] Dynamic Time Warping :
http://en.wikipedia.org/wiki/Dynamic_time_warping

[47] E. J. Keogh, and M. J. Pazzani
Derivative dynamic time warping.
SIAM International Conference on DATA MINING (SDM2001), 2001

[48] I. L. Dryden, and K. V. Mardia
Statistical shape analysis (1998), 1998

[49] E. Hsu, K. Pulli, and J. Popovic,
Style translation for human motion.
ACM Trans. Graph 2005, pp.1082-1089, 2005

[50] Human Sensing Lab (Carnegie Mellon University).
`http://www.cs.cmu.edu/~ftorre/`

[51] 本村陽一, 西田佳史:
日常環境における支援技術のための行動理解,
人工知能学会誌, 20(5), pp.587-594, 2005

[52] Y. Nishida, Y. Motomura, G. Kawakami, N. Matsumoto, and H. Mizoguchi
Spatio-tempora semantic map for acquiring and retargeting knowledge on everyday life behavior,
Lecture Notes in Artificial Intelligence, JSAI 2007 Conference and Workshops, Revised Selected papers, pp.63-75, 2008

[53] 川上悟郎, 西田佳史, 本村陽一, 溝口博:
ロケーション EMG センサを用いた行動の時空間展開記述に基づく日常生活行動モデリング手法,
知能情報ファジィ学会誌, 20 (2), pp.190-200, 2008.

[54] 本村陽一, 西田佳史:
ベイズ推定における事前分布のグラフ構造モデリングと実生活行動理解,
情報処理学会論文誌コンピュータビジョンとイメージメディア CVIM, 18, pp.43-56, 2007

[55] A. V. McCree, and T. P. Barnwell III
A mixed excitation LPC vocoder model for low bit rate speech coding
IEEE Trans. Speech and Audio Processing, vol.3, no.4, pp.242-250, 1995.

[56] M. Nishiguchi, K. Iijima, and J. Matsumoto
Harmonic vector excitation coding of speech at 2.0 kbps
IEEE Workshop on Speech Coding for Telecommunications pp.39-40, 1997.

[57] Starner, T., Pentland A., and Weaver, J.
Real-time American Sign Language recognition using desk and wearable computer based video
IEEE Trans. PAMI, Vol.20, No.12, pp.1371-1375, 1998

[58] Ong, S.C.W., and Ranganath, S.
Automatic Sign language Analysis: A Survey and the Future beyond Lexical Meaning
IEEE Trans. PAMI, Vol.27,No.6, pp.873-891, 2005

[59] 北 研二,
言語と計算 4 確率的言語モデル
東京大学出版会

[60] Xen project
http://xen.org/

[61] open-iSCSI
http://www.open-iscsi.org/

[62] J. Viega, M. Messier, and P. Chandra
Openssl
"Network Security with OpenSSL", O'Reilly (2009)

[63] Openswan
https://www.openswan.org/projects/openswan/

Appendix A

System Setup

A.1 Visual C++ 2008

1. Install Visual C++ 2008 from following URL

```
http://www.microsoft.com/japan/msdn/vstudio/2008/product/express/
(Visual C++ 2008 Web)
```

2. Install OpenCV form following URL

```
http://sourceforge.net/projects/opencvlibrary/
( Download "OpenCV_1.1pre1a.exe")
```

3. Setup OpenCV environment on Visual C++ 2008. The test image data path of OpenCV is as follow.

```
C:\Program Files\OpenCV\sample\c
```

If the image data does not appears correctly, please go to step 4.

4. How to set the path of test image data
set Environment Variables of System Property as follow.

```
";C:\Program Files\OpenCV\bin"
```

Don't forget to add ";" at beginning of the line.

A.2 Network Camera

Install following two softwares included in CD-ROM attached with Network Camera kit.
"IPCam Admin Utility", "PCI Network Camera Viewer"

A.3 SunSPOT

1. Insert set up CD-ROM of SunSPOT to Windows XP PC.
The following files are inside the CD-ROM
".ant", ".Demos", ".jdks", ".NetBeans", ".sdk", "SunSPOTSDKnstaller.jar"

2. Check the java version on the PC.

```
"java -version"
```

If java is installed, go to step 4, otherwise, go to step 3.

3. Install java.

```
a.  Click jdk-1_5_0_07-windows-i586-p.exe in ".jdks".
b. Select accept.
c. Select Custom setup.
e. Register the browser.
f. Done
g. Check the java version with the command java -version

Check the JAVA_HOME.
Input "set" command to command prompt, if "JAVA_HOME" is included the output list,
go to step 4, otherwise, do following procedure.

set Environment Variables of System Property as follow.
Variable : "JAVA_HOME"
Path : "C:Program Files\Java\jdk 1.5.0_07"
```

4. Install SunSPOT SDK.
Click "SunSPOTSDKInstaller.jar" included in CD-ROM.

5. How to control SunSPOT
Utilize "NetBeans 5.5" installed.
The program is in following path.
"C:Program Files/Sun/SunSPOT/Demos"

Appendix B

Human Activity Recognition System Program

B.1 Header file definition

```
////////////////////////////////////////////////////////////////////////
//////////////////////     define        ///////////////////////////////
////////////////////////////////////////////////////////////////////////
#define _CRT_SECURE_NO_DEPRECATE 1  /* VisualC++2005 での警告抑制 */
#define _CRTDBG_MAP_ALLOC // メモリリーク検知

#define THRESHOLD 15 // 差分を求める際の閾値
#define THRESHOLD_MAX_VALUE 255 // 2 値化の際に使用する最大値
#define CAPTURE_OFF 0 // 画像のキャプチャを中止するフラグ値
#define CAPTURE_ON 1 // 画像のキャプチャを開始するフラグ値
#define GRAY_DIFFERENCE     0 //  グレースケールにおける差分算出
#define NOISE_MORPHOLOGY 0 // モルフォロジー演算によるノイズ除去のフラグ値
#define SPHERE 10 //   RGB 背景差分法の閾値
#define CONT 500

// 変更所 //
#define THRESHOLD_AVI 2 //   avi の条件の閾値//ここを変えてみる !!
#define SENSOR_INTERVAL 0.1 //  この数以上の時、センサのノードを返す（閾値２）全部の物体共通にする
//#define SENSOR_INTERVAL3    0.3         //   (いす)
#define THRESHOLD_SENSOR    2          //  センサの条件の閾値
#define OBJECT_NUMBER     5 // 定義物体の個数（行動の個数）
#define DROP_NUMBER 10 // 1 秒に何回コマ落としするか（数字は使う回数 !! ）
#define MOVIE_DROP 10 // 動画データのコマ落とし

#define THRESHOLD_MARKOV 10 // マルコフの時の閾値（近づきがいくつを超えたらいいか)
// センサファイルのビットもセットする

//以下、肌色抽出用の閾値軍
#define HMAX 20
#define HMIN 0
#define SMAX 255*1
#define SMIN 255*0.2
#define VMAX 255*1.0
#define VMIN 255*0

// フィルタ用（縦と横ブレの行列用）
#define MAX_SIZE 100000
////////////////////////////////////////////////////////////////////////
//////////////////////     struct        ///////////////////////////////
////////////////////////////////////////////////////////////////////////

// 定義物体の構造体
typedef struct _Object{
int ObNumber; // 物体番号
char ObName[256]; // 物体名
char ObPic[32]; // 物体画像
IplImage *ObImage;
int OBcounter;
int OBDefTime;
CvPoint center_c1; // 定義物体の中心座標（構造体になっている）(カメラ 1)
CvPoint center_c2;
_Object *pNext;
_Object *pPrev;
}Object;

// 出力するかどうかのビット（それぞれの物体が 1 個ずつ持ってる） 出力する->1, 出力しない->0
typedef struct _OutputBit{
int R_000;
int R_001;
int R_010;
```

```
int R_011;
int R_100;
int R_101;
int R_110;
int R_111;
}OutputBit;

// 動詞、活用など、出力関係の情報はここに格納
typedef struct _TempBayes{
char ObNumber[8];
char ObName[128];
char Verb[128];
char PostPositional[2]; // 助詞
char Conjugation[128]; // 活用
OutputBit a_1; // この中に出力するかどうかが入ってる
_TempBayes *pNext;
_TempBayes *pPrev;
}TempBayes;

// FILE 型を持っている構造体（sensor_test 用）
typedef struct _File_struct{
int n; // 何番目の構造体か
int flag; // センサが入っているかのビット（1->あり、0->なし）
FILE *fp_in; // 入力用ファイル
FILE *fp_out; // 出力用ファイル
FILE *fp_node;
char text_in[32];
char text_out[32];
char text_node[32];

int c_flag; // カメラ 1 と 2 に映ってるかどうか（10 01 11）

}File_struct;

////////////////////////////////////////////////////////////////////////
//////////////////////     global 変数        //////////////////////////
////////////////////////////////////////////////////////////////////////
//-------------------------------------------------
extern Object g_object[32]; // 定義物体の番号と名前を格納する配列

extern Object *pFirstObject1;
extern Object *pFirstObject2;
extern Object *pLastObject1;
extern Object *pLastObject2;

extern TempBayes *pFirstBayes;
extern TempBayes *pLastBayes;

extern OutputBit g_outputbit[OBJECT_NUMBER];
extern File_struct g_file_sensor[OBJECT_NUMBER];

//-------------------------------------------------
extern CvSize g_ImageSize1; // 画像のサイズ
extern CvSize g_ImageSize2; // 画像のサイズ
extern CvPoint g_Pos[4];
//extern CvPoint g_pos_new[2][OBJECT_NUMBER][4]; // 物体座標保持用（マウスを使わない時用）
extern CvPoint g_pos_new1[OBJECT_NUMBER][4];
extern CvPoint g_pos_new2[OBJECT_NUMBER][4];

extern CvPoint2D32f center1; //  重心座標ポインタ
extern CvPoint2D32f center2; //  重心座標ポインタ

extern IplImage *defineImage1; // 定義画像用 IplImage
extern IplImage *defineImage2; // 定義画像用 IplImage
extern IplImage *defMaskRedImage1; // 赤色塗りつぶしマスク画像用の IplImage
extern IplImage *defMaskRedImage2; // 赤色塗りつぶしマスク画像用の IplImage
extern IplImage *backImage1; // 背景画像用 IplImage
extern IplImage *backImage2; // 背景画像用 IplImage

extern IplImage *defObImage1[10]; // 個別定義物 2 値化画像（以下同）
extern IplImage *defObImage2[10]; // 個別定義物 2 値化画像（以下同）

extern IplImage *frameImage1; // フレーム毎の保持用
extern IplImage *frameImage2; // 〃

//-------------------------------------------------
extern int g_PointCnt;
extern int g_time; //  ファイル出力時間
extern int iObnum1; // 定義物体の数を格納
extern int iObnum2; // 定義物体の数を格納

// 入力と出力ファイル名（textver 用）
extern char sunspot_text_in[32];
extern char sunspot_text_out[32];

// センサノードの text ファイル（ファイル名の前半）
extern char sensor_node[32];

extern char cText[];
extern char *res_node;   // g_time ごとの画像のノード

extern char *file_memo;

extern char ob_NAME[OBJECT_NUMBER][16]; // 物体名を保持しておくグローバル変数
```

```
// センサ１のノードカウンタ（ドア用：s1，いす用：s3）
extern int s_counter[OBJECT_NUMBER];

extern int g_flag[OBJECT_NUMBER];
extern int g_flag_timer[OBJECT_NUMBER];

extern int s_count; // センサファイル用カウンタ

extern int fun_flag; // main 関数の中身を実行するかのフラグ

extern FILE *fp_for_prog; // メモ用ファイル
extern char *f_name_prog; // ファイル名

extern int thr; // フィルタリング用（処理画素数）
extern int filter_flag; // どのフィルタを使うか
extern int pre_put; // 連続して同じ言語化をしないようにするため、前の言語化の ob_number を保持しておくためのもの

// マルコフ関係
extern int mar_flag; // マルコフの書き込みバージョンと読込バージョンの切り替えフラグ
extern int b_or_h_flag; // ベイズか HMM かを切り替えるフラグ（デフォルトはベイズ）


// マルコフ：遷移の記録用
extern int pre_tra_c[OBJECT_NUMBER]; // カメラの遷移
extern int pre_tra_a[OBJECT_NUMBER]; // 加速度の遷移

// マルコフコマ落とし
extern int drop_movie; // 動画
extern int drop_sensor; // 加速度

// 動画の開始時刻が、どれだけずれてるか
extern int fr_dif; // フレームの差分

////////////////////////////////////////////////////////////////////////////
//////////////////////////       関数定義         ///////////////////////////
////////////////////////////////////////////////////////////////////////////

// 輪郭抽出と重心計算
void maskContour(int w_flag, IplImage *paintImage, IplImage *maskoutImage,
IplImage *contourImage, IplImage *momentImage, int area_line,
FILE *fp, CvPoint2D32f center, CvSize imageSize, Object *pFirst,
FILE *d_fp1, FILE *d_fp2);

// マウスによる座標指定 （クリックごとに座標値を取得）
void Mouse1(int event, int x, int y, int flags, void *param);
void Mouse2(int event, int x, int y, int flags, void *param);

// 繰り返し表示用マウス
void Mouse( int event, int x, int y, int flags ,void *param);

// 差分画像生成
void GrayScaleDifference(IplImage *currentImage, IplImage *backgroundImage, IplImage *resultImage, CvSize imageSize );

// 定義物体のリストからファイル出力（リストは解放する）
void OutPutObject(Object *pFirst);

// 画素値を得る
int Elemtypetraits(IplImage *img, int x, int y, int ch);

// 構造体をフリーする関数
void FreeObject(Object *pNode);
void FreeBayes(TempBayes *pNode);

//構造体の領域を確保して、物体定義ファイルから一行ずつ読みとって、格納。そのファイルに定義されている物体の数を返す
int SetObjectFromFile(char *FileName, int flag);

// マウスで物体を定義する関数
void MakeObjectNode_auto(char *WNDef1, char *WNDef2, char *RedPic1, char *RedPic2);
//  定義物体の中心を求める関数（引数：何番目の物体か）
void new_get_center(int n_i, int c_flag);

// 構造体格納関数
void MakeNode(char *ObName, int iObnum, int flag);

// 定義物体のリスト構造からファイルに書き出す
void PutFromListToFile(char *FileName, Object *pFirst);

// 定義域内に重心が含まれているか
void matching(Object *pFirst, CvSize imageSize, CvPoint2D32f center);

// 定義物体のマスク画像のファイル名を取得し cImageName に格納
void GetMaskFileName(char *cImageName, int iObnum, int flag);

// 指定された定義物体の画像を膨張処理し、保存、格納する。
void GetObjectMaskImage( IplImage *defMaskRedImage, IplImage *ObImage, char *cRed, char *str, CvSize g_ImageSize, int mov_number/*0 or 1*/, int ob_n);

//-------------------------------------------------
void new_Verbalization2(FILE *fp);  // ベイズの定理を使って、A_i の大小比べをきちんとしたやつ。(fp は書き込み用ファイル)

void new_Verbalization_m(FILE *fp_Text, FILE *s_fp_main);  // マルコフ ver 距離が書いてあるファイルを読み込んで解析 ver

void new_SetBayesFromFile2( char *FileName1, char *FileName2);     // csv ファイルからベイズのノードを読んで、別の csv ファイルに書き込み。
```

```
BOOL new_IsOutPut2( Object *pObject1, Object *pObject2, TempBayes *pMove, FILE *fp, int *r3_node/*配列になってる*/);   // TempBayes に
出力ビットが入ってる形

float calc_p(float R_A[][OBJECT_NUMBER], OutputBit *outputbit, int ob_n, FILE *fp_memo); // 出力するかどうかのビットを構造体に埋め込んで
返す関数。

int check_n(int n, int *arr, int e_n);

// text -> text ver
//  (引数  :  区切る回数  、  最大値か最小値か)
int scene_drop_text(int n, int max_or_min, char *fname1, char *fname2); // 1 秒につき、何回か (n 回) で区切る

// センサーの差を２行ずつ取って、node を設定し、ファイル
// interval : 差の閾値 fname1 : 入力ファイル fname2 : 出力ファイル
int set_sensor_node(int n, float interval, char *fname1, char *fname2);

// センサ処理一般化用（文字列結合でファイル名生成）
// pre_name + n_of_file -> str_out
void make_file_name(char *pre_name, int n_of_file, char *str_out);

//  センサファイルが存在するかをセットする関数
void set_file_bit();

// マウス定義してた座標設定
void set_g_pos();

// filter 用関数

//IplImage *smooth(IplImage *img, int thr); // 平滑化
void blur_len(IplImage *img, int thr); // ブレた画像（縦）
void blur_side(IplImage *img, int thr); // ブレた画像（横）
void resolution(IplImage *img, int thr); // 解像度

// ブレ画像用、配列作成関数
float make_len_array(float *array_len, int size_of_array); // ブレ画像の配列作成用
float make_side_array(float *array_side, int size_of_array); // ブレ画像の配列作成用

// マルコフの距離用ファイルの調整
void new_set_HMM_state(FILE *fp);

//プライバシ保護用のフィルタをかける
//アニメ風にする関数
void cv_CartoonFilter(IplImage* src, IplImage* dst, int BlockSize, double ContThre, int PyrLevel, double SegmentThre, int MedianLevel);
```

B.2 main function

```
//メモリリーク検知
#include <stdlib.h>
#include <crtdbg.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <cv.h>
#include <cxcore.h>
#include <highgui.h>
#include <iostream>
#include <windows.h>

// インクルードファイル
#include "hedder.h"

using namespace std; // 元々ついてたからとりあえず入れておく

//グローバル変数
//-------------------------------------------------
Object g_object[32]; // 定義物体の番号と名前を格納する配列

Object *pFirstObject1 = NULL;
Object *pFirstObject2 = NULL;
Object *pLastObject1 = NULL;
Object *pLastObject2 = NULL;

TempBayes *pFirstBayes = NULL;
TempBayes *pLastBayes = NULL;

OutputBit g_outputbit[OBJECT_NUMBER]; // 配列の形で物体の個数個だけ用意しておく。
File_struct g_file_sensor[OBJECT_NUMBER]; // 物体の形だけ用意しておく。

//-------------------------------------------------
CvSize g_ImageSize1; // 画像のサイズ
CvSize g_ImageSize2; // 画像のサイズ
CvPoint g_Pos[4];
//CvPoint g_pos_new[2][OBJECT_NUMBER][4];
CvPoint g_pos_new1[OBJECT_NUMBER][4];
CvPoint g_pos_new2[OBJECT_NUMBER][4];


CvPoint2D32f center1; //  重心座標ポインタ
CvPoint2D32f center2; //  重心座標ポインタ

IplImage *defineImage1; // 定義画像用 IplImage
```

```
IplImage *defineImage2; // 定義画像用 IplImage
IplImage *defMaskRedImage1; // 赤色塗りつぶしマスク画像用の IplImage
IplImage *defMaskRedImage2; // 赤色塗りつぶしマスク画像用の IplImage
IplImage *backImage1; // 背景画像用 IplImage
IplImage *backImage2; // 背景画像用 IplImage

IplImage *defObImage1[10]; // 個別定義物 2 値化画像（以下同）
IplImage *defObImage2[10]; // 個別定義物 2 値化画像（以下同）

IplImage *frameImage1;
IplImage *frameImage2;
//-----------------------------------------------
int g_PointCnt = 0;
int g_time = 0; //  ファイル出力時間
int iObnum1 = 0; // 定義物体の数を格納
int iObnum2 = 0; // 定義物体の数を格納

// 入力と出力ファイル名（textver 用）
// ファイル名の前半
char sunspot_text_in[32] = "t_sensor_in_";
char sunspot_text_out[32] = "t_sensor_out_";

// sensor ノードの text ファイル
// ファイル名の前半
char sensor_node[32] = "sensor_node_";

char cText[] = "Result.txt"; // ←おかしかったら main の中に戻す !!
char *res_node = "Result_node.txt";   // g_time ごとの画像のノード

char *file_memo = "R_A_memo.txt";

char ob_NAME[OBJECT_NUMBER][16];

// センサ 1 のノードカウンタ（ドア用：s1，いす用：s3）じゃなくて物体数だけ
int s_counter[OBJECT_NUMBER];


int g_flag[OBJECT_NUMBER];
int g_flag_timer[OBJECT_NUMBER];

// マルコフ設定
int s_count = 0;
int fun_flag = 1;
int mar_flag = 0; // 0:距離書き込み (bayes の時もこっち !! )  1:読込&解析
int b_or_h_flag = 0; // bayes か HMM か。(0:bayes 1:HMM)
int drop_movie = 10; // 動画のコマ落とし（マルコフ）
int drop_sensor = 10; // センサのコマ落とし（マルコフ）

// フレームズレ設定
int fr_dif = 2*10;

FILE *fp_for_prog;
char *f_name_prog = "for_prog.txt";

// フィルタ定義
int thr=15;
// 0:フィルタなし 1:平滑化 2:縦ブレ 3:横ブレ 4:解像度
int filter_flag = 0;

int pre_put=0; //言語化出力制御用

// マルコフ：遷移の記録用
int pre_tra_c[OBJECT_NUMBER]; // カメラの遷移
int pre_tra_a[OBJECT_NUMBER]; // 加速度の遷移
////////////////////////////////////////////////////


//main 関数

int main( int argc, char *argv[] ){

    int a; // 終了用（特に意味はない。）

CvCapture *Capture1 = NULL; // カメラキャプチャ用の構造体
CvCapture *Capture2 = NULL; // カメラキャプチャ用の構造体

int key; // キー入力用の変数
bool bIsPush = FALSE; // 指定のボタンが押されたかの判断用
bool bIsNewAVI = FALSE; //

char pic_DefMaskRed1[] = "defMaskRed1.jpg"; // 赤色塗りつぶしマスク画像の名前
char pic_DefMaskRed2[] = "defMaskRed2.jpg"; // 赤色塗りつぶしマスク画像の名前

char WNDef1[] = "Define1"; // 前回定義画像を表示するウインドウの名前
char WNDef2[] = "Define2"; // 前回定義画像を表示するウインドウの名前
char WNCapt1[] = "Camera1";
char WNCapt2[] = "Camera2";
char WNCon1[] = "Contour1";
char WNCon2[] = "Contour2";
char WNmask1[] = "Painted1";
char WNmask2[] = "Painted2";
char cDefObjFile1[] = "Object1.csv"; // 定義物体を保存する CSV 形式のファイル名
char cDefObjFile2[] = "Object2.csv"; // 定義物体を保存する CSV 形式のファイル名
char cMmtFile1[] = "Moment1.csv"; // 重心の位置を保存する
```

```
char cMmtFile2[] = "Moment2.csv"; // 重心の位置を保存する
char cBayesFile[] = "Bayes_in.csv"; // テンプレートとその確率が保存してあるファイル
char cBayesFile2[] = "Bayes_out.csv";     // ベイジアンネットの学習用の出力ファイル。結局使ってないけど。

char VideoName1[] = "Y:\\img\\rec\\test1.avi";
char VideoName2[] = "Y:\\img\\rec\\test2.avi";

IplImage *currentImage1;
IplImage *currentImage2;
IplImage *pre_Image1;
IplImage *pre_Image2;

IplImage *resultImage1;
IplImage *resultImage2;
IplImage *maskoutImage1;
IplImage *maskoutImage2;
IplImage *contourImage1;
IplImage *contourImage2;
IplImage *momentImage1;
IplImage *momentImage2;

IplImage *paintImage1;
IplImage *paintImage2;

IplImage *paintImage1_mono;
IplImage *paintImage2_mono;

FILE *f_Text;
FILE *f_Moment1;
FILE *f_Moment2;


FILE *d_fp1; //距離書き込み用ファイル
FILE *d_fp2;
char *fname1 = "distance1.txt";
char *fname2 = "distance2.txt";

FILE *tra_fp;
char *fname_tra = "trance.txt";

int captureOn = CAPTURE_ON; // 背景差分を行う画像を更新するかどうか
int differenceMode = GRAY_DIFFERENCE; // 差分の計算モード
int noiseMode = NOISE_MORPHOLOGY; // ノイズを除去するモード

// メモ用ファイルを開く
if( (fp_for_prog = fopen(f_name_prog,"w")) == NULL ){
printf("出力ファイルを開けません\n");
return -1;
}

//while(fr_dif > 0){
//----------------<<概要 1:キャプチャの初期化>>--------------
if( (Capture1 = cvCaptureFromAVI(VideoName1)) == NULL ){
// ファイルが見つからなかった場合
printf( "%s が見つかりません\n", VideoName1 );
return -1;
}
if( (Capture2 = cvCaptureFromAVI(VideoName2)) == NULL ){
// ファイルが見つからなかった場合
printf( "%s が見つかりません\n", VideoName2 );
return -1;
}

if(fun_flag == 1){ // 1:中身を実行, 2:マウス座標の表示

//--------------------<<概要 1:おわり>>----------------------

// 初期画像を 1 枚キャプチャし背景, 定義用として保存する
frameImage1 = cvQueryFrame( Capture1 );
frameImage2 = cvQueryFrame( Capture2 );

////////////////////////////////////////////////
// filter_flag != 0 ならいずれかのフィルタ処理を行う。
////////////////////////////////////////////////

if(filter_flag!=0){
if(filter_flag==1){ // 平滑化
cvSmooth(frameImage1, frameImage1, CV_BLUR, thr, 0, 0, 0); // フィルタ処理
cvSmooth(frameImage2, frameImage2, CV_BLUR, thr, 0, 0, 0); // フィルタ処理
}else if(filter_flag==2){ // 縦ブレ
blur_len(frameImage1, thr);
blur_len(frameImage2, thr);
}else if(filter_flag==3){ // 横ブレ
blur_side(frameImage1, thr);
blur_side(frameImage2, thr);
}else if(filter_flag==4){ // 解像度
resolution(frameImage1, thr);
resolution(frameImage2, thr);
}
}

backImage1 = cvCloneImage( frameImage1 );
backImage2 = cvCloneImage( frameImage2 );
```

```
///-------------------------------------------------------------------------論文用
cvSaveImage("frameImage1.jpg",frameImage1);
cvSaveImage("frameImage2.jpg",frameImage2);


//画像サイズの保存
g_ImageSize1 = cvSize( frameImage1->width, frameImage1->height);
g_ImageSize2 = cvSize( frameImage2->width, frameImage2->height);

//IplImage の初期化
defineImage1 = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 3 );
defineImage2 = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 3 );
currentImage1 = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 3 );
currentImage2 = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 3 );

pre_Image1 = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 3 );
pre_Image2 = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 3 );

resultImage1 = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 1 );
resultImage2 = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 1 );
maskoutImage1 = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 3 );
maskoutImage2 = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 3 );
contourImage1 = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 3 );
contourImage2 = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 3 );
momentImage1 = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 3 );
momentImage2 = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 3 );

paintImage1 = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 3 );
paintImage2 = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 3 );

// 白黒用の出力画像
paintImage1_mono = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 3 );
paintImage2_mono = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 3 );


//--------------上に IplImage の初期化をまとめて書いておこう

//定義用ウィンドウ生成
cvNamedWindow( WNDef1, CV_WINDOW_AUTOSIZE );
cvNamedWindow( WNDef2, CV_WINDOW_AUTOSIZE );


if( !(((defMaskRedImage1 = cvLoadImage( pic_DefMaskRed1, CV_LOAD_IMAGE_ANYDEPTH | CV_LOAD_IMAGE_ANYCOLOR )) != NULL)
&& ((defMaskRedImage1->width) == (frameImage1->width))
&& ((defMaskRedImage1->height) == (frameImage1->height)))
){
defMaskRedImage1 = cvCreateImage( g_ImageSize1, IPL_DEPTH_8U, 3 );
cvSetZero(defMaskRedImage1);
printf("Set Mask Img1 0\n");
}


if( !(((defMaskRedImage2 = cvLoadImage( pic_DefMaskRed2, CV_LOAD_IMAGE_ANYDEPTH | CV_LOAD_IMAGE_ANYCOLOR )) != NULL)
&& ((defMaskRedImage2->width) == (frameImage2->width))
&& ((defMaskRedImage2->height) == (frameImage2->height)))
){
defMaskRedImage2 = cvCreateImage( g_ImageSize2, IPL_DEPTH_8U, 3 );
cvSetZero(defMaskRedImage2);
printf("Set Mask Img2 0\n");
}

//-----------------------------------------------------<<概要 1-2:おわり>>-----------------------------------------------------------
cvShowImage( WNDef1, frameImage1 );
cvShowImage( WNDef2, frameImage2 );

// いろいろ定義する！
// ベイズ処理の値をセット（出力ビットもセット）（物体名もセット）
new_SetBayesFromFile2( cBayesFile, cBayesFile2);
// センサファイルの有無のビットセット
set_file_bit();
set_g_pos();

//----------------<<概要 2：定義部分表示>>-------------------------------------------------------------------------------------

printf("\n ********** input '1' to show Defined Object Image \n");

key =cvWaitKey(0);

if(key == '1'){
//読み込んでいた構造体をフリーする
FreeObject(pFirstObject1);
FreeObject(pFirstObject2);
pFirstObject1 = NULL;
pFirstObject2 = NULL;
cvSetZero( defMaskRedImage1 );
cvSetZero( defMaskRedImage2 );
cvAddWeighted( backImage1, 0.7, defMaskRedImage1, 0.3, 0, defineImage1 ); //(背景)7 割:(塗りつぶし画像)3 割、重み付き合成で defineImage
に
cvAddWeighted( backImage2, 0.7, defMaskRedImage2, 0.3, 0, defineImage2 ); //(背景)7 割:(塗りつぶし画像)3 割、重み付き合成で defineImage
に
cvShowImage( WNDef1, defineImage1 );
cvShowImage( WNDef2, defineImage2 );
iObnum1 = 0;
iObnum2 = 0;
```

```
MakeObjectNode_auto( WNDef1, WNDef2, pic_DefMaskRed1, pic_DefMaskRed2 );

for(a=0; a<OBJECT_NUMBER; a++){
printf("main %d : %d, %d\n\n", a, g_object[a].center_c1.x, g_object[a].center_c1.y);
printf("main %d : %d, %d\n\n", a, g_object[a].center_c2.x, g_object[a].center_c2.y);
}
bIsPush = TRUE;

}

printf("Input 'q' to quit while movie is running\n");
printf("if it's ok input any key to start\n");
cvWaitKey(0);
//-----------------------------------------------------<<概要 2:おわり>>-----------------------------------------------------------

//すべての定義物体を表示
printf( "***定義***\n" );
//映像１の処理
printf("%s\n",WNDef1);
OutPutObject(pFirstObject1);
//映像２の処理
printf("%s\n",WNDef2);
OutPutObject(pFirstObject2);

cvDestroyWindow( WNDef1 );
cvDestroyWindow( WNDef2 );

PutFromListToFile( cDefObjFile1, pFirstObject1 );
PutFromListToFile( cDefObjFile2, pFirstObject2 );

//------------------------------------------------------ while 文に入る前の定義終わり ----------------------------------------------------------

//  重心データを書き込むファイルを開く

//動画１のファイル
if( (f_Moment1 = fopen(cMmtFile1,"w")) == NULL ){
printf("出力ファイルを開けません\n");
return -1;
}
fprintf( f_Moment1, "time, center.x, center.y\n" );


//動画２のファイル
if( (f_Moment2 = fopen(cMmtFile2,"w")) == NULL ){
printf("出力ファイルを開けません\n");
return -1;
}
fprintf( f_Moment2, "time, center.x, center.y\n" );


// 距離書き込み用ファイルを開く
if(mar_flag == 0){
if( (d_fp1 = fopen(fname1,"w")) == NULL ){
printf("距離用出力ファイル 1 を開けません (maskCountour)\n");
return -1;
}

if( (d_fp2 = fopen(fname2,"w")) == NULL ){
printf("距離用出力ファイル 2 を開けません (maskCountour)\n");
return -1;
}
}else if(mar_flag == 1){ //読み込んで解析バージョン

// 読込用ファイルを開く
if( (tra_fp = fopen(fname_tra,"r")) == NULL ){
printf("state.txt を開けません (before while) \n");
return -1;
}

}else{
printf("error in markov_file_open\n");
}

cvNamedWindow( WNCapt1, CV_WINDOW_AUTOSIZE );
cvNamedWindow( WNCapt2, CV_WINDOW_AUTOSIZE );
cvNamedWindow( WNCon1, CV_WINDOW_AUTOSIZE );
cvNamedWindow( WNCon2, CV_WINDOW_AUTOSIZE );

cvNamedWindow( WNmask1, CV_WINDOW_AUTOSIZE );
cvNamedWindow( WNmask2, CV_WINDOW_AUTOSIZE );



if( (f_Text = fopen(cText,"w")) == NULL ){
printf("\"%s\" ファイルを開けません\n");
return -1;

}else{

printf("*******************************\n");
////////////////////////////////////////////////////////////////////////////////
//////////////////                            ////////////////////////////////////
//////////////////    センサ関係の処理        ////////////////////////////////////
//////////////////                            ////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////////////////

// □　センサデータを n(回)/1(秒) でコマ落としたファイルを用意する。(n_p は DROP_NUMBER) と同じ
// センサ処理一般化後

for(s_count=0; s_count<OBJECT_NUMBER; s_count++){

if(g_file_sensor[s_count].flag == 1){ // ファイルが存在する時だけ
// ファイル名生成（第３引数に出力）（入力用と出力用）
make_file_name(sunspot_text_in, s_count+1, g_file_sensor[s_count].text_in);
make_file_name(sunspot_text_out, s_count+1, g_file_sensor[s_count].text_out);

// コマ落とし
scene_drop_text(DROP_NUMBER, 1, g_file_sensor[s_count].text_in, g_file_sensor[s_count].text_out);

}
}

// □　センサデータのビットをセット
// センサ処理一般化後
for(s_count=0; s_count<OBJECT_NUMBER; s_count++){
g_file_sensor[s_count].n = s_count+1; // 通し番号をセットしておく

if(g_file_sensor[s_count].flag == 1){
// ファイル名作成
make_file_name(sensor_node, s_count+1, g_file_sensor[s_count].text_node);
set_sensor_node(2, SENSOR_INTERVAL, g_file_sensor[s_count].text_out, g_file_sensor[s_count].text_node);

// ファイルを開いておいて、〜.n　をセット
if (fopen_s(&g_file_sensor[s_count].fp_in, g_file_sensor[s_count].text_node,"r")!=0){ //読み取り用ファイル
puts( "ファイルが開けません" );
return 1;
}
}
}

// センサのカウンタを 0 に初期化しておく。

for(s_count = 0; s_count<OBJECT_NUMBER; s_count++){
s_counter[s_count] = 0;
g_flag[s_count] = 1;
g_flag_timer[s_count] = 0;
}
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////

printf("*******************************\n 以上の定義物体と人の行動について言語化します。\n");
printf("\n【言語化】\n");
if((b_or_h_flag == 1)&&(mar_flag == 0)){
printf("data processing : HMM - making_file\n");
}else if((b_or_h_flag == 1)&&(mar_flag == 1)){
printf("data processing : HMM - analyzing\n");
}else if((b_or_h_flag == 0)&&(mar_flag == 0)){
printf("data processing : Bayes Classifier\n");
}else{
printf("something wrong with define!!\n");
}
// ファイルが開けなかったら、表示されない！

// 最初の差分用画像
backImage1 = cvCloneImage(frameImage1);
backImage2 = cvCloneImage(frameImage2);


// フレームのズレ
while(fr_dif > 0){
fr_dif--;
}

while( 1 ){
g_time++;

// 1 枚前のフレームを backImg に代入
backImage1 = cvCloneImage(pre_Image1);
backImage2 = cvCloneImage(pre_Image2);


// 画像が取得出来なかったら (ビデオが終了) 処理を終える

if( (frameImage1 = cvQueryFrame( Capture1 )) == NULL || (frameImage2 = cvQueryFrame( Capture2 )) == NULL ){
printf("End Files...\n");
break;
}

if( captureOn != 0){
if((currentImage1 != NULL) && (currentImage2 != NULL)){
cvReleaseImage( &currentImage1 );
cvReleaseImage( &currentImage2 );
}
currentImage1 = cvCloneImage( frameImage1 );
currentImage2 = cvCloneImage( frameImage2 );

////////////////////////////////////////////////
```

```
// filter_flag != 0 ならいずれかのフィルタ処理を行う。
//////////////////////////////////////////
if(filter_flag!=0){
if(filter_flag==1){ // 平滑化
cvSmooth(currentImage1, currentImage1, CV_BLUR, thr, 0, 0, 0); // フィルタ処理
cvSmooth(currentImage2, currentImage2, CV_BLUR, thr, 0, 0, 0); // フィルタ処理
}else if(filter_flag==2){ // 縦ブレ
blur_len(currentImage1, thr);
blur_len(currentImage2, thr);
}else if(filter_flag==3){ // 横ブレ
blur_side(currentImage1, thr);
blur_side(currentImage2, thr);
}else if(filter_flag==4){ // 解像度
resolution(currentImage1, thr);
resolution(currentImage2, thr);
}
}
}

// 差の計算方法の切り替え
if( differenceMode == GRAY_DIFFERENCE ){
// 成分ごとに評価をする
GrayScaleDifference( currentImage1, backImage1, resultImage1, g_ImageSize1 );
GrayScaleDifference( currentImage2, backImage2, resultImage2, g_ImageSize2 );

}
// ノイズ除去
if( noiseMode == NOISE_MORPHOLOGY ){
cvErode( resultImage1, resultImage1 ); // 膨張
cvDilate( resultImage1, resultImage1 ); // 収縮
cvErode( resultImage2, resultImage2 ); // 膨張
cvDilate( resultImage2, resultImage2 ); // 収縮
}


//  差分結果を用いて原画像をマスク処理
cvSetZero( maskoutImage1 );
cvCopy(currentImage1, maskoutImage1, resultImage1);
cvSetZero( maskoutImage2 );
cvCopy(currentImage2, maskoutImage2, resultImage2);


//膨張処理
cvDilate( maskoutImage1, maskoutImage1 , NULL, 5 );
cvDilate( maskoutImage2, maskoutImage2 , NULL, 5 );
//収縮処理
cvErode( maskoutImage1, maskoutImage1, NULL, 7 );
cvErode( maskoutImage2, maskoutImage2, NULL, 7 );

/*
// 出力確認
if(g_time==120){
cvSaveImage("mask1.jpg", maskoutImage1);
cvSaveImage("cur1.jpg", currentImage1);
cvSaveImage("res1.jpg", resultImage1);
cvSaveImage("mask2.jpg",maskoutImage2);
}*/

//  2 値化し、輪郭線、重心を求める
contourImage1 = cvCloneImage( maskoutImage1 );
momentImage1 = cvCloneImage( maskoutImage1 );
contourImage2 = cvCloneImage( maskoutImage2 );
momentImage2 = cvCloneImage( maskoutImage2 );

// frameImage をコピーして連れていく。(塗りつぶし用)
paintImage1 = cvCloneImage( frameImage1 );
paintImage2 = cvCloneImage( frameImage2 );

paintImage1_mono = cvCloneImage( frameImage1 );
paintImage2_mono = cvCloneImage( frameImage2 );


// 輪郭と重心を描写したものを momentImage へ
if(mar_flag==0){ // 距離書き込み ver
maskContour(1, paintImage1,
     maskoutImage1,
 contourImage1,
 momentImage1,
 500,
 f_Moment1,
 center1,//  重心座標ポインタ
 g_ImageSize1,
 pFirstObject1,
 d_fp1,
 d_fp2);

maskContour(2, paintImage2,
 maskoutImage2,
 contourImage2,
 momentImage2,
 500,
 f_Moment2,
 center2,//  重心座標ポインタ
 g_ImageSize2,
```

```
 pFirstObject2,
 d_fp1,
 d_fp2);

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////          処理判定          ////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// ベイズの場合（後で、if 文の形にする !! ）
if(b_or_h_flag==0){
new_Verbalization2(f_Text);     // センサ込みの判定処理
}else if(b_or_h_flag==1){
if(mar_flag==1){
// マルコフの場合（引数：読込ファイル、書き込みファイル）
new_Verbalization_m(tra_fp, f_Text);
}
}else{
printf("error in b_or_h_flag \n");
}


// 定義物体の重心
for(a=0; a<OBJECT_NUMBER; a++){
cvCircle(currentImage1, g_object[a].center_c1, 3, CV_RGB(0,0,255), 1, 8, 0);
cvCircle(currentImage2, g_object[a].center_c2, 3, CV_RGB(0,0,255), 1, 8, 0);
}

cvShowImage( WNCapt1, currentImage1 ); //  現在の動画像
cvShowImage( WNCon1, momentImage1 ); //  背景差分→重心画像
cvShowImage( WNCapt2, currentImage2 ); //  現在の動画像
cvShowImage( WNCon2, momentImage2 ); //  背景差分→重心画像

// キー入力判定
key = cvWaitKey( 1 );
if( key == 'q' ){
// 'q' キーが押されたらループを抜ける
break;
} else if( key == 'b' ){
// 'b' キーが押されたらその時点での画像を背景画像とする
if( backImage1 != NULL ){
cvReleaseImage( &backImage1 );
}
if( backImage2 != NULL ){
cvReleaseImage( &backImage2 );
}
backImage1 = cvCloneImage( frameImage1 );
backImage2 = cvCloneImage( frameImage2 );
printf( "背景画像更新\n" );
}else if( key == ' ' ){
// スペースキーが押されたら画像取得の ON/OFF を変更する
captureOn = 1 - captureOn;
if( captureOn == CAPTURE_ON ){
printf( "画像取得: ON\n" );
}else if( captureOn == CAPTURE_OFF ){
printf( "画像取得: OFF\n" );
}
}

//   1 枚前のフレームを保持しとく
pre_Image1 = cvCloneImage(currentImage1);
pre_Image2 = cvCloneImage(currentImage2);


// 解放
cvReleaseImage( &currentImage1 );
cvReleaseImage( &currentImage2 );

}   // while(1) 終わり


// 書き込み ver だったら
if((mar_flag == 0)&&(b_or_h_flag == 1)){
// 距離用ファイルを閉じて、
fclose(d_fp1);
fclose(d_fp2);

// ファイルの調整用に読込 ver で開く
// new_set_HMM_state の中でファイルを開いたり閉じたりする。
/* if( (d_fp1 = fopen(fname1,"r")) == NULL ){
printf("距離用読込ファイル 1 を開けません (after while)\n");
return -1;
}

if( (d_fp2 = fopen(fname2,"r")) == NULL ){
printf("距離用読込ファイル 2 を開けません (after while)\n");
return -1;
}
*/

// マルコフ用、距離ファイルの整理と加速度フラグ書き込み
new_set_HMM_state(f_Text);
```

```
}else if((mar_flag == 1)&&(b_or_h_flag == 1)){ // 読込 ver だったら、
fclose(tra_fp);
}


for(s_count=0; s_count<OBJECT_NUMBER; s_count++){
if(g_file_sensor[s_count].flag == 1){
fclose(g_file_sensor[s_count].fp_in);  // センサノード付ファイルを閉じる。
}
}

fclose(f_Text);
} // else 終わり

fclose(f_Moment1);
fclose(f_Moment2);

FreeObject(pFirstObject1);
FreeObject(pFirstObject2);

FreeBayes(pFirstBayes);

cvReleaseImage( &frameImage1 );
cvReleaseImage( &frameImage2 );
cvReleaseImage( &backImage1 );
cvReleaseImage( &backImage2 );
cvReleaseImage( &defineImage1 );
cvReleaseImage( &defineImage2 );
cvReleaseImage( &currentImage1 );
cvReleaseImage( &currentImage2 );
cvReleaseImage( &resultImage1 );
cvReleaseImage( &resultImage2 );

cvReleaseImage( &defMaskRedImage1);  //0503
cvReleaseImage( &defMaskRedImage2);  //0503
cvReleaseImage( &maskoutImage1);  //0503
cvReleaseImage( &maskoutImage2);  //0503
cvReleaseImage( &contourImage1);  //0503
cvReleaseImage( &contourImage2);  //0503
cvReleaseImage( &momentImage1);  //0503
cvReleaseImage( &momentImage2);  //0503

cvReleaseImage( &paintImage1);
cvReleaseImage( &paintImage2);

cvReleaseImage( &paintImage1_mono);
cvReleaseImage( &paintImage2_mono);



cvDestroyWindow( WNCapt1 );
cvDestroyWindow( WNCapt2 );
cvDestroyWindow( WNCon1 );
cvDestroyWindow( WNCon2 );

cvDestroyWindow( WNmask1 );
cvDestroyWindow( WNmask2 );

_CrtDumpMemoryLeaks();  //メモリリーク検知

}else if(fun_flag == 0){
printf("***************** output mouse point mode *****************\n");


// 初期画像を 1 枚キャプチャし背景, 定義用として保存する
frameImage1 = cvQueryFrame( Capture1 );
frameImage2 = cvQueryFrame( Capture2 );

// window 生成
cvNamedWindow( WNDef1, CV_WINDOW_AUTOSIZE );
cvNamedWindow( WNDef2, CV_WINDOW_AUTOSIZE );


cvShowImage(WNDef1, frameImage1);
cvShowImage(WNDef2, frameImage2);

printf("----- Window 1 -----\n");
while(1){
cvSetMouseCallback( WNDef1, Mouse);
key = cvWaitKey(10);
if(key == 's'){
break;
}

}

printf("----- Window 2 -----\n");
while(1){
cvSetMouseCallback( WNDef2, Mouse);
key = cvWaitKey(10);
if(key == 's'){
break;
}
}
```

```
cvDestroyWindow(WNDef1);
cvDestroyWindow(WNDef2);

fclose(fp_for_prog);

} //fun_flag が 0 の時の処理
// 終了する前に一旦止めるため。(数字キーで終了)
printf("finish ?\n");
scanf_s("%d", &a);

return 0;
}
```

B.3 function definition

B.3.1 maskContour

```
////////////////////////////////////////////////////////////////////////////////////////
// 輪郭抽出  と  重心計算 (cvPoint2D32f center : 重心ポインタ)
////////////////////////////////////////////////////////////////////////////////////////
void maskContour(int w_flag, IplImage *paintImage, IplImage *maskoutImage, IplImage *contourImage, IplImage *momentImage, int area_line, FILE *fp,
CvPoint2D32f center, CvSize imageSize, Object *pFirst, FILE *d_fp1, FILE *d_fp2){

//  2 値化画像用 IplImage
IplImage *binaryImage = cvCreateImage(imageSize,IPL_DEPTH_8U,1);
IplImage *grayImage = cvCreateImage(imageSize,IPL_DEPTH_8U,1); // 表示用の 2 値化用

//  グレースケール化
cvCvtColor( maskoutImage, binaryImage, CV_BGR2GRAY);

//  グレースケール化→ 2 値化
cvThreshold( binaryImage, binaryImage, 10, 255, CV_THRESH_BINARY);

// 輪郭抽出用のメモリを確保する
CvMemStorage* storage = cvCreateMemStorage( 0 ); // 抽出された輪郭を保存する領域 (0 だと、自動的にメモリを割り当てる)
CvSeq* find_contour = NULL; // 輪郭へのポインタ


// 2 値画像中の輪郭を見つけ、その数を返す
int find_contour_num = cvFindContours(
binaryImage,// 入力画像 (8 ビットシングルチャンネル)
storage,// 抽出された輪郭を保存する領域
&find_contour,// 一番外側の輪郭へのポインタへのポインタ
sizeof( CvContour ),// シーケンスヘッダのサイズ
CV_RETR_CCOMP,// 抽出モード
CV_CHAIN_APPROX_NONE,// 推定手法
cvPoint( 0, 0 ) // オフセット
);

// 物体の輪郭を赤色で描画する
CvScalar red = CV_RGB( 255, 0, 0 );

//  すべての領域の輪郭線描画 (表示)
cvDrawContours(
contourImage,// 輪郭を描画する画像
find_contour,// 最初の輪郭へのポインタ
red,// 外側輪郭線の色
red,// 内側輪郭線 (穴) の色
2,// 描画される輪郭の最大レベル
1,// 描画される輪郭線の太さ
8,// 線の種類
cvPoint( 0, 0 )  // オフセット
);

//  輪郭線が描画された contourImage を
// 重心描画で使用のため momentImage にコピー
cvCopy(contourImage,momentImage,NULL);


//  輪郭抽出物体の重心計算の定義
double area; // 面積値
double check; // 走査点の位置
CvRect rect; //  輪郭領域の矩形

int measure_dist; //  もっとも近い輪郭までの距離
int i,j; //  x,y 方向の走査
int sum_x, sum_y, sum_b; // もっとも近い輪郭までの距離

CvPoint2D32f pt; // 走査ポインタ

// 塗りつぶし用定義
int k=0;
int l=0;
int m=0;
double check_int;
CvScalar s_color;
s_color.val[0] = 0;
s_color.val[1] = 0;
s_color.val[2] = 0;
```

```
int th=0; // 塗りつぶしの範囲

char sFileName1[64];
char sFileName2[64];

// 人と物体の距離用定義
double line_len[2][OBJECT_NUMBER]; // 人の重心と定義物体の重心の距離
int n_i;

// 初期化
for(n_i=0; n_i<OBJECT_NUMBER; n_i++){
line_len[0][n_i] = 0.0;
line_len[1][n_i] = 0.0;
}

//
//  重心計算
//
while(find_contour !=NULL){

//  輪郭線 (find_contour) で囲まれた物体の面積
area = fabs(cvContourArea(find_contour,CV_WHOLE_SEQ));

//   find_contour で囲まれた矩形の認識
rect = cvBoundingRect(find_contour,0);

//  面積一定以上 (引数 : area_line) について重心を求める
if(area>area_line){
//  走査点の初期化
sum_x =0; //   x 座標
sum_y =0; //   y 座標
sum_b =0; //  座標数 (重心計算の分母となる)
measure_dist =0; //  もっとも近い輪郭線までの距離

//  面積内の走査
//  縦 (y 方向) に走査
for(j=rect.y;j<rect.y+rect.height+1;j++){
pt.y = (float)j;
//  横 (x 方向) に走査
for(i=rect.x;i<rect.x+rect.width+1;i++){
pt.x = (float)i;

//  走査点 pt が輪郭内にある  +1
//     輪郭上にある    0
//     輪郭外にある  -1
check = cvPointPolygonTest(find_contour,pt,measure_dist);

////////////////////////////

//fprintf(fp_for_prog, "g_time : %d, check : %lf\n\n", g_time, check);
// cvSet2D(frameImage, x, y, (0,0,0));
// 関数か何かで一回一回 find_contour を連れていって処理 ??
////////////////////////////


//  輪郭の外に出たとき
if(check<0){
sum_x += i;
sum_y += j;
sum_b += 1;
}
}
}

//  輪郭内の重心を計算
center.x = (float)sum_x/(float)sum_b;
center.y = (float)sum_y/(float)sum_b;

//  求めた重心座標を画像上に描画
cvCircle(momentImage,cvPoint((int)center.x,(int)center.y),3,CV_RGB(0,0,255),1,8,0);


//  ファイルに重心データを書き込む
//   (10 フレーム毎に 1 回)
fprintf(fp,"%d,%d,%d\n",g_time,(int)center.x,(int)center.y);

// マルコフの時
if(b_or_h_flag == 1){

// 人の重心と定義物体の中心の距離 !!
for(n_i=0; n_i<OBJECT_NUMBER; n_i++){
if(w_flag==1){
line_len[0][n_i] = sqrt(double((center.x - g_object[n_i].center_c1.x)*(center.x - g_object[n_i].center_c1.x)) +
double((center.y - g_object[n_i].center_c1.y)*(center.y - g_object[n_i].center_c1.y)));

}else if(w_flag==2){
line_len[1][n_i] = sqrt(double((center.x - g_object[n_i].center_c2.x)*(center.x - g_object[n_i].center_c2.x)) +
double((center.y - g_object[n_i].center_c2.y)*(center.y - g_object[n_i].center_c2.y)));
}

}

// ファイルに書き込み (カメラ 1)
for(n_i=0; n_i<OBJECT_NUMBER; n_i++){
```

```
if(w_flag==1){

// 状態設定（物体毎に）
if(line_len[0][n_i]<=50){
// g_time, ob_number, distance, state
fprintf(d_fp1, "%3d,%2d,%.2f,1\n", g_time, n_i, line_len[0][n_i]);
//printf("%d,%d,%.2f,1\n", g_time, n_i, line_len[0][n_i]);
}else if(line_len[0][n_i]<=150){
fprintf(d_fp1, "%3d,%2d,%.2f,2\n", g_time, n_i, line_len[0][n_i]);
//printf("%d,%d,%.2f,2\n", g_time, n_i, line_len[0][n_i]);
}else{
fprintf(d_fp1, "%3d,%2d,%.2f,3\n", g_time, n_i, line_len[0][n_i]);
//printf("%d,%d,%.2f,3\n", g_time, n_i, line_len[0][n_i]);
}
}

if(w_flag==2){

// 状態設定（物体毎に）
if(line_len[1][n_i]<=50){
// g_time, ob_number, distance, state
fprintf(d_fp2, "%3d,%2d,%.2f,1\n", g_time, n_i, line_len[1][n_i]);
}else if(line_len[1][n_i]<=150){
fprintf(d_fp2, "%3d,%2d,%.2f,2\n", g_time, n_i, line_len[1][n_i]);
}else{
fprintf(d_fp2, "%3d,%2d,%.2f,3\n", g_time, n_i, line_len[1][n_i]);
}
}
}

} //if(b_or_h_flag == 1) 終わり



} // if(area > area_line) 終わり


//for(n_i=0; n_i<OBJECT_NUMBER; n_i++){
//fprintf(fp_for_prog,"\n\ng_time : %d\n", g_time);
//fprintf(fp_for_prog, "ob_number : %d, line_len_1 : %f, line_len_2 : %f\n",n_i, line_len[0][n_i], line_len[1][n_i]);

//  定義域内に重心が含まれているか確認
matching(pFirst,imageSize,center);

//次の輪郭物体へ
find_contour = find_contour->h_next;
}

// 1. フィルタ（アニメ風）
// 引数（src, dst, 輪郭の細かさ（小さいほど), 輪郭の多さ（小さいほど), 同色の領域の細かさ（小さいほど), 同色の領域の大きさ（大きいほど), ノイズ除去レベル）
cv_CartoonFilter(paintImage, paintImage, 11, 5.0, 5, 30.0, 5);

// 2. 二値化_1:
/* cvCvtColor(paintImage, grayImage, CV_BGR2GRAY);
cvThreshold (grayImage, grayImage, 65, 255, CV_THRESH_BINARY);
paintImage = cvCloneImage(grayImage);
*/
// 3. ２値化（大津の手法）
/*cvCvtColor(paintImage, grayImage, CV_BGR2GRAY);
cvSmooth (grayImage, grayImage, CV_GAUSSIAN, 5);
cvThreshold (grayImage, grayImage, 0, 255, CV_THRESH_BINARY | CV_THRESH_OTSU);
paintImage = cvCloneImage(grayImage);
*/

// 3. 二値化_2: 輪郭
/* cvCvtColor(paintImage, grayImage, CV_BGR2GRAY);
cvAdaptiveThreshold (grayImage, grayImage, 255, CV_ADAPTIVE_THRESH_MEAN_C, CV_THRESH_BINARY, 11, 10);
paintImage = cvCloneImage(grayImage);
*/

if(w_flag==1){
cvShowImage("Painted1", paintImage);
sprintf(sFileName1, "../../images/AR_privacy/p_im1_%d.bmp", g_time);
cvSaveImage(sFileName1, paintImage);
//cvSaveImage("p_im1.jpg", paintImage);
}else if(w_flag==2){
cvShowImage("Painted2", paintImage);
sprintf(sFileName2, "../../images/AR_privacy/p_im2_%d.bmp", g_time);
cvSaveImage(sFileName2, paintImage);
//cvSaveImage("p_im2.jpg", paintImage);
}
cvReleaseImage( &binaryImage );


}
```

B.3.2 Mouse1, Mouse2, Mouse

```
////////////////////////////////////////////////////////////////////////////////////////////
```

```
// マウスによる座標指定 （クリックごとに座標値を取得）
// カメラ入力、2 台分
////////////////////////////////////////////////////////////////////////////////////////
void Mouse1( int event, int x, int y, int flags ,void *param = NULL ){
switch(event){

case CV_EVENT_LBUTTONDOWN:

if( g_PointCnt == 0 ){ //  1 点目をクリック
g_Pos[ g_PointCnt ].x = x;
g_Pos[ g_PointCnt ].y = y;
g_PointCnt++;

printf("[%d] : %d\t%d\n", g_PointCnt, x, y);

}else if( g_PointCnt < 4 ){ //  ２～４点目をクリック
g_Pos[ g_PointCnt ].x = x;
g_Pos[ g_PointCnt ].y = y;
//  ポインタした点と、直前に指定した点を線で結ぶ
cvLine( defineImage1, g_Pos[g_PointCnt-1], g_Pos[g_PointCnt], CV_RGB(0, 0, 255), 2, CV_AA , 0 );
if( g_PointCnt == 3 ){
cvLine( defineImage1, g_Pos[ g_PointCnt ], g_Pos[0], CV_RGB(0, 0, 255), 2, CV_AA, 0 );
}
g_PointCnt++;
printf("[%d] : %d\t%d\n", g_PointCnt, x, y);
}

cvShowImage("Define1",defineImage1);
break;

default:
break;
} // switch  終わり

}

////////////////////////////////////////////////////////////////////////////////////////

void Mouse2( int event, int x, int y, int flags ,void *param = NULL ){
switch(event){

case CV_EVENT_LBUTTONDOWN:

if( g_PointCnt == 0 ){ //  1 点目をクリック
g_Pos[ g_PointCnt ].x = x;
g_Pos[ g_PointCnt ].y = y;
g_PointCnt++;

printf("[%d] : %d\t%d\n", g_PointCnt, x, g_ImageSize1.height - y);

}else if( g_PointCnt < 4 ){ //  ２～４点目をクリック
g_Pos[ g_PointCnt ].x = x;
g_Pos[ g_PointCnt ].y = y;
//  ポインタした点と、直前に指定した点を線で結ぶ
cvLine( defineImage2, g_Pos[g_PointCnt-1], g_Pos[g_PointCnt], CV_RGB(0, 0, 255), 2, CV_AA , 0 );
if( g_PointCnt == 3 ){
cvLine( defineImage2, g_Pos[ g_PointCnt ], g_Pos[0], CV_RGB(0, 0, 255), 2, CV_AA, 0 );
}
g_PointCnt++;
printf("[%d] : %d\t%d\n", g_PointCnt, x, g_ImageSize1.height - y);

}
cvShowImage("Define2",defineImage2);
break;


default:
break;
}

}
////////////////////////////////////////////////////////////////////////////////////////
// マウスによる座標指定 （クリックごとに座標値を取得）
// カメラ入力、2 台分
// 繰り返し用
////////////////////////////////////////////////////////////////////////////////////////

void Mouse( int event, int x, int y, int flags ,void *param = NULL){
switch(event){
case CV_EVENT_LBUTTONDOWN:
printf("(%d,%d) \n",x,y);
break;

default:
break;
}
}
```

B.3.3 GrayScaleDifference

```
////////////////////////////////////////////////////////////////////////////////////////
```

```
// 差分画像生成
////////////////////////////////////////////////////////////////////////////////////

void GrayScaleDifference( IplImage *currentImage, IplImage *backgroundImage, IplImage *resultImage, CvSize imageSize ){

// 差分画像用 IplImage
IplImage *differenceImage = cvCreateImage( imageSize, IPL_DEPTH_8U, 3 );

// 現在の背景との差の絶対値を成分ごとに取る
cvAbsDiff( currentImage, backgroundImage, differenceImage );

// 現在の背景との差の絶対値を成分ごとに取る
cvCvtColor( differenceImage,resultImage,CV_BGR2GRAY );

// グレースケールから 2 値に変換する
cvThreshold( resultImage, resultImage, THRESHOLD, THRESHOLD_MAX_VALUE, CV_THRESH_BINARY );

//  RGB 背景差分法の適用
//  画像内を走査
for(int y = 0; y < currentImage->height; y++){
for (int x =0; x < currentImage->width; x++){
//  2 値化画像 (resultImage) から画素値を取得
//  黒 (差分なし) は 0
//  白 (差分あり) は 255  の値を取得
const unsigned char gray = Elemtypetraits(resultImage, x,y,3);

if(gray != 0){ //  差分があった場合
//  現画像の RGB それぞれの画素値を取得
const unsigned char fb = Elemtypetraits(currentImage, x,y,0);
const unsigned char fg = Elemtypetraits(currentImage, x,y,1);
const unsigned char fr = Elemtypetraits(currentImage, x,y,2);

//  背景画像の RGB それぞれの画素値を取得
const unsigned char gb = Elemtypetraits(backgroundImage, x,y,0);
const unsigned char gg = Elemtypetraits(backgroundImage, x,y,1);
const unsigned char gr = Elemtypetraits(backgroundImage, x,y,2);

//  RGB それぞれの差分を求める (絶対値)
const unsigned char b = abs(fb - gb);
const unsigned char g = abs(fg - gg);
const unsigned char r = abs(fr - gr);

//  任意の座標において差分が閾値以下であるとき
//  差分はないとみなして、2 値化画像 (resultImage) の画素値を 0(黒) に置き換える
if(b<SPHERE && g<SPHERE && r<SPHERE){
(resultImage -> imageData + resultImage -> widthStep * y )[x] = 0;
}
}
}
}

// メモリを解放する
cvReleaseImage( &differenceImage );
}
```

B.3.4 OutPutObject

```
////////////////////////////////////////////////////////////////////////////////////

// 定義物体のリストから出力

////////////////////////////////////////////////////////////////////////////////////
void OutPutObject(Object *pFirst){

Object *pMove;

if((pMove = pFirst) == NULL){
printf("定義されていません\n");
}else{
while( pMove != NULL ){
printf( "[%d] : %s\n", pMove->ObNumber, pMove->ObName );
pMove = pMove->pNext;
}
}
}
```

B.3.5 Elemtypetraits

```
////////////////////////////////////////////////////////////////////////////////////

// 画素値を得る
//   引数 :   img   : 対象画像
//          x,y   : 対象座標
//            ch    : 求める種別 (2 値化、R、G、B)

////////////////////////////////////////////////////////////////////////////////////
int Elemtypetraits(IplImage *img, int x, int y, int ch){
//  画素値の定義
unsigned char elem;

//  ch=0～2   RGB の画素値
```

```
//   ch=3      2 値化の画素値
//  それぞれの場合の画素値を求める
if(ch != 3){
elem = (img -> imageData + img -> widthStep * y )[x*3 + ch];
}
else{
//elem = (img -> imageData + img -> widthStep * y )[x];
elem = (img->imageData + img->widthStep * y)[x];
}

//  求めた画素値を返す
return elem;

}
```

B.3.6 FreeObject, FreeBayes

```
////////////////////////////////////////////////////////////////////////////////////////

// 構造体   object と bayse   を解放する関数

////////////////////////////////////////////////////////////////////////////////////////
void FreeObject(Object *pNode){

if(pNode!=NULL){
FreeObject(pNode->pNext);
cvReleaseImage(&pNode->ObImage);
free(pNode);
}
}

void FreeBayes(TempBayes *pNode)
{
if(pNode!=NULL){
FreeBayes(pNode->pNext);
free(pNode);
}
}
```

B.3.7 SetObjectFromFile

```
////////////////////////////////////////////////////////////////////////////////////////

//構造体の領域を確保して、物体定義ファイルから一行ずつ読みとって、格納
//そのファイルに定義されている物体の数を返す

////////////////////////////////////////////////////////////////////////////////////////
int SetObjectFromFile(char *FileName,int flag ){
//変数定義
char str[512];
FILE *fp;
char cTemp[32];
int num = 0;

Object *pNode;
Object *Ret = NULL;

// ファイルを開く
if( (fp = fopen( FileName, "r" )) == NULL ){
printf("定義物体ファイルが存在しません\n");
return num;


}else{
while((fgets(str, sizeof(str), fp)) != NULL ){

//構造体格納部分
if(( pNode = (Object*)malloc(sizeof(Object)) ) != NULL){ // object 用のメモリ確保
memset(pNode, 0, sizeof(Object));

if( sscanf( str, "%[^,],%[^,],%[^,]", &cTemp, pNode->ObName, pNode->ObPic ) == 3 ){
//構造体初期値を設定
pNode->ObNumber = atoi(cTemp);
GetMaskFileName( pNode->ObPic, num+1, flag );//何故か LoadImage で文句言われるので一時的対処

//csv で読み込んだ画像名だとなぜか NG... 文字コードとか？
pNode->ObImage = cvLoadImage(pNode->ObPic, CV_LOAD_IMAGE_ANYDEPTH | CV_LOAD_IMAGE_GRAYSCALE);

pNode->OBcounter = 1;
pNode->OBDefTime = 0;

pNode->pNext = NULL;
pNode->pPrev = NULL;
num++;

if(flag==1){
if(!pFirstObject1){ // 1 個目の物体の定義だったら、
pFirstObject1 = pNode;
}else{ // 1 個目の物体の定義じゃなかったら、
pNode->pPrev = pLastObject1;
pLastObject1->pNext = pNode;
```

```
}
pLastObject1 = pNode;

}else if(flag==2){
if(!pFirstObject2){
pFirstObject2 = pNode;
}else{
pNode->pPrev = pLastObject2;
pLastObject2->pNext = pNode;
}
pLastObject2 = pNode;
}
} // if(fscan) 終わり
}else{
printf("malloc at PutDefinedObject Error...\n");
}
} // while 終わり
fclose(fp);
}
return num;

}
```

B.3.8 MakeObjectNode_auto

```
////////////////////////////////////////////////////////////////////////////////////////////

// マウスで物体を定義する関数

////////////////////////////////////////////////////////////////////////////////////////////
void MakeObjectNode_auto( char *WNDef1, char *WNDef2, char *RedPic1, char *RedPic2 ){

// 変数名
char str[32];
int key;
int flag = 0;
int n_counter;

for(n_counter=0; n_counter<OBJECT_NUMBER; n_counter++){
// if(key == 's'){
//マウスによる座標指定
//----------------------------動画１の処理

//  指定域内を塗りつぶす
g_Pos[0].x = g_pos_new1[n_counter][0].x;
g_Pos[0].y = g_pos_new1[n_counter][0].y;
g_Pos[1].x = g_pos_new1[n_counter][1].x;
g_Pos[1].y = g_pos_new1[n_counter][1].y;
g_Pos[2].x = g_pos_new1[n_counter][2].x;
g_Pos[2].y = g_pos_new1[n_counter][2].y;
g_Pos[3].x = g_pos_new1[n_counter][3].x;
g_Pos[3].y = g_pos_new1[n_counter][3].y;

// 定義物体の重心座標を Object 型に入れておく
// g_bject 構造体の CvPoint に代入
new_get_center(n_counter, 1);
printf("%d : %d, %d\n\n", n_counter, g_object[n_counter].center_c1.x, g_object[n_counter].center_c1.y);

cvFillConvexPoly( defMaskRedImage1, g_Pos, 4, CV_RGB(255,0,0), 8, 0 );   ///////////////////////
cvAddWeighted( backImage1, 0.7, defMaskRedImage1, 0.3, 0, defineImage1); ///////////////////////← !!
cvShowImage( WNDef1, defineImage1 );
iObnum1++;
flag = 1;                                                                 ///////////////////////← !!

MakeNode(ob_NAME[n_counter], iObnum1, flag);
GetObjectMaskImage( defMaskRedImage1, pLastObject1->ObImage, RedPic1, pLastObject1->ObPic, g_ImageSize1, 0, pLastObject1->ObNumber);


//----------------------------動画２の処理
//  指定域内を塗りつぶす

g_Pos[0].x = g_pos_new2[n_counter][0].x;
g_Pos[0].y = g_pos_new2[n_counter][0].y;
g_Pos[1].x = g_pos_new2[n_counter][1].x;
g_Pos[1].y = g_pos_new2[n_counter][1].y;
g_Pos[2].x = g_pos_new2[n_counter][2].x;
g_Pos[2].y = g_pos_new2[n_counter][2].y;
g_Pos[3].x = g_pos_new2[n_counter][3].x;
g_Pos[3].y = g_pos_new2[n_counter][3].y;

// 定義物体の重心座標を Object 型に入れておく
new_get_center(n_counter, 2);
printf("%d : %d, %d\n\n", n_counter, g_object[n_counter].center_c2.x, g_object[n_counter].center_c2.y);

cvFillConvexPoly( defMaskRedImage2, g_Pos, 4, CV_RGB(255,0,0), 8, 0 );
cvAddWeighted( backImage2, 0.7, defMaskRedImage2, 0.3, 0, defineImage2);
cvShowImage( WNDef2, defineImage2 );
iObnum2++;
flag = 2;

MakeNode(ob_NAME[n_counter], iObnum2, flag);
GetObjectMaskImage( defMaskRedImage2, pLastObject2->ObImage, RedPic2, pLastObject2->ObPic, g_ImageSize2, 1, pLastObject2->ObNumber);
```

```
//cvSaveImage( RedPic2, defMaskRedImage2 );
}
}
```

B.3.9 new_get_center

```
////////////////////////////////////////////////////////////////////////////////////////////

// 定義物体の中心

////////////////////////////////////////////////////////////////////////////////////////////
void new_get_center(int n_i, int c_flag){
CvMemStorage *storage = cvCreateMemStorage(0);
CvSeq *seq = cvCreateSeq( CV_SEQ_POLYGON, sizeof( CvSeq), sizeof( CvPoint), storage);

CvPoint2D32f pc;
    CvMoments moment;

int i;

// 4 点を格納
for(i=0; i<4; i++){
cvSeqPush( seq, &g_Pos[i]);
}

cvMoments( seq, &moment);
pc = cvPoint2D32f( moment.m10/moment.m00, moment.m01/moment.m00);

// どっちのカメラか
if(c_flag == 1){
g_object[n_i].center_c1.x = pc.x;
g_object[n_i].center_c1.y = pc.y;
//printf("%d : %d, %d\n\n", n_i, g_object[n_i].center_c1.x, g_object[n_i].center_c1.x);
}else if(c_flag == 2){
g_object[n_i].center_c2.x = pc.x;
g_object[n_i].center_c2.y = pc.y;
//printf("%d : %d, %d\n\n", n_i, g_object[n_i].center_c2.x, g_object[n_i].center_c2.x);
}
return;
}
```

B.3.10 MakeNode

```
////////////////////////////////////////////////////////////////////////////////////////////
// 構造体格納関数
////////////////////////////////////////////////////////////////////////////////////////////

void MakeNode( char *ObName, int iObnum, int flag ){

Object *pNode;

//構造体格納部分
if(( pNode = (Object*)malloc(sizeof(Object)) ) != NULL){

//構造体初期値を設定
strcpy(pNode->ObName, ObName);
pNode->ObNumber = iObnum;
GetMaskFileName( pNode->ObPic, iObnum, flag );
pNode->ObImage = cvLoadImage( pNode->ObPic, CV_LOAD_IMAGE_ANYDEPTH | CV_LOAD_IMAGE_GRAYSCALE);

pNode->OBcounter = 1;
pNode->OBDefTime = 0;
pNode->pNext = NULL;
pNode->pPrev = NULL;

if(flag == 1){
if(!pFirstObject1){
pFirstObject1 = pNode;
}else{
pNode->pPrev = pLastObject1;
pLastObject1->pNext = pNode;
}
pLastObject1 = pNode;

}else if(flag == 2){
if(!pFirstObject2){
pFirstObject2 = pNode;
}else{
pNode->pPrev = pLastObject2;
pLastObject2->pNext = pNode;
}
pLastObject2 = pNode;
}
}else{
printf("malloc at PutDefinedObject Error...\n");
}
}
```

B.3.11 PutFromListToFile

```
////////////////////////////////////////////////////////////////////////////////////////////

// 定義物体のリスト構造からファイルに書き出す

////////////////////////////////////////////////////////////////////////////////////////////

void PutFromListToFile( char *FileName, Object *pFirst ){

FILE *fp;
Object *pMove;

if( (fp = fopen( FileName, "w" )) == NULL ){
printf("定義物体ファイルが存在しません\n");

}else{
pMove = pFirst;

while(pMove != NULL){
fprintf( fp, "%d,%s,%s\n", pMove->ObNumber, pMove->ObName, pMove->ObPic );
pMove = pMove->pNext;
}
fclose(fp);
}

FreeObject( pMove );  //0501
}
```

B.3.12 matching

```
////////////////////////////////////////////////////////////////////////////////////////////

// 定義域内に重心が含まれているか

////////////////////////////////////////////////////////////////////////////////////////////

void matching(Object *pFirst, CvSize imageSize,CvPoint2D32f center){

//  それぞれの定義物のカウンターの確認
Object *pMove;

//  画素値が白（定義内）にあるとき（マウス指定の形に従う）
pMove = pFirst;

while(pMove != NULL){
//  重心座標の 2 値化画素値を得る
int maskno = 0;

maskno = Elemtypetraits(pMove->ObImage, (int)center.x, (int)center.y, 3);

if(maskno > 50 ){
//  前回 counter を増やしたときとの時間の開きを考慮
if(pMove->OBcounter == 1 || (g_time - pMove->OBDefTime)< 10){
//  動画時間を構造体の時間に入れる
pMove->OBDefTime = g_time; //deftime というよりは、ひらきが存在するかの検査ってこと。
}
else{
//  開きがあれば counter を初期化
pMove->OBcounter = 0;
}
//  カウンタを増やす
pMove->OBcounter++;
}
pMove = pMove->pNext;
}
}
```

B.3.13 GetMaskFileName

```
////////////////////////////////////////////////////////////////////////////////////////////

// 定義物体のマスク画像のファイル名を取得し cImageName に格納

////////////////////////////////////////////////////////////////////////////////////////////

void GetMaskFileName( char *cImageName, int iObnum, int flag ){

char str[32] = "defmaskbin";
char cNum[8] = ""; //８桁まで定義物体画像の保存

sprintf_s( cNum, "%d", iObnum );
switch( flag ){
case 1:
strcat_s( str, "1_" );
break;
case 2:
strcat_s( str, "2_" );
break;
default:
printf("GetMaskFileName Error...\n");
```

```
break;
}

strcat_s( str, cNum );
strcat_s( str, ".jpg" );

strcpy(cImageName,str);

}
```

B.3.14 GetObjectMaskImage

```
////////////////////////////////////////////////////////////////////////////////////////////

// 指定された定義物体の画像を膨張処理し、保存、格納する。

////////////////////////////////////////////////////////////////////////////////////////////

void GetObjectMaskImage( IplImage *defMaskRedImage, IplImage *ObImage, char *cRed, char *str, CvSize g_ImageSize, int m_n, int ob_n){

IplImage *maskImage;
IplImage *saveImage;

saveImage = cvCreateImage( g_ImageSize, IPL_DEPTH_8U, 1 );
maskImage = cvCreateImage( g_ImageSize, IPL_DEPTH_8U, 3 );

cvSetZero( maskImage );
if(m_n == 0){
g_Pos[0].x = g_pos_new1[ob_n-1][0].x;
g_Pos[0].y = g_pos_new1[ob_n-1][0].y;
g_Pos[1].x = g_pos_new1[ob_n-1][1].x;
g_Pos[1].y = g_pos_new1[ob_n-1][1].y;
g_Pos[2].x = g_pos_new1[ob_n-1][2].x;
g_Pos[2].y = g_pos_new1[ob_n-1][2].y;
g_Pos[3].x = g_pos_new1[ob_n-1][3].x;
g_Pos[3].y = g_pos_new1[ob_n-1][3].y;

cvFillConvexPoly( maskImage, g_Pos, 4, CV_RGB(255,0,0), 8, 0 );
cvFillConvexPoly( defMaskRedImage, g_Pos, 4, CV_RGB(255,0,0), 8, 0 ); //保存用の IplImage に今回定義された物体領域を赤で塗りつぶす。
}else if(m_n == 1){

g_Pos[0].x = g_pos_new2[ob_n-1][0].x;
g_Pos[0].y = g_pos_new2[ob_n-1][0].y;
g_Pos[1].x = g_pos_new2[ob_n-1][1].x;
g_Pos[1].y = g_pos_new2[ob_n-1][1].y;
g_Pos[2].x = g_pos_new2[ob_n-1][2].x;
g_Pos[2].y = g_pos_new2[ob_n-1][2].y;
g_Pos[3].x = g_pos_new2[ob_n-1][3].x;
g_Pos[3].y = g_pos_new2[ob_n-1][3].y;

cvFillConvexPoly( maskImage, g_Pos, 4, CV_RGB(255,0,0), 8, 0 );
cvFillConvexPoly( defMaskRedImage, g_Pos, 4, CV_RGB(255,0,0), 8, 0 );
}else{
printf("error in GetObjectMaskImage\n");

}
//------この時点で maskImage は赤の物体と黒の背景の画像になっている。
//  グレースケール化
cvCvtColor( maskImage ,saveImage, CV_BGR2GRAY );
//  グレースケール化→ 2 値化
cvThreshold( saveImage, saveImage, 10, 255, CV_THRESH_BINARY);
//  膨張処理 (領域の拡張)
cvDilate( saveImage, saveImage, cvCreateStructuringElementEx(3,3,1,1,CV_SHAPE_ELLIPSE,NULL), 7);

cvSaveImage( str, saveImage );
cvSaveImage( cRed, defMaskRedImage );
ObImage = cvCloneImage(saveImage);

cvReleaseImage( &saveImage );
cvReleaseImage( &maskImage );
}
```

B.3.15 new_Verbalization2, new_Verbalization_m

```
////////////////////////////////////////////////////////////////////////////////////////////
// ファイルは、書き込み用、センサ 1 、センサ 2
//  ベイズ適用後。
////////////////////////////////////////////////////////////////////////////////////////////

void new_Verbalization2(FILE *fp){
Object *pMove1,*pMove2;
TempBayes *pTemp;

pMove1 = pFirstObject1;

int s_bit[OBJECT_NUMBER]; // センサのビット

int r3_node[OBJECT_NUMBER];     // 物体の個数だけ持つ。
int e_n;
int i;  // カウント用
```

```
// 初期化
for(s_count=0; s_count<OBJECT_NUMBER; s_count++){
s_bit[s_count] = 0;
r3_node[s_count] = 0;
}

// 一つずつの物体に対して、センサからノードを一行ずつ読み込む。
////////////////////////////////////////////////////////////////////////////////
//////////////////        センサのノード設定        ////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//    センサのフラグがどれか一つでも０以上になったら、１プラスする。
//    （ｘ、ｙ、ｚごとにフラグが別れているけど、有効利用は後で考える。）
//    センサのフラグの合計が（センサ用の）閾値を超えたら、ノードを返す.
//    ただし、０が２個以上続いたらカウンタを初期化する。
////////////////////////////////////////////////////////////////////////////////
// センサノードのカウンタがセンサの条件の閾値を超えているか（ドアという）じゃなくて一つだけ
int time_pre, time_aft, s_flag;
float dif_x, dif_y, dif_z;

int flag_time;

////////////////////////////////////////////////////////////////////////////////
//////////////////        コマ落としの時に捨てるデータを入れとく配列        //////////
////////////////////////////////////////////////////////////////////////////////
// 捨てる数を入れとく配列（コマ数は１０と仮定）
int thrown[10];

//  加速度のコマ落とし（固定）////////////////////////////////////////////////////////////
if(DROP_NUMBER==10){
thrown[0] = -1;

e_n = 1;
}else if(DROP_NUMBER == 9){
thrown[0] = 0;

e_n = 1;
}else if(DROP_NUMBER == 8){
thrown[0] = 0;
thrown[1] = 1;

e_n = 2;
}else if(DROP_NUMBER == 7){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;

e_n = 3;
}else if(DROP_NUMBER == 6){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;

e_n = 4;
}else if(DROP_NUMBER == 5){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;

e_n = 5;
}else if(DROP_NUMBER == 4){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;

e_n = 6;
}else if(DROP_NUMBER == 3){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;
thrown[6] = 6;

e_n = 7;
}else if(DROP_NUMBER == 2){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;
thrown[6] = 6;
thrown[7] = 7;

e_n = 8;
}else if(DROP_NUMBER == 1){
```

```
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;
thrown[6] = 6;
thrown[7] = 7;
thrown[8] = 8;

e_n = 9;
}else{
printf("error in new_verbalization2 (thrown)\n");
}
////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////
//////////////////                一行ずつ読み込み                          //////////
//////////////////          タイマー、カウンタ、出力確認処理              //////////
////////////////////////////////////////////////////////////////////////////////////

// g_time を DROP_NUMBER で割った余りが、thrown 配列の中になければ以下を実行する！
if(check_n((g_time % 10), thrown, e_n) == 0){

for(s_count=0; s_count<OBJECT_NUMBER; s_count++){ // 物体の数だけ
if((g_flag[s_count] == 1) && (g_file_sensor[s_count].flag == 1)){

// 一行読み込み
fscanf_s(g_file_sensor[s_count].fp_in,
"%3d   %4d-%4d   %f,  %f,  %f   %d", &flag_time, &time_pre, &time_aft, &dif_x, &dif_y, &dif_z, &s_flag);

// ノード処理////////////////////////////////////////////
if(s_flag != 0){ // s_flag が０だと割れないから
s_counter[s_count] = s_counter[s_count] + ((s_flag)/(s_flag));
}else{    // s_flag1 が１じゃなかったら、０にする
s_counter[s_count] = 0;
}
// 閾値を超えたらビットを立てる
if(s_counter[s_count] >= THRESHOLD_SENSOR){
s_bit[s_count] = 1;
s_counter[s_count] = 0;
}

// 値が飛んでいた時のタイマー設定
g_flag_timer[s_count] = time_aft - time_pre;
if(g_flag_timer[s_count]<0){ // 秒の変わり目だったら、
g_flag_timer[s_count] = time_aft; // 後半の数字を入れれば OK
}


// 出力確認処理
fprintf(fp, "*****************************************************************************************************\n");
for(i=0; i<e_n; i++){
fprintf(fp, "thrown[%d] : %d\n", i, thrown[i]);
}

fprintf(fp, "OBJECT_NUMBER : %d\n", OBJECT_NUMBER);
fprintf(fp, "g_time : %d, g_time と 10 : %d, 要素数：%d\n", g_time, g_time%10, e_n);
fprintf(fp, "g_flag_timer[%d] : %d, time_pre : %d, time_aft : %d, g_flag[%d] : %d\n", s_count, g_flag_timer[s_count],
time_pre, time_aft, s_count, g_flag[s_count]);
fprintf(fp, "センサ [%d] の読込後：%3d   %3d-%3d   %f,  %f,  %f   %d\n", s_count+1, flag_time, time_pre, time_aft,
dif_x, dif_y, dif_z, s_flag);
fprintf(fp, "s_counter[%d]：%d   s_bit[%d]：%d\n", s_count, s_counter[s_count], s_count, s_bit[s_count]);
fprintf(fp, "*****************************************************************************************************\n");

} // if(g_flag == ...) 終わり

// タイマーが切れたらフラグ ON (1)
if(g_flag_timer[s_count] == 1){
g_flag[s_count] = 1;
}else{
g_flag[s_count] = 0;
}
g_flag_timer[s_count]--;

} // for(s_count...) 終わり

} // if(check...) 終わり

// これは、for 文の最後でセットする
for(s_count=0; s_count<OBJECT_NUMBER; s_count++){
r3_node[s_count] = s_bit[s_count];
}
////////////////////////////////////////////////////////////////////////////////////
//////////////////        出力処理                ////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////


while( pMove1 != NULL ){
pMove2 = pFirstObject2;
while(pMove2!= NULL){
if(strcmp(pMove1->ObName,pMove2->ObName) == 0){
pTemp = pFirstBayes;
while( pTemp != NULL ){
```

```
if( strcmp(pTemp->ObName,pMove1->ObName) == 0 ){
fprintf(fp, "\n\n*******\n");
fprintf(fp, "c1_g_time : %d\n ob_number : %d\n pMove1->OBounter : %d\n", g_time, pMove1->ObNumber, pMove1->OBcounter);
fprintf(fp, "c2_g_time : %d\n ob_number : %d\n pMove2->OBounter : %d\n", g_time, pMove2->ObNumber, pMove2->OBcounter);

if( new_IsOutPut2(pMove1,pMove2,pTemp,fp, r3_node) != FALSE ){  // 新しい出力の形。

fprintf(fp,"\n\n 時刻：%d\t 人が %s %s %s。\n\n\n",g_time, pMove1->ObName, pTemp->PostPositional, pTemp->Verb);
fprintf(fp_for_prog,"\n\n 時刻：%d\t 人が %s %s %s。\n\n\n",g_time, pMove1->ObName, pTemp->PostPositional, pTemp->Verb);


//連続して表示してしまうなら"*"にする
if(pre_put == pMove1->ObNumber){
printf("*\n");
}else{ // 前と違う言語化なら
printf("\n\n 時刻：%d\t 人が %s %s %s。\n\n",g_time, pMove1->ObName, pTemp->PostPositional, pTemp->Verb);
}
pre_put = pMove1->ObNumber; // 今出力した物体番号を保持

pMove1->OBcounter = 0;
pMove2->OBcounter = 0;
// ここでセンサのノードも0に初期化する。
break;
}
}
pTemp = pTemp->pNext;
}
break;
}
pMove2 = pMove2->pNext;

}
pMove1 = pMove1->pNext;
}
}

////////////////////////////////////////////////////////////////////////////////////////////

// マルコフバージョン
// state.txt から状態を物体数ずつ読み込んで、

////////////////////////////////////////////////////////////////////////////////////////////

// 引数は（読込用距離ファイル、書き出し用結果ファイル）
void new_Verbalization_m(FILE *tra_fp, FILE *fp){
Object *pMove1,*pMove2;
TempBayes *pTemp;

pMove1 = pFirstObject1;

int e_n;
int i;  // カウント用

int tra_time;
float tra_dist;
int st_c[OBJECT_NUMBER];
int tra_a[OBJECT_NUMBER];
int tra_c[OBJECT_NUMBER];
int rate[OBJECT_NUMBER]; // どれくらい近づいてるか

int output_flag[OBJECT_NUMBER]; // 言語化出力する場合のフラグ

////////////////////////////////////////////////////////////////////////
//////////////////        state.txt から遷移を読込              //////////
////////////////////////////////////////////////////////////////////////
// ここで使ってる pre_変数はグローバル
// この関数に入る前に、初期値が必要 (main の while 前で要設定 !! )
for(i = 0; i<OBJECT_NUMBER; i++){
fscanf_s(tra_fp, "%3d,%f,%1d,%d,%1d,%d", &tra_time, &tra_dist, &st_c[i], &tra_a[i], &tra_c[i], &rate[i]);
}

// とりあえず、4 が 2 回以上続いたら OK
for(i = 0; i<OBJECT_NUMBER; i++){
output_flag[i] = 0;
}

for(i = 0; i<OBJECT_NUMBER; i++){
if(g_file_sensor[i].flag == 1){ // SunSPOT 付きなら tra_a も考慮
if((tra_c[i] == 4)&&(tra_a[i]>0)&&(rate[i] > THRESHOLD_MARKOV)){
output_flag[i] = 1;
printf("g_time : %d ob : %d\n", g_time, i+1);
fprintf(fp, "verbalie g_time : %d ob : %d\n", g_time, i+1);
}
}else{
if((tra_c[i] == 4)&&(rate[i] > THRESHOLD_MARKOV)){
output_flag[i] = 1;
printf("g_time : %d ob : %d\n", g_time, i+1);
fprintf(fp, "verbalie g_time : %d ob : %d\n", g_time, i+1);
}
}
}
////////////////////////////////////////////////////////////////////////////////
//////////////////        出力処理                 ////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
```

```
// 一つ前を保持
for(i = 0; i<OBJECT_NUMBER; i++){
pre_tra_c[i] = tra_c[i];
pre_tra_a[i] = tra_a[i];
}
}
```

B.3.16 new_SetBayesFromFile2

```
////////////////////////////////////////////////////////////////////////////////////////////

//   CPT からベイズの定理で計算して、最大値、出力するかどうかを保持してあるファイル。
//   FileName は読み込み用、FileName2 は書き込み用

////////////////////////////////////////////////////////////////////////////////////////////

void new_SetBayesFromFile2( char *FileName, char *FileName2){

//変数定義
char str[512];
FILE *fp;
FILE *fp2;  // 出力用
FILE *fp_memo;  // 動作確認用
TempBayes *pNode;
TempBayes *Ret = NULL;

int i, j, k, l;  // calc_P 呼び出し時のカウント用
float R_A[3][OBJECT_NUMBER];  // R が 3 * A が 3 (R=1 の値)
char temp[8];


//  定義が保存されているファイルを呼び出す
if( (fp = fopen( FileName, "r" )) == NULL ){
printf("定義物体ファイルが存在しません\n");
}else if(fopen_s(&fp_memo, file_memo, "w") != 0){
printf("file_memo open error!!\n");

}else{
//  ファイルから 1 行呼び込む
fgets( str,sizeof(str), fp ); // ←この 1 行は最初の一行。名前のやつ。だから、捨てちゃう。
fprintf(fp_memo,"%s\n", str);
//構造体格納部分
for(k=0; k<OBJECT_NUMBER; k++){ // 物体の個数分だけ（行数）

if(( pNode = (TempBayes*)malloc(sizeof(TempBayes)) ) != NULL){
memset(pNode, 0, sizeof(TempBayes));

fgets( str,sizeof(str), fp );
fprintf(fp_memo,"%s\n", str);

if(sscanf(str,"%[^,],%[^,],%[^,],%[^,],%[^,],%s",
pNode->ObNumber, pNode->ObName, pNode->Verb, pNode->Conjugation, pNode->PostPositional, &str ) == 6 ){
fprintf(fp_memo,"pNode->ObNumber : %s\t pNode->PostPositional:%s\t pNode->Verb:%s\t pNode->Conjugation:%s\n\n",
pNode->ObNumber, pNode->PostPositional, pNode->Verb, pNode->Conjugation);
strcpy(ob_NAME[atoi(pNode->ObNumber)-1], pNode->ObName);

for(i=0; i<3; i++){
for(j=0; j<OBJECT_NUMBER; j++){
sscanf(str,"%[^,],%s", &temp, &str);
R_A[i][j] = atof(temp);
fprintf(fp_memo, "R_A[%d][%d] : %f\n", i, j, R_A[i][j]);
}
}

if(!pFirstBayes){ // pFirstBayes が空なら（リストの先頭なら）
pFirstBayes = pNode;
}else{
pNode->pPrev = pLastBayes; // リストの途中なら
pLastBayes->pNext = pNode;
}
pLastBayes = pNode;
}
// 一行分読んだら、処理
calc_p(R_A, &pNode->a_1, k, fp_memo);
fprintf(fp_memo, "[k=%d] : calc_p was called\n", k);
fprintf(fp_memo, "outputbit(a_1) : %d, %d, %d, %d, %d, %d, %d, %d\n", pNode->a_1.R_000, pNode->a_1.R_001,
pNode->a_1.R_010, pNode->a_1.R_011, pNode->a_1.R_100, pNode->a_1.R_101, pNode->a_1.R_110, pNode->a_1.R_111);
}else{
printf("malloc at SetBayesFromFile Error...\n");
}

} //読み込みの繰り返し終わり

fclose(fp);
}

if( (fp2 = fopen( FileName2, "w" )) == NULL ){
printf("定義物体ファイルが存在しません\n");
}else{
fprintf(fp2,"%s", str);
```

```
}
fclose(fp2);
fclose(fp_memo);
}
```

B.3.17 new_IsOutPut2

```
////////////////////////////////////////////////////////////////////////////////////////

// 出力処理判定

////////////////////////////////////////////////////////////////////////////////////////

BOOL new_IsOutPut2( Object *pObject1, Object *pObject2, TempBayes *pMove, FILE *fp, int *Node3){
BOOL bRet = FALSE;
//TempBayes *pMove;
int Node1 = 0;
int Node2 = 0;
int i; // カウント用

FILE *fp2;

if ((fopen_s(&fp2, res_node, "a")) != 0) {
printf("file open error!!\n");
return 1;
}
//この時点で、pObject1,pObject2,pMove の ObName がおなじ（同じ定義物体について指している）

//カウンタが閾値を超えていた場合、D1 = 1
if( pObject1->OBcounter >= THRESHOLD_AVI ){
Node1 = 1;
}
//カウンタが閾値を超えていた場合、D2 = 1
if( pObject2->OBcounter >= THRESHOLD_AVI ){

Node2 = 1;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////   動画データのコマ落とし処理用     ////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////

int thrown[10]; // movie 用（捨てるコマを保持）
int e_n = 0;

//  動画のコマ落とし（固定）//////////////////////////////////////////////////////////////
if(MOVIE_DROP==10){
thrown[0] = -1;

e_n = 1;
}else if(MOVIE_DROP == 9){
thrown[0] = 0;

e_n = 1;
}else if(MOVIE_DROP == 8){
thrown[0] = 0;
thrown[1] = 1;

e_n = 2;
}else if(MOVIE_DROP == 7){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;

e_n = 3;
}else if(MOVIE_DROP == 6){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;

e_n = 4;
}else if(MOVIE_DROP == 5){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;

e_n = 5;
}else if(MOVIE_DROP == 4){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;

e_n = 6;
}else if(MOVIE_DROP == 3){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
```

```
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;
thrown[6] = 6;

e_n = 7;
}else if(MOVIE_DROP == 2){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;
thrown[6] = 6;
thrown[7] = 7;

e_n = 8;
}else if(MOVIE_DROP == 1){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;
thrown[6] = 6;
thrown[7] = 7;
thrown[8] = 8;

e_n = 9;
}else{
printf("error in new_IsOutPut (thrown)\n");
}
////////////////////////////////////////////////////////////////////////////

if(check_n((g_time % 10), thrown, e_n) == 1){ // もし、今のコマが thrown に含まれていれば、カメラノード 2 つを 0 にする。
Node1 = 0;
Node2 = 0;
}

fprintf(fp,"\n\n in new_IsOutPut \n\n g_time : %d \n\n R3_node : %d   Ob_Neme : %s  Ob_number1 : %d   Ob_number2 : %d\n",
g_time, Node3[atoi(pMove->ObNumber)-1], pMove->ObName, pObject1->ObNumber, pObject2->ObNumber);
fprintf(fp, "*********************************************************************************************\n");
for(i=0; i<e_n; i++){
fprintf(fp, "thrown[%d] : %d\n", i, thrown[i]);
}
fprintf(fp, "*********************************************************************************************\n");

////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////   ここから RRR の場合分け     ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////
if( Node1 == 0 && Node2 == 0 ){

if(Node3[atoi(pMove->ObNumber)-1] == 0){                //////    000    ////////////////////////////////////////////////////
if(pMove->a_1.R_000 == 1){
bRet = TRUE;
}
fprintf(fp,"g_time : %d \n %s\n", g_time, "000");

}else if(Node3[atoi(pMove->ObNumber)-1] == 1){ //////    001    ////////////////////////////////////////////////////
if(pMove->a_1.R_001 == 1){
bRet = TRUE;
}
fprintf(fp,"g_time : %d \n %s\n", g_time, "001");
}

}else if( Node1 == 0 && Node2 == 1 ){

if(Node3[atoi(pMove->ObNumber)-1] == 0){ //////    010    ////////////////////////////////////////////////////
if(pMove->a_1.R_010 == 1){
bRet = TRUE;
}
fprintf(fp,"g_time : %d \n %s\n", g_time, "010");

}else if(Node3[atoi(pMove->ObNumber)-1] == 1){ //////    011    ////////////////////////////////////////////////////
if(pMove->a_1.R_011 == 1){
bRet = TRUE;
}
fprintf(fp,"g_time : %d \n %s\n", g_time, "011");
}

}else if( Node1 == 1 && Node2 == 0 ){

if(Node3[atoi(pMove->ObNumber)-1] == 0){ //////    100    ////////////////////////////////////////////////////
if(pMove->a_1.R_100 == 1){
bRet = TRUE;
}
fprintf(fp,"g_time : %d \n %s\n", g_time, "100");

}else if(Node3[atoi(pMove->ObNumber)-1] == 1){ //////    101    ////////////////////////////////////////////////////
if(pMove->a_1.R_101 == 1){
bRet = TRUE;
}
fprintf(fp,"g_time : %d \n %s\n", g_time, "101");
}
```

```
}else if( Node1 == 1 && Node2 == 1 ){

if(Node3[atoi(pMove->ObNumber)-1] == 0){ //////    110    ////////////////////////////////////////////////////
if(pMove->a_1.R_110 == 1){
bRet = TRUE;
}
fprintf(fp,"g_time : %d \n %s\n", g_time, "110");
}else if(Node3[atoi(pMove->ObNumber)-1] == 1){ //////    111    ////////////////////////////////////////////////////
if(pMove->a_1.R_111 == 1){
bRet = TRUE;
}
fprintf(fp,"g_time : %d \n %s\n", g_time, "111");
}
}else{
printf("IsOutPut Error.....\n");
}

fclose(fp2);
return bRet;
}
```

B.3.18 calc_p

```
////////////////////////////////////////////////////////////////////////////////////////

// CPT 計算用

////////////////////////////////////////////////////////////////////////////////////////
float calc_p(float R_A[][OBJECT_NUMBER], OutputBit *outputbit, int ob_n, FILE *fp_memo){

int i, j;  // カウント用
float prob;
float pre_prob = 0.0;
float max_prob = 0.0;  // 現時点の最大値
float main_prob; // 大小比べに使う。

fprintf(fp_memo, "物体番号 : %d\n", ob_n+1);

// 000 ////////////////////////////////////////////////////////////////////
fprintf(fp_memo, "**********000*****\n");
for(j=0; j<OBJECT_NUMBER; j++){   // 物体 A の数だけ
prob = (1.0 - R_A[0][j]) * (1.0 - R_A[1][j]) * (1.0 - R_A[2][j]);
fprintf(fp_memo, "A[%d] : %f\n", j+1, prob);
// max かどうか確認
if(prob > max_prob){  // max_prob の初期値は 0
max_prob = prob;
}
if(j==ob_n){  // 物体番号が一致したら main_prob
main_prob = prob;
}
pre_prob = prob;
}
if((main_prob == max_prob)&&(main_prob != 0.0)){  // main_prob が最大値だった時だけ、出力 bit=1
outputbit->R_000 = 1;
}
max_prob = 0.0;


// 001 ////////////////////////////////////////////////////////////////////
fprintf(fp_memo, "**********001*****\n");
for(j=0; j<OBJECT_NUMBER; j++){   // 物体 A の数だけ
prob = (1.0 - R_A[0][j]) * (1.0 - R_A[1][j]) * R_A[2][j];
fprintf(fp_memo, "A[%d] : %f\n", j+1, prob);
// max かどうか確認
if(prob > max_prob){  // max_prob の初期値は 0
max_prob = prob;
}
if(j==ob_n){  // 物体番号が一致したら main_prob
main_prob = prob;
}
pre_prob = prob;
}
if((main_prob == max_prob)&&(main_prob != 0.0)){  // main_prob が最大値だった時だけ、出力 bit=1
outputbit->R_001 = 1;
}
max_prob = 0.0;


// 010 ////////////////////////////////////////////////////////////////////
fprintf(fp_memo, "**********010*****\n");
for(j=0; j<OBJECT_NUMBER; j++){   // 物体 A の数だけ
prob = (1.0 - R_A[0][j]) * R_A[1][j] * (1.0 - R_A[2][j]);
fprintf(fp_memo, "A[%d] : %f\n", j+1, prob);
// max かどうか確認
if(prob > max_prob){  // max_prob の初期値は 0
max_prob = prob;
}
if(j==ob_n){  // 物体番号が一致したら main_prob
main_prob = prob;
}
pre_prob = prob;
```

```
}
if((main_prob == max_prob)&&(main_prob != 0.0)){  // main_prob が最大値だった時だけ、出力 bit=1
outputbit->R_010 = 1;
}
max_prob = 0.0;


// 011 ////////////////////////////////////////////////////////////////////////
fprintf(fp_memo, "**********011*****\n");
for(j=0; j<OBJECT_NUMBER; j++){   // 物体 A の数だけ
prob = (1.0 - R_A[0][j]) * R_A[1][j] * R_A[2][j];
fprintf(fp_memo, "A[%d] : %f\n", j+1, prob);
// max かどうか確認
if(prob > max_prob){  // max_prob の初期値は 0
max_prob = prob;
}
if(j==ob_n){  // 物体番号が一致したら main_prob
main_prob = prob;
}
pre_prob = prob;

if((main_prob == max_prob)&&(main_prob != 0.0)){  // main_prob が最大値だった時だけ、出力 bit=1
outputbit->R_011 = 1;
}
max_prob = 0.0;


// 100 ////////////////////////////////////////////////////////////////////////
fprintf(fp_memo, "**********100*****\n");
for(j=0; j<OBJECT_NUMBER; j++){   // 物体 A の数だけ
prob = R_A[0][j] * (1.0 - R_A[1][j]) * (1.0 - R_A[2][j]);
fprintf(fp_memo, "A[%d] : %f\n", j+1, prob);
// max かどうか確認
if(prob > max_prob){  // max_prob の初期値は 0
max_prob = prob;
}
if(j==ob_n){  // 物体番号が一致したら main_prob
main_prob = prob;
}
pre_prob = prob;
prob = 1.0;
}
if((main_prob == max_prob)&&(main_prob != 0.0)){  // main_prob が最大値だった時だけ、出力 bit=1
outputbit->R_100 = 1;
}
max_prob = 0.0;



// 110 ////////////////////////////////////////////////////////////////////////
fprintf(fp_memo, "**********101*****\n");
for(j=0; j<OBJECT_NUMBER; j++){   // 物体 A の数だけ
prob = R_A[0][j] * (1.0 - R_A[1][j]) * R_A[2][j];
fprintf(fp_memo, "A[%d] : %f\n", j+1, prob);
// max かどうか確認
if(prob > max_prob){  // max_prob の初期値は 0
max_prob = prob;
}
if(j==ob_n){  // 物体番号が一致したら main_prob
main_prob = prob;
}
pre_prob = prob;
}
if((main_prob == max_prob)&&(main_prob != 0.0)){  // main_prob が最大値だった時だけ、出力 bit=1
outputbit->R_101 = 1;
}
max_prob = 0.0;

// 110 ////////////////////////////////////////////////////////////////////////
fprintf(fp_memo, "**********110*****\n");
for(j=0; j<OBJECT_NUMBER; j++){   // 物体 A の数だけ
prob = R_A[0][j] * R_A[1][j] * (1.0 - R_A[2][j]);
fprintf(fp_memo, "A[%d] : %f\n", j+1, prob);
// max かどうか確認
if(prob > max_prob){  // max_prob の初期値は 0
max_prob = prob;
}
if(j==ob_n){  // 物体番号が一致したら main_prob
main_prob = prob;
}
pre_prob = prob;
}
if((main_prob == max_prob)&&(main_prob != 0.0)){  // main_prob が最大値だった時だけ、出力 bit=1
outputbit->R_110 = 1;
}
max_prob = 0.0;


// 111 ////////////////////////////////////////////////////////////////////////
fprintf(fp_memo, "**********111*****\n");
for(j=0; j<OBJECT_NUMBER; j++){   // 物体 A の数だけ
prob = R_A[0][j] * R_A[1][j] * R_A[2][j];
fprintf(fp_memo, "A[%d] : %f\n", j+1, prob);
// max かどうか確認
```

```
if(prob > max_prob){  // max_prob の初期値は0
max_prob = prob;
}
if(j==ob_n){  // 物体番号が一致したら main_prob
main_prob = prob;
}
pre_prob = prob;
}
if((main_prob == max_prob)&&(main_prob != 0.0)){  // main_prob が最大値だった時だけ、出力 bit=1
outputbit->R_111 = 1;
}
max_prob = 0.0;
return 0.0;
}
```

B.3.19 check_n

```
////////////////////////////////////////////////////////////////////////////////////////////

// n が配列 arr[] の中に含まれてれば1、含まれていなければ0を返す関数

////////////////////////////////////////////////////////////////////////////////////////////

int check_n(int n, int *arr, int e_n){

int i;

for(i=0; i<e_n; i++){
if(n == arr[i]){
return 1;
}
}
return 0;
}
```

B.3.20 scene_drop_text

```
////////////////////////////////////////////////////////////////////////////////////////////

// text から読み取って、text で出力
// n : 一秒につき何回で区切るか。

////////////////////////////////////////////////////////////////////////////////////////////

int scene_drop_text(int n, int max_or_min, char *fname1, char *fname2){

float x, y, z;
int ye,mo,d,h,mi,s,ms;
int pre_s, pre_ms;
int i=0;

// FILE 型の変数
FILE *file1; // = g_file_sensor[s_count].fp_in;
FILE *file2; //= g_file_sensor[s_count].fp_out;

int nps1 = 1000/n;  // n(回) per second (1 個分)
int nps = nps1;     // 読み込むたびに変化していく。
int j=0;            // ファイルの最初に表示する、「s_1」とかの数字の所
int k=0;            // カウント用

int data;

if(fopen_s(&file1, fname1,"r")!=0){ //読み取り用ファイル
puts( " ファイルが開けません" );
return 1;
}

fopen_s(&file2, fname2, "w"); // 書き込み用ファイル

//最初の一行を取り出す
fscanf_s(file1,"%f,%f,%f,%4d,%2d,%2d,%2d,%2d,%2d,%3d",&x,&y,&z,&ye,&mo,&d,&h,&mi,&s,&ms);
float xx = x;
float yy = y;
float zz = z;


//2行目から読み出し
for(i=1; !feof(file1); ++i){
// 一つ前の情報を pre_s として保持しておく。
pre_s = s;
pre_ms = ms;

// ms の値がとんだ時の処理
for(k=0; pre_ms > nps; k++){
j++;
nps = nps + nps1;
}

fscanf_s(file1,"%f,%f,%f,%4d,%2d,%2d,%2d,%2d,%2d,%3d",&x,&y,&z,&ye,&mo,&d,&h,&mi,&s,&ms);
```

```
  // □   s が変わらず、かつ、ms が nps より小さいの時 => 大きい方の加速度を保持
if((s == pre_s) && (ms < nps)){

if(max_or_min == 1){
if(abs(x)>abs(xx)) xx = x;
if(abs(y)>abs(yy)) yy = y;
if(abs(z)>abs(zz)) zz = z;
}else if(max_or_min == 0){
if(abs(x)<abs(xx)) xx = x;
if(abs(y)<abs(yy)) yy = y;
if(abs(z)<abs(zz)) zz = z;

}else{
printf("error in max_or_min\n");
return 1;
}

  // □   ms が nps を超えたら、nps 1個分だけ増やす
}else if((s == pre_s) && (ms >= nps)){
j++;
//printf("%f ,  %f , %f ,%4d,%2d,%2d,%2d,%2d,%2d,%3d,%3d\n",pre_s, j, xx,yy,zz,ye,mo,d,h,mi, pre_s, pre_ms, j);
data= fprintf_s(file2,"%f ,  %f , %f ,%4d,%2d,%2d,%2d,%2d,%2d,%3d,%3d\n",xx,yy,zz,ye,mo,d,h,mi, pre_s, pre_ms, j);

xx = x;
yy = y;
zz = z;

nps = nps1 * (j+1);

  // □   秒が変わった時
}else if(s != pre_s){
j++;
//printf("%f ,  %f , %f ,%4d,%2d,%2d,%2d,%2d,%2d,%3d,%3d\n",pre_s, j, xx,yy,zz,ye,mo,d,h,mi, pre_s, pre_ms, j);
data= fprintf_s(file2,"%f ,  %f , %f ,%4d,%2d,%2d,%2d,%2d,%2d,%3d,%3d\n",xx,yy,zz,ye,mo,d,h,mi, pre_s, pre_ms, j);

j=0;
nps = nps1;

   // □   想定外
}else{
fprintf(file2, "%d  %d%  ", nps, nps1);
data = fprintf_s(file2, "%s\n", "例外！!");
}

}


// 最後の一行を表示
j++;
data= fprintf_s(file2,"%f ,  %f , %f ,%4d,%2d,%2d,%2d,%2d,%2d,%3d,%3d\n",xx,yy,zz,ye,mo,d,h,mi, pre_s, pre_ms, j);

fclose(file1);
fclose(file2);

printf("****** text_file : %s was made ******\n", fname2);

return n;
}
```

B.3.21 set_sensor_node

```
////////////////////////////////////////////////////////////////////////////////////////
// センサーのデータを２行ずつ差を取ってノードを付けて、ファイル出力する関数。
// n : ファイルの中で、何行ずつ取るか　（とりあえず２行ずつ取る。）
// interval : x, y, z のそれぞれの差
////////////////////////////////////////////////////////////////////////////////////////

int set_sensor_node(int n, float interval, char *fname1, char *fname2){

FILE *fp1;
FILE *fp2;

int i=0;
float x, y, z;
int year, month, day, h, m, s, ms, flag;

float pre_x, pre_y, pre_z;
float dif_x, dif_y, dif_z; // x,y,z のそれぞれの差
int pre_flag;

// node の決め方
// x-1, y-2, z-4
// xyz  = 000   001   010   011   100   101   110   111
// node =   0     4     2     6     1     5     3     7
int node;
int node_x = 1;
int node_y = 2;
int node_z = 4;

if(fopen_s(&fp1, fname1, "r") != 0){
printf("in set_sensor_node  file1 open error!!\n");
}
```

```
if(fopen_s(&fp2, fname2, "w") != 0){
printf("in set_sensor_node  file2 open error!!\n");
}

// 1行目を読む
fscanf(fp1, "%f ,  %f , %f ,%4d,%2d,%2d,%2d,%2d,%2d,%3d,%3d", &x, &y, &z, &year, &month, &day, &h, &m, &s, &ms, &flag);

while(!feof(fp1)){
pre_x = x;
pre_y = y;
pre_z = z;
pre_flag = flag;
node = 0;   // ノードを初期化

fscanf(fp1, "%f ,  %f , %f ,%4d,%2d,%2d,%2d,%2d,%2d,%3d,%3d", &x, &y, &z, &year, &month, &day, &h, &m, &s, &ms, &flag);

// 差を求めて
dif_x = x-pre_x;
dif_y = y-pre_y;
dif_z = z-pre_z;


// node を設定する
if(abs(dif_x) >= interval){
node = node + node_x;
}
if(abs(dif_y) >= interval){
node = node + node_y;
}
if(abs(dif_z) >= interval){
node = node + node_z;
}


if(pre_flag > flag){
i++;
}
fprintf(fp2, "%3d   %4d-%4d   %f,  %f,  %f   %d\n",i , pre_flag, flag, dif_x, dif_y, dif_z, node);
}


printf("******  sensor_node file 「%s」 was made!!  ******\n", fname2);

fclose(fp1);
fclose(fp2);

return 0;
}
```

B.3.22 set_sensor_node

```
////////////////////////////////////////////////////////////////////////////////////////////

//   センサ処理用ファイルに名前を付ける関数

////////////////////////////////////////////////////////////////////////////////////////////

void make_file_name(char *pre_name, int n_of_file, char *str_out){
sprintf(str_out, "%s%d.txt", pre_name, n_of_file);
}
```

B.3.23 set_file_bit, set_g_pos

```
////////////////////////////////////////////////////////////////////////////////////////////

//  センサファイルが存在するかをセットする関数

////////////////////////////////////////////////////////////////////////////////////////////
void set_file_bit(){

g_file_sensor[0].flag = 0;
g_file_sensor[1].flag = 1;
g_file_sensor[2].flag = 1;
g_file_sensor[3].flag = 1;
g_file_sensor[4].flag = 1;
//g_file_sensor[5].flag = 1;

// カメラに映ってるかもセット
g_file_sensor[0].c_flag = 11;
g_file_sensor[1].c_flag = 10;
g_file_sensor[2].c_flag = 11;
g_file_sensor[3].c_flag = 11;
g_file_sensor[4].c_flag = 11;
//g_file_sensor[5].c_flag = 11;
}

////////////////////////////////////////////////////////////////////////////////////////////

//  マウス定義してた座標をここで指定
// しょうがないから、全部手動で指定。。。
```

```
// g_pos_new[video_number][Object_number][which_position]

////////////////////////////////////////////////////////////////////////////////////////
void set_g_pos(){

Object *pMove1;
Object *pMove2;

////////////////////////////////////////////////// video_1
// object_1
g_pos_new1[0][0].x = 184; // 184
g_pos_new1[0][0].y = 135; // 135
g_pos_new1[0][1].x = 302;
g_pos_new1[0][1].y = 234;
g_pos_new1[0][2].x = 318;
g_pos_new1[0][2].y = 166;
g_pos_new1[0][3].x = 266;
g_pos_new1[0][3].y = 125;

// object_2
g_pos_new1[1][0].x = 57;
g_pos_new1[1][0].y = 123;
g_pos_new1[1][1].x = 132;
g_pos_new1[1][1].y = 115;
g_pos_new1[1][2].x = 135;
g_pos_new1[1][2].y = 228;
g_pos_new1[1][3].x = 61;
g_pos_new1[1][3].y = 225;

// object_3
g_pos_new1[2][0].x = 13;
g_pos_new1[2][0].y = 97;
g_pos_new1[2][1].x = 56;
g_pos_new1[2][1].y = 96;
g_pos_new1[2][2].x = 55;
g_pos_new1[2][2].y = 170;
g_pos_new1[2][3].x = 24;
g_pos_new1[2][3].y = 175;

// object_4
g_pos_new1[3][0].x = 242;
g_pos_new1[3][0].y = 37;
g_pos_new1[3][1].x = 316;
g_pos_new1[3][1].y = 39;
g_pos_new1[3][2].x = 312;
g_pos_new1[3][2].y = 129;
g_pos_new1[3][3].x = 233;
g_pos_new1[3][3].y = 136;

// object_5
g_pos_new1[4][0].x = 130;
g_pos_new1[4][0].y = 42;
g_pos_new1[4][1].x = 190;
g_pos_new1[4][1].y = 43;
g_pos_new1[4][2].x = 188;
g_pos_new1[4][2].y = 179;
g_pos_new1[4][3].x = 129;
g_pos_new1[4][3].y = 176;


////////////////////////////////////////////////// video_2

// object_1
g_pos_new2[0][0].x = 229;
g_pos_new2[0][0].y = 158;
g_pos_new2[0][1].x = 319;
g_pos_new2[0][1].y = 174;
g_pos_new2[0][2].x = 319;
g_pos_new2[0][2].y = 148;
g_pos_new2[0][3].x = 279;
g_pos_new2[0][3].y = 139;

// object_2
g_pos_new2[1][0].x = 212;
g_pos_new2[1][0].y = 174;
g_pos_new2[1][1].x = 266;
g_pos_new2[1][1].y = 209;
g_pos_new2[1][2].x = 287;
g_pos_new2[1][2].y = 172;
g_pos_new2[1][3].x = 211;
g_pos_new2[1][3].y = 149;

// object_3
g_pos_new2[2][0].x = 2;
g_pos_new2[2][0].y = 134;
g_pos_new2[2][1].x = 19;
g_pos_new2[2][1].y = 127;
g_pos_new2[2][2].x = 28;
g_pos_new2[2][2].y = 219;
g_pos_new2[2][3].x = 2;
g_pos_new2[2][3].y = 236;

// object_4
g_pos_new2[3][0].x = 218;
```

```
g_pos_new2[3][0].y = 51;
g_pos_new2[3][1].x = 253;
g_pos_new2[3][1].y = 52;
g_pos_new2[3][2].x = 250;
g_pos_new2[3][2].y = 145;
g_pos_new2[3][3].x = 220;
g_pos_new2[3][3].y = 171;

// object_5
g_pos_new2[4][0].x = 58;
g_pos_new2[4][0].y = 41;
g_pos_new2[4][1].x = 117;
g_pos_new2[4][1].y = 45;
g_pos_new2[4][2].x = 123;
g_pos_new2[4][2].y = 206;
g_pos_new2[4][3].x = 78;
g_pos_new2[4][3].y = 213;

}
```

B.3.24 cvSmooth, blur_len, make_len_array, blur_side, make_side_array, resolution

```
////////////////////////////////////////////////////////////////////////////////////////

// 動画データを読み込んだ後すぐに、フレームに対して、filter 処理を施す。

////////////////////////////////////////////////////////////////////////////////////////

// 平滑化する（ぼやけた感じになる）

/* 本質の部分は以下の関数だけなので、直接使った。
cvSmooth(img, ds_img, CV_BLUR, thr, 0, 0, 0);
*/

////////////////////////////////////////////////////////////////////////////////////////

// 縦ブレ

////////////////////////////////////////////////////////////////////////////////////////

void blur_len(IplImage *img, int thr){

int i = 0; // カウンタ

float array_len[MAX_SIZE];

for(i=0; i<MAX_SIZE; i++){
array_len[i] = 0.0;
}

// 配列作成
*array_len = make_len_array(array_len, thr);

CvMat kernel_len = cvMat(thr, thr, CV_32F, array_len);

    // (1) 画像の読み込
// (2) カーネルの正規化と，フィルタ処理
cvNormalize(&kernel_len, &kernel_len, 1.0, 0, CV_L1);
cvFilter2D(img, img, &kernel_len, cvPoint(-1, -1));
}


////////////////////////////////////////
//
// 配列を作る関数（縦）
//
float make_len_array(float *array_len, int size_of_array){
int i=0;
int j=0;

for(j=0; j<size_of_array; j++){
for(i=0; i<size_of_array; i++){
array_len[i + size_of_array*j] = 0.0;
if(i == (size_of_array/2)){
array_len[i + size_of_array*j] = 1.0 / float(size_of_array);
}
}
}

return *array_len;
}

////////////////////////////////////////////////////////////////////////////////////////

// 横ブレ

////////////////////////////////////////////////////////////////////////////////////////
void blur_side(IplImage *img, int thr){

int i = 0; // カウンタ
```

```
float array_side[MAX_SIZE];

for(i=0; i<MAX_SIZE; i++){
array_side[i] = 0.0;
}

// 配列作成
*array_side = make_side_array(array_side, thr);

CvMat kernel_side = cvMat(thr, thr, CV_32F, array_side);

    // (1) 画像の読み込み
// (2) カーネルの正規化と，フィルタ処理
cvNormalize(&kernel_side, &kernel_side, 1.0, 0, CV_L1);
cvFilter2D(img, img, &kernel_side, cvPoint(-1, -1));
}

////////////////////////////////////////
//
// 配列を作る関数（縦）
//
float make_side_array(float *array_side, int size_of_array){

int i=0;
int j=0;

for(j=0; j<size_of_array; j++){
for(i=0; i<size_of_array; i++){
array_side[i + size_of_array*j] = 0.0;

if(j == size_of_array/2){
array_side[i + size_of_array*j] = 1.0 / float(size_of_array);
}
}
}

return *array_side;
}

////////////////////////////////////////////////////////////////////////////////////////

// 解像度

////////////////////////////////////////////////////////////////////////////////////////
void resolution(IplImage *img, int thr){
int blue, green, red;
int tmp1, tmp2;
int x, y;
int i, j;

int sub_h; // 解像度処理用の仮の height
int sub_w;

// 中央の値を保持
int middle = thr / 2;


// 縦と横のサイズを３で割ってあまり１にする（たぶんプログラムの問題...）
int w_size = (img->width) % 3;
int h_size = (img->height) % 3;

if((w_size!=1) || (h_size!=1)){
if(w_size == 0){
sub_w = img->width - 2;
}else if(w_size == 2){
sub_w = img->width - 1;
}

if(h_size == 0){
sub_h = img->height - 2;
}else if(h_size == 2){
sub_h = img->height - 1;
}
}
////////////////////////////////////////////// ここまで画素の操作

for(y=0; y<sub_h; y+=thr){
for(x=0; x<sub_w; x+=thr){

blue = 0;
green = 0;
red = 0;

for(i=0; i<thr; i++){
if(sub_h < (y+i)){ // モザイク画素の位置が縦辺を超えたら終わり
break;
}
tmp1 = i;

for(j=0; j<thr; j++){
if(sub_w < (x+j)){
break;
}
```

```
// 中央の画素まできたら、それを代入
if((j == middle) && (i == middle)){
blue = (unsigned char)img->imageData[img->widthStep * (y + i) + (x + j) * 3];
green = (unsigned char)img->imageData[img->widthStep * (y + i) + (x + j)* 3 + 1];
red = (unsigned char)img->imageData[img->widthStep * (y + i) + (x + j) * 3 + 2];
}
tmp2 = j;
}
}

tmp1++;
tmp2++;

for(i=0; i<tmp1; i++){
for(j=0; j<tmp2; j++){
img->imageData[img->widthStep * (y + i) + (x + j) * 3] = blue;
img->imageData[img->widthStep * (y + i) + (x + j) * 3 + 1] =green;
img->imageData[img->widthStep * (y + i) + (x + j) * 3 + 2] = red;
}
}


}
}
}
```

B.3.25 new_set_HMM_state

```
////////////////////////////////////////////////////////////////////////////////////////
// ファイルは、書き込み用、カメラ 1 の距離、カメラ 2 の距離
//  マルコフの時の state のセット（ファイルの整理 & 加速度ノードの合体）（書き込み ver）
// とりあえず、カメラ 1 とカメラ 2 の距離を足し算して、加速度ファイルを読込んで、state をセットしなおす
// この関数に入った時点で、g_time は解析したフレーム数になってる！(そのまま使う)
////////////////////////////////////////////////////////////////////////////////////////

void new_set_HMM_state(FILE *fp){
int time_c1, time_c2; // カメラ 1 とカメラ 2 の g_time
int ob_c1, ob_c2; // 物体番号（0 スタート）
float dist_c1[OBJECT_NUMBER], dist_c2[OBJECT_NUMBER]; // 距離
int st_r_c1[OBJECT_NUMBER], st_r_c2[OBJECT_NUMBER]; // 読込時の状態

// 一つ前の状態
int pre_time_c1, pre_time_c2; // カメラ 1 とカメラ 2 の g_time
int pre_ob_c1, pre_ob_c2; // 物体番号（0 スタート）

float pre_dist_c1[OBJECT_NUMBER], pre_dist_c2[OBJECT_NUMBER]; // 距離
int pre_st_r_c1[OBJECT_NUMBER], pre_st_r_c2[OBJECT_NUMBER]; // 読込時の状態

// マルコフで状態遷移を記録するため変数
int pre_st_c[OBJECT_NUMBER]; // 直前のカメラの state
int pre_st_a[OBJECT_NUMBER]; // 直前の加速度の state

int st_c[OBJECT_NUMBER]; // カメラの state
int st_a[OBJECT_NUMBER]; // 加速度の state

// マルコフ：遷移の記録用
int tra_c[OBJECT_NUMBER]; // カメラの遷移
int rate[OBJECT_NUMBER]; // 近づき度

// 加速度データ用
int flag_time;
int time_pre, time_aft, s_flag;
float dif_x, dif_y, dif_z;

// 書き込み用 state
int state;
float dist_c;  //カメラ 2 台の合計距離の合計

// カウント用
int i, j, k;

FILE *d_fp1; // 読込用ファイル
FILE *d_fp2;

FILE *d_out1; // g_time 整理後の出力ファイル
FILE *d_out2;

FILE *s_fp; // 加速度データ込みファイル
FILE *tra_fp; // 遷移付き

char *fname_d1 = "d_out1.txt";
char *fname_d2 = "d_out2.txt";

char *fname_st = "state.txt";
char *fname_tra = "trance.txt";

// 読込用ファイル open
if(fopen_s(&d_fp1, "distance1.txt", "r") != 0){
printf("距離用出力ファイル 1 を開けません (in new_set_HMM_state)\n");
}
```

```
if(fopen_s(&d_fp2, "distance2.txt", "r") != 0){
printf("距離用出力ファイル 2 を開けません (in new_set_HMM_state)\n");
}

// 書き込み用ファイル open
if( (d_out1 = fopen(fname_d1,"w")) == NULL ){
printf("距離用出力ファイル 1 を開けません (in new_set_HMM_state)\n");
}

if( (d_out2 = fopen(fname_d2,"w")) == NULL ){
printf("距離用出力ファイル 2 を開けません (in new_set_HMM_state)\n");
//return -1;
}

////////////////////////////////////////////////////////////////////////////////
//////////////////        まずは、2 枚の距離ファイルの g_time を整理          //////////
////////////////////////////////////////////////////////////////////////////////

// 最初の一行を読み込んで書き込み（g_time = 1）(物体の数だけ)
for(i=0; i<OBJECT_NUMBER; i++){

// カメラ 1 読込
fscanf_s(d_fp1, "%3d,%2d,%f,%1d", &pre_time_c1, &pre_ob_c1, &pre_dist_c1[i], &pre_st_r_c1[i]);

// カメラ 2 読込
fscanf_s(d_fp2, "%3d,%2d,%f,%1d", &pre_time_c2, &pre_ob_c2, &pre_dist_c2[i], &pre_st_r_c2[i]);
}

//
// カメラ 1 の距離ファイル処理
//
for(i=1; !feof(d_fp1); i++){

// まず、1行読み込んで
for(j=0; j<OBJECT_NUMBER; j++){
fscanf_s(d_fp1, "%3d,%2d,%f,%1d", &time_c1, &ob_c1, &dist_c1[j], &st_r_c1[j]);
}

if(pre_time_c1+1 == time_c1){ // 直後の time だったら、pre は書いて OK
for(j=0; j<OBJECT_NUMBER; j++){
fprintf(d_out1, "%3d,%2d,%.2f,%1d\n", pre_time_c1, j, pre_dist_c1[j], pre_st_r_c1[j]);
}
}else if(pre_time_c1 == time_c1){ // g_time がダブってたら（近い方の距離を採用する）
for(j=0; j<OBJECT_NUMBER; j++){
if(dist_c1[j] >= pre_dist_c1[j]){ // 小さい方を保持！書き込みは次する！
dist_c1[j] = pre_dist_c1[j];
st_r_c1[j] = pre_st_r_c1[j];
}
}
}else if(pre_time_c1 < time_c1){ // time が飛んでたら、来るべきだった time を書き込む。前の行と同じ。(読込はしない)
//  とりあえず、pre を書く。
for(j=0; j<OBJECT_NUMBER; j++){
fprintf(d_out1, "%3d,%2d,%.2f,%1d\n", pre_time_c1, j, pre_dist_c1[j], pre_st_r_c1[j]);
}

while(time_c1 - pre_time_c1 != 1){ // 差が 1 になるまで書き込みを繰り返す
for(j=0; j<OBJECT_NUMBER; j++){
fprintf(d_out1, "%3d,%2d,%.2f,%1d\n", pre_time_c1+1, j, pre_dist_c1[j], pre_st_r_c1[j]);
}
pre_time_c1++;
}
}else{
printf("想定外 !! (in new_set_HMM_state c1)\n");
}

// 一つ前のデータを保持
  pre_time_c1 = time_c1;
for(j=0; j<OBJECT_NUMBER; j++){
pre_dist_c1[j] = dist_c1[j];
pre_st_r_c1[j] = st_r_c1[j];
}
} // while(!feof(d_fp1)) 終わり

// 最後の行を書いとく
for(j=0; j<OBJECT_NUMBER; j++){
fprintf(d_out1, "%3d,%2d,%.2f,%1d\n", pre_time_c1, j, pre_dist_c1[j], pre_st_r_c1[j]);
}
fclose(d_out1);

//
// カメラ 2 の距離ファイル処理
//

//while(!feof(d_fp1)){ // ファイルが終わるまで
for(i=1; !feof(d_fp2); i++){

// まず、1行読み込んで
for(j=0; j<OBJECT_NUMBER; j++){
fscanf_s(d_fp2, "%3d,%2d,%f,%1d", &time_c2, &ob_c2, &dist_c2[j], &st_r_c2[j]);
}

if(pre_time_c2+1 == time_c2){ // 直後の time だったら、pre は書いて OK
for(j=0; j<OBJECT_NUMBER; j++){
```

```
fprintf(d_out2, "%3d,%2d,%.2f,%1d\n", pre_time_c2, j, pre_dist_c2[j], pre_st_r_c2[j]);
}
}else if(pre_time_c2 == time_c2){ // g_time がダブってたら（近い方の距離を採用する）
for(j=0; j<OBJECT_NUMBER; j++){
if(dist_c2[j] >= pre_dist_c2[j]){ // 小さい方を保持！書き込みは次する！
dist_c2[j] = pre_dist_c2[j];
st_r_c2[j] = pre_st_r_c2[j];
}
}
}else if(pre_time_c2 < time_c2){ // time が飛んでたら、来るべきだった time を書き込む。前の行と同じ。（読込はしない）
//  とりあえず、pre を書く。
for(j=0; j<OBJECT_NUMBER; j++){
fprintf(d_out2, "%3d,%2d,%.2f,%1d\n", pre_time_c2, j, pre_dist_c2[j], pre_st_r_c2[j]);
}

while(time_c2 - pre_time_c2 != 1){ // 差が 1 になるまで書き込みを繰り返す
for(j=0; j<OBJECT_NUMBER; j++){
fprintf(d_out2, "%3d,%2d,%.2f,%1d\n", pre_time_c2+1, j, pre_dist_c2[j], pre_st_r_c2[j]);
}
pre_time_c2++;
}
}else{
printf("想定外 !! (in new_set_HMM_state c2)\n");
// ダブってたら何もしない -> ここに落ちる。
}

// 一つ前のデータを保持
  pre_time_c2 = time_c2;
for(j=0; j<OBJECT_NUMBER; j++){
pre_dist_c2[j] = dist_c2[j];
pre_st_r_c2[j] = st_r_c2[j];
}
} // while(!feof(d_fp2)) 終わり

// 最後の行を書いとく
for(j=0; j<OBJECT_NUMBER; j++){
fprintf(d_out2, "%3d,%2d,%.2f,%1d\n", pre_time_c2, j, pre_dist_c2[j], pre_st_r_c2[j]);
}

fclose(d_out2);
fclose(d_fp1);
fclose(d_fp2);

////////////////////////////////////////////////////////////////
//////////////////    加速度と同期して書き出し      //////////
////////////////////////////////////////////////////////////////

// 読込用用ファイル open
if( (d_out1 = fopen(fname_d1,"r")) == NULL ){
printf("距離読込ファイル 1 を開けません (in new_set_HMM_state)\n");
}

if( (d_out2 = fopen(fname_d2,"r")) == NULL ){
printf("距離読込ファイル 2 を開けません (in new_set_HMM_state)\n");
}

// 加速度のノードファイルを開く
for(i=0; i<OBJECT_NUMBER; i++){
if(g_file_sensor[i].flag == 1){ // SunSPOT 付き物体だったら
if( (g_file_sensor[i].fp_node = fopen(g_file_sensor[i].text_node,"r")) == NULL ){
printf("加速度センサ読込用ファイルを開けません (in new_set_HMM_state)\n");
}
}
}

// 書き込み用ファイル
if( (s_fp = fopen(fname_st,"w")) == NULL ){
printf("状態出力ファイルを開けません (in new_set_HMM_state)\n");
}

for(i=1; i<g_time; i++){
// 物体の数だけ
for(j=0; j<OBJECT_NUMBER; j++){
// カメラ 2 台の距離読込
if(fscanf_s(d_out1, "%3d,%2d,%f,%1d", &time_c1, &ob_c1, &dist_c1[j], &st_r_c1[j]) == NULL){
time_c1 = g_time;
dist_c1[j] = 0.0;
st_r_c1[j] = 3;
}

if(fscanf_s(d_out2, "%3d,%2d,%f,%1d", &time_c2, &ob_c2, &dist_c2[j], &st_r_c2[j]) == NULL){
time_c2 = g_time;
dist_c2[j] = 0.0;
st_r_c2[j] = 3;
}

// 加速度
if(g_file_sensor[j].flag == 1){ // SunSPOT 付物体だったら、
// 加速度データ読込
fscanf_s(g_file_sensor[j].fp_node, "%3d   %4d-%4d   %f,  %f,  %f   %d", &flag_time, &time_pre, &time_aft, &dif_x, &dif_y, &dif_z, &s_flag);
}else{
s_flag = -1;
}
```

```
// カメラ 2 台の state 設定
if(g_file_sensor[j].c_flag == 11){ // 両方のカメラに写ってるなら
dist_c = dist_c1[j] + dist_c2[j];
}else if(g_file_sensor[j].c_flag == 10){ // 片方のカメラのみに写ってるなら
dist_c = dist_c1[j]*2;
}else if(g_file_sensor[j].c_flag == 01){
dist_c = dist_c2[j]*2;
}

if(dist_c <= 100.0){
fprintf(s_fp, "%3d,%.2f,1,%1d\n", time_c1, dist_c, s_flag);
}else if(dist_c <= 300.0){
fprintf(s_fp, "%3d,%.2f,2,%1d\n", time_c1, dist_c, s_flag);
}else{
fprintf(s_fp, "%3d,%.2f,3,%1d\n", time_c1, dist_c, s_flag);
}

}
}

// ファイルを閉じる
fclose(d_out1);
fclose(d_out2);
fclose(s_fp);

// 加速度のノードファイルを閉じる
for(i=0; i<OBJECT_NUMBER; i++){
if(g_file_sensor[i].flag == 1){ // SunSPOT 付き物体だったら
fclose(g_file_sensor[i].fp_node);
}
}

////////////////////////////////////////////////////////////
//////////////////   遷移状態の書き出し             //////////
////////////////////////////////////////////////////////////

// 読込用ファイル
if((s_fp = fopen(fname_st,"r")) == NULL ){
printf("状態読込ファイルを開けません (in new_set_HMM_state)\n");
}

// 書き込み用ファイル
if((tra_fp = fopen(fname_tra,"w")) == NULL ){
printf("遷移状態出力ファイルを開けません (in new_set_HMM_state)\n");
}
////////////////////////////////////////////////////////////////////////////////////
//////////////////        コマ落としの時に捨てるデータを入れとく配列        //////////
////////////////////////////////////////////////////////////////////////////////////
// 捨てる数を入れとく配列（コマ数は１０と仮定）
int thrown[10]; // movie
int thrown2[10]; // accel
int e_n = 0; // 要素数（movie）
int e_n2 = 0; // 要素数（accel）

// movie のコマ落とし（固定）
if(drop_movie==10){
thrown[0] = -1;

e_n = 1;
}else if(drop_movie == 9){
thrown[0] = 0;

e_n = 1;
}else if(drop_movie == 8){
thrown[0] = 0;
thrown[1] = 1;

e_n = 2;
}else if(drop_movie == 7){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;

e_n = 3;
}else if(drop_movie == 6){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;

e_n = 4;
}else if(drop_movie == 5){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;

e_n = 5;
}else if(drop_movie == 4){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
```

```
thrown[4] = 4;
thrown[5] = 5;

e_n = 6;
}else if(drop_movie == 3){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;
thrown[6] = 6;

e_n = 7;
}else if(drop_movie == 2){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;
thrown[6] = 6;
thrown[7] = 7;

e_n = 8;
}else if(drop_movie == 1){
thrown[0] = 0;
thrown[1] = 1;
thrown[2] = 2;
thrown[3] = 3;
thrown[4] = 4;
thrown[5] = 5;
thrown[6] = 6;
thrown[7] = 7;
thrown[8] = 8;

e_n = 9;
}else{
printf("error in new_verbalization2 (thrown)\n");
}


// accel のコマ落とし（固定）
if(drop_sensor==10){
thrown2[0] = -1;

e_n2 = 1;
}else if(drop_sensor == 9){
thrown2[0] = 0;

e_n2 = 1;
}else if(drop_sensor == 8){
thrown2[0] = 0;
thrown2[1] = 1;

e_n2 = 2;
}else if(drop_sensor == 7){
thrown2[0] = 0;
thrown2[1] = 1;
thrown2[2] = 2;

e_n2 = 3;
}else if(drop_sensor == 6){
thrown2[0] = 0;
thrown2[1] = 1;
thrown2[2] = 2;
thrown2[3] = 3;

e_n2 = 4;
}else if(drop_sensor == 5){
thrown2[0] = 0;
thrown2[1] = 1;
thrown2[2] = 2;
thrown2[3] = 3;
thrown2[4] = 4;

e_n2 = 5;
}else if(drop_sensor == 4){
thrown2[0] = 0;
thrown2[1] = 1;
thrown2[2] = 2;
thrown2[3] = 3;
thrown2[4] = 4;
thrown2[5] = 5;

e_n2 = 6;
}else if(drop_sensor == 3){
thrown2[0] = 0;
thrown2[1] = 1;
thrown2[2] = 2;
thrown2[3] = 3;
thrown2[4] = 4;
thrown2[5] = 5;
thrown2[6] = 6;
```

```
e_n2 = 7;
}else if(drop_sensor == 2){
thrown2[0] = 0;
thrown2[1] = 1;
thrown2[2] = 2;
thrown2[3] = 3;
thrown2[4] = 4;
thrown2[5] = 5;
thrown2[6] = 6;
thrown2[7] = 7;

e_n2 = 8;
}else if(drop_sensor == 1){
thrown2[0] = 0;
thrown2[1] = 1;
thrown2[2] = 2;
thrown2[3] = 3;
thrown2[4] = 4;
thrown2[5] = 5;
thrown2[6] = 6;
thrown2[7] = 7;
thrown2[8] = 8;

e_n2 = 9;
}else{
printf("error in new_verbalization2 (thrown2)\n");
}

////////////////////////////////////////////////////////////////////////////////////////

for(j=0; j<OBJECT_NUMBER; j++){
pre_st_c[j] = 3;
pre_st_a[j] = 0;
tra_c[j] = 0;
rate[j] = 0;
}

for(j=0; j<e_n; j++){
fprintf(fp, "%d\n", thrown[j]);
}


for(i=1; i<g_time; i++){

// 物体の数だけ
for(j=0; j<OBJECT_NUMBER; j++){
fscanf_s(s_fp, "%3d,%f,%1d,%d", &time_c1, &dist_c, &st_c[j], &st_a[j]);

// コマ落とし
// g_time(i) を 10 で割った余りが、thrown 配列の中にあれば、
// 加速度付いてたら
if(g_file_sensor[j].flag == 1){
if(check_n((i % 10), thrown2, e_n2) == 1){
st_a[j] = 0;
fprintf(fp, "g_time : %d, ob_n : %d, drop_sensor : %d, reset_accel 0 \n", i, j, drop_sensor);
}
}
// 動画のコマ落とし
if(check_n((i % 10), thrown, e_n) == 0){

// 遷移の記録
//  st_c:    3 -> 3 -> 2 -> 1 -> 1 -> 1 -> 3
//  tra_c:     1    2    3    4    4    1
// 他は 0

// ただの遷移のノード
if(st_c[j] == 3){
if(pre_st_c[j] == 3){ // 3 から来たなら
tra_c[j] = 1;
}else{
tra_c[j] = 0;
}
}else if(st_c[j] == 2){
if((pre_st_c[j] == 2)||(pre_st_c[j] == 3)){ // 2, 3 から来たなら
tra_c[j] = 2;
}else{
tra_c[j] = 0;
}
}else if(st_c[j] == 1){
if(pre_st_c[j] == 1){
tra_c[j] = 4;
}else{
tra_c[j] = 3;
}
}else{
printf("error in new_Ver_m\n");
}

// 近づき
if(st_c[j] == 3){
if(pre_st_c[j] == 3){ // 3 から来たなら
rate[j] = 0;
}else if(pre_st_c[j] == 2){
```

```
rate[j] = rate[j]-2;
}else if(pre_st_c[j] == 1){
rate[j] = rate[j]-4;
}
}else if(st_c[j] == 2){
if(pre_st_c[j] == 3){ // 3 から来たなら
rate[j] = rate[j]+2;
}else if(pre_st_c[j] == 2){
rate[j] = rate[j]+1;
}else if(pre_st_c[j] == 1){
rate[j] = rate[j]-2;
}
}else if(st_c[j] == 1){
if(pre_st_c[j] == 3){ // 3 から来たなら
rate[j] = rate[j] + 4;
}else if(pre_st_c[j] == 2){
rate[j] = rate[j] + 2;
}else if(pre_st_c[j] == 1){
rate[j]++;
}
}else{
printf("error in new_Ver_m\n");
}
}

fprintf(tra_fp, "%3d,%.2f,%1d,%1d,%d,%d\n", time_c1, dist_c, st_c[j], st_a[j], tra_c[j], rate[j]);

// マルコフよりも大きくなったらリセット。
if(rate[j] > (int)(THRESHOLD_MARKOV*3/2)){
rate[j] = 0;
}

} //for 終わり

for(j=0; j<OBJECT_NUMBER; j++){
pre_st_c[j] = st_c[j];
pre_st_a[j] = st_a[j];
}
}

fclose(s_fp);
fclose(tra_fp);
}
```