Spatio-temporal Keypoint Extraction based on Local Correlation and Full-HD 60 fps FPGA Implementation using Gradient Histogram for Cloud Video Recognition

Takahiro SUZUKI

Graduate School of Information, Production and Systems

Waseda University

July, 2017

Acknowledgements

I would like to express my gratitude to all those who helped me during my master's and doctoral coursework and doctor course. I gratefully acknowledge the help of my supervisor, Professor. Takeshi Ikenaga, who has offered me valuable advices in my academic study. Without his patient instruction, insightful criticism, and expert guidance, the completion of this thesis and my current achievements would not have been possible.

I also owe my thanks to the professors that have guided me in improving and finishing this dissertation by providing much useful and remarkable advice, including Professor Tetsunori Kobayashi, Professor Takahiro Watanabe, Professor Takahata Kiyoto, Professor Keiichi Koyanagi and all the Professors in integrated system field of IPS.

Additionally, I am extremely grateful to all my lab mates. I would especially like to express my gratitude to the senior members who have graduated from the Tokyo laboratory: Mr. Hiroyuki Sekiguchi, Mr. Ryosuke Araki, Mr. Shiina Yuhi, and Mr. Yoshiyasu Shimizu. They guided me not only in research but also in daily life. Special thanks should also be given to my lab colleagues: Mr.Youhei Mikami and Mr. Yuichiro Kitao. Our mutual

collaboration has contributed to the completion of this thesis. In addition, I would like to express my gratitude to the members of the Kitakyusyu laboratory: Dr. Lei Sun and Mr. Yoshimasa Iwasaki. I attended conferences with them, and their discussions helped guide me through many challenges. Also, Zhenyu Pei gave me support and valuable comments at the Kitakyusyu laboratory. I would like to thank the lab members, including Mr. Gaoxing Chen, Ms. Xina Cheng, Mr. Songlin Du, Ms. Fanglu Xie, Ms. Yilin Hou, Ms. Ziwei Deng, Ms. Tingting Hu, Mr. Zhennan Wang, Mr. Yuan Hu, Mr. Yinkai Liu, Mr. Tianming Rao, Mr. Guannan Wu, Mr. Zijie Wang, Mr. Yang Liu, Mr. Yiming Zhao, Mr. Yuhao Xu, Mr. Yuchen Yang, Ms. Haoxuan Tan, and Mr. Guixing Liang. I greatly appreciate their support during my dissertation preparation.

I owe a special debt of gratitude to superiors: Akihiro Otaka, Ken-Ichi Suzuki, Jun-ichi Kani, and Sang-Yuep Kim in NTT Access Network Service Systems Laboratory. They accepted and supported my pursuit of a doctoral, as well as gave me a lot of guidance on how to proceed with research. Without their understanding and guidance, I could not have completed this thesis.

I would like to deliver my gratitude to all of my other friends whose names are not listed here. They supported me in my life and study during both university and the working world.

Finally, I would like to dedicate this thesis to my family. They have constantly supported and encouraged me, and it would have

been impossible to achieve so much without them.

Takahiro SUZUKI

July, 2017

Abstract

In recent years, the Internet of Things (IoT), by which various devices are connected to the Internet to send and receive data, is spreading for various information services. IoT services will likely expand to require not only sensor data with small data demands, but also multimedia such as images and video and it is expected to spread to a wide range of applications in daily life from now on. In fact, in speech applications, a speech communication system such as Speech Interpretation and Recognition Interface (Siri) has been put to practical use and is implemented in mobile terminals. Siri sends the voice data caught by the microphone of the terminal to the cloud server via the network. The server recognizes the meaning of the speech dialogue. After that, it extracts the appropriate information via the Internet, and presents it to the terminal side. It is a computer interface that a wide range of people can use regardless of age. Meanwhile, video information also plays important roles in various scenes in everyday life, and it is expected to be widely applied development. In recent years, due to the advancement of image recognition technology, practical technology for surveillance camera, in-vehicle camera, augmented reality etc. are being established. From now on, in the field of

video recognition, it is expected that a system in which user terminals connect to the cloud via the network and recognize acquired data will spread.

The video information is much bigger in data capacity than audio and other multimedia data and it is a big problem in realizing video recognition systems using the cloud. For this reason, it is not realistic to send video information as it is to the server, and it is essential for the terminal side to extract important features that are keys of recognition from images acquired by the camera. Furthermore, in the conventional feature extraction technique, it is a problem to extract not only the object to be recognized but also many unnecessary keypoints for image recognition in videos having complicated background. This is because many keypoint extraction algorithms extract keypoints only from large local spatial features and it cannot obtain the whole recognition object regionally. Therefore, it is expected to realize a keypoint extraction algorithm that can obtain keypoints regionally from recognition objects. On the other hand, in many video recognition applications, real-time processing is required. However, in order to obtain descriptor robustly to image deformation and invariant descriptor such as rotation under practical environments, the keypoint extraction algorithm calculates the second order derivative and gradient histogram, and the amount of calculation is very large. For this reason, high-speed operation is anticipated assuming hardware implementation such as Field-Programmable Gate Array (FPGA).

Furthermore, since conventional keypoint extraction processing includes a lot of sequence dependency and sequential processing of calculation, it is important to improve algorithms for hardware implementation and to study hardware architecture with high parallelism.

This dissertation summarizes the results of research on keypoint extraction algorithms for cloud video recognition system and its real-time FPGA implementation to solve the above problems. In the keypoint extraction algorithm, this dissertation proposes a keypoint extraction method considering local correlation in addition to conventional spatial information and temporal information. Each keypoint is weighted by the spatio-temporal features. Only the keypoints with larger weights are selected. After that, clustering is performed using the inter-keypoint distance and it becomes possible to detect keypoints regionally. In real-time FPGA implementation, this dissertation proposes algorithms suitable for hardware implementation by keypoint extraction using gradient histograms and lower computational complexity of local correlation by density clustering. Furthermore, this dissertation proposes a hardware architecture in which keypoint detection using gradient histogram and descriptor generation are parallelized, and real-time processing on FPGA evaluation board is realized.

Chapter 1 explains the present state of recognition system, cloud video recognition, and its problem that the amount of data

which are transmitted is large. After that, this dissertation describes the problems of the conventional keypoint extraction method, the background of the research, and the points of focus and the objectives of this paper.

Chapter 2 proposes a spatio-temporal keypoint extraction based on local correlation. In the conventional keypoint extraction method, since keypoints are detected from locations having large local intensity gradient values, it is a problem that keypoints are detected from other than the object in video recognition. In the proposed method, keypoints are detected regionally from the recognition target by using correlation between the weights of the spatio-temporal information and the distance between keypoints for locally obtained keypoint candidates. First, keypoint candidates are detected using spatio-temporal information. Thereafter, the graph cut algorithm is applied to the keypoint candidates, and as the distance between the keypoints is smaller with respect to the smooth term, a larger weighting is performed. As a result, it is possible to detect keypoints only from regions with a high keypoint candidate density. The evaluation results show that the F-measure showing comprehensive evaluation criteria for accuracy and coverage has improved from 17 to 45% when comparing with the Harris detector (Optik, 2014) widely used in recent years, and assuming surveillance camera, augmented reality and in-vehicle camera application in video recognition. It also shows that the proposed algorithm is possible to reduce the 93% keypoints on average.

Chapter 3 proposes a hardware-friendly algorithm and its FPGA implementation method in order to realize real-time processing at the user terminal in cloud video recognition. As a hardware-friendly algorithm, this dissertation proposes keypoint detection based on gradient histogram and density clustering. In conventional keypoint extraction algorithms, computation using pixels around keypoint is required to be performed separately in keypoint detection and descriptor generation. Thus, the amount of computation is large. Regarding calculation of local correlation, the amount of computation is also large because global optimization computation is necessary. In the proposed algorithm, threshold processing is applied to the gradient histogram, and keypoints are detected for portions having large gradient intensities for a plurality of directions. As a result, the second order differential calculation of the luminance using the pixels around the keypoint in the keypoint detection is reduced. For local correlation, by using density clustering, all the keypoints in the grid are detected if the number of keypoints in the grid-divided region is larger than the threshold. If it is smaller than the threshold value, the keypoints in the grid are deleted. This reduces global optimization operations. On the other hand, in the FPGA implementation, parallel architecture of keypoint detection and descriptor generation is proposed based on a gradient histogram. In the conventional implementation method, it is necessary to sequentially perform descriptor generation, and the descriptor generation unit waits for the detection information

output from the keypoint detection unit and calculates in the entire flow of keypoint extraction. It is an issue that needs to be started. In the proposed method, the number of clocks is reduced by using the features of creation of histogram dividing the descriptor area during descriptor generation, parallel computation is performed for every equally spaced pixel that do not enter the same dimension of the histogram. Furthermore, by removing the order dependency of the detection information and executing every clock feature amount generation, parallelization of keypoint detection and feature amount generation is realized. Compared with SIFT (IJCV, 2004) which is a representative keypoint extraction method by software simulation evaluation, repeatability showing the robustness of keypoint detection for image scale change and rotation change is +21% to -13 %, which means that there is no significant performance difference as the keypoint detection performance. In addition, as a result of evaluation of the proposed method including spatio-temporal information and local correlation on the FPGA, it is 427 times faster than the software processing of SIFT using only spatial information, and 18 times faster SIFT's FPGA implementation (ICMCS 2014). Real-time processing at Full-HD 60 fps is realized. Furthermore, it demonstrates that it can operate under the real environment by constructing a demonstration system of the whole video processing connecting the camera and the display to the FPGA board.

In chapter 4, as a conclusion, this dissertation and future stud-

ies are summarized. This dissertation shows the spatio-temporal keypoint extraction based on local correlation for cloud recognition system and FPGA implementation method. Through these proposals and their evaluations, the algorithms that can reduce the amount of data to be sent to the cloud and its real-time verification have been achieved. These results contribute to the realization of cloud video recognition systems as a key technology for processing on the user terminal side. In the future, evaluation of recognition accuracy when applying the proposed algorithm to each application and real-time performance evaluation when considering the entire network are performed.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Cloud-based systems 5

1.3 Problems of conventional methods 7

1.3.1 Detection of unnecessary keypoints for video recognition 8

1.3.2 Large computational demands 9

1.3.3 Serial processing due to data dependency 10

1.4 Proposed concepts 11

1.5 Organization of dissertation 13

2 Spatio-temporal keypoint extraction based on local correlation **15**

2.1 Overview of this chapter 15

2.2 Introduction of keypoint extraction 18

2.2.1 SIFT 19

2.2.1.1 Keypoint detection by the DoG 19

2.2.1.2 SIFT descriptor computation 20

2.2.2 Extension methods of SIFT 21

2.2.3 Keypoint matching 23
2.2.4 Space-time interest point 24
2.2.5 Keypoint extraction based on Harris detector and SIFT descriptor 25
2.2.5.1 The approximation of Hessian detector and using the integral image 26
2.2.5.2 Utilization of multi-scale images in the database 28
2.2.6 Threshold optimization 30
2.2.6.1 Performance evaluation 31
2.3 Spatio-temporal keypoint extraction based on local correlation . 35
2.3.1 Spatial information 37
2.3.2 Temporal information 38
2.3.3 Calculation of KOI utilizing spatio-temporal information 39
2.3.4 Connectivity of adjacent keypoints 40
2.3.5 Calculation of camera-motion invariant optical flow by camera motion estimation 42
2.4 Evaluation results 45
2.5 Applications based on the proposed algorithm 51
2.6 Conclusion 54

3 Full-HD 60 fps FPGA implementation using gradient histogram 56

3.1 Overview of this chapter 56
3.2 Hardware-friendly algorithm based on gradient histogram and density clustering 58

3.2.1 Keypoint detection with descriptor calculation utilizing dual threshold for a gradient histogram 60
3.2.2 The KOI algorithm utilizing gradient histogram-based detection and weighting keypoints and dense-clustering-based connectivity of an adjacent keypoint 61
3.2.2.1 Spatial information 62
3.2.2.2 Temporal information 63
3.2.2.3 Connection of adjacent keypoints 63
3.2.3 Approximation based on SAD, bit-shift and LUT 65
3.3 FPGA implementation based on parallelized gradient histogram 68
3.3.1 60-fps spatial-feature-based keypoint extraction hardware 70
3.3.1.1 RGB to gray converter module 71
3.3.1.2 Gaussian filter module 71
3.3.1.3 Line memory module 71
3.3.1.4 Corner detection module 71
3.3.1.5 Orientation computation module 72
3.3.1.6 SIFT descriptor computation module 73
3.3.2 Architecture of keypoint detection based on the gradient histogram of the descriptor computation module 76
3.3.3 Parallelization of detection module and descriptor module 77
3.3.4 Parallelization of descriptor computation 77
3.3.5 Pipelining by fetching pixel lines from block RAM 78
3.3.6 Matching process by keypoint pipelining 79
3.3.7 Architecture of temporal information calculation and key-point connectivity calculation 79

3.4 Evaluation 83
3.4.1 Evaluation of keypoint detection performance 83
3.4.1.1 Detector performance 83
3.4.1.2 Keypoint clustering performance 87
3.4.2 Evaluation of processing performance 88
3.4.2.1 Evaluation condition 89
3.4.2.2 Evaluation results 89
3.4.2.3 Evaluation of spatial-feature-based keypoint extraction 91
3.4.2.4 Evaluation of spatio-temporal keypoint extraction 93
3.5 Conclusion 97

4 Conclusion **99**

References **i**

Publications **ix**

5

List of Figures

1.1 IoT platform for various application. 2

1.2 Cloud video recognition system 3

1.3 Effectiveness targets of the present paper 5

1.4 Position of this study. 7

1.5 Keypoints detected by spatial keypoint detection and spatiotemporal keypoint extraction 9

1.6 Workflow of keypoint extraction. 10

1.7 Data dependency and serial processing of keypoint extraction. . 11

1.8 Proposed concept. 12

2.1 The conceptual difference of the proposed algorithm. 16

2.2 The flow of keypoint extraction. 19

2.3 The DoG detector. 21

2.4 The SIFT descriptor. 22

2.5 Rate of the computational complexity of SIFT. 23

2.6 Flowchart of the proposed algorithm. 26

2.7 Illustration of the integral image. 27

2.8 Approximated Filter (L_{xx}, L_{yy}, L_{xy}). 28

2.9 The relation of scale change and precision. 29

LIST OF FIGURES

2.10 Proposed matching process. 30

2.11 The comparison between the proposed algorithm and SIFT. (a) Number of correct nearest neighbour matches (Boat sequence). (b) Repeatability score for scale change (Boat sequence). (c) precision-recall curve (Wall sequence). (d) precision-recall curve (Bikes sequence). 33

2.12 The software simulation of keypoint detection by proposal (top) and SIFT (bottom). 34

2.13 The workflow of entire processing 36

2.14 The workflow of weighting on keypoints 39

2.15 The connection of keypoints 41

2.16 Calculation of camera-motion invariant optical flow. 44

2.17 Test sequences 46

2.18 The comparison between (a) conventional keypoint extraction and (b) proposed algorithm in Zoom1, (c) conventional keypoint extraction and (d) proposed algorithm in Pan 49

2.19 The comparison between conventional keypoint extraction and proposed algorithm of recall-precision curve in (a) Fixed1, (b) Track, (c) Fixed3 and (d) In-vehicle 50

2.20 Illustration of calculating the histograms and the result 52

2.21 Histogram for gait recognition. 53

3.1 The flow of proposed algorithm. 60

3.2 Keypoint extraction based on a gradient histogram. 62

3.3 Density clustering of KOI smoothing. 65

3.4 The approximation of the arctan calculation. 67
3.5 The structure of the proposed hardware. 69
3.6 The block diagram of entire structure. 74
3.7 The structure of keypoint extraction module. 75
3.8 Parallelization of histogram computation. 78
3.9 The structure of keypoint matching module. 79
3.10 The difference of software implementation and FPGA implementation. 81
3.11 Memory utilization of keypoint matching. 82
3.12 Grid-region based parallelization of density clustering. 82
3.13 The comparison between the proposed algorithm and SIFT. (a) Number of correct nearest neighbour matches (Boat sequence). (b) Repeatability score for scale change (Boat sequence). (c) ROC curve (Wall sequence). (d) ROC curve (Bikes sequence). . 84
3.14 Performance of keypoint detection with the proposed algorithm and SIFT. 85
3.15 Keypoints which are detected by the proposed algorithm (top) and SIFT (bottom). 86
3.16 Performance of the MRF based method and proposal. 87
3.17 Keypoints of the MRF based method (red keypoints) and proposal (green keypoints). 88
3.18 FPGA (TB-5V-LX330) 90
3.19 Option board (TB-SUB-DVI) 91
3.20 FPGA environment 91
3.21 A system utilizing FPGA board. 97

List of Tables

2.1 The number of keypoints and processing times by the conventional method and the proposed method 47

3.1 The parameter on evaluation condition. 88

3.2 Specification of FPGA board (TB-5V-LX330-DDR2) 90

3.3 Available resources of FPGA device (XC5VLX330-1FF1760C) . 92

3.4 Comparison of processing time (SW or HW/VGA or Full-HD) . 92

3.5 Resource comparison of conventional work and proposal. 93

3.6 Processing time of keypoint extraction. 96

3.7 Performance comparison of conventional works and proposal. . . 96

Chapter 1

Introduction

1.1 Motivation

The Internet of Things (IoT) [1, 2] shown in Fig. 1.1, connects a wide variety of devices to each other through the Internet, sends and receives data, and provides numerous services. The IoT is expected to expand in capacity to work not only with sensor data, but also with multimedia. Because many Internet-connected devices will exchange information with each other, this expansion will increase both downstream and upstream traffic.

Cloud-based services will create particularly large network demands. These services use cloud servers to analyze user data and provide processing such as search, storage, and recognition. Practical cloud services such as Siri have been developed for voice recognition. These services send speech data from user devices to a cloud server. The cloud server classifies the data and sends recognition results to users.

No practical cloud-based video recognition service currently exists. Research on video recognition has examined the use of this technology in applications such as surveillance cameras, in-vehicle cameras, and augmented

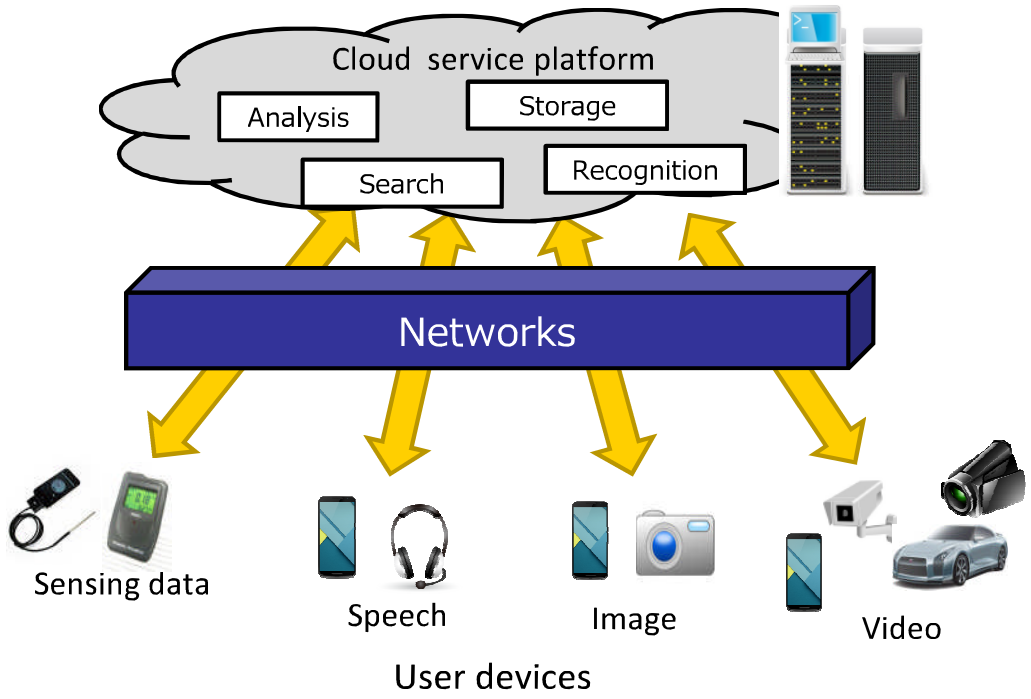

Figure 1.1: IoT platform for various application.

reality (AR) [3, 4, 5, 6, 7, 8, 9, 10]. However, the use of the cloud for these services is not yet practical because of the need to send large amounts of video data. To reduce the amount of data transfer, keypoints can be extracted from images or videos and sent to cloud servers, as shown in Fig. 1.2. Keypoints are points obtained from robust correspondence parts even for transformed images by using image features. The data can then be identified by the cloud servers through machine learning from large databases, and the results can be returned to the users.

Several existing keypoint extraction methods are based only on spatial features and extract many unnecessary keypoints for video recognition. Thus, large amounts of video keypoint data are transferred between user devices and

cloud servers. Moreover, video recognition systems must calculate keypoints from user devices in real time. In summary, current keypoint extraction algorithms have two main problems:

- Detection of unnecessary keypoints for recognition
- Real-time processing

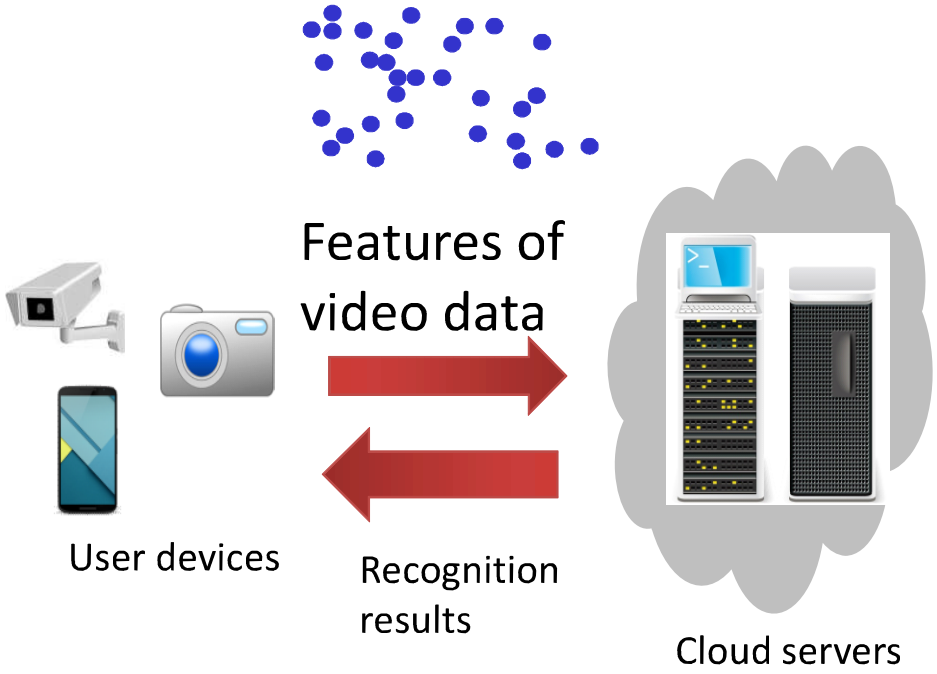

Figure 1.2: Cloud video recognition system

Many conventional keypoint extraction methods [11, 12, 13, 14, 15, 16, 17] use only spatial features. These methods detect unnecessary keypoints for video recognition. Several spatiotemporal feature-based keypoint extraction methods [18, 19, 20] have been proposed. These methods eliminate unnecessary keypoints, including spatial features. Therefore, methods using local correlation are required.

For real-time processing, keypoint extraction is computationally complex. Keypoint extraction consists of two parts: keypoint detection and descriptor generation. Keypoint detection calculates convolution using a filter that processes entire images. Descriptor generation calculates 128-dimension histograms based on intensity gradients. High-speed software implementation is difficult, even using a GPU [21, 22, 23] and a hardware implementation is necessary for real-time processing. This approach requires a low-complexity and hardware-friendly algorithm.

To realize hardware implementation, keypoint extraction must show data dependency between keypoint detection and descriptor generation. The positions of keypoints and their surrounding pixel data must be sent from keypoint detection to descriptor generation. In histogram calculation, the pixel data around keypoints are entered serially into bin calculation and histogram addition modules before calculation. A parallel-architecture method is thus necessary to achieve high-speed processing. The proposed effectiveness target is shown in Fig. 1.3.

Therefore, this thesis proposes a spatiotemporal feature and local correlation based algorithm that extracts only keypoints of interest (KOI). KOI candidates are detected by the Kanade-Lucas-Tomasi (KLT) tracker. KOI are selected by two kinds of features: intensity gradient and optical flow. However, target regions in video recognition applications do not necessarily include large motion and gradient information. Thus, I propose noise reduction using a Markov random field (MRF), which connects adjacent keypoints and determines the keypoint class. The proposed algorithm is also hardware-friendly due to its use of gradient histograms and density clustering [24]. Filter computation

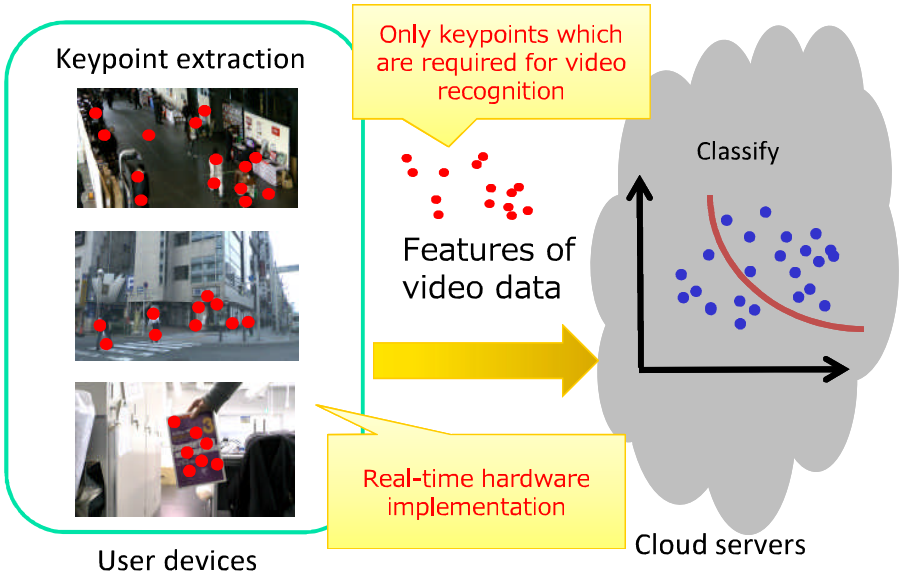

Figure 1.3: Effectiveness targets of the present paper

of keypoint detection is replaced with a gradient histogram, which is reused in descriptor generation. Local correlation is replaced with hardware-friendly density clustering. Finally, I propose an architecture including parallelization of detection and descriptor generation. Conventional methods include a buffer between keypoint detection and descriptor generation. I remove this buffer and parallelize the modules. The performance of the proposed keypoint extraction method is then evaluated through comparison with the processing time and hardware resources of conventional methods.

1.2 Cloud-based systems

This section describes the position of this research. IoT applications using the cloud have been studied in various fields. Several methods have been proposed to reduce the amount of data sent to the cloud. Examples of those related to

speech recognition, sensing data analysis, and image recognition are shown in Fig 1.4.

In speech recognition services such as Siri and Syabetteconcier are already in practical use. They transmit speech data to the cloud, identify them, and return additional information to the user.

In Sensing data analysis, when aggregating information such as air pollution data and system logs in the cloud, several methods [25, 26, 27, 28, 29] for reducing the amount of transmission data by compression and extracting necessary data have been proposed. These are proposals for the problem that the amount of data flowing into the Internet increases comprehensively when a large number of devices are connected to the Internet although the data amount of each device is small. Most of the studies of IoT applications in recent years focus on analysis of sensor data like these.

Regarding image recognition, there is only one related study [30]. This method assumes application to surveillance cameras, detects the face area from the image, and reduces the amount of data by sending only the face area to the cloud. This method does not use time information, in particular, it is a still image. In addition, since only the face area is sent to the cloud, the application is limited to face authentication. These applications have lower real-time requirements, and the amount of data sent to the cloud is not large.

This study is for video recognition systems. This study deals with a method combining time information in addition to spatial information in order to study algorithms for images, which is different from the method of still images. Unlike still images, since images are continuously transmitted, the amount of data sent to the cloud is also very large. Furthermore, this study assumes a

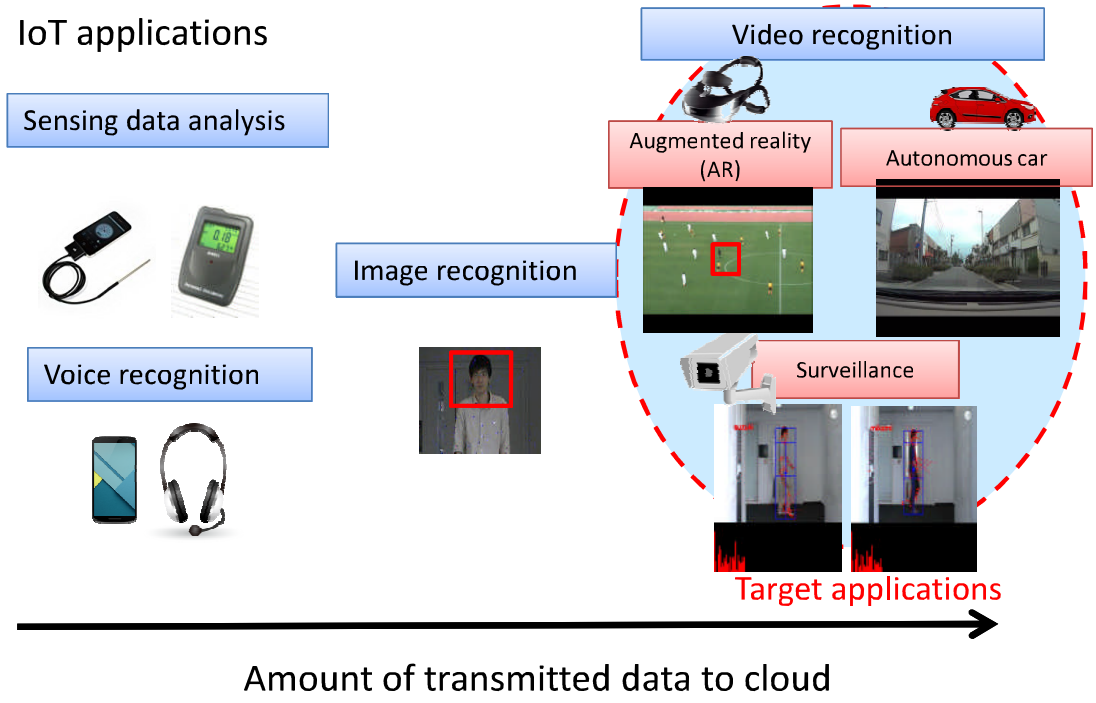

Figure 1.4: Position of this study.

more future applications such as AR, surveillance camera, autonomous car. In consideration of these applications, many studies have been made to improve recognition accuracy. However, few studies aim to realize the entire system. In short, techniques for cloud video recognition have not been studied yet, and for the first time, this study proposes keypoint extraction for region of cloud video recognition.

1.3 Problems of conventional methods

Conventional methods have three main problems for the realization of real-time video recognition: the detection of unnecessary keypoints for video recognition,

large computational demands, and serial processing due to data dependency. The following section describes these problems in detail.

1.3.1 Detection of unnecessary keypoints for video recognition

Conventional keypoint extraction methods based on spatial features detect unnecessary keypoints for video recognition. These methods extract keypoints for both the recognition targets and parts of the images that include large textures. Several spatiotemporal feature based keypoint extraction methods eliminate necessary keypoints that include spatial features. Figure 1.5 presents the keypoints detected by both types of evaluation. The blue points are obtained by spatial keypoint detection, and the red points are obtained by spatiotemporal keypoint extraction. A suitable method must detect keypoints from necessary parts and eliminate keypoints from unnecessary parts.

Several keypoint extraction methods use spatiotemporal information. For example, Laptev and Lindeberg proposed a method using space-time interest points [18]. The methods of Chen *et al.* [19], Willems *et al.* [20] and Huang [31] also employ spatiotemporal information for keypoint extraction. However, these methods are geared toward event detection and extract keypoints only where large gradients move widely, collide with each other, or collide with edges. In addition, a background subtraction method [32] that extracts foregrounds has been proposed. However, much of the texture of still objects is ignored by these methods. No current method continuously extracts keypoints from both moving objects such as humans and objects with unusual texture. One proposed method [33] uses queries to select the keypoints required for

recognition. However, this method is not practical for general object recognition.

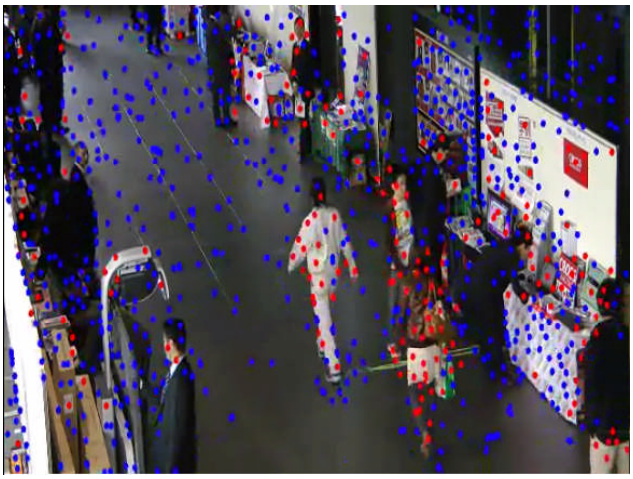

Figure 1.5: Keypoints detected by spatial keypoint detection and spatiotemporal keypoint extraction

1.3.2 Large computational demands

As shown in Fig. 1.6, keypoint extraction consists of two processes: keypoint detection and descriptor generation. Keypoint detection employs filter calculation and threshold decision to determine keypoints by the calculated value of the filter. Descriptor generation calculates multidimensional histograms based on the direction and magnitude of gradients. These two calculations are computationally complex, and their computational demands increase as the square of filter size. A hardware implementation is necessary, and several hardware implementations [34, 35, 36, 37, 38] have been proposed. However, these methods are targeted for spatial keypoint extraction, and no implementation has been developed for spatiotemporal keypoint extraction. The target

video consists of small VGA images at a low frame rate. For implementation, a low-complexity and hardware-friendly algorithm is necessary.

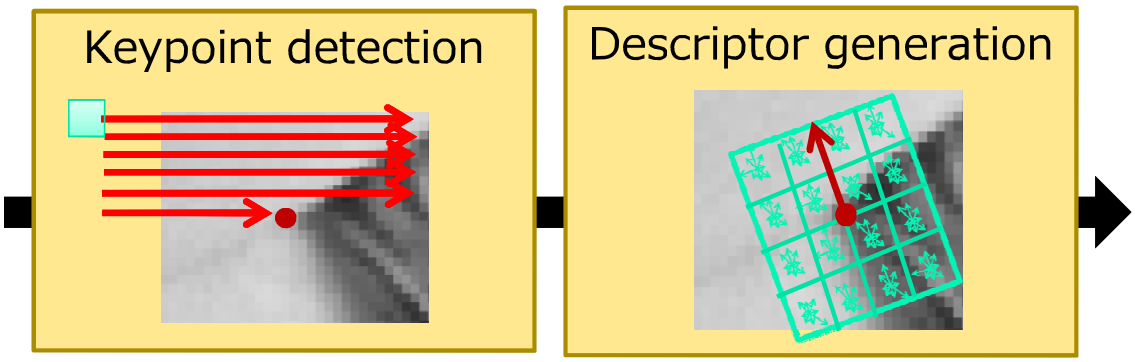

Figure 1.6: Workflow of keypoint extraction.

1.3.3 Serial processing due to data dependency

Figure 1.7 shows the data dependency and serial processing of keypoint extraction. Keypoint extraction exhibits data dependency between keypoint detection and descriptor generation. The data consist of the pixels around the detected keypoints; large amounts must be contained over a long clock time. The histogram generation serially accesses pixel data around the keypoints. Bins are calculated using gradient directions, and the magnitudes of the gradients are added to histogram. For this reason, many existing methods store pixels around the feature point in memory after extraction, read data sequentially from the memory, and calculate the feature amount. This approach decreases hardware implementation speed. A parallel architecture method is thus necessary to achieve high-speed processing.

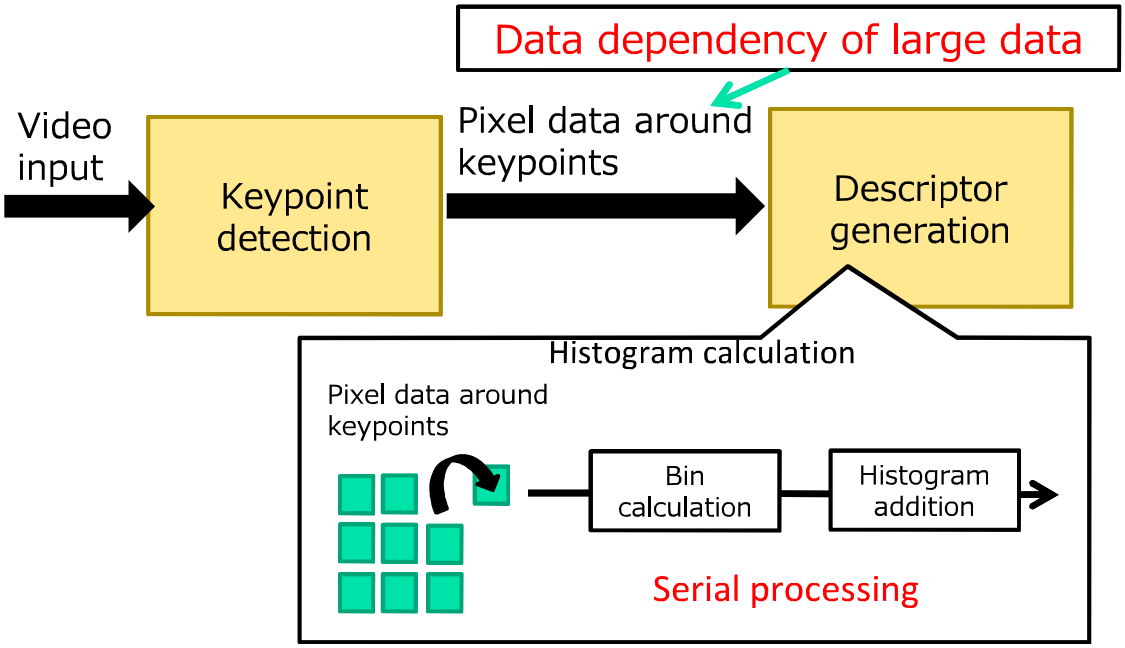

Figure 1.7: Data dependency and serial processing of keypoint extraction.

1.4 Proposed concepts

Figure 1.8 shows overall of the proposed concepts.

First, this paper proposes an algorithm that combines local correlation with spatiotemporal features to realize a keypoint detection method that detects only the necessary keypoints for video recognition, referred to as KOI. The method can detect keypoints from regions that include large motion and gradient features. The proposed algorithm weighs each keypoint by luminance and motion information. The method acquires the gradient based on luminance and acquires the optical flow based on motion using the KLT tracker. Using the obtained weights, the method extracts the KOI candidate points. Subsequently, using MRF and the graph cut algorithm, the method assigns strong weights to nearby keypoints and performs clustering.

Second, I propose a hardware-friendly algorithm to realize real-time hardware implementation. The algorithm employs a gradient histogram, which is normally calculated during descriptor generation for the calculation of keypoint detection. The gradient histogram is efficiently reused for two processing steps. Local correlation, which is difficult to implement on hardware, is replaced with density clustering, which has lower complexity. Using the gradient histogram for keypoint detection, the method obtains keypoints at positions equivalent to those obtained by general corner detection [39] using two thresholds. For density clustering, I propose an algorithm that divides the image into a plurality of grids. Keypoints in each grid are determined to be KOI when they exceed the threshold value.

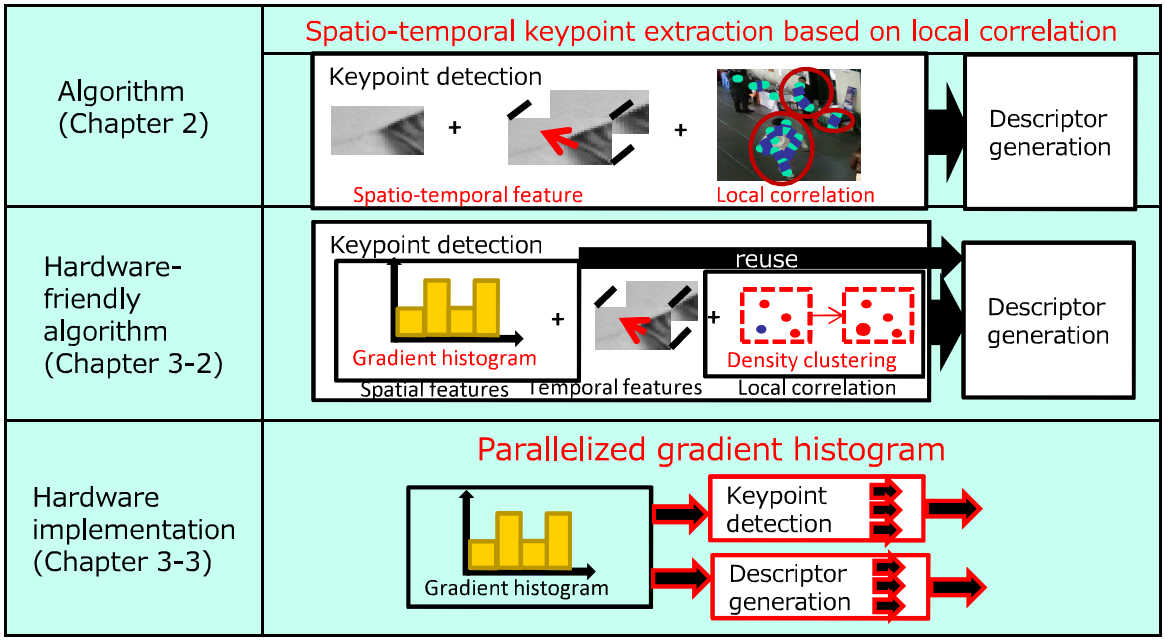

Figure 1.8: Proposed concept.

Finally, I propose a hardware implementation based on bufferless architecture with parallelization of detection and descriptor generation. The proposed

architecture calculates the gradient histogram at the beginning of the flow, then sends it to the keypoint detection module and descriptor generation module in parallel. Because pixel data is first converted into gradient histogram data, the number of registers and computing units held in the pipeline can be reduced. For descriptor generation, pixels that do not depend on other data are processed in parallel using the grid division characteristics of the descriptors. This approach eliminates the data dependency of keypoint extraction to realize a low-resource parallel architecture.

1.5 Organization of dissertation

This thesis is composed of each chapter below.

Chapter 1 provides background for the study, including the IoT, cloud video recognition systems, the issues of existing keypoint extraction methods, and an outline of the proposed method. In the cloud video recognition system, there are two problems of large amount of video data to be sent from user terminal to cloud server and real-time processing of keypoint extraction in user terminal. Chapter 2 has solved the first problem of amount of data to be sent and chapter 3 has solved the second problem of real-time processing. This thesis consists of three full papers. Capter 2 consists of one paper [40] and chapter 3 consists of two papers [41, 42] whose contents are hardware implementation of keypoint extraction and its expansion to the spatio-temporal keypoint extraction.

Chapter 2 describes the proposed spatio-temporal keypoint extraction algorithm based on local correlation, which detects only the necessary keypoints for video recognition. Keypoint extraction combining temporal and spatial

features with local correlation is described in detail. The proposed method is based on a low-computation algorithm. Therefore, after describing a low-complexity algorithm using spatial features, I describe a method combining spatio-temporal features and local correlation.

Chapter 3 presents a FPGA implementation including a hardware-friendly algorithm and its parallelized architecture for real-time processing using gradient histogram. This chapter first describes the hardware-friendly algorithm and then explains the hardware architecture using a detailed block diagram. The proposed method is characterized by a bufferless architecture that employs parallelization of gradient histogram and low computational complexity.

Chapter 4 concludes this paper and describes future research directions. The proposed method could reduce feature quantity in cloud video recognition and realize real-time processing on the terminal side. I also introduce the recognition process, which is a future subject of research.

Chapter 2

Spatio-temporal keypoint extraction based on local correlation

2.1 Overview of this chapter

Conventional keypoint extraction [11, 12, 13, 14, 15, 16, 17] utilize only local spatial features. These methods detect unnecessary keypoints for video recognition. Several spatio-temporal feature based keypoint extractions [18, 19, 20] have been proposed. These methods eliminate necessary keypoints which include spatial feature. Thus, the proposed method detects many keypoints from necessary parts for video recognition using local correlation in addition to spatio-temporal features. Figure 2.1 presents the conceptual difference of the proposal.

In this paper, we first introduce keypoint extraction. In introduction, we explain about SIFT which is a representative keypoint extraction method and describe its extension methods. Furthermore, before proposing the KOI extraction method, we propose a low computational keypoint extraction method

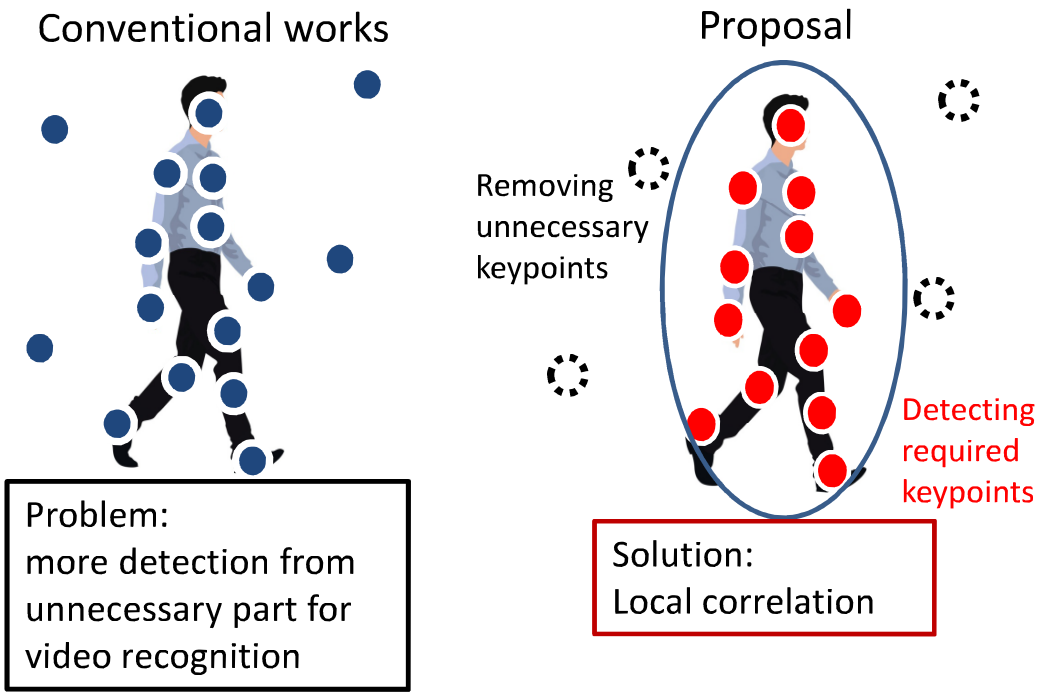

Figure 2.1: The conceptual difference of the proposed algorithm.

which is easy to give spatio-temporal information. In this study, we propose KOI extraction method based on the algorithm.

This chapter mainly describes the proposed keypoint extraction algorithm that detects only KOI based on spatio-temporal features and the Markov Random Field (MRF). The KOI extraction is composed of three elements: spatial information, temporal information and connectivity of adjacent keypoints. This algorithm contains a keypoint selection part between the keypoint detection part and the descriptor generation part. The proposed method includes an approximated KLT tracker to calculate positions of keypoints and optical flows simultaneously. This algorithm calculates weights at each keypoint using two kinds of features, namely, intensity gradient and optical flow. Then, these

values are arranged and keypoint class is determined by a threshold. However, the results include a large amount of noise because required parts do not necessarily include motion and gradient. It reduces noise of extraction by comparing states of the intended keypoint with states of surrounding keypoints. Camera motion estimation is added to the algorithm and it calculates camera-motion invariant optical flows.

The evaluation results using eight videos including pan, zoom and track show that the proposed algorithm achieves 93% reduction of keypoints and 76% reduction of computational complexity in comparison with a conventional keypoint extraction. KOI are extracted in the region whose motion and gradient are large. The results also confirm that the proposed algorithm extracts a number of keypoints from defined parts.

2.2 Introduction of keypoint extraction

Keypoint extraction is utilized for recognition and finding corresponding point between two images. Keypoints are points obtained from robust correspondence parts even for transformed images by using image features. By matching the keypoints between two images, we check the objects in the image. Normally, keypoints are detected at places where the gradient is large with respect to two directions as in corner detection. As shown in Fig. 2.2, the algorithm is divided into following two key parts.

- Keypoint detection
- Descriptor generation

The keypoint detection is a process which decides keypoint's position near characterized region. Keypoint detection employs filter calculation. The output value of filter is binarized by threshold and decides keypoints. The SIFT descriptor generation calculates histograms with information about neighboring region. It calculates multi-dimension histogram based on direction and magnitude of gradient. SIFT [11] divides one region into 4×4 and calculates 8 dimension histogram. It generates total of 128 dimension vector. The evaluation of this paper utilizes the SIFT descriptor as a conventional method. SIFT is cited by large number of papers and used as a benchmark of keypoint extraction.

This section shows the algorithms of SIFT, expanded methods of SIFT and keypoint extraction based on Harris detector and SIFT descriptor. Before proposing the spatio-temporal keypoint extraction based on local correlation,

we proposed simplified keypoint extraction which is easy to extend for spatio-temoral methods. The keypoint extraction based on Harris detector and SIFT descriptor is simple and it has low number of dependencies between keypoint detection and descriptor generation. The algorithm has no dependency of scale like SIFT by dealing with scale transformation with database. In addition, the Harris detector is easy to extend for time information acquisition which is used by KOI extraction. Thus, our algorithm is based on it.

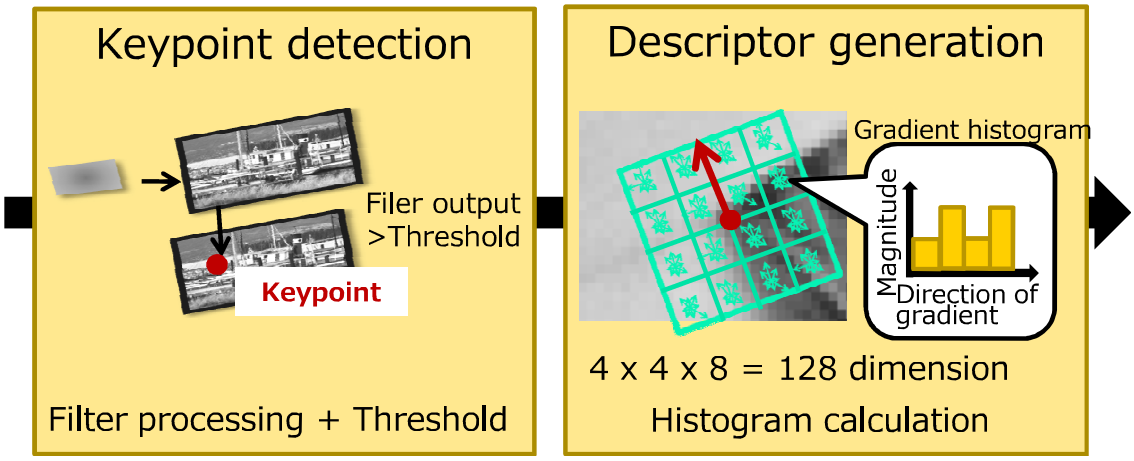

Figure 2.2: The flow of keypoint extraction.

2.2.1 SIFT

2.2.1.1 Keypoint detection by the DoG

SIFT detects scale invariant keypoints by the DoG function. DoG function computes difference of images convolved by Gaussian filters. An image, $I(x, y)$, a variable-scale Gauss function, $G(x, y, \sigma)$, and a smoothed images, $L(x, y, \sigma)$,

define the DoG image, $D(x, y, \sigma)$:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma), \end{aligned} \tag{2.1}$$

where $*$ is the convolution operation. $D(x, y, \sigma)$ is repeatedly computed by a constant multiplicative factor k. Its computational complexity becomes higher and higher when σ increases. Thus, this process is very complex. After this, detections of extreme value and localizations of keypoints are performed. They also require high computational complexity because localizations use matrix calculation. Figure 2.3 is a schema of the DoG detector. The smoothed images are on the top and DoG images are at the bottom. Figure 2.5 shows the rate of the computational complexity of SIFT. We can see the DoG process which is the detection part requires large amount processing time. The processing time depends on the number of keypoints. However, this figure shows the average rate. Thus, we proposed the algorithm based on this result in next chapter.

2.2.1.2 SIFT descriptor computation

In computation of SIFT descriptor, firstly, keypoint's orientation is obtained. The histogram is calculated by gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$:

$$m(x, y) = \sqrt{L_x(x, y) + L_y(x, y)}, \tag{2.2}$$

$$\theta(x, y) = tan^{-1}\frac{L_y(x, y)}{L_x(x, y)}. \tag{2.3}$$

When its sum of magnitude is max, the orientation becomes the keypoint's one. After this, SIFT descriptor is computed. The region is rotated by the

Figure 2.3: The DoG detector.

keypoint's orientation. The size of region depends on scale obtained by the DoG detector. It is divided into 4×4 and histogram is computed by 8 directions in each region. Total 128 dimension vector, SIFT descriptor, is generated. This process's computational complexity changes depending on keypoint's scale. Figure 2.4 is schema of SIFT descriptor. The length and direction of arrows shows the magnitude and quantized direction of the gradient.

The computational complexity of SIFT descriptor depends on the number of keypoints. Thus, if the number of keypoint is large, the processing time also becomes long.

2.2.2 Extension methods of SIFT

There are many method which expand the SIFT algorithm.

Speeded-Up Robust Features (SURF) [12] and approximated SIFT [13] are proposed as the speed-up methods. Especially, the SURF is low complexity

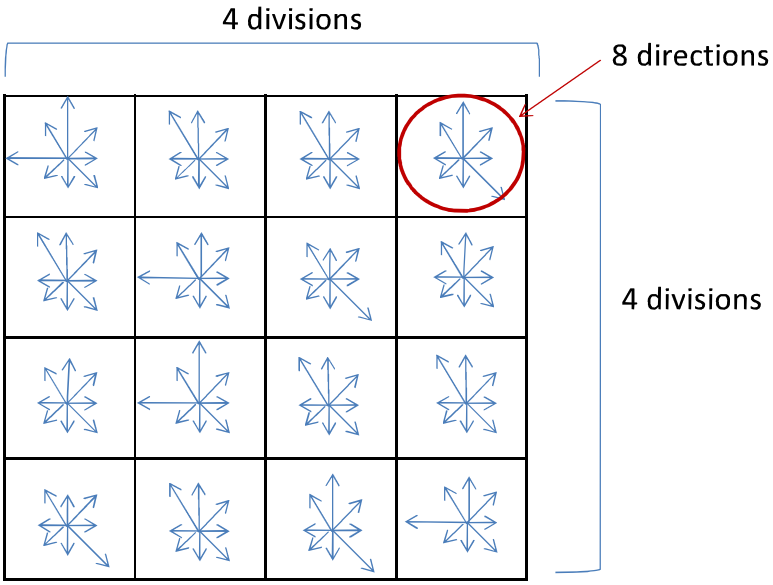

Figure 2.4: The SIFT descriptor.

keypoint extraction algorithm using the box filter and integral image. DoG processing was performed in SIFT, DoH (Determinant of Hessian) is used in the case of SURF. When calculating the Hessian matrix, filtering for calculating second-order differentials is approximated as a Box filter and filtering of various sizes is performed by using an integral image. As a result, the feature extraction is much faster than SIFT. Calculate the brightness gradient by calculating Haar-like as the feature quantity. Orientation is calculated while rotating Haar-like and adopts the angle at which the value becomes the largest. In the feature quantity description, a square region centered on a feature point is divided into 4 $\times$ 4, a luminance gradient is calculated in each grid, and a four-dimensional vector is calculated. Therefore, a total of 64 dimensional feature vectors are obtained. There is also a method called U-SURF that speeds up at the expense of this rotation invariance of SURF. In software, it can per-

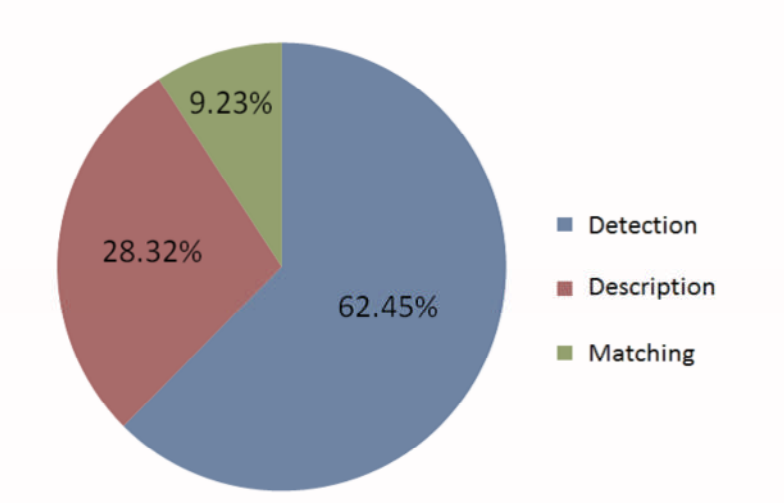

Figure 2.5: Rate of the computational complexity of SIFT.

form faster than SIFT. However, integral image utilizes a lot of memory. Thus, when it is difficult to implement it on low-cost FPGA. SIFT which includes many filtering process is better to implement.

GLOH [14], PCA-SIFT [15], CSIFT [16] and ASIFT [17] are proposed as the method considering accuracy. Especially, PCA-SIFT is used to reduce the dimension of the descriptor. In this algorithm, SIFT descriptor is reduced into 36 dimension vector. PCA is also used for recognition process. Usually, the number of dimension of histograms becomes large. In this case, the dimension is reduced by PCA. In the application of recognition, this algorithm is often used. In the chapter of evaluation results, the application using PCA is shown.

2.2.3 Keypoint matching

It is the keypoint matching that associates the obtained keypoints with each other. The feature quantities described in each are 128 dimensions, but consider this as a feature vector and calculate the distance d with all registered

keypoints v. It is expressed by the following equation.

$$d = \sqrt{\left(v_i^1 - v_r^1\right)^2 + \left(v_i^2 - v_r^2\right)^2 + \cdots + \left(v_i^{128} - v_r^{128}\right)^2}, \tag{2.4}$$

where v_i is the descriptor of the keypoint obtained from the input image, and v_r is the descriptor of the keypoint obtained from the registered image. The number on the right shoulder is the dimension number. It is a process with a relatively small amount of computation on software. In the general nearest neighbor search, the feature vector with the smallest d in this equation is taken as the nearest neighbor.

2.2.4 Space-time interest point

There are several keypoint extraction methods utilizing spatio-temporal information. Laptev and Lindeberg proposed space-time interest points [18]. The methods of Chen *et al.* [19] and Willems *et al.* [20] also utilize the spatio-temporal information for keypoint extraction.

The methods utilize 3D filter which expands the Harris corner detector to temporal dimension. The detector is calculated by

$$\mu = \mathbf{G}(\cdot\,; \sigma_l^2, \tau_l^2) * \begin{bmatrix} L_x^2 & L_xL_y & L_xL_t \\ L_xL_y & L_y^2 & L_yL_t \\ L_xL_t & L_yL_t & L_t^2 \end{bmatrix}, \tag{2.5}$$

$$H = det(\mu) - \omega tra^3(\mu). \tag{2.6}$$

It can extract keypoints in only parts which large gradient moves widely, large gradients collide with each other and large gradients collide with edges. It is applied to the recognition of human action, for example, walk, hand waving,

etc. Thus, their target is event detection. It cannot detect the keypoints which do not move. Much texture of still objects is ignored and there is no method which extracts keypoints from the moving objects like humans continuously and objects which have outstanding texture.

2.2.5 Keypoint extraction based on Harris detector and SIFT descriptor

Keypoint extraction based on Harris detector and SIFT descriptor which is simple method and easy to extend for time information acquisition is shown. Our proposed keypoint extraction which is shown in next section is based on this method. We shows the evaluation comparing with SIFT. The algorithm utilizes two techniques below.

- Approximation of Hessian detector and using the integral image
- Utilization of multi-scale images in the database

The flowchart which summarizes the process of this algorithm is shown in Fig. 2.14. The DoG detector is the highest computational complexity part in SIFT algorithm as shown in section 2. However, the keypoints obtained by DoG is positioned near corners in an image. Therefore, we propose that the DoG is replaced with corner detection. The computational complexity is drastically reduced by this because corner detection is relatively low complexity. The gaussian filter in Fig. 2.14 is used for the noise reduction. When SIFT descriptor is computed, the size of described regions is a constant 15×15. This also simplifies SIFT, because the regions increase in size depending on scale in the case of SIFT.

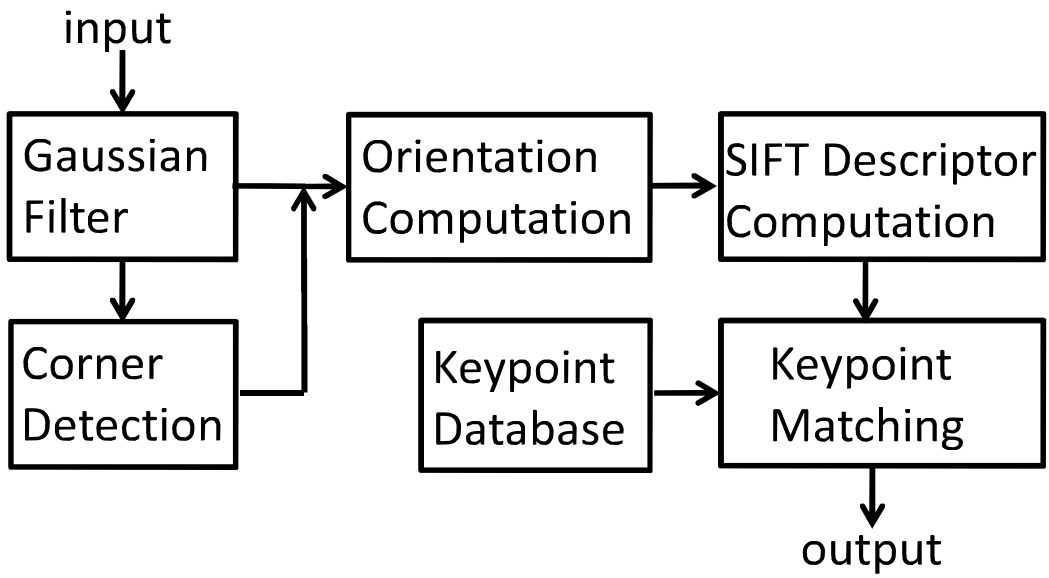

Figure 2.6: Flowchart of the proposed algorithm.

2.2.5.1 The approximation of Hessian detector and using the integral image

Hessian-based keypoint detector [43] is one of the corner detection methods. Its positioning corner is very suitable. It uses filter which computes 2nd-order difference of adjacent pixels. It needs to refer many adjacent pixels during detection from general images with noise. According to the number of referred pixels, the processing time becomes very long. Thus, an integral image and box filter are utilized for speeding up.

First, we describe an integral image. The integral image, II, is the sum of pixels from top left corner of image to intended pixel (x, y):

$$II(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,j). \tag{2.7}$$

When the sum of the pixels of rectangular region S in Fig. 2.7 is calculated,

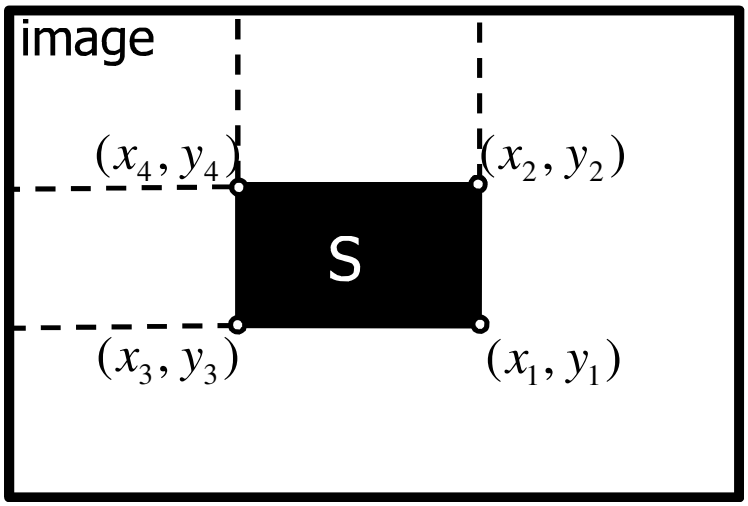

Figure 2.7: Illustration of the integral image.

it is represented by

$$S = II(x_1, y_1) - II(x_2, y_2) - II(x_3, y_3) + II(x_4, y_4). \tag{2.8}$$

This method accelerates the calculation of rectangular region's sum. It can be processed by only 4 accesses of the integral image. Moreover, there is a merit that the process time does not depend on the size of region. It has many merits during software processing. However, in the case of hardware, it is difficult to reserve memory because integral image uses a lot of memory. Thus, it does not use integral image.

Next, corner detection by box filter is shown. We use Hessian matrix, $\mathbf{H}$, which is composed of elements are the 2nd-order difference of adjacent pixels:

$$\mathbf{H} = \mathbf{G}(\sigma) \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix}. \tag{2.9}$$

In general, their elements are weighed by Gaussian function, $\mathbf{G}(\sigma)$. However, it is not suitable for an integral image because weighing is detail. Thus, this

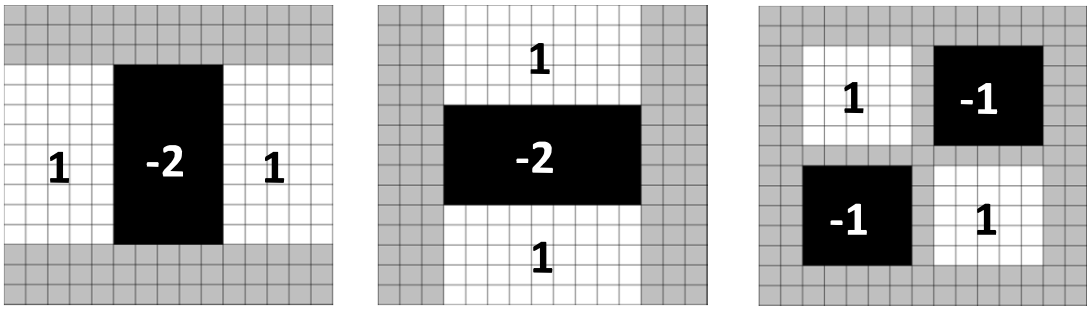

Figure 2.8: Approximated Filter (L_{xx}, L_{yy}, L_{xy}).

filter is approximated and it becomes easy to compute by integral image. Approximated filter is shown in Fig. 2.8. L_{xx}, L_{yy}, L_{xy} are obtained by filter process of integral image. After that, they are used to compute the function which decides corners. When the position is a corner, it satisfies the equation,

$$det(\mathbf{H}) - \omega tra(\mathbf{H}) > T, \tag{2.10}$$

where ω is a parameter and T is a threshold. If the threshold becomes larger, corners decreases and becomes better position as corners. It is adjusted to keep the number of keypoints optimal.

2.2.5.2 Utilization of multi-scale images in the database

This proposed algorithm removed the DoG detector of SIFT. In other words, it does not compute scale of each keypoint. It is impossible to deal with scale changes as it is. Thus, next, we propose the solution that prepares various sizes images in the database and decides the scale during keypoint matching. SIFT descriptor can deal with some scale-changes because it is very robust for various image changes or transformations. Experimental result shows SIFT descriptor whose described region is fixed deals with 0.5-1.5 scale object. Considering this feature, we prepare three images of various size at regular inter-

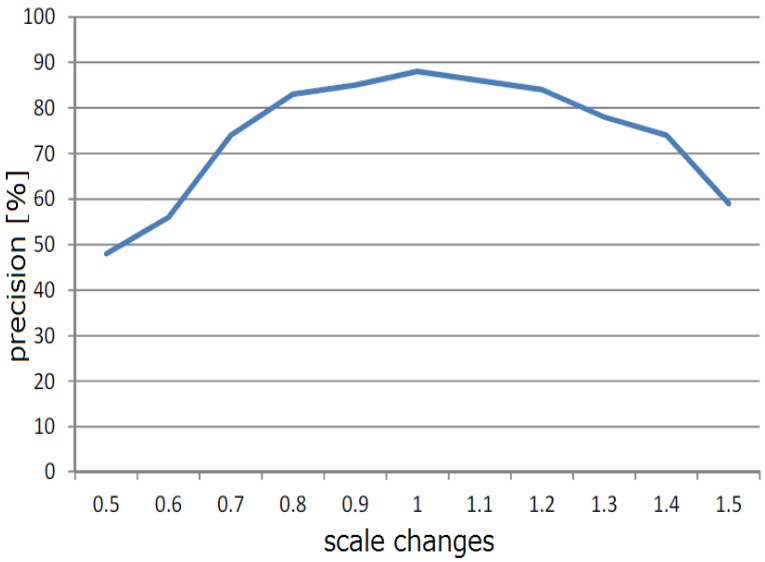

Figure 2.9: The relation of scale change and precision.

vals. For example, images of three sizes (1, 1.5, 0.5) are registered as objects of matching. These values are selected by the experiments in Fig. 2.9 by one template matching. It shows the precision is 50% when the scale is 0.5 or 1.5. In these cases, near template image is selected and the compensates the precision. Keypoints are extracted from them and tag is added to keypoints. The tag (=1, 1.5, 0.5) shows which size image the keypoint is obtained from. In keypoint matching process, tag is checked and counted if registered keypoint matches with input image's kepoints. The tag with the most matches is considered as nearly input image's scale. Finally, the input image matched with the registered image of decided tag. Keypoint matching process is speed-up by Approximated Nearest Neighbor (ANN) [44] in comparison with Nearest Neighbor (NN). NN is a computation of distance between two feature vectors. In the case of NN, it searches all keypoints, but the process can be avoided by approximation of ANN. Software processing uses ANN, but hardware uses NN

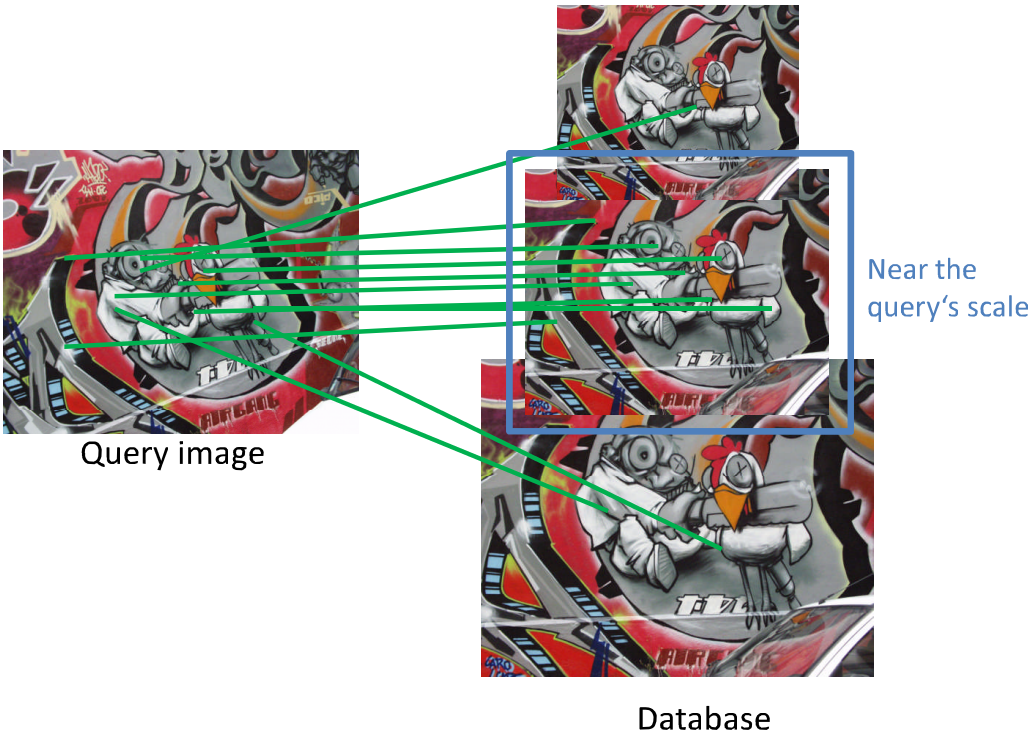

Figure 2.10: Proposed matching process.

to simplify its structure. Figure 2.10 is a schema of this process.

2.2.6 Threshold optimization

SIFT descriptor is computed for all keypoints in this process. Thus, process time depends on the number of keypoints. The more keypoints we extract, the higher computational complexity becomes. However, there is an appropriate number of keypoints to accomplish keypoints matching with sufficient accuracy. It is not necessary to extract too many keypoints. Therefore, keypoints are localized by optimizing T of equation (2.13) in every frame. It is

the current frame and $t-1$ is the last frame. T is optimized as the equation:

$$T(t) = \begin{cases} T(t-1) & \text{if} \quad n_l < n < n_h \\ T(t-1) + \Delta T & \text{if} \quad n \geq n_h \\ T(t-1) - \Delta T & \text{if} \quad n \leq n_l \end{cases} . \tag{2.11}$$

n_h, n_l are upper bound and lower bound of number of keypoints. ΔT is parameter which adjusts T.

2.2.6.1 Performance evaluation

The keypoint extraction based on Harris detector and SIFT descriptor and SIFT are examined in performance on the test sequences. For evaluation of correct matches and repeatability in Fig. 2.11 (a) and (b), we use the framework proposed by Mikolajczyk *et al.* [45]. The boat image shown in the paper is used as a test sequence. It includes both scale changes and rotation. The correct transformation matrix is obtained by RANdom SAmple Consensus (RANSAC) [46]. When L2 distance between the correct value and experimental value is under 10, the pair is regarded as a successful match. Figure 2.11 (a) shows the correct matches. The stability of descriptor is measured from this. The proposed algorithm is a little unstable but obtains more correct correspondences on many scales compared with SIFT. Figure 2.11 (b) is the repeatability. It means if keypoints are detected at the same positions between two images with a viewpoint change. It shows performance of detector. We can observe that the proposed corner detection obtained a lot of correspondences and better performance than the DoG detector of SIFT. Figure 2.12 is the software simulations of keypoint detection. The proposal is almost same performance with SIFT, but it is the difference that the proposal detects

more keypoints at corner. It has a problem that the proposal does not detect many keypoints from images that do not include many texture. To measure the 1-precision vs. recall, we use the framework proposed by Mikolajczyk and Schmid [47]. Figure 2.11 (c) is an evaluation on wall images. This images include many gradients. In this case, SIFT has the ideal curve and the proposed algorithm also has the comparable curve. Figure 2.11 (d) is an evaluation in bike images. This images are natural scenes. Both algorithms are not ideal results. The curve of the proposed algorithm is the more gradual curve. SIFT has better performance in these scene.

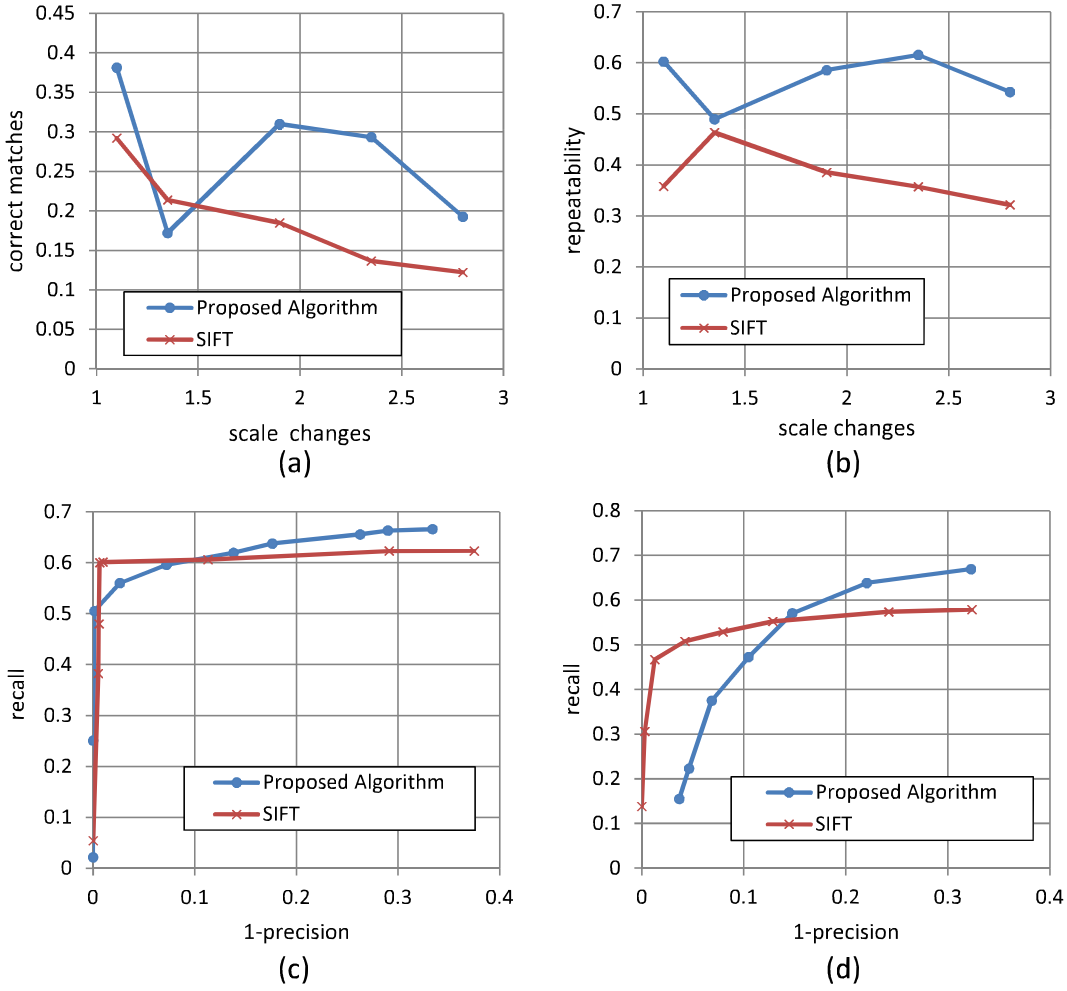

Figure 2.11: The comparison between the proposed algorithm and SIFT. (a) Number of correct nearest neighbour matches (Boat sequence). (b) Repeatability score for scale change (Boat sequence). (c) precision-recall curve (Wall sequence). (d) precision-recall curve (Bikes sequence).

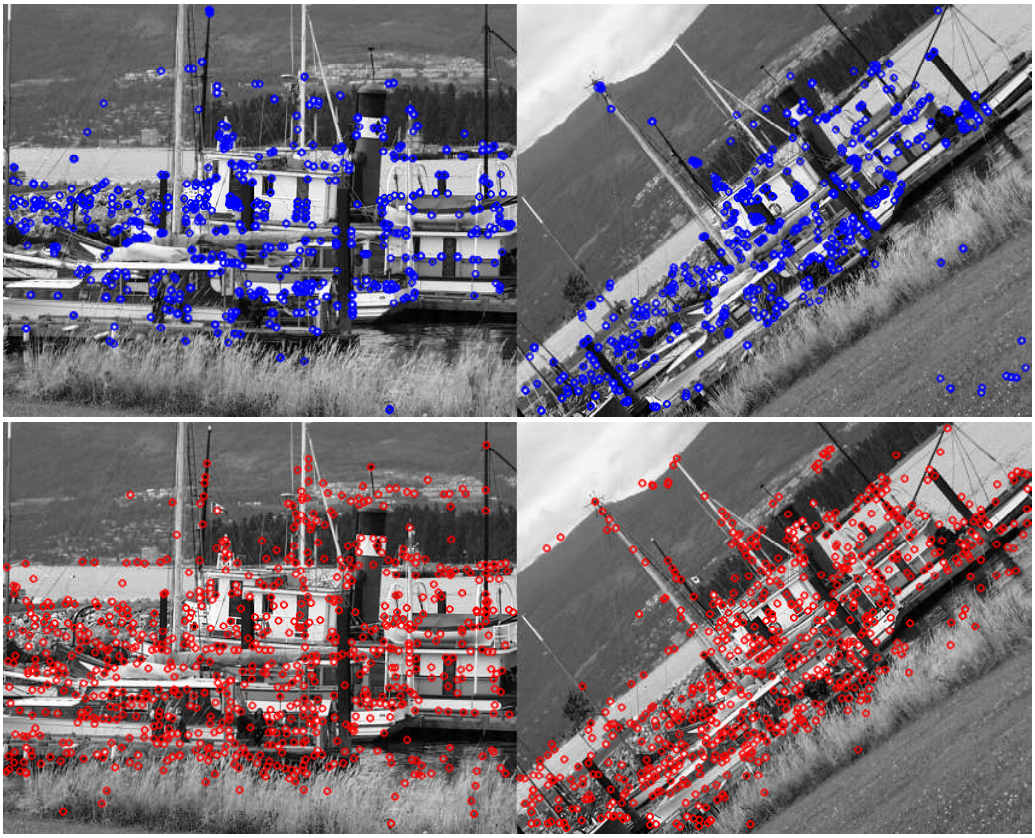

Figure 2.12: The software simulation of keypoint detection by proposal (top) and SIFT (bottom).

2.3 Spatio-temporal keypoint extraction based on local correlation

In this section, we show the method that extracts KOI from input videos. KOI extraction is composed of three elements below.

- Spatial information
- Temporal information
- Connectivity of adjacent keypoints

Spatial information is often used to extract image transformation invariant keypoints. This paper adds other two elements. In videos, temporal information is added to extract keypoints in the parts which include motions. In addition, this paper utilizes the connectivity of adjacent keypoints because a number of keypoints must be extracted from one object part.

The entire workflow containing these elements is shown in **Fig.** 2.13. We choose the KLT tracker [48, 49] as a keypoint detection method because it simultaneously calculates positions of keypoints and optical flow which is utilized in keypoint selection part. This algorithm contains a keypoint selection part between the keypoint detection part and the descriptor generation part. In the keypoint selection part, first, this algorithm weights keypoint by two elements and calculates values which describe likelihood of KOI at each keypoint. Then, these values are arranged and keypoint class is determined by a threshold. However, the results include a large amount of noise because required parts do not necessarily include motion and gradient. Thus, keypoints are connected by MRF and the graph cut algorithm is used to reduce

noise from the output keypoints. In addition, to deal with moving cameras, motion compensation is executed by camera motion estimation and camera-motion invariant optical flows are extracted. The subtractions of influenced optical flows by camera motions and obtained optical flow are minimized. As results, the camera motion and the camera motion-invariant optical flows are obtained. The SIFT descriptor is calculated at only selected keypoints. This section shows each algorithm in more detail based on three elements.

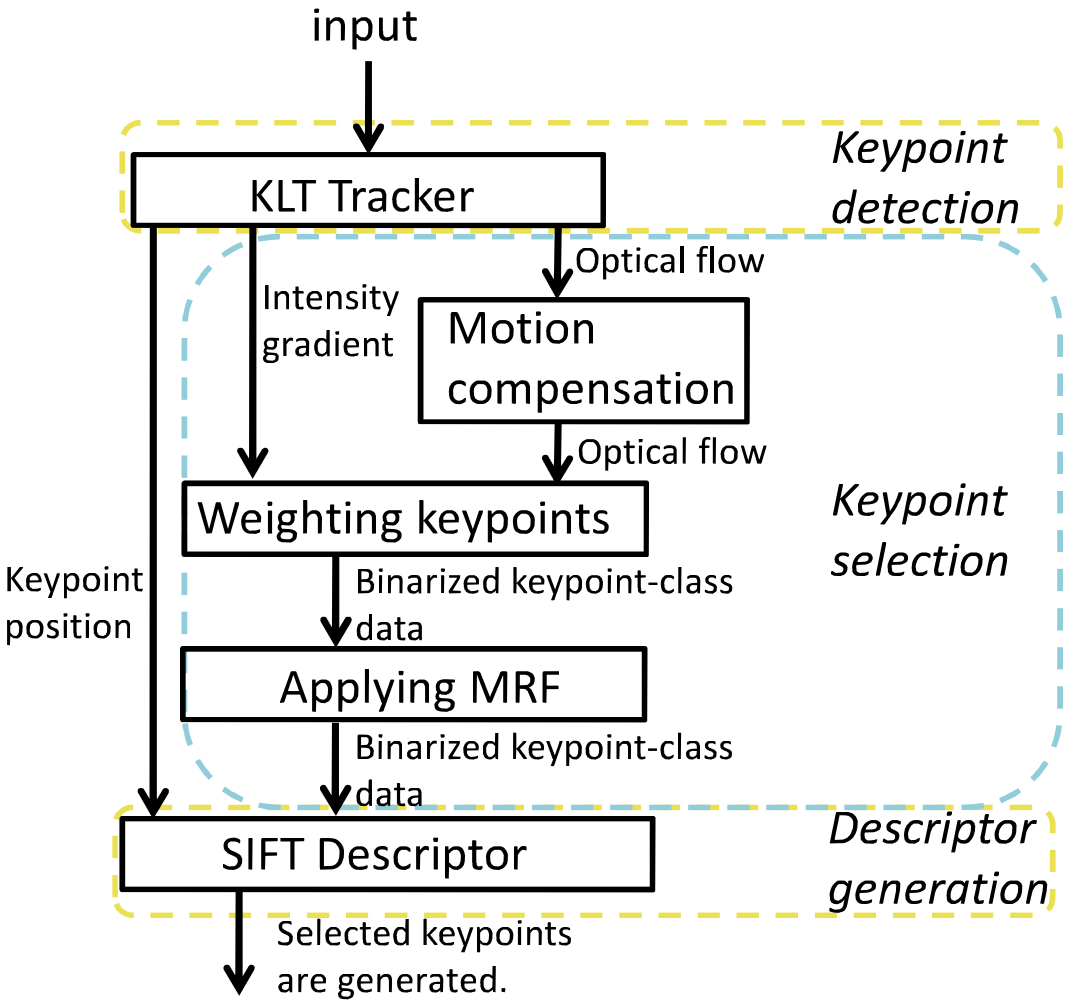

Figure 2.13: The workflow of entire processing

2.3.1 Spatial information

As keypoint detection method, the KLT tracker is utilized. A KLT tracker is one of the algorithms which detect keypoints and calculate optical flows. It uses filters which compute second-order difference of adjacent pixels. It needs to refer many adjacent pixels during detection from general images with noise. According to the number of referred pixels, the processing time becomes long. Thus, an integral image and a box filter are utilized for speeding up.

First, keypoint detection by a box filter is shown. We use Hessian matrix, $\mathbf{H}$, which is composed of elements are the second-order difference of adjacent pixels:

$$\mathbf{H} = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix}. \tag{2.12}$$

In general, their elements are weighted by Gaussian function. However, it is not suitable for an integral image because weights have to be determined at each pixel. Thus, this paper utilizes approximated BOX filters and it becomes easy to compute by an integral image. This approximation is also used by SURF. L_{xx}, L_{yy}, L_{xy} are obtained by filter process of integral image. After that, they are used to compute the function which decides corners. When the position is a corner, it satisfies the equation,

$$V = det(\mathbf{H}) - \omega\ tra(\mathbf{H}) > T, \tag{2.13}$$

where ω is a parameter and T is a threshold. If the threshold becomes larger, corners decreases. It is adjusted to keep the number of keypoints optimal.

2.3.2 Temporal information

After the keypoint detection, the KLT tracker calculates optical flows. Optical flows are also calculated by second-order difference of adjacent pixels. Thus, the Hessian matrix is reused. A optical flow, $[u, v]$, is calculated by the equation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix}^{-1} \begin{bmatrix} L_{xt} \\ L_{yt} \end{bmatrix}. \tag{2.14}$$

In addition to gradient information, it utilizes the frame difference. This calculation also uses the integral image.

In this paper, we choose two elements for weighting keypoints. The elements are an intensity gradient and an optical flow. With respect to the intensity gradient, there is a high possibility that objects with many intensity gradients is the recognition targets. For example, book covers, posters and traffic signs are pointed out. With respect to the optical flow, there is a high possibility that objects with large motion are the recognition targets. For example, human, animals and vehicles are pointed out. Conventional keypoint extraction algorithms generally utilize only gradient information. Thus, it is expected to extract important keypoints including motion information if we use the temporal information. The weights of two elements are calculated at each keypoint which is obtained by the KLT tracker. The two different weights are normalized and summed up. This flow is described in **Fig.** 2.14.

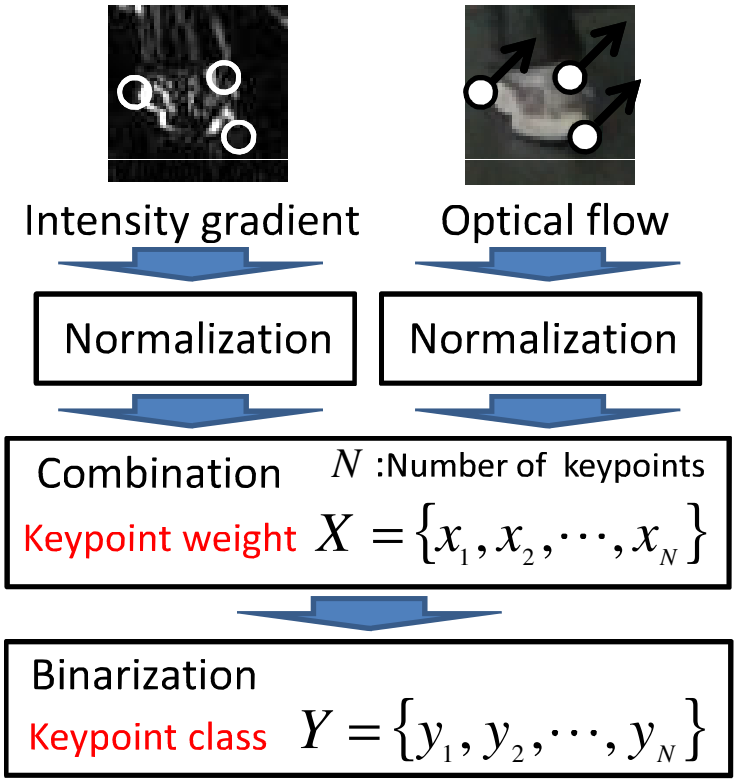

Figure 2.14: The workflow of weighting on keypoints

2.3.3 Calculation of KOI utilizing spatio-temporal information

A way to obtain the weights of keypoints are shown next. The weight of intensity gradients are calculated by the Hessian detector [43]. The value, V, has already calculated in **Eq.** (2.12) and **Eq.** (2.13). V of **Eq.** (2.13) describes the strength of intensity gradient. It is obtained by the corner detection part of keypoint extraction. On the other hand, weights of optical flows are calculated by norm of optical flow. The value is obtained by **Eq.** (2.14). This calculation is a low complexity because the values have been already calculated. These two values are calculated at each keypoint and summed up after

normalization. The weight is quantized data: $\mathbf{X} = \{x_1, x_2, \cdots, x_N\}$ where $x_i \in \{0, 1, \cdots, 255\}$. It is because that quantization does not influence the accuracy of results and it also reduces the computational time. This weight data is binarized by a threshold. The threshold is arranged by the number of KOI which the applications require. This process generates keypoint class $Y = \{y_1, y_2, \cdots, y_N\}$ at each keypoint where $y_i \in \{0, 1\}$. If the value of y_i is 1, the keypoint i is KOI.

The important regions do not necessarily include motion and gradient. For example, gradient of human body is not large. It depends on their clothes. Several KOI can be extracted because the gradient of contour contains large values by proposed method in this section. Next, MRF is applied to reduce noise data and smoothing the keypoint class using the result of this section and adjacent keypoint data. The keypoint class is integrated on the each region.

2.3.4 Connectivity of adjacent keypoints

To solve the problem that keypoints in important regions do not necessarily include large motion and gradient values, this paper applies MRF [50, 51] to keypoint class. MRF is usually used to reduce the noise of image in the region of image processing. MRF is a graph structure which represents the dependence between nodes. In this case, the nodes are keypoints and the dependency is defined in this section. Keypoints are connected by the weight of the distance from each other because the candidate keypoint whose adjacent keypoints are KOI tends to be KOI. The example of connections is shown in **Fig.** 2.15. In the circle, the keypoints are connected and they are easy to become same class. We utilize the graph cut to reduce noise and determine keypoint classes.

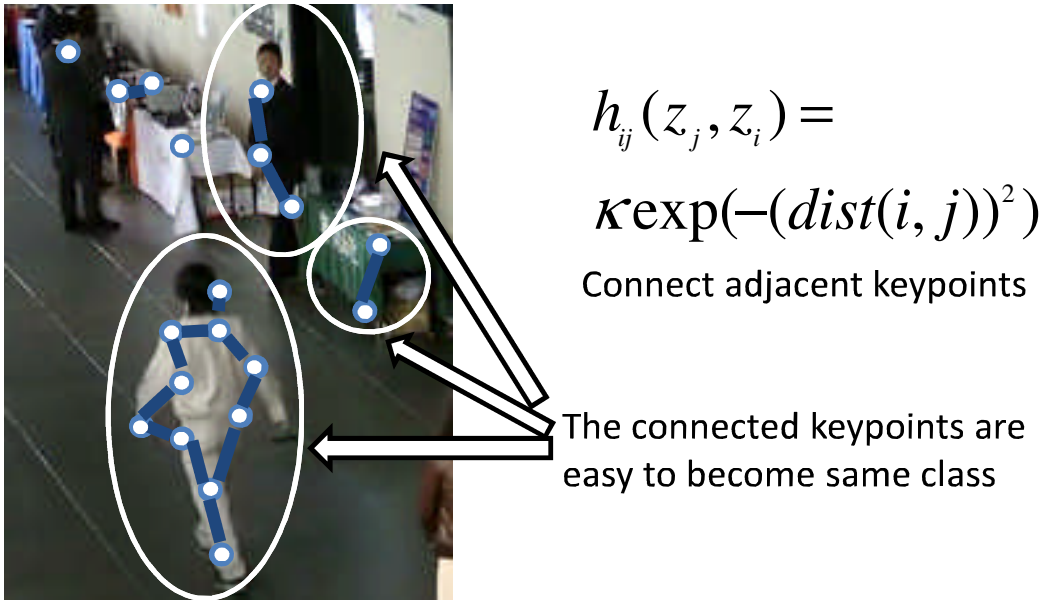

Figure 2.15: The connection of keypoints

The graph cut algorithm changes keypoint class z_i and minimizes the energy equation:

$$E(Z) = \sum_i g_i(z_i) + \sum_{i,j} h_{ij}(z_j, z_i). \tag{2.15}$$

In this case, global solution is calculated because the keypoint class is binary. To solve this minimization problem, Min-Cut/Max-flow algorithm is used. Each function is defined by **Eq.** (2.16) and **Eq.** (2.17).

$$g_i(z_i) = \lambda |y_i - z_i| \tag{2.16}$$

$$h_{ij}(z_j, z_i) = \kappa \exp(-(dist(i,j))^2) \tag{2.17}$$

Eq. (2.16) is data term. The outputted z_i is changed to approximate inputted y_i. **Eq.** (2.17) is smoothing term. The strength of connection depends on distance between keypoints. We assume it is gaussian distribution. The nearer the

keypoint distance, the stronger connection this function generates. $dist(i, j)$ represents the distance between keypoint i and keypoint j. λ and κ are parameters which are determined experimentally. They determine the strength of data term and smoothing term. If λ is larger than κ, the result approximates to the inputted data. If κ is larger than λ, the result approximates to the majority class of inputted keypoints. The calculated $Z = \{z_1, z_2, \cdots, z_N\}$ where $z_i \in \{0, 1\}$ is the output keypoint class. If the value of z_i is 1, the keypoint i is KOI. This calculation is faster than noise reduction of image which each node is a pixel because there are fewer nodes of the proposed method.

2.3.5 Calculation of camera-motion invariant optical flow by camera motion estimation

In the practical scenes, cameras move like motions of pan or zoom. There are a large number of scenes of zoom and pan in surveillance or in-vehicle cameras. To apply this algorithm to moving cameras, this paper proposes calculation of camera-motion invariant optical flows by camera motion estimation not to obtain large weight from the parts which do not move in fact. The overall flow is shown in Fig. 3.10 including zoom scenes. Optical flows are obtained by the KLT tracker at each keypoint. However, they include the influence of camera motions. For example, a number of optical flows which contain radical directions are generated like Fig. 3.10 from the parts which do not move in fact. Thus, we calculate the camera motion from these optical flows and the optical flows which are influenced by only camera motion is estimated. These are subtracted and the optical flows without influence of camera motion are obtained. Next, the method that estimates a camera motions is shown.

A camera motion is estimated by all obtained optical flow by the KLT tracker. The motion vector of camera is defined as $\mathbf{T} = [t_x, t_y, t_z]^T$. The coordinate of the keypoint i is defined as $\mathbf{x}_i = [x_i, y_i, z_i]^T$. The optical flow, $\mathbf{v}_i = [u_i, v_i]^T$, is calculated by

$$\begin{aligned} u_i &= \frac{x_i t_z}{z_i} - \frac{f t_x}{z_i}, \\ v_i &= \frac{y_i t_z}{z_i} - \frac{f t_x}{z_i}. \end{aligned} \tag{2.18}$$

$\mathbf{T}$ is estimated by minimizing the function J:

$$J = \sum_{i=1}^{N} (\hat{u}_i - u_i)^T (\hat{u}_i - u_i), \tag{2.19}$$

where u_i is the calculated optical flow by Eq. (2.18) and $\hat{u}_i$ is the calculated optical flow by the KLT tracker. $\mathbf{T}$ is changed to minimize J. The result is substituted for Eq. (2.18) again. Estimated optical flows and obtained optical flows from inputted video are subtracted. The result is the camera-motion invariant optical flows.

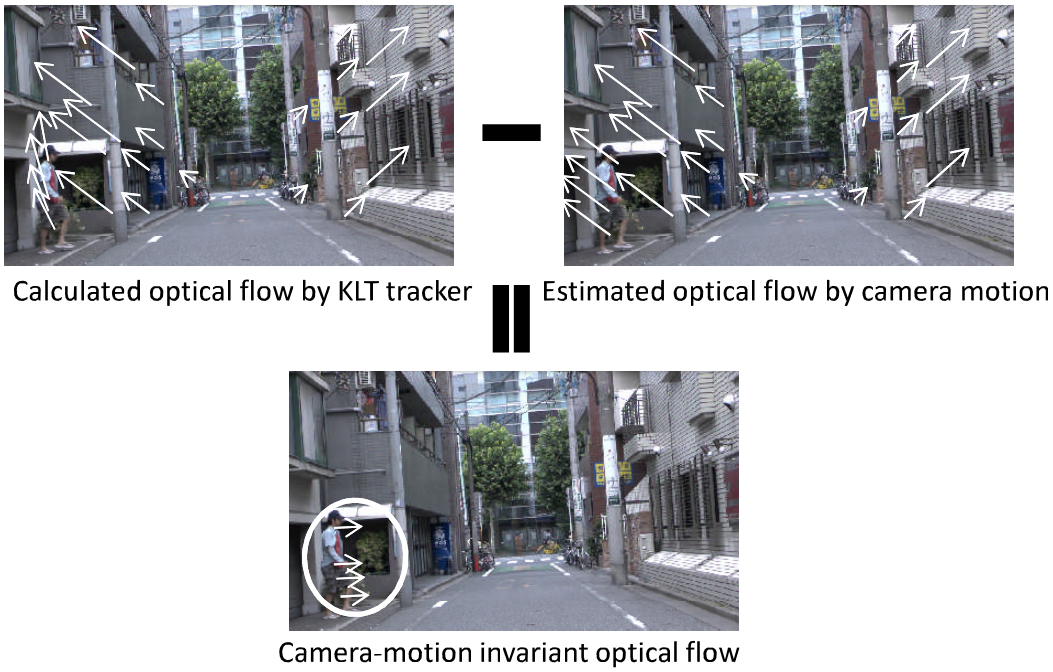

Figure 2.16: Calculation of camera-motion invariant optical flow.

2.4 Evaluation results

This section shows the evaluation results that compare the proposed method with the general keypoint extraction which utilizes the corner detector and SIFT descriptor [11, 41, 43, 52]. The method includes same algorithm except for the keypoint selection part and KLT tracker. The development environment on software is Visual Studio C++ 2008. CPU is Intel Core i7-2600 CPU 3.40GHz. In this thesis, we evaluated eight test sequences to confirm effectiveness of KOI extraction dealing with camera motion: fixed camera, zoom, pan and track. The test sequences are shown in **Fig.** 2.17. The name of the sequences indicates the kind of camera motion. In-vehicle is a video taken by an in-vehicle camera. We utilize the surveillance scene in Fixed1-2, Zoom1-2 and Pan and they include scenes that people walk on paths. As application, it is assumed that the motions of human are analyzed and recognition. In Fixed1 and Zoom1, three people walk. In Fixed2, Zoom2 and Pan, one person walks. We also utilize the shields sequence which is generally used in the field of video compression as a sequence of Track because everyone can get this sequence and evaluate. In this video, one person walks. In Fixed3 and In-vehicle, not only human but also objects are evaluated. As application, it is assumed that the kind of books is identified in Fixed. In In-vehicle, it is assumed that motions of human are analyzed and the traffic signs are identified. In Fixed3, one book is taken. In In-vehicle, two people walk and there are three traffic sighs and signboards of parking. The resolution of the videos we used is Full-HD (1920×1080 pixels) 60 fps. Only Track is HD (1280×720 pixels) 50 fps. The applications of these scenes are shown in next section.

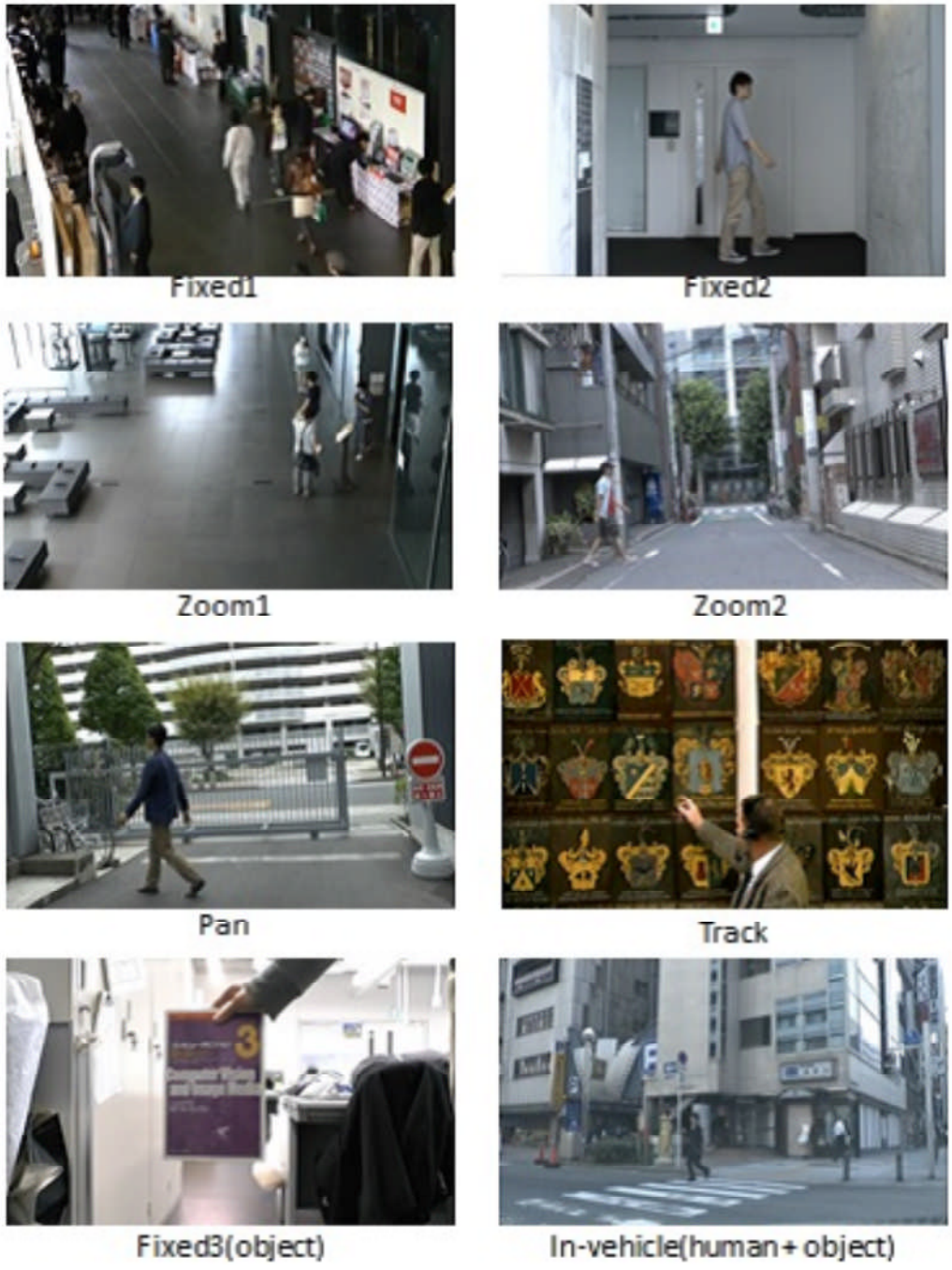

Figure 2.17: Test sequences

Table 2.1: The number of keypoints and processing times by the conventional method and the proposed method

		Conventional	Proposed
Fixed1	Number of keypoints	1192	66
	Processing time	754	190
Fixed2	Number of keypoints	1188	53
	Processing time	821	179
Zoom1	Number of keypoints	1202	47
	Processing time	759	183
Zoom2	Number of keypoints	1205	63
	Processing time	851	204
Pan	Number of keypoints	1143	108
	Processing time	771	198
Track	Number of keypoints	1037	140
	Processing time	723	175
Fixed3	Number of keypoints	1125	136
	Processing time	795	193
In-vehicle	Number of keypoints	1212	55
	Processing time	876	188

First, the number of keypoints which are detected by both methods are compared in Tab. 2.1. It shows the average among all frames of the movie. The proposed algorithm achieves the 94%, 96%, 96%, 95%, 90%, 86%, 88% and 95% reduction of keypoints in each movie. Almost same results were obtained among all videos. However, Pan, Track and Fixed3 are lower reduction comparing with others. It is considered that the movies include complex texture that has large intensity gradients in background. In Fixed1, several keypoints are extracted in poster or other display items whose gradient is large. In all videos, the reduction of keypoints was confirmed.

In addition, processing time is compared. The proposed algorithm reduces about 75%, 78%, 76%, 76%, 74%, 76%, 76% and 79% computational complexity than the conventional keypoint extraction in each movie. In all videos, the reduction of computational complexity is confirmed.

Fig. 2.18 shows the output of results of the conventional method and the proposed algorithm. The white circles are the keypoint obtained by each algorithm. It shows the proposal detects keypoints from only human which moves largely and outstanding texture whose gradient is large. In the other video, the proposed algorithm extracts a large number of keypoints from human body and the part including outstanding texture. By using only these keypoints, it is expected to analyze human or other outstanding object behaviors in surveillance and in-vehicle camera combining motion features.

Fig. 2.19 shows the recall-precision curves when the correct KOI are defined. We define the correct KOI by considering applications that KOI extraction can be applied. In Fixed1-2, Zoom1-2 and Track on Fig. 2.17, the keypoints in human parts are defined as the correct KOI. In Fixed3, the keypoints in book part are defined as the correct KOI. In In-vehicle, the keypoints in humans, traffic signs and the signboards of parking area are defined as the correct KOI. The values of precision and recall are calculated by average of all frames in each video.

Precision and recall are defined by **Eq.** (2.20) and (2.21).

$$prescision = \frac{TP}{TP + FN} \tag{2.20}$$

$$recall = \frac{TP}{TP + FP} \tag{2.21}$$

Each value means,

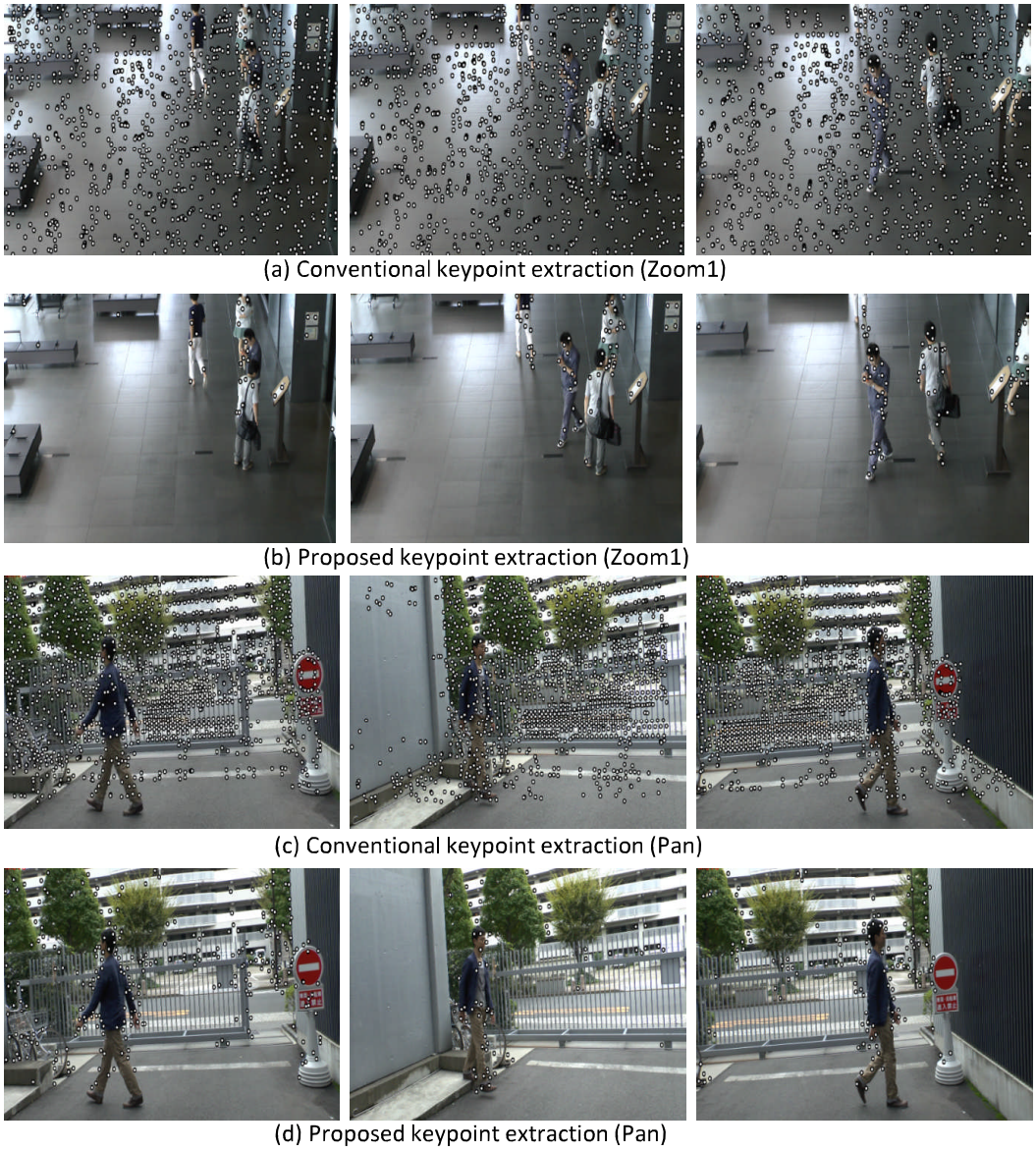

Figure 2.18: The comparison between (a) conventional keypoint extraction and (b) proposed algorithm in Zoom1, (c) conventional keypoint extraction and (d) proposed algorithm in Pan

- TP: Number of points which are detected correctly

- TN: Number of points which are not detected correctly

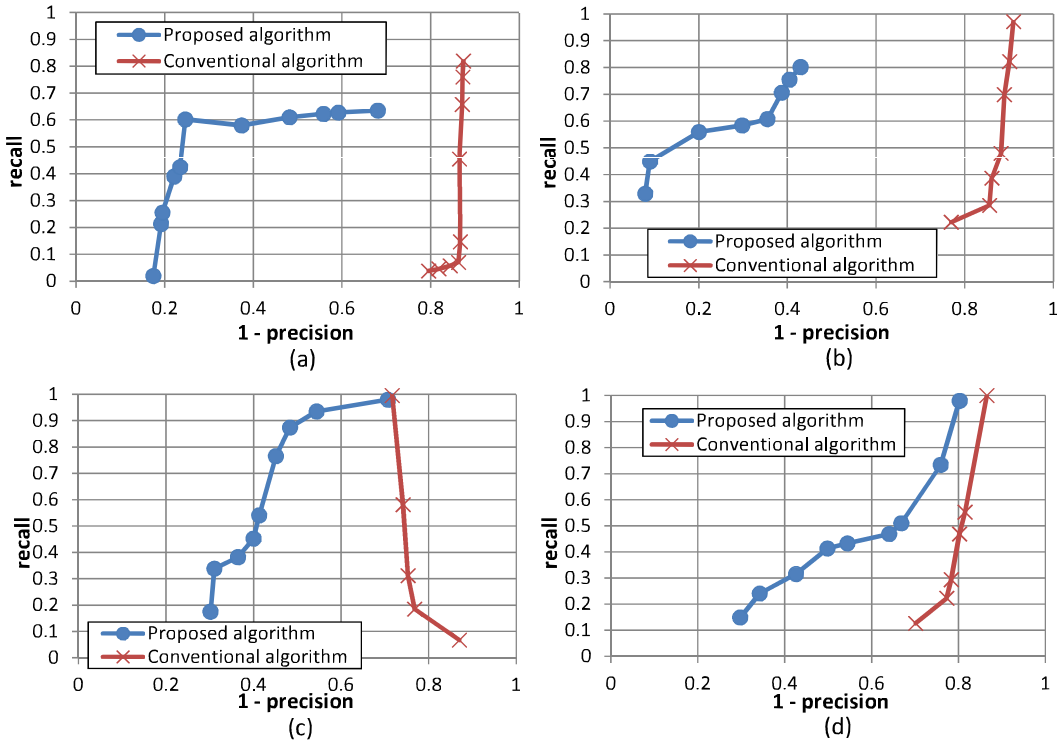

Figure 2.19: The comparison between conventional keypoint extraction and proposed algorithm of recall-precision curve in (a) Fixed1, (b) Track, (c) Fixed3 and (d) In-vehicle

- FP: Number of detected points which do not have to be detected
- FN: Number of not detected points which have to be detected.

The evaluation results show the proposed algorithm contains higher recall when 1-precision is low comparing with conventional algorithm in all videos. We can see that the proposed algorithm extract more keypoints from the intended parts comparing with conventional algorithm. Especially, **Fig.** 2.19 (a) and **Fig.** 2.19 (b) show the better results. In **Fig.** 2.19 (c), the curve of the conventional algorithm is downward to the right. It is because more keypoints in defined parts decrease than keypoints in undefined parts when keypoints

decrease by the conventional algorithm. In **Fig.** 2.19 (d), the curve of the proposed algorithm is not better results comparing with other sequences. It is considered that plural parts are defined as KOI and the proposed algorithm extracts more keypoints from other parts.

2.5 Applications based on the proposed algorithm

The proposed algorithm can be applied to a large number of applications by utilizing the characteristics that it extracts keypoints from only objects. For examples, we can consider following three applications.

- Gait recognition in surveillance scenes
- Recognition of in-vehicle systems
- Marker-less identification

In the surveillances scenes, the evaluation results show the proposed algorithm can extracts keypoints from the parts of human. Thus, by using their optical flow, it is possible to analyze the motion in each part for gait recognition. In regard to in-vehicle systems, it is possible to recognize humans, several traffic signs and other signboards. **Fig.** 2.17 In-vehicle video is one of the scenes which the proposed algorithm is applied. It is possible to recognize humans and give warning of their rushing out, and tell traffic sign and its meaning, and tell other information, for example, there are parking areas. In regard to marker-less identification, for example, it is possible to hold books to the

camera and recognize the cover like **Fig.** 2.17 Fixed3 video in library without markers. This paper shows how to achieve gait recognition in more detail.

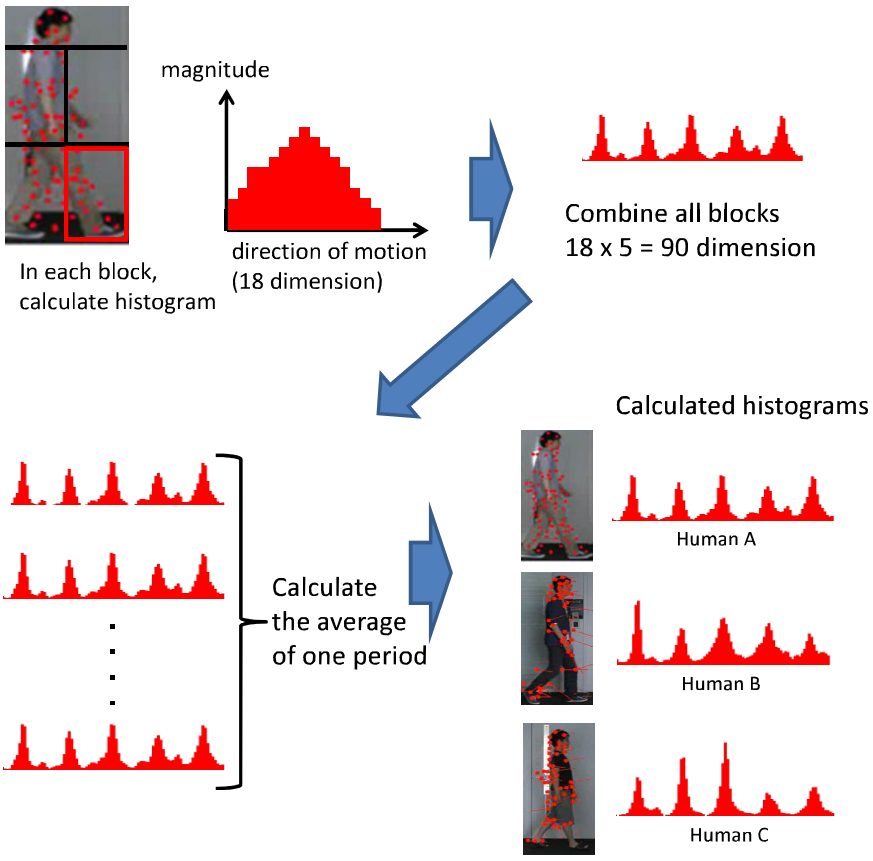

Figure 2.20: Illustration of calculating the histograms and the result

One example of calculations of gait recognition and experiment which we implemented is shown. KOI are extracted from human part in **Fig.** 2.17, Fixed2 by the proposed algorithm. Thus, the region of humans is calculated by combination of noise reduction and the average position of the keypoints. After that, the region is divided into five blocks. It is to divide the parts of head, right arm and leg, left arm and leg and analyze independently. In each block, the direction of motion is binarized to 18 dimensions. 18×5=90

dimension histogram is generated. Finally, the average of the period of their walks is calculated and average histogram of the period is obtained on one person. **Fig.** 2.20 shows the method that calculates the histogram. The evaluation result that histograms are calculated from three different persons is shown in Fig. 2.21. The experiments of recognition using Support Vector Machine (SVM) show the result that three persons can be identified perfectly. We can consider that more persons can be identified and it can be applied to other complex scenes by expanding this method.

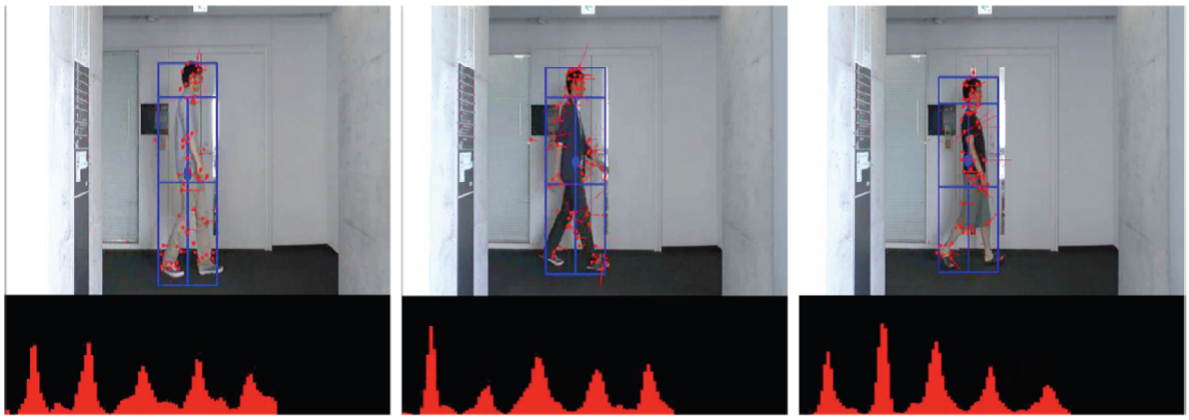

Figure 2.21: Histogram for gait recognition.

2.6 Conclusion

Reduction of keypoint data and the reduction of computational complexity are required for cloud applications. Conventional keypoint extractions utilize only spatial information and extract a large number of unnecessary keypoints. To solve the problem, this chapter proposed the keypoint extraction algorithm that detects only KOI based on spatio-temporal features and the MRF.

First, this chapter proposed the low complexity keypoint extraction algorithm which is easy to expand into spato-temporal feature based method. We detect keypoints by the low complexity corner detection and generated SIFT descriptor. Multi-scale database images are used to deal with scale changes. Second, this chapter describes the proposed keypoint selection algorithm from a number of keypoints including unnecessary ones based on spatio-temporal features and MRF considering camera motions. The KOI extraction is composed of three elements: spatial information, temporal information and connectivity of adjacent keypoints. The proposed method includes an approximated KLT tracker to calculate positions of keypoints and optical flows. It calculates weights at each keypoint using an intensity gradient and an optical flow reducing noise by comparing with states of surrounding keypoints. Camera motion estimation is added to the algorithm and it calculates camera-motion invariant optical flows.

The evaluation result have shown that the proposed algorithm achieved about 93% reduction of keypoints and 76% reduction of computational complexity on average. KOI are extracted from human bodies that move widely and target objects whose gradient is large. This algorithm is expected to be

applied to surveillance cameras and in-vehicle cameras when cloud systems start to utilize it. As an example of applications, this chapter shows the gait recognition. Three people can be recognized by the shown extraction algorithm perfectly. It can been seen that the proposed keypoint extraction's potential is high.

Chapter 3

Full-HD 60 fps FPGA implementation using gradient histogram

3.1 Overview of this chapter

The conventional works of FPGA implementation realize only up to spatial keypoint extraction for small size video such as VGA. The reason is that most methods buffer keypoint and its surrounding pixels after keypoint detection and serially executes descriptor generation. This paper proposes the bufferless architecture for high-speed implementation [42]. To realize bufferless architecture, we need hardware-friendly algorithm and parallel architecture in order to reduce amount of resource utilization.

This chapter describes the proposed hardware-friendly algorithm shown in chapter 2 and its real-time FPGA implementation using gradient histogram. The proposed hardware-friendly algorithm makes it possible to implement in a realistic FPGA resource by commonly using modules and reducing computational complexity. Furthermore, in the FPGA implementation, parallelization

of the proposed algorithms realizes resource reduction and enhancement of processing time. Finally, FPGA implementation is evaluated from the viewpoint of performance concerning feature point extraction and speed.

Regarding the proposed technique, the algorithm is based on dual threshold keypoint detection by gradient histogram and parallelization of connectivity of adjacent keypoint-utilizing register counters. The algorithm utilizes histogram based detection and keypoint-matching based calculation of motion information and dense-clustering based keypoint smoothing. The FPGA architecture is composed of a detection module utilizing descriptor, and grid-region-parallelization based density clustering. The detection and the descriptor generation modules are placed in parallel. The descriptor generation module is parallelized by considering grid placements. The processing time of descriptor computation in this hardware is independent of the number of keypoints because its descriptor generation is pipelining structure of pixel.

Finally, the evaluation results of FPGA implementation show that the implemented hardware achieves Full-HD (1920x1080)-60 fps spatio-temporal keypoint extraction on field-programmable gate array (FPGA). Further, it is 47 times faster than low complexity keypoint extraction on software and 12 times faster than spatio-temporal keypoint extraction on software, and the hardware resources are almost the same as SIFT implementation on FPGA, maintaining accuracy.

3.2 Hardware-friendly algorithm based on gradient histogram and density clustering

To realize bufferless architecture, efficient utilization of hardware resource is important. The problems of conventional works [40, 41, 53] and their solutions in this paper are shown below.

1. High resources of keypoint detection module

   ⇒ Gradient-histogram-based keypoint detection algorithm and implementation

2. Memory utilization of optical flow calculation by KLT tracker

   ⇒ Keypoint matching based optical calculation and implementation

3. Complex iteration calculation of graph-cut algorithm

   ⇒ Dense-clustering-based connectivity of adjacent keypoint

4. Large resources of temporal information calculation and keypoint connectivity calculation

   ⇒ Grid-region-based parallel implementation and memory utilization method

The detection module of hardware requires significant resources. Thus, this paper proposes a gradient-histogram based keypoint detection which utilizes a descriptor computation module. The KLT tracker utilizes previous frame data and it requires a large memory. Thus, this paper proposes a keypoint-matching based optical calculation which utilized extracted keypoints. Only keypoints

in the previous frame are memorized. The graph-cut algorithm includes a complex iteration calculation which is hard to implement on hardware that has a time constraint. Thus, this paper proposes a dense-clustering based connectivity of an adjacent keypoint. Finally, significant resources of temporal information calculation and keypoint connectivity calculation is a problem. Thus, grid-region based parallel implementation and memory utilization method are proposed.

This section shows an algorithm with low amount of computations for FPGA implementation. Mainly, this paper proposes two methods for hardware-friendly algorithms.

- Keypoint detection with descriptor calculation utilizing dual threshold for a gradient histogram

- The KOI algorithm utilizing gradient histogram based detection and weighting keypoints and dense-clustering based connectivity of an adjacent keypoint

The flow of the proposed algorithm is shown in Fig. 3.1. First, descriptors are calculated. Then, keypoints are detected based on the gradient histogram which is obtained during descriptor calculation. Temporal information is obtained by keypoint matching. Keypoints are weighted by the gradient histogram and inter-frame keypoint distance. Finally, the smoothing process is performed and the algorithm outputs whether the pixel is a KOI. The details of the algorithms are shown next.

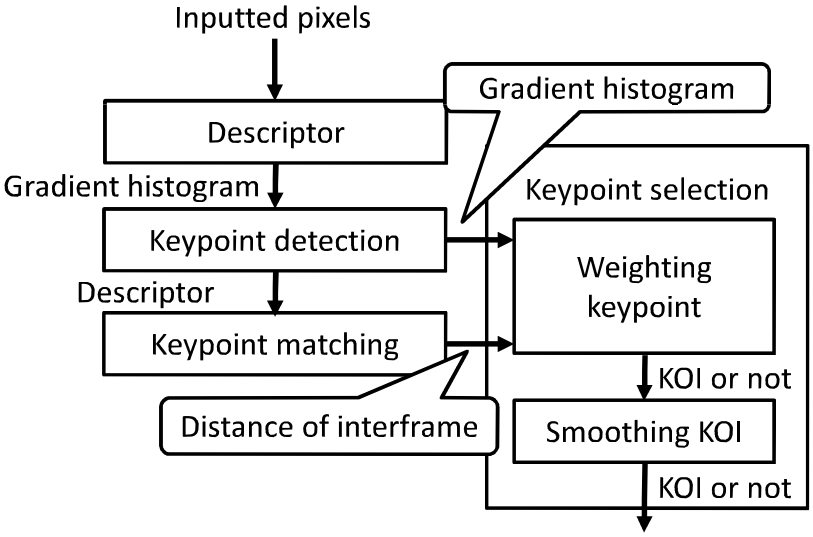

Figure 3.1: The flow of proposed algorithm.

3.2.1 Keypoint detection with descriptor calculation utilizing dual threshold for a gradient histogram

This section shows a low complexity keypoint extraction utilizing a gradient histogram. In general, there is no method which extracts keypoints utilizing descriptor because amount of computations is reduced by calculating only extracted keypoints on software. The proposed algorithm utilizes the high-speed descriptor-computation engine effectively and achieves a low-resource design.

The gradient histogram which is h is utilized in the descriptor computation part. The histogram is calculated pixel by pixel. During FPGA implementation, the dimension of the gradient histogram is 32 (5 bit) data. This paper utilized them effectively and also utilized dual thresholds T_1 and T_2. Keypoints

which satisfy both of the two inequalities below are detected.

$$max(h_1, h_2, \cdots, h_{32}) > T_1 \tag{3.1}$$

$$\sum_{i=1}^{32} H(h_i) \geq N \tag{3.2}$$

where max is the function which selects the maximum value. H is defined by

$$H(h_i) = \begin{cases} 1 \ (h_i > T_2) \\ 0 \ (h_i \leq T_2) \end{cases} \tag{3.3}$$

A pixel whose magnitude is larger than threshold is counted. The parameters, T_1, T_2 and N, are utilized when the number of keypoints is optimized. When the N increases, the keypoints which have gradients to more directions are detected. When the T_1 and T_2 increase, the keypoints which have larger feature represented by Eq. (3.3) are detected. The $N = 2, 3$ are suitable values in this algorithm to detect keypoints in corners which has plural directional gradient generally. We experimentally decided that $N = 2$ and $T_2 = 0.8 \times T_1$. This thesis utilized these values in implementation and experiments. The schematic diagram of the process is shown in Fig. 3.2. The length of the arrows show the scale of magnitude and the direction of the arrows show the direction of the gradient.

3.2.2 The KOI algorithm utilizing gradient histogram-based detection and weighting keypoints and dense-clustering-based connectivity of an adjacent keypoint

First, a method which obtains motion information on each keypoint is shown. A spatio-temporal keypoint method utilizes an optical flow by the KLT tracker.

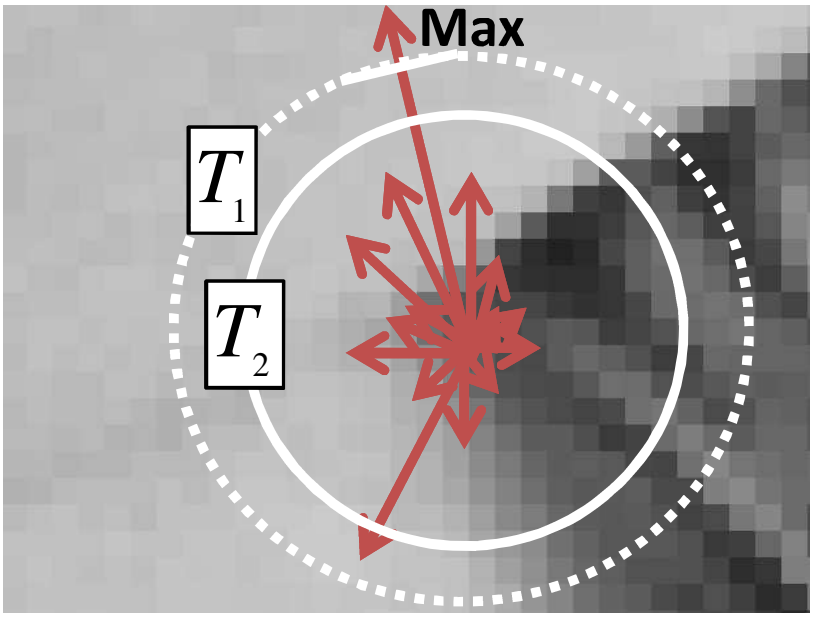

Figure 3.2: Keypoint extraction based on a gradient histogram.

This paper describes an algorithm for hardware along with three elements which compose the spatio-temporal keypoint extraction. The spatial information is calculated by the gradient histogram of the descriptor. The temporal information is calculated by keypoint matching on the inter-frame difference. The connectivity is obtained by density clustering on a grid division of an image region.

3.2.2.1 Spatial information

This thesis proposes a descriptor-based temporal information calculation. The conventional algorithms [40, 41] calculate gradient information by a Hessian matrix. The calculation of the Hessian matrix has high computational complexity because it includes matrix calculations and several multiplications. The SIFT descriptor of keypoints also includes a number of gradient information

points. Thus, this paper proposes a method by utilizing the descriptor. The maximum value of the gradient histogram which weights the magnitude of the gradient on each orientation is calculated on a keypoint. The value is utilized as a weight of temporal information.

3.2.2.2 Temporal information

This thesis proposes a keypoint-matching based motion calculation. Keypoints in different frames are matched by a distance calculation with the descriptor. The distance is calculated between a keypoint on a current frame and all keypoints on a previous frame. The L1-norm of x-y coordinates is calculated between a keypoint on a current frame and the keypoint on a previous frame which is the minimum distance of the descriptor. The L1-norm of x-y coordinates of the two keypoints, M, is utilized as motion information for weighting keypoints. Each keypoint is weighted by spatio-temporal information. The value which decides KOI is defined by,

$$\omega_1 max(h_1, h_2, \cdots, h_{32}) + \omega_2 M, \tag{3.4}$$

where ω_1 and ω_2 are parameters which are decided by experiments.

3.2.2.3 Connection of adjacent keypoints

As the calculation of local correlation, density clustering of keypoint class is proposed in order to reduce computational complexity while maintaining accuracy. In this section, division means the operation which divides a region into plural regions. The conventional method [40] utilizes MRF and the graph-cut algorithm. The method requires all keypoint connections on MRF. In addition, it includes an optimization problem with iterations. The calculation

is not suitable for FPGA implementation because it cannot be analyzed when the calculation is finished. The computational complexity of the graph cut is $O(V \cdot E^2)$, where E is the number of branches and V is the number of nodes. It depends on three variants.

This thesis proposed a density clustering 1 dimension filter whose computational complexity is $O(V)$. First, the algorithm divides an image into several regions. The regions are 2 dimensions and aligned at equal spaces. The candidates of KOI which are calculated by Eq. (3.3) in each region are counted. Each keypoint has a flag which represents whether the keypoint is KOI or not and the flag is checked during counting. The counted values are contained in variants whose number is equal to the number of regions. When the number of the candidates of KOI is larger than the threshold, all keypoints of the region are redefined as KOI. The flags of keypoints in the region also are rewritten. Otherwise, all keypoints of the region are redefined as not KOI. The pixel which is in i-th region r_i is decided as KOI utilizing threshold, T_s, when the condition below is satisfied:

$$\sum_{j=1,\ k_j \in r_i}^{M} k_j \geq T_s, \tag{3.5}$$

$$k_j = \begin{cases} 1\ (KOI) \\ 0\ (not\ KOI) \end{cases} \tag{3.6}$$

where k_j is the flag which represents whether the j-th keypoint is a candidate of KOI or not. M is the number of extracted keypoints. Figure 3.3 shows an example of the proposed process. First, an image is divided into plural region. In each region, there are keypoints which are candidates of KOI (red keypoins) and not candidates of KOI (blue keypoints). Then, flags of each keypoints k_j

are checked. If Eq. (3.5) is satisfied, flags of all keypoints in the region are updated as KOI.

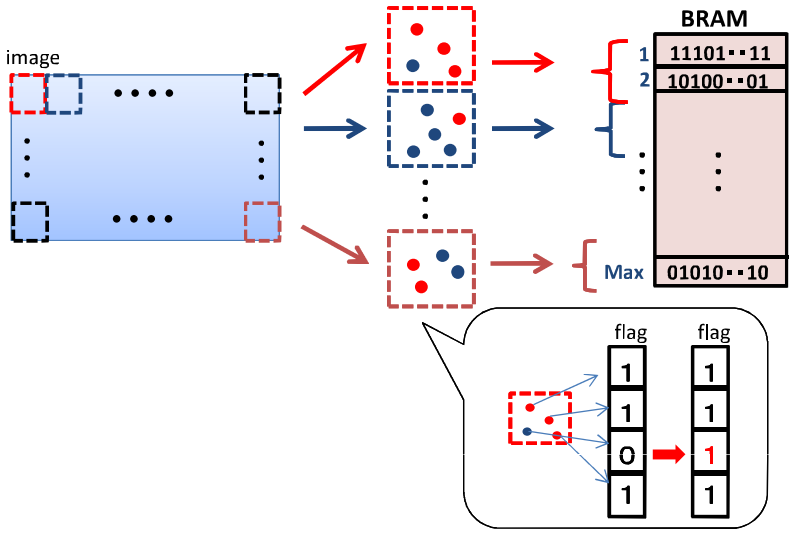

Figure 3.3: Density clustering of KOI smoothing.

3.2.3 Approximation based on SAD, bit-shift and LUT

To this point, the proposed low complexity algorithm remains relatively complex set of calculations. This consumes a large amount of hardware resources. The approximated calculators reduce the use of resources and enhances frequency of circuits. This also deals with **P1**.

First, equation (2.2) incorporates a square root. To solve the square root, many methods have been proposed, such as extraction of square root. However, these methods use iterations, which have the possibility that generates hardware delays because its number of iteration is unknown. Thus, the magnitude of gradient is considered. It is replaced by SAD because the magnitude is a weight of histogram computation. By this, magnitude of gradient in Eq.

(2.2) is redefined as

$$m(x,y) \quad = \quad |L_x(x,y)| + |L_y(x,y)|, \tag{3.7}$$

where L_x, L_y are the gradients of the pixels on each direction. Similarly, the distance d calculation at the time of keypoint matching is also approximated as follows.

$$d = |v_i^1 - v_r^1| + |v_i^2 - v_r^2| + \cdots + |v_i^{128} - v_r^{128}|, \tag{3.8}$$

where v shows the descriptor on each dimension. It is calculated more simply. The distance of two descriptors is computed similarly during matching process.

Next, the direction of gradient in Eq. (2.3) is computed approximately. It includes division and arctan. It occurs in two steps. First, the division is approximated by bit shift. Concretely, numerator is shifted by the value of denominator. Second, the arctan is computed by the Look Up Table (LUT). the value of arctan is decided by each result of the division. The direction of magnitude is quantized in 32 directions. Figure 3.4 is the detail of this approximation. The software simulations using these approximation do not show largely precision loss. It is considered that this algorithm does not require high precision computation because of quantization.

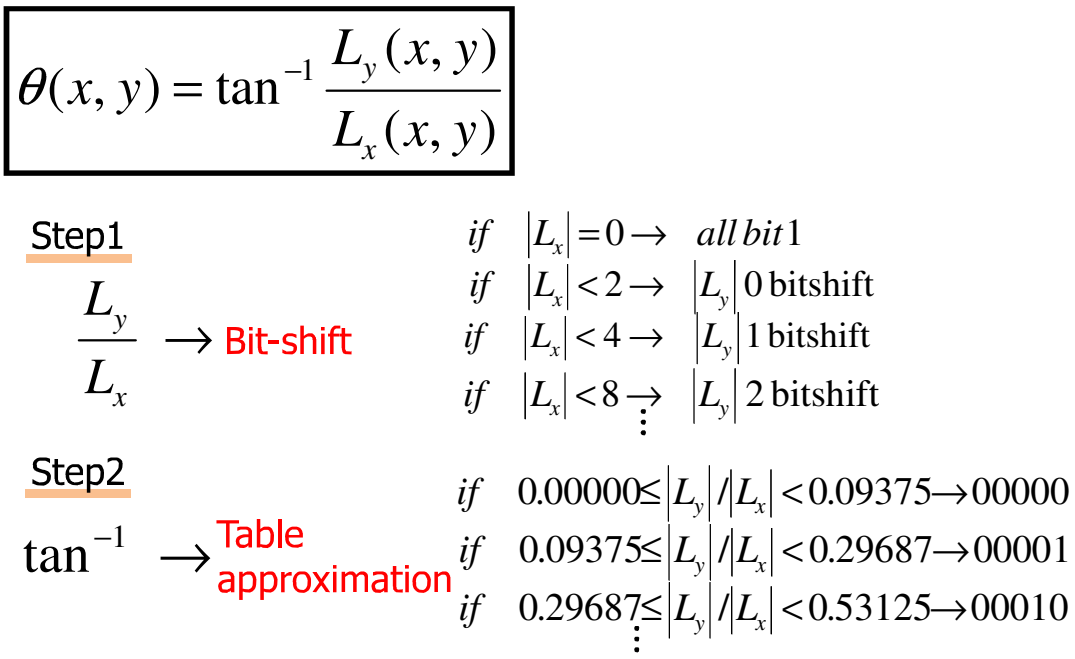

Figure 3.4: The approximation of the arctan calculation.

3.3 FPGA implementation based on parallelized gradient histogram

This section shows low-complexity architecture of spatio-temporal keypoint extraction. The architecture is based on 60-fps spatial-feature-based keypoint extraction hardware [41]. First, we shows the architecture. This thesis proposes a gradient-histogram based keypoint extraction hardware to remove detection modules, which is paralleled with the descriptor module. In addition, when the keypoint is matched in the user's device, a plural matching process is required. In that case, a lot of memory must be utilized. To solve this, parallelization of the keypoint matching module which is based on Least Recently Used (LRU) is proposed. Moreover, the grid-division based method requires a number of processes. Thus, this paper proposes a parallel counter on each division-based implementation of dense clustering by grid division. The proposed architecture is shown in Fig. 3.5. The details of the FPGA implementation are shown next.

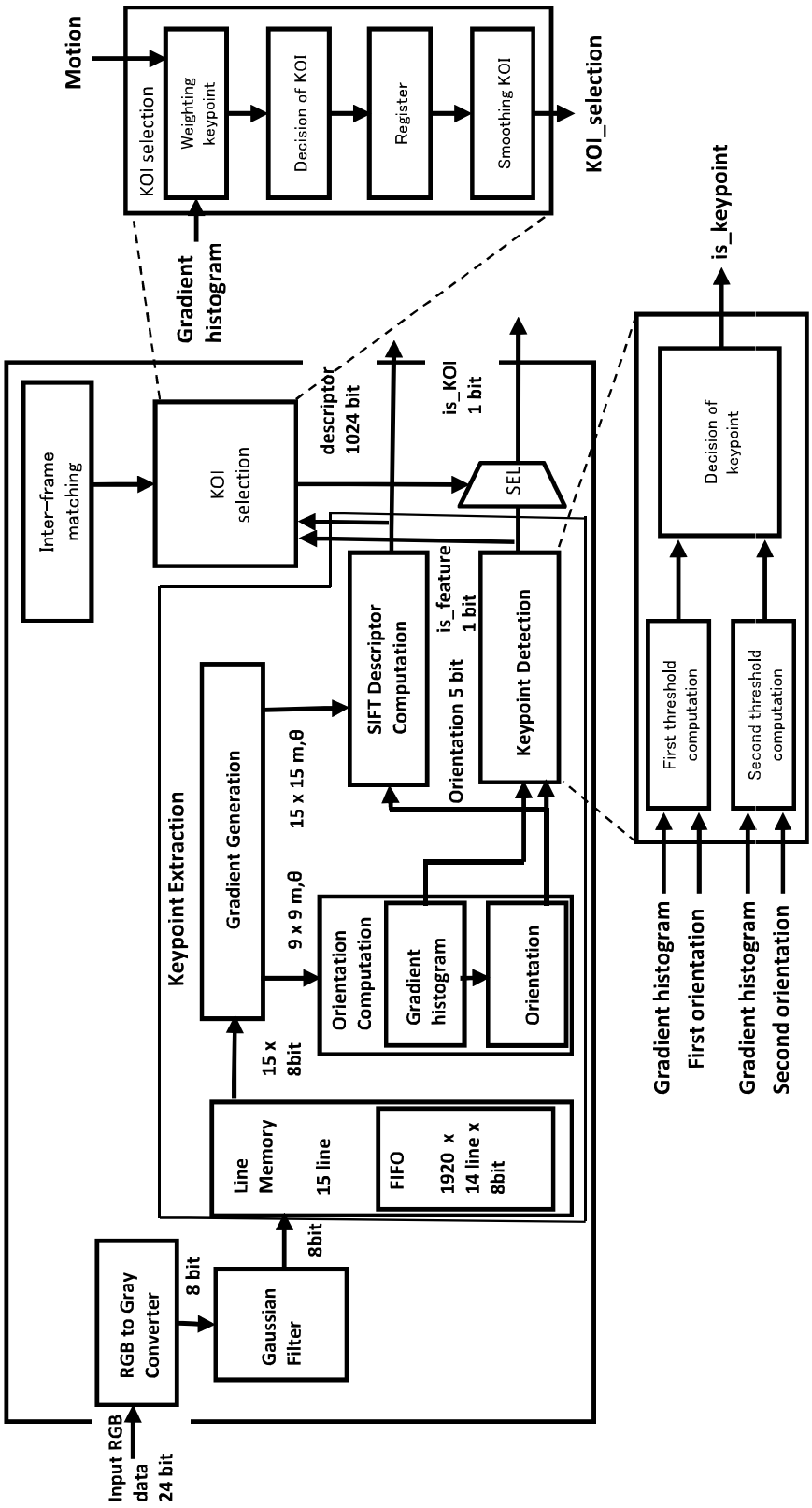

Figure 3.5: The structure of the proposed hardware.

3.3.1 60-fps spatial-feature-based keypoint extraction hardware

The entire block diagram is expressed in Fig. 3.6. The inputted RGB data is entered into the keypoint extraction module. In the keypoint extraction module (Fig. 3.7), 1 descriptor is outputted for 1 RGB data. The RGB data is converted to the gray scale data. It is accumulated in 5×5 and smoothed by gaussian coefficients. Next, these are entered into line buffer and vertical 15 pixels are kept. These are inputted to the detection module and the descriptor module in parallel. In the detection module, 15×15 pixel data is kept and weighted by the filter in Fig. 2.8. In the descriptor module, after the orientation computation (9×9 region), the SIFT descriptor (8 bit×128 D=1024 bit) is generated in 15×15 region. The descriptor is generated in all pixels because the structure is a pixel pipelining. If the pixel is detected as a keypoint (the signal of is_keypoint is 1), the position and descriptor data are memorized in block RAM.

Keypoint matching process is performed with 1 frame delay in keypoint matching module (Fig. 3.9). The descriptor data in the block RAM are fetched and Sum of Absolute Difference (SAD) is computed. It is a distance of feature vector. The minimum data between 1 inputted keypoint and all database keypoints is kept and sent to block RAM. In all clock, the address of block RAM is computed. Finally, an output image is generated by using the kepoint data and matching results. The fundamental modules are described in more detail next.

3.3.1.1 RGB to gray converter module

RGB to gray converter module converts grayscale RGB data (24 bit) of input into output grayscale data (8 bit). Pipeline processing for each pixel is performed. Each input RGB data is assumed to be a fixed point, and approximate calculation is performed by 1/3 with 16 bit precision by bit shift and addition. This process takes three clocks.

3.3.1.2 Gaussian filter module

The Gaussian filter module holds 1920 × 4 pieces of 8 bit data using FIFO memory using the input grayscale data, and performs a convolution operation. 15 × 15 are processed in order. It calculates the Gaussian filter with a distribution of 1.6 with the coefficient fixed for the region of 15 × 15. Again, the operation is a fixed decimal point and the operation is represented by only bit shift and addition. This process ends the operation with five clocks.

3.3.1.3 Line memory module

The line memory module functions as a buffer for pipeline processing of inputs of gray scale pixels (8 bits) smoothed by the Gaussian filter on a line-by-line basis. The memory is FIFO, 1920 entries are held and output in the order of input. By arranging this in 14 lines, you can output data of 8 bit *times* 15 line in order. This is done with one clock delay.

3.3.1.4 Corner detection module

The corner detection module performs a process of detecting a corner with respect to an input image. Processing is performed in a pipeline system for

each pixel line (8 bits × 15 line). It prepares registers and hold 15 rows of pixel data, so we will process 15 × 15 area. As processing, we first calculate the sum of rectangular regions by weighting like the Box filter. Since the amount of data to be held in the integral image becomes enormous and it is difficult to realize the hardware, it is not used. The sum of the rectangular areas is obtained by dividing into four clocks. Since it is necessary to multiply here, use DSP. The calculation result is compared with the threshold value, and 1 bit data is_ feature of whether or not the target pixel is a corner is outputted. All processing can be calculated with 9 clocks, but the output result must be held in the register because it is necessary to output it in synchronization with the descriptor module.

3.3.1.5 Orientation computation module

In the orientation computation module, the orientation which is the direction of the luminance gradient of the keypoint is calculated. Data of a pixel line (8 bits × 9 line) is input to this module, and processing is performed in a pipeline system. The input data are sequentially processed. First, calculate the absolute value of the difference, and then use it to calculate the arc tangent which is sum and direction that is intensity in parallel. Thereafter, histogram creation is performed, but in this case, unlike creation of feature amount histogram, there are no multiple grids that cannot be said to have equal dimensions. For that reason, nine histogram registers are prepared and assigned sequentially to the histogram, and then the addition is performed for each dimension. After finishing the histogram creation, it searches for the orientation with the maximum weight. This is only a matter of finding the maximum value, but

if you try to do it in a simple way, it must do 32 comparisons, which cannot operate at high speed. Therefore, the maximum value in the tournament formula such as comparing 32 pieces of data two by two and keeping the larger one was searched. This requires five clocks, but since it is possible to reduce the number of calculations sequentially performed for one clock, a high-speed circuit can be maintained. Then, it outputs an orientation with the obtained maximum intensity and sends it to the descriptor module.

3.3.1.6 SIFT descriptor computation module

In the SIFT descriptor computation module, it receives the input 5-bit orientation data, determines the dimension as a descriptor for all grids in the description area, and creates a feature amount histogram. Processing is done to the region of 15× 15 by the pipeline method for each pixel line (8 bits × 15 line). As with the orientation computation module, hold the data for each grid until the orientation input is made, after the strength and orientation of the luminance gradient are obtained and held in the register. Upon input, trigonometric functions are computed using the data. When the trigonometric function is calculated, for each grid of the descriptor description area, calculate which position of the SIFT descriptor is divided into 4 × 4. In parallel therewith, the orientation of each grid is subtracted according to the direction obtained from each grid, and the process of rotating the description area is advanced. Subsequently, as suggested above, values are assigned to the descriptor register in parallel, and the sum for each dimension is calculated to complete the SIFT descriptor. The calculation of the orientation and the description processing of the SIFT feature amount are completed in 22 clocks in

total.

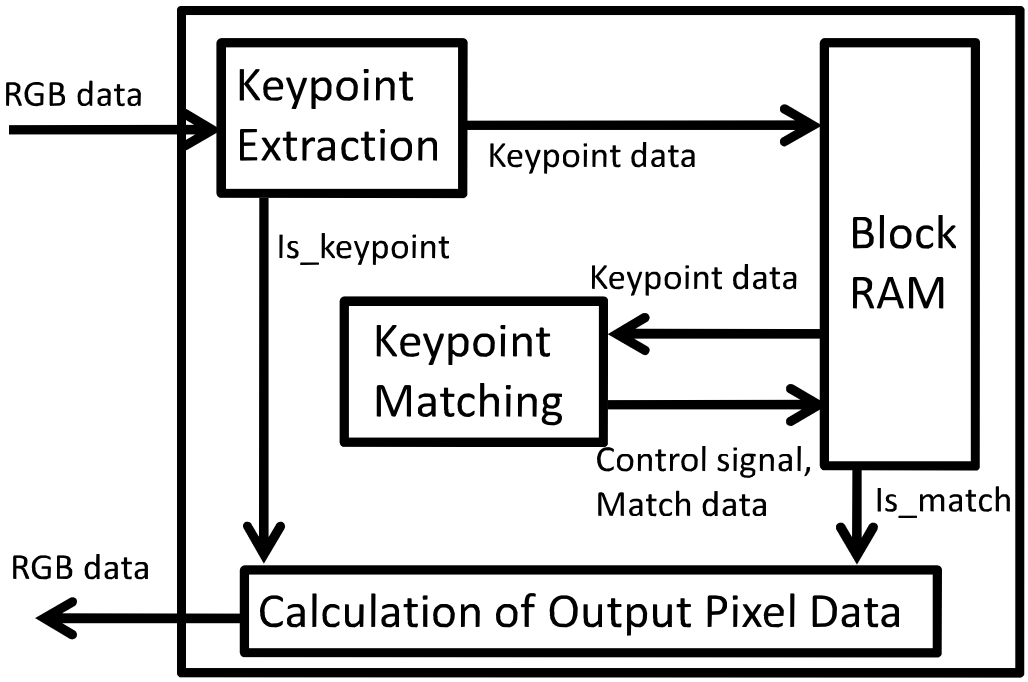

Figure 3.6: The block diagram of entire structure.

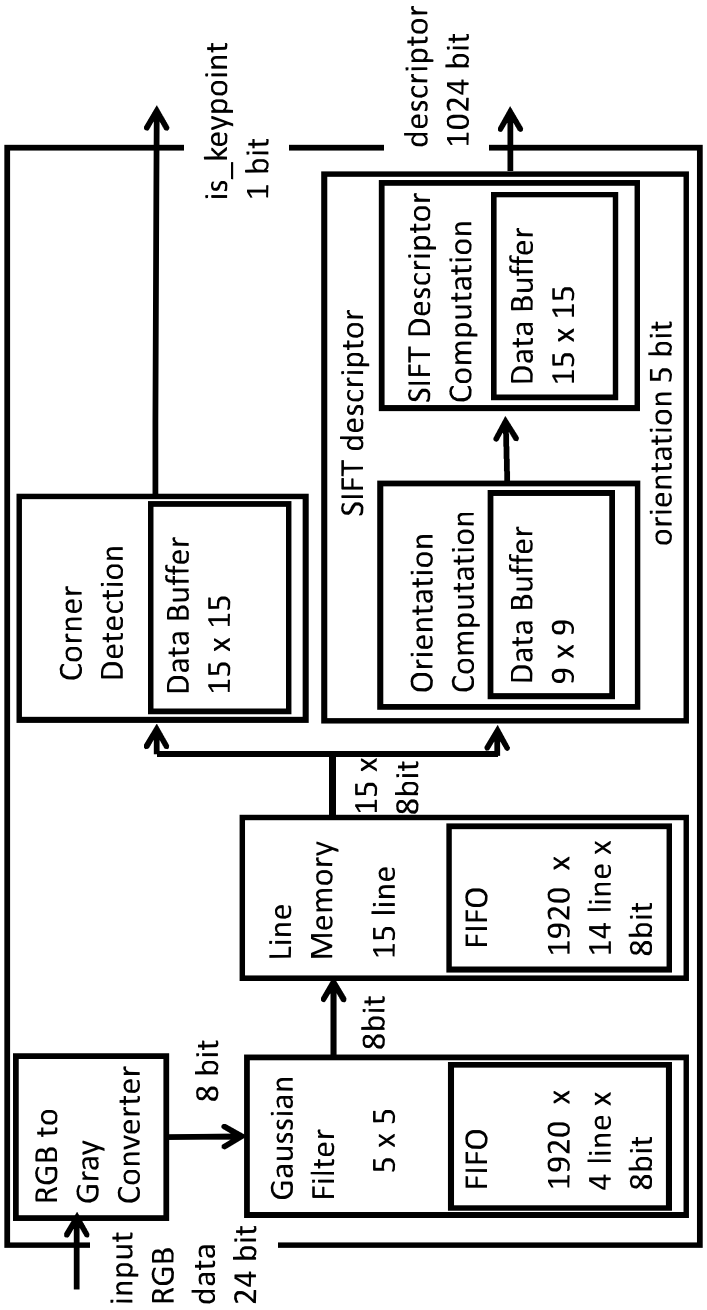

Figure 3.7: The structure of keypoint extraction module.

3.3.2 Architecture of keypoint detection based on the gradient histogram of the descriptor computation module

The conventional approach [41] implements parallel design of the detection module and descriptor module on 60 fps video. In an approach of this thesis, the detection module which calculates the determinant and trace of the Harris matrix is deleted by the proposed algorithm of the previous section. The gradient generation module calculates the magnitude and orientation of the gradient on each inputted pixel. The orientation computation module calculates the gradient histogram whose magnitude is added together on each quantized orientation by 9x9 magnitude and orientation of pixels. In the SIFT descriptor computation module, the 15x15 gradient information is inputted and the descriptor of a pixel is outputted. The keypoint detection module calculates binary which indicates if the pixel is a keypoint or not by the gradient histogram and orientation data. Then, the $is\_feature$ which expresses if the pixel is keypoint or not and descriptor signal are outputted from the keypoint extraction module in parallel.

In the keypoint detection module, the sum of the magnitude of each bin of the gradient histogram is calculated by comparators and the largest bin and second largest bin are selected. Thresholds to these bins of decision of the keypoint module decide whether a pixel is a keypoint or not by Eq. (2.13).

3.3.3 Parallelization of detection module and descriptor module

SIFT has the dependency between keypoint detection and computation of descriptor because a scale has to be computed and it determines the descriptor region. However, the proposed algorithm does not compute a scale. Therefore, it becomes possible to parallelize keypoint detection and computation of descriptor. It reduces clocks to compute. 15 pixel data are inputted from line memory. After that, detection and description are computed simultaneously. After the results are obtained, they are synchronized.

3.3.4 Parallelization of descriptor computation

The SIFT descriptor is computed by histogram computation. In general, it is serial processing because a conflict between registers occurs. It takes many clocks to generate descriptor. Concretely, 225 clocks are required because it uses 15×15 region. To solve this, SIFT descriptor computation is parallelized. SIFT descriptor divides the region into 4×4 grids. The bins of histogram do not have a dependence on each other. Thus, it is possible to parallelize distinct grid. The simulation result shows that 9 pixels at regular intervals do not depend on each other. It computes 15×15 keypoint region by placing 25 descriptor registers. After registers have values, they are added together in each bins. It computes descriptor in 4 clocks with finality. Fig. 3.8 is a schema that shows this processing.

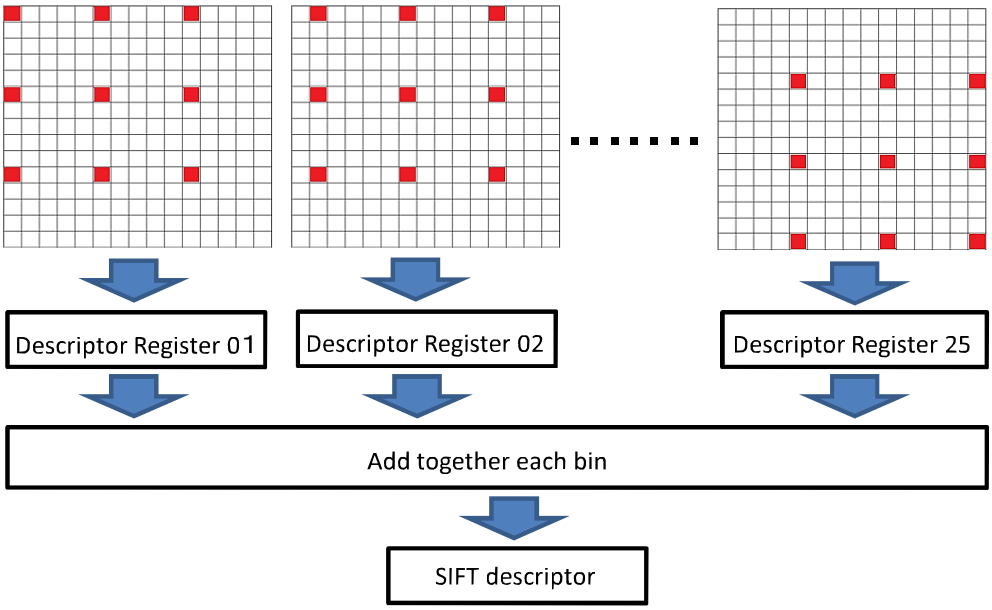

Figure 3.8: Parallelization of histogram computation.

3.3.5 Pipelining by fetching pixel lines from block RAM

To calculate SIFT descriptor in each pixel, it is necessary to process 9×9 region for orientation and 15×15 region for descriptor in 1 clock. It requires 9×9+15×15=306 calculators if it is computed simultaneously. To reduce the hardware resources , pipelining using 15 pixel lines is proposed. It deals with **P1**. When the vertical 15 pixel is inputed, these are calculated and m, θ are generated. After that, obtained values are kept in 14 clocks by registers. Only 9+15=24 calculator is required by this pipelining. This architecture is shown in the line buffer of Fig. 3.5.

3.3.6 Matching process by keypoint pipelining

Each keypoint has the descriptor data (1024 bit) and the position data (22 bit). It is very long bit data. If plural keypoints is processed simultaneously, a lot of hardware resources are required. To deal with **P4**, 1 keypoint data is fetched from block RAM in 1 clock and entered into the matching module of pipelining structure. The nearest neighbor search is performed. The obtained keypoint pairs are memorized in block RAM. The structure is shown in Fig. 3.9.

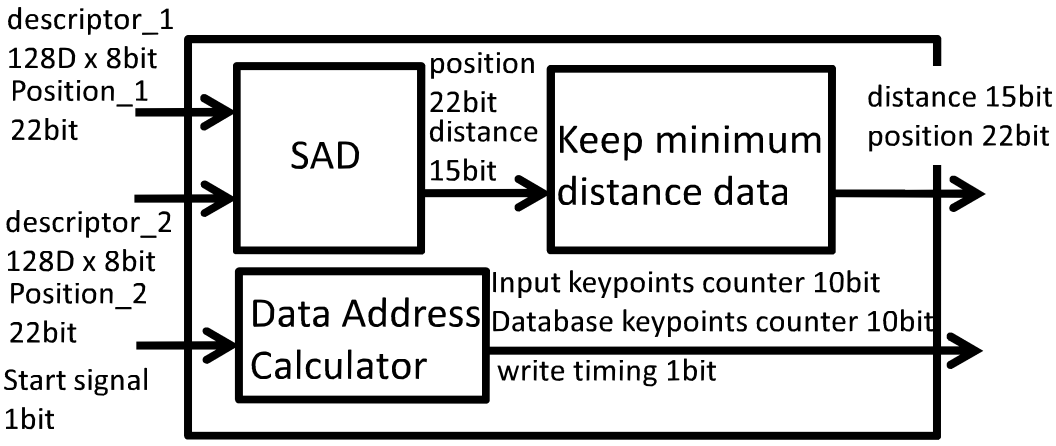

Figure 3.9: The structure of keypoint matching module.

3.3.7 Architecture of temporal information calculation and keypoint connectivity calculation

For the spatial information, the gradient histogram is added to the weight of keypoints to decide whether the pixel is a KOI. In the gradient histogram, the maximum value of all bins is utilized and is sent from the SIFT descriptor computation module to the KOI selection module. For the temporal infor-

mation, the motion of each keypoint is obtained by an inter-frame keypoint matching. The keypoints of the previous frame in BRAM are read by the inter-frame matching module. The L1-norm of x-y coordinates of the two keypoints is calculated between one detected keypoint and all keypoints in the previous frame. The L1-norm on the keypoint with the smallest distance is memorized in BRAM. It is inter-frame motion and utilized in the KOI selection module. The obtained spatio-temporal information is weighted and added together. Then, the value is binarized by the threshold, and the KOI-selection signal is generated. The difference in software implementation and FPGA implementation is shown in Fig. 3.10.

First, this subsection describes a LRU-based memory utilization of inter-frame-keypoint matching. The proposed algorithm requires memories which contain keypoints in the current frame and previous frame. Fig. 3.11 shows the block diagram. WE represents write enable signal. The matching process calculates the Sum of Absolute Difference (SAD) to obtain the distance of the descriptor. BRAM1 and BRAM2 are utilized to keep descriptor data and coordinates of the keypoints. Keypoints' data in $k-2$ are expelled when data in k frame are inputted utilizing LRU. The input of data is detected by vertical synchronization signal which represents effective timing of pixel data obtained from video signal. The hardware contains counter which represents the number of inputted frames. The SAD module is performed utilizing BRAM1 and BRAM2 in parallel and recognizes which data is obtained from current frame by the counter. The minimum-distance pair is memorized in BRAM3 utilizing the WE controller. Keypoints in the current frame are memorized in the least recently used memory of BRAM1 and BRAM2.

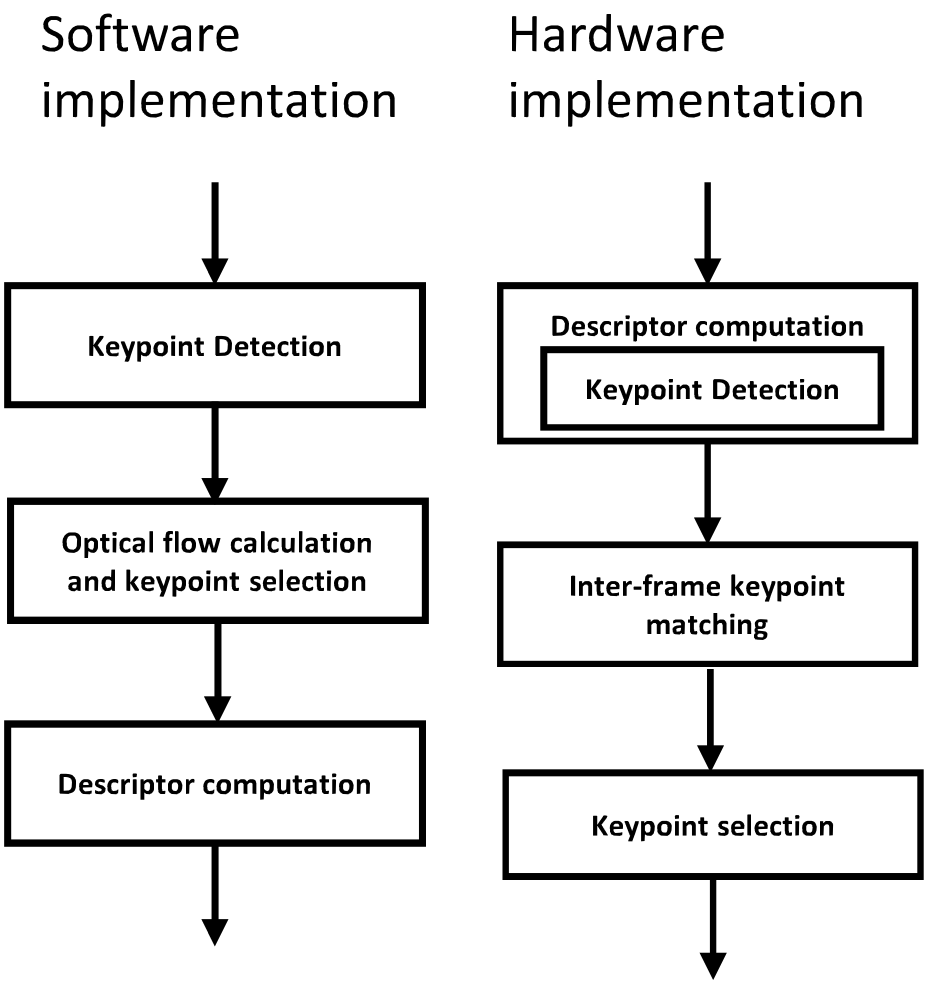

Figure 3.10: The difference of software implementation and FPGA implementation.

Next, for implementation of the connectivity of adjacent keypoints, implementation of dense clustering by parallel counter on each division is shown. The decision of the KOI module calculates candidates of KOI. Candidates of KOI are smoothed by dense clustering. Fig. 3.12 shows the block diagram of parallel implementation. The signal which shows if the pixel is KOI or not is sent to the counter depending on a region which the pixel belongs to by axis of candidates of KOI. Counter modules calculate the number of candidates of

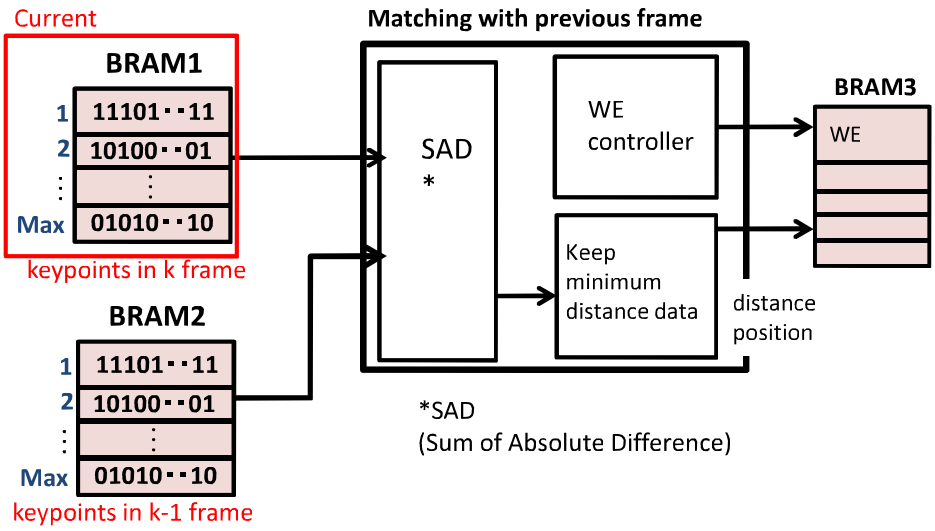

Figure 3.11: Memory utilization of keypoint matching.

KOI. The decision modules decide the KOI region utilizing counters. Finally, a KOI map shows all the KOI regions on a grid division. Utilizing the map, candidates of KOI are updated and outputted. The keypoints of next frame are decided by the results on each grid.

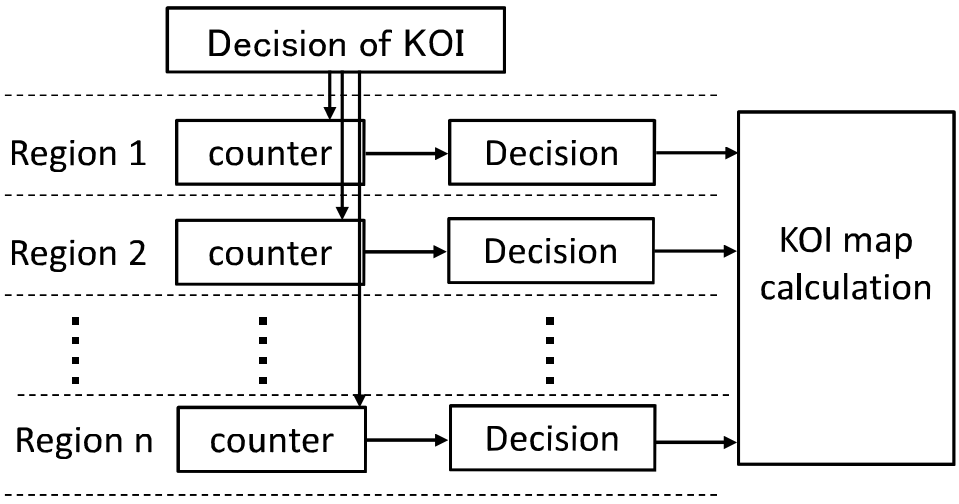

Figure 3.12: Grid-region based parallelization of density clustering.

3.4 Evaluation

3.4.1 Evaluation of keypoint detection performance

The proposed algorithm is compared with SIFT with respect to performance. The development environment for software is Visual Studio C++ 2008. CPU is Intel Core i5 CPU M 450 2.40GHz.

3.4.1.1 Detector performance

First, performances of implemented keypoint extraction which is composed of Harris detector and SIFT descriptor are examined on test sequences. For evaluation of correct matches and repeatability in Fig. 3.13 (a) and (b), we use the framework proposed by Mikolajczyk *et al.* [45]. The boat image shown in the paper is used as a test sequences. It includes both scale changes and rotation. The correct transformation matrix is obtained by RANdom SAmple Consensus (RANSAC) [46]. When L2 distance between the correct value and experimental value is under 10, the pair is regarded as a successful match. Figure 3.13 (a) shows the correct matches. The stability of descriptor is measured from this. The proposed algorithm is a little unstable but obtains more correct correspondences on many scales compared with SIFT. Figure 3.13 (b) is the repeatability. It means if keypoints are detected at the same positions between two images with a viewpoint change. It shows performance of detector. We can observe that the proposed corner detection obtained a lot of correspondences and better performance than the DoG detector of SIFT. Figure 3.13 (c) is an evaluation on wall images. This images include many gradients. In this case, SIFT has the ideal ROC curve and the proposed

algorithm also has the comparable ROC curve. Figure 3.13 (d) is an evaluation in bike images. This images are natural scenes. Both algorithms are not ideal results. The ROC curve of the proposed algorithm is the more gradual curve. SIFT has better performance in these scene.

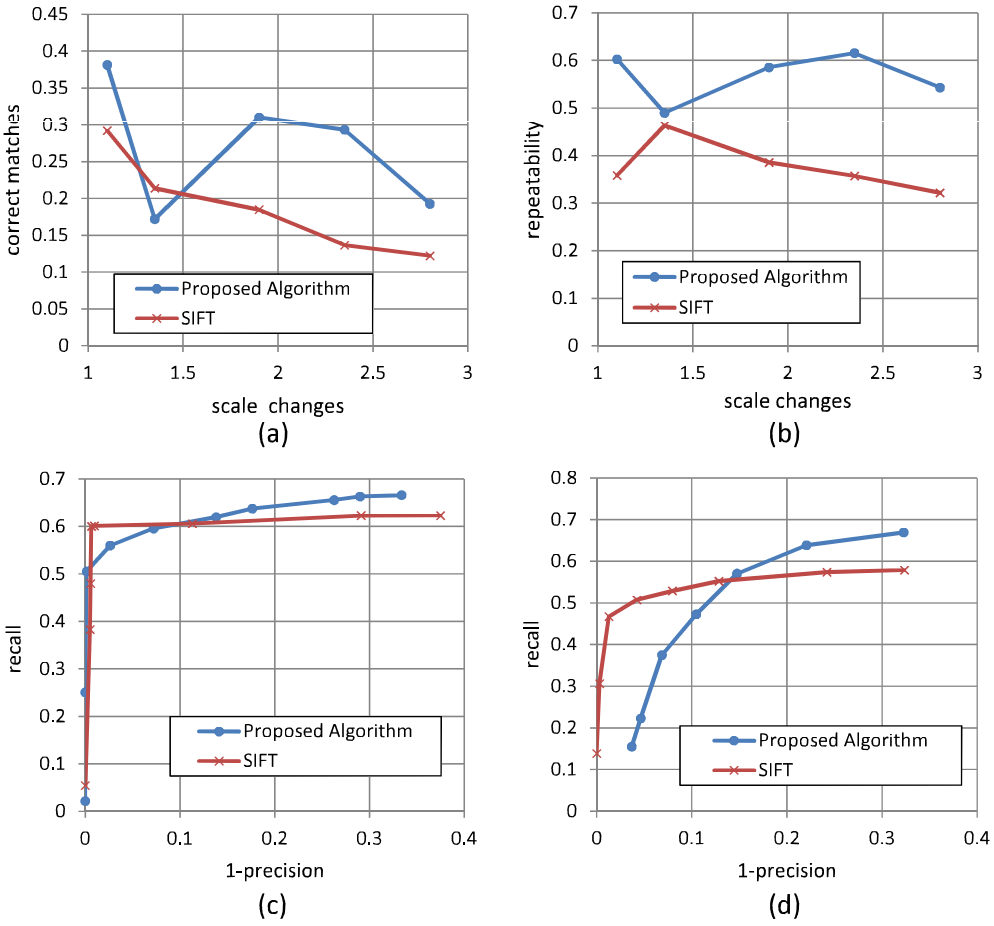

Figure 3.13: The comparison between the proposed algorithm and SIFT. (a) Number of correct nearest neighbour matches (Boat sequence). (b) Repeatability score for scale change (Boat sequence). (c) ROC curve (Wall sequence). (d) ROC curve (Bikes sequence).

Figure 3.14 shows the performance of the keypoint detection of the pro-

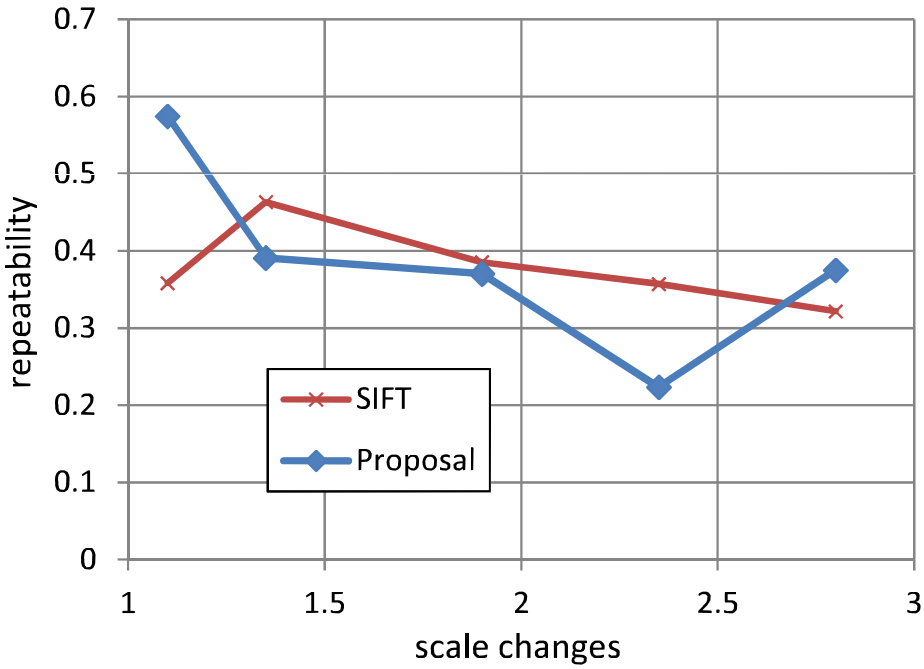

Figure 3.14: Performance of keypoint detection with the proposed algorithm and SIFT.

posed algorithm and conventional algorithm, SIFT [11]. Repeatability was evaluated. We used the framework proposed by Mikolajczyk *et al.* [45]. The boat image shown in the paper was used as a test sequence. It includes both scale changes and rotation. The detected keypoints in the images are shown in Fig. 3.15. The transformation matrix $\mathbf{H}$ for definition of correct matching is obtained by RANSAC [46]. Repeatability is defined as

$$Repeatability(\epsilon) = \frac{R(\epsilon)}{min(n_1, n_2)}, \tag{3.9}$$

$$R(\epsilon) = \{(\mathbf{x}_1, \mathbf{x}_2) | dist(\mathbf{H}\mathbf{x}_1, \mathbf{x}_2) < \epsilon\}, \tag{3.10}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are numbers of keypoints of evaluated two images. $mini()$ function selects the lower value from two input numbers. $dist(\mathbf{H}\mathbf{x}_1, \mathbf{x}_2)$ rep-

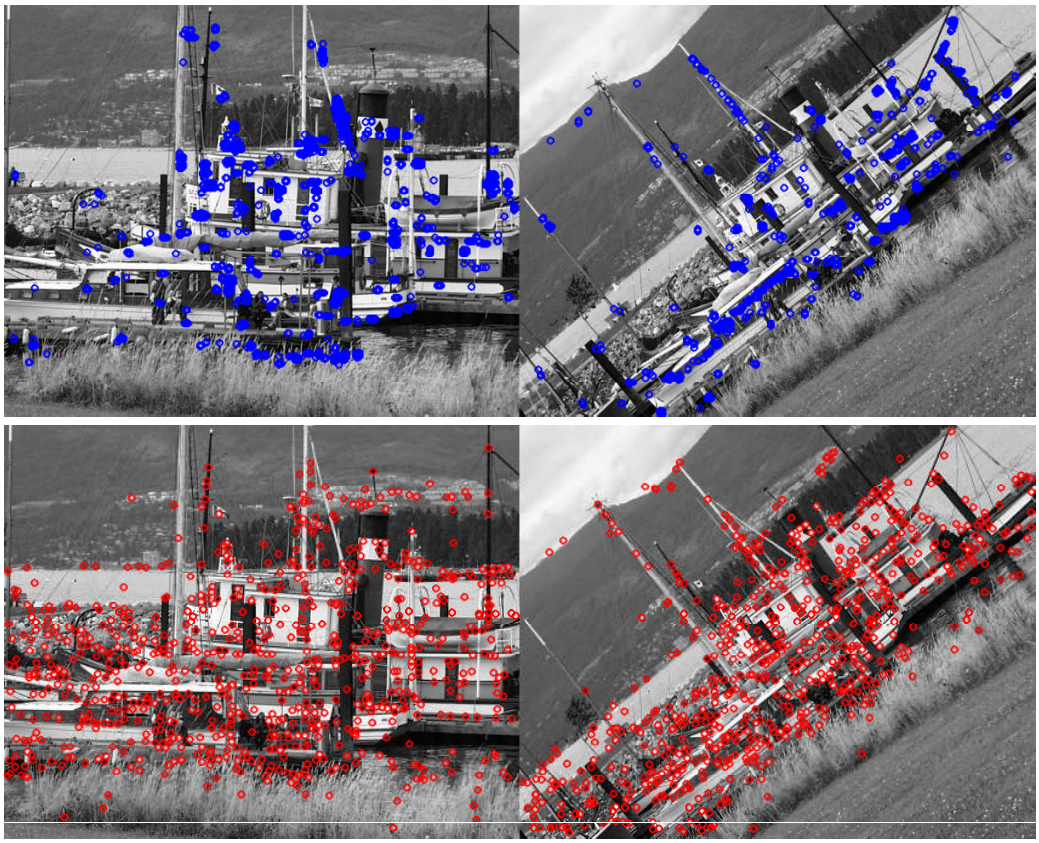

Figure 3.15: Keypoints which are detected by the proposed algorithm (top) and SIFT (bottom).

resents detection errors. In this experiment, we fixed $\epsilon = 10$. The proposed method shows the results of the keypoint matching with multi-scale (0.5, 1.0 and 2.0 times) images of the same objects. Scale is the size of the target objects compared with one image. The result shows the proposed algorithm has almost the same accuracy compared with SIFT. It is considered that the stability of detected keypoints is not greatly changed because the proposed method also utilized gradient-based keypoint detection and it greatly does not differ from conventional methods which calculate gradient on a region.

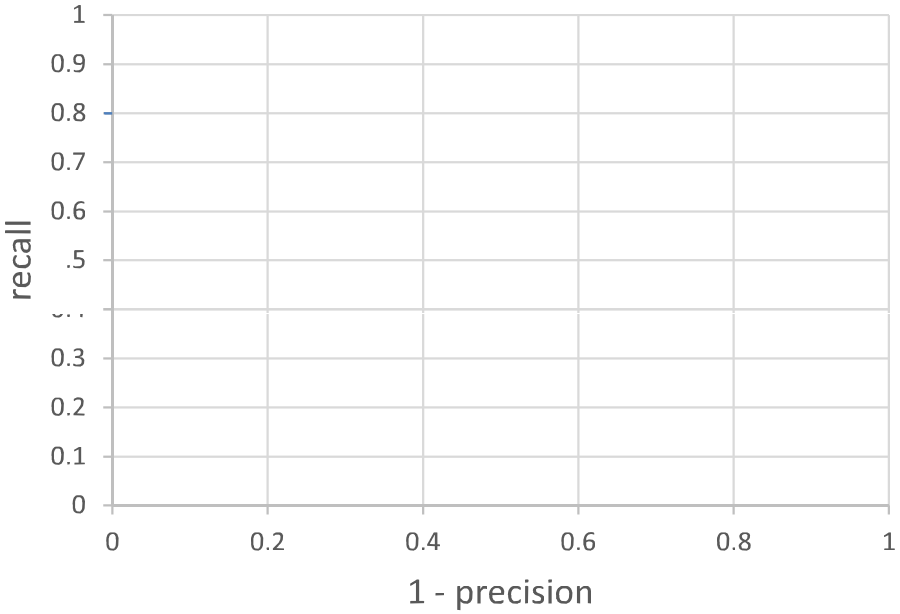

Figure 3.16: Performance of the MRF based method and proposal.

3.4.1.2 Keypoint clustering performance

Figure 3.16 shows comparison of smoothing KOI methods between MRF based algorithm [40] and proposal utilizing precision and recall. The KOI which are obtained by MRF based algorithm are defined as correct detection points. The keypoints which are obtained by proposal were evaluated. We utilized the video in surveillance scene and smoothing methods of KOI are compared shown as Fig. 3.17. *Recall* is higher than 0.8 when $1 - precision$ has lower value. This means that keypoints which are obtained by MRF based method and proposal are located in almost same position.

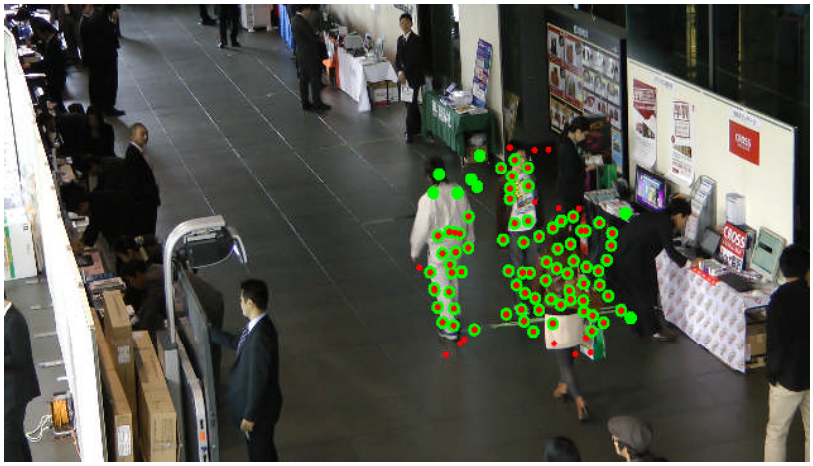

Figure 3.17: Keypoints of the MRF based method (red keypoints) and proposal (green keypoints).

3.4.2 Evaluation of processing performance

This section shows the evaluation results of the processing speeds and utilized resources. First, evaluation conditions are described. Then, evaluation results are shown. The evaluation conditions show the parameters of implemented hardware. The evaluation results show the results of processing time and utilized resource of the proposed FPGA architecture compared with conventional methods.

Table 3.1: The parameter on evaluation condition.

Block	Parameter
Gaussian filter	5 [pixel] x 5 [pixel] x 5 [module]
keypoint memory	1024 [bit] x 1024 x 2
Register of grid division	8 x 8 [grid] x 4 [bit]

3.4.2.1 Evaluation condition

The evaluation condition is shown in Tab. 3.1. The input data is 24bit of RGB color. The calculation of conversion, RGB to Gray, is Y of YUV color space. The data is converted to 8 bits. Five Gaussian filter modules of smoothing processing for detection are utilized for stable keypoint extraction. Line memory memorizes pixels in 15 x 15 regions. Then, the 9-bit magnitude and 5-bit orientation are calculated in the gradient generation. The orientation calculation generates a 32-dimension histogram. The SIFT descriptor calculation generates an 128-dimension descriptor. The Keypoint detection module generates a 1-bit signal which shows if a point is a keypoint or not. In the inter-frame matching module, a database keypoint memory and two-keypoint memory are utilized. In the KOI selection module, a 4-bit register on each 8x8 grid is utilized during the smoothing of the KOI. Finally, the selector generates a 1-bit signal which shows whether the pixel is a KOI or not.

3.4.2.2 Evaluation results

The proposed algorithm is compared with SIFT with respect to performance and speed. The development environment for software is Visual Studio C++ 2008. CPU is Intel Core i5 CPU M 450 2.40GHz. Vertex-5 (XC5VLX330-1FF1760C, Fig 3.18) as FPGA offered by Xilinx, Inc. and option boards (TB-SUB-DVI) are used for hardware evaluation. They are connected each others, input videos are entered into the FPGA and the calculation results are displayed in monitor like Fig. 3.20. Specifications of the boards are shown in Tab. 3.2 and 3.3. The logic synthesis on FPGA is performed by ISE14.7. The evaluation is shown by two steps. First step is the FPGA implementation of

spatial-feature-based keypoint extraction. Second is that of spatio-temporal keypoint extraction.

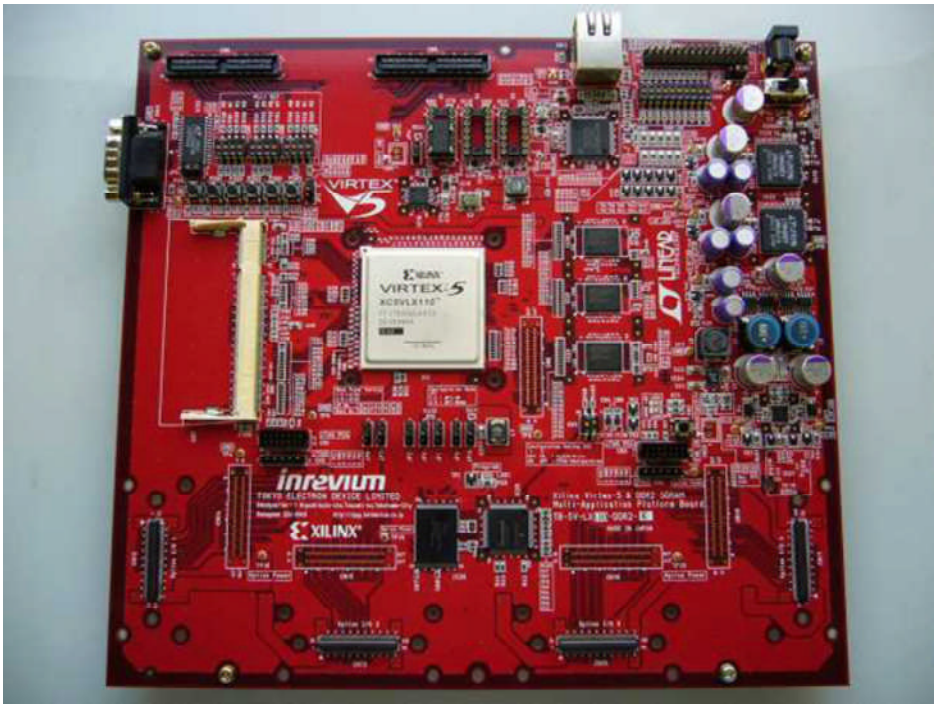

Figure 3.18: FPGA (TB-5V-LX330)

Table 3.2: Specification of FPGA board (TB-5V-LX330-DDR2)

FPGA	XC5VLX330-1FF1760
DDR2SDRAM memory	512Mbit × 3
DDR2SDRAM SO-DIMM	Maximum 2GB
Ethernet	10/100 Base Ethernet MAC & PHY
Serial communication	RS232C
I/O port	LVDS I/O, generic I/O, option I/O

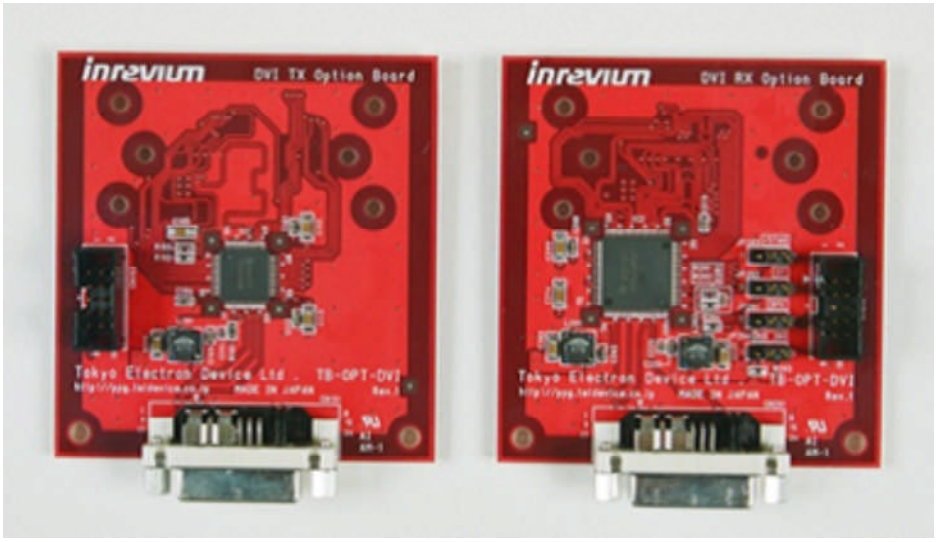

Figure 3.19: Option board (TB-SUB-DVI)

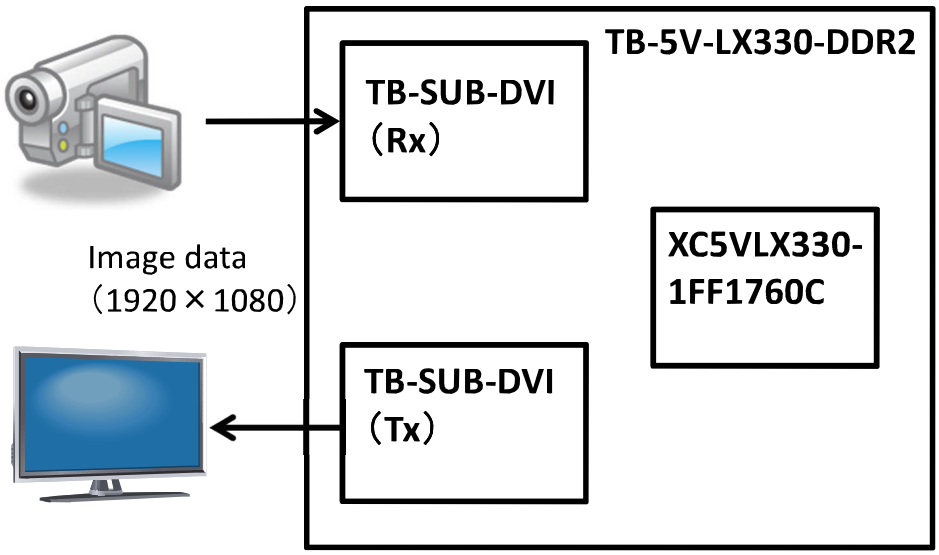

Figure 3.20: FPGA environment

3.4.2.3 Evaluation of spatial-feature-based keypoint extraction

Hardware of only spatial-feature-based keypoint extraction on the FPGA and the algorithm on software are compared with SIFT and SURF. SIFT imple-

Table 3.3: Available resources of FPGA device (XC5VLX330-1FF1760C)

device	XC5VLX330
array	240 × 108
slice	51,840
maximum variance RAM (Kb)	3,420
DSP48E slice	192
maximum block RAM (Kb)	10,368
all I/O bank	33
maximum user I/O	1,200

Table 3.4: Comparison of processing time (SW or HW/VGA or Full-HD)

			Processing time [ms]	Number of keypoints
SIFT	SW	VGA	1026	252
SURF			184	254
Proposal			**85**	258
SIFT		Full -HD	6840	986
SURF			2043	980
Proposal			982	950
Proposal	HW		**16**	1024

mented by Hess [54] and SURF in OpenCV2.1 library are used. The evaluation is performed by VGA (640× 480) and Full-HD (1920× 1080). The number of keypoints is almost same in each resolution. Table 3.4 is a result comparing processing time per frame between SIFT and the proposals. The result comparing proposal (SW/VGA) with SIFT (SW/VGA) shows that the proposed algorithm is about 12 times faster than SIFT on software. Moreover, it is about 2 times faster than SURF. It performs 10 fps keypoint extraction and match-

ing for VGA video. The result comparing proposal (HW/Full-HD) with SIFT (SW/Full-HD), proposal (SW/Full-HD) shows that the proposed hardware on FPGA is about 427 times faster than SIFT and about 61 times faster than the proposed algorithm on software. The proposed hardware performs at 60 fps keypoint extraction and matching for Full-HD video. The number of keypoints is also shown, but the processing time of keypoint extraction does not depend on the number of keypoints due to pixel pipelining. On the other the hand, keypoint matching depends on the number of keypoints due to keypoint pipelining. We have verified that it is possible to perform keypoint matching up to 1024 keypoints on the FPGA.

3.4.2.4 Evaluation of spatio-temporal keypoint extraction

Table 3.5: Resource comparison of conventional work and proposal.

		Conventional work		Proposed architecure	
FPGA resource	Available	Used	Utilization	Used	Utilization
slice register	207,360	55,407	26%	50,466	24%
slice LUTs	207,360	108,322	52%	113,657	54%
occupied slices	51,840	32,741	63%	32,274	62%
BlockRAM/FIFO	288	89	30%	150	52%
DSP48Es	192	3	1%	0	0%
Maximum frequency		168.464 MHz		160.197 MHz	

The hardware resources including spatial and spatio-temporal methods are shown in Tab. 3.5. It shows that the resource of the proposed spatio-temporal keypoint extraction FPGA architecture are almost the same as that of the conventional keypoint extraction hardware [41] which utilizes only spatial in-

formation. Utilization of the slice register, occupied slice and DSP48Es are 24%, 62% and 0%, respectively. They are lower than the resource utilization of conventional FPGA architecture. DSP48Es is completely removed by the proposed detection algorithm. Utilization of slice LUTs and BlockRAM/FIFO are 54% and 52%, respectively. They are slightly higher than the resource utilization of conventional FPGA architecture. In particular, BlockRAM/FIFO is utilized in the inter-frame matching of the keypoints part by the proposed FPGA architecture. Implementation on lower and cheaper resourced devices is expected because all utilization is lower than available resources.

Next, table 3.6 shows processing time. SW and HW of the spatio-temporal keypoint extraction are the proposed methods. Regarding software implementation, processing time of the conventional keypoint extraction algorithm [41] is 754 ms per frame. Processing time of the spatio-temporal keypoint extraction algorithm [40, 53] is 190 ms per frame. Thus, the proposed keypoint extraction hardware is 47 times faster than the spatial-feature-based keypoint extraction of the software and 12 times faster than the spatio-temporal keypoint extraction of the software. In this case, the number of keypoints which can be memorized is 1024 to compare performance utilizing same conditions with other methods. However, if the sizes of the BRAM1 and BRAM2 are expanded, it is possible to increase the number of keypoints which can be processed. The number of keypoints can be increased up to $n_{(t-1)} \cdot n_t \leq 2073600$, where n_t represents the number of keypoints in t frame, because the keypoint matching between $(t-1)$-frame keypoints and t-frame keypoins must be processed in one frame.

Table 3.7 shows the processing speed of the proposed FPGA architecture

and conventional methods [36, 37, 38]. The proposed FPGA architecture achieves keypoint extraction processing of Full-HD (1920x1080 pixel) 60 fps. There is no hardware proposal for spatio-temporal keypoint extraction. Thus, implementation of the spatial-feature-based keypoint extraction is shown as the conventional works. The proposed FPGA architecture of spatio-temporal keypoint extraction is a higher processing speed. This is because in the conventional method, peripheral information of keypoints is buffered in a memory after keypoint extraction, and then feature amounts are generated sequentially. In the proposed method, the keypoint extraction, the number of resources of the descriptor generation module is reduced and the pipeline structure is implemented to operate at the same frequency, and it operates with the video clock.

Table 3.6: Processing time of keypoint extraction.

	Number of keypoint	Number of KOI	Processing time [ms]
keypoint extraction (SW)	1192	N/A	754
Spatio-temporal keypoint extraction (SW)	1192	64	190
Spatio-temporal keypoint extraction (HW)	up to 1024	up to 1024	16

Table 3.7: Performance comparison of conventional works and proposal.

	Proposal	[34]	[35]	[41]	[37]	[38]
Clock frequency [MHz]	160.197	100	145.0	168.464	100	134
Keypoint extraction	Spatio-temporal	Spatial (only detection)	Spatial	Spatial	Spatial	Spatial
Resolution	Full-HD	QVGA	VGA	Full-HD	VGA	VGA
Frame rate [fps]	60	30	40	60	30	30
Device	Virtex-5	Stratix II	Virtex-5	Virtex-5	DE2-70	Virtex-5

Figure 3.21 shows a system utilizing an FPGA board. The system is composed of FPGA board, camera and monitor. It shows the real-time performance of keypoint matching by the proposed spatio-temporal keypoint extraction.

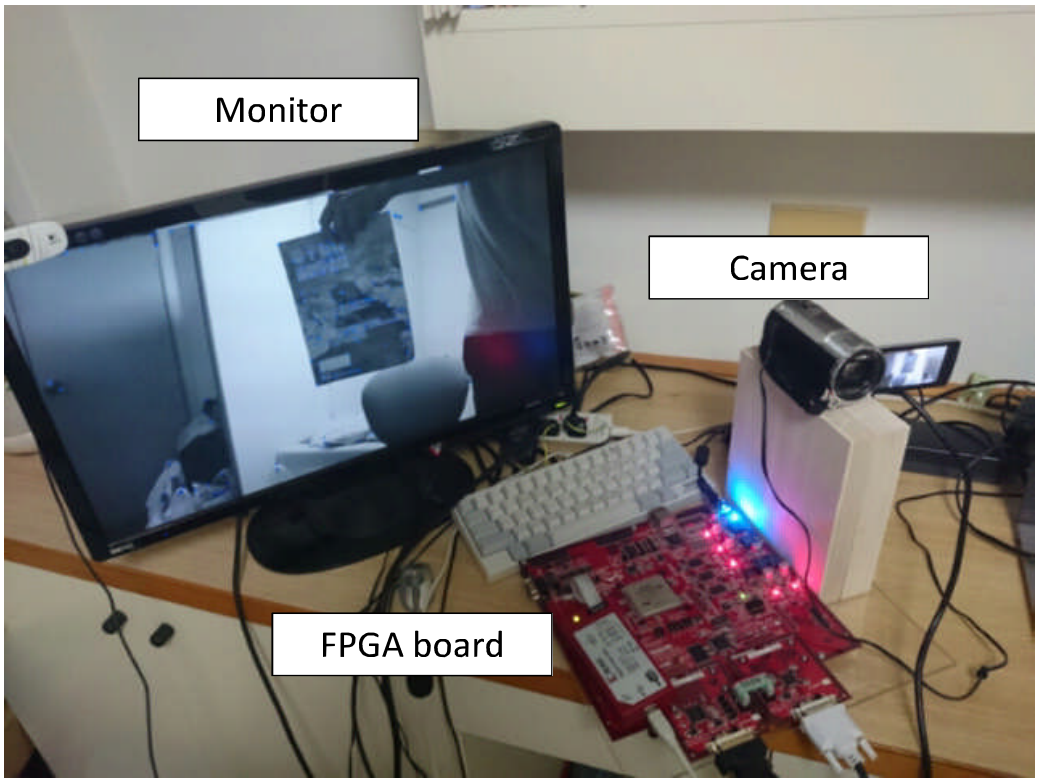

Figure 3.21: A system utilizing FPGA board.

3.5 Conclusion

This chapter described the proposed hardware-friendly algorithm of spatio-temporal keypoint extraction and its FPGA implementation. The proposed algorithm and implementation method are evaluated by implementation on real FPGA. These proposals realize FPGA resource reduction and enhance-

ment of processing speed by combining.

The algorithm is based on optical flow computation parallelization of memory utilization and dual threshold keypoint detection for real-time cloud-based recognition systems to achieve high-speed systems and reduce hardware resources containing the specification of processing. The KLT tracker which obtains motion information is replaced with inter-frame keypoint matching, and a detection algorithm which is processed with the descriptor calculation in parallel is replaced with a gradient-histogram based algorithm. A FPGA implementation of memory utilization and grid division, and dense-clustering based keypoint smoothing method are shown.

Finally, the evaluation results of the FPGA implementation showed that the implemented hardware achieves Full-HD (1920x1080)-60 fps spatio-temporal keypoint extraction and is 47 times faster than the low-complexity keypoint extraction on the software and 12 times faster than spatio-temporal keypoint extraction on the software, and the hardware resource is almost the same as with SIFT implementation, maintaining accuracy.

Chapter 4

Conclusion

This thesis summarizes the technology for reducing the amount of data to be transmitted to the cloud and the technology for realizing real-time processing at the user terminal in the cloud video recognition system. The cloud systems start to be utilized for services which analyze user's data in the region of multimedia. In video recognition, there ares few services that has been put to practical use yet. The system is assumed that keypoints are extracted from videos, the data is sent to cloud, and the data is used for identification processing in cloud server. The system includes a keypoint extraction part in user terminal and a identification part in cloud server. In the keypoint extraction part of the system, there are two problems of large number of keypoints to be sent to cloud server and real-time processing of keypoint extraction in user terminal. Chapter 2 has solved the first problem and chapter 3 has solved the second problem. The achievements obtained by the research are summarized below.

Chapter 2 proposed a keypoint extraction algorithm to reduce the amount of data sent to the cloud by combining spatio-temporal keypoint extraction with local correlation. Along with the progress of IoT, the number of termi-

nals connected to the edge in the network increases, and it is thought that the amount of data flowing in the network will increase. For the first time, this chapter proposed the keypoint extraction algorithm to reduce data amount for cloud video recognition in the IoT era and have presented its effectiveness. This chapter proposed spatio-temporal keypoint extraction based on local correlation to reduce the number of keypoints. The algorithm is composed of three elements: spatial information, temporal information and connectivity of adjacent keypoints. The proposed method includes an approximated KLT tracker to calculate positions of keypoints and optical flows simultaneously. This algorithm calculates weights at each keypoint using two kinds of features, namely, an intensity gradient and an optical flow. It reduces noise of extraction by comparing states of the intended keypoint with states of surrounding keypoints. Camera motion estimation is added to the algorithm and it calculates camera-motion invariant optical flows. Evaluation showed the recall-precision performance and number of keypoints by the proposed method to confirm that the number of keypoints to be sent to cloud server is reduced. The proposed algorithm achieves about 93% reduction of keypoints compared with conventional keypoint extraction while detecting required keypoints (17-45% enhancement of F-measure). These results shows that reduction of the number of keypoints to be sent to cloud server is realized while detecting necessary keypoints for cloud video recognition.

Chapter 3 proposed a hardware-friendly algorithm and its FPGA implementation which realizes real-time keypoint extraction in user terminal on cloud system by utilizing gradient histogram effectively and parallelizing them. There is no conventional FPGA implementation of keypoint extraction for

cloud video recognition. In the hardware of keypoint extraction, there is no work whose target is real-time processing for VGA. For the first time, this chapter proposed the Full-HD 60 fps FPGA implementation of keypoint extraction for cloud video recognition and demonstrates it by real FPGA. This chapter proposed the hardware-friendly KOI algorithm with low amount of computations and its real-time FPGA implementation based on dual threshold keypoint detection by gradient histogram and parallelization of connectivity of adjacent keypoint-utilizing register counters. The algorithm utilizes dual-histogram based detection and keypoint-matching based calculation of motion information and dense-clustering based keypoint smoothing. The hardware architecture is composed of a detection module utilizing descriptor, and grid-region-parallelization based density clustering. The detection and the descriptor generation modules are placed in parallel. The descriptor generation module is parallelized by considering grid placements. The processing time of descriptor computation in this hardware is independent of the number of keypoints because its descriptor generation is pipelining structure of pixel. Evaluation showed the hardware resource amount, processing time and keypoint detection performance to confirm real-time performance. The proposed hardware achieves Full-HD 60 fps spatio-temporal keypoint extraction on FPGA. Further, it is 47 times faster than low complexity keypoint extraction on software and 12 times faster than spatio-temporal keypoint extraction on software, and the hardware resources are almost the same as SIFT hardware implementation, maintaining accuracy. These results shows that real-time processing of keypoint extraction for cloud video recognition is realized.

By these proposals, the keypoint extraction algorithm for cloud video recog-

nition and its real time system have been realized. The algorithm and hardware are expected to be applied to various applications such as surveillance cameras and in-vehicle cameras by cloud video recognition. Two main future works are remaining to be done in order to realize the entire cloud video recognition system. Firstly, evaluation of recognition accuracy is necessary to realize applications. The framework of the recognition consists of keypoint extraction and identification. It is necessary to evaluate recognition accuracy by combining the proposed keypoint extraction with identification processing. As identification processing, support vector machine (SVM) is widely used. Since multiple kinds of applications are assumed, evaluation of optimizing parameters of the proposed algorithm and SVM is required for each application. Secondly, it is necessary to evaluate the real-time performance including the entire cloud system. In the proposed hardware, it is limited to achieving real-time performance in the user terminal. The actual cloud video recognition system includes a process in which keypoint flows through the network, a process of identifying, and a process of returning information to the user. These processes also cause dela l-time verification via the network is necessary to confirm whether real-time performance can be achieved in the whole system.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] M.-y. Chen and A. Hauptmann, "Mosift: Recognizing human actions in surveillance videos," Technological report, vol. CMU-CS-09-161, pp. 1–16, 2009.

[4] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. IEEE, 2005, pp. 65–72.

[5] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," Transportation Research Part C: Emerging Technologies, vol. 6, no. 4, pp. 271–288, 1998.

REFERENCES

[6] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua, "Fully automated and stable registration for augmented reality applications," in Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality. IEEE Computer Society, 2003, p. 93.

[7] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette, "Real-time markerless tracking for augmented reality: the virtual visual servoing framework," IEEE Transactions on visualization and computer graphics, vol. 12, no. 4, pp. 615–628, 2006.

[8] X. Zhang, S. Fronz, and N. Navab, "Visual marker detection and decoding in ar systems: A comparative study," in Proceedings of the 1st International Symposium on Mixed and Augmented Reality. IEEE Computer Society, 2002, p. 97.

[9] S.-S. Huang, C.-J. Chen, P.-Y. Hsiao, and L.-C. Fu, "On-board vision system for lane recognition and front-vehicle detection to enhance driver's awareness," in Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, vol. 3. IEEE, 2004, pp. 2456–2461.

[10] C. Yang, H. Hongo, and S. Tanimoto, "A new approach for in-vehicle camera obstacle detection by ground movement compensation," in 2008 11th International IEEE Conference on Intelligent Transportation Systems. IEEE, 2008, pp. 151–156.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. Journal of Computer Vision, vol. 60, pp. 91–110, 2004.

[12] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in European conference on computer vision. Springer, 2006, pp. 404–417.

[13] G. Michael, G. Helmut, and B. Horst, "Fast approximated SIFT," Proc. of ACCV,, pp. 918–927, 2006.

[14] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," IEEE Transactions on Pattern Analysis, vol. 10, no. 27, pp. 1615–1630, 2005.

[15] Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, vol. 2. IEEE, 2004, pp. II–II.

[16] A. E. Abdel-Hakim and A. A. Farag, "Csift: A sift descriptor with color invariant characteristics," in Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, vol. 2. IEEE, 2006, pp. 1978–1983.

[17] J.-M. Morel and G. Yu, "Asift: A new framework for fully affine invariant image comparison," SIAM Journal on Imaging Sciences, vol. 2, no. 2, pp. 438–469, 2009.

[18] I. Laptev and T. Lindeberg, "Space-time interest points," in ICCV, 2003, pp. 432–439.

[19] C. M, H. A, and L. H, Informedia @ TRECVID2009: analyzing video motions. In: Proc. of the TRECVID workshop, 2009.

[20] G. Willems, T. Tuytelaars, and L. VanGool, "An efficient dense and scale-invariant spatio-temporal interest point detector," Proc European Conf. Computer Vision, vol. II, pp. 650–663, 2008.

[21] L. Juan and O. Gwun, "A comparison of sift, pca-sift and surf," International Journal of Image Processing (IJIP), vol. 3, no. 4, pp. 143–152, 2009.

[22] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "Gpu-based video feature tracking and matching," in EDGE, Workshop on Edge Computing Using New Commodity Architectures, vol. 278, 2006, pp. 4321–4335.

[23] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "Feature tracking and matching in video using programmable graphics hardware," Machine Vision and Applications, vol. 22, no. 1, pp. 207–217, 2011.

[24] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 1, no. 3, pp. 231–240, 2011.

[25] P. Derbeko, S. Dolev, E. Gudes, and J. D. Ullman, "Concise essence-preserving big data representation," in Big Data (Big Data), 2016 IEEE International Conference on. IEEE, 2016, pp. 3662–3665.

[26] F. Hadiatna, H. Hindersah, D. Yolanda, and M. A. Triawan, "Design and implementation of data logger using lossless data compression method for

internet of things," in System Engineering and Technology (ICSET), 2016 6th International Conference on. IEEE, 2016, pp. 105–108.

[27] M. Amarlingam, P. K. Mishra, K. D. Prasad, and P. Rajalakshmi, "Compressed sensing for different sensors: A real scenario for wsn and iot," in Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on. IEEE, 2016, pp. 289–294.

[28] E. Zimos, J. F. Mota, M. R. Rodrigues, and N. Deligiannis, "Internet-of-things data aggregation using compressed sensing with side information," in Telecommunications (ICT), 2016 23rd International Conference on. IEEE, 2016, pp. 1–5.

[29] C. J. Deepu, C.-H. Heng, and Y. Lian, "A hybrid data compression scheme for power reduction in wireless sensors for iot," IEEE transactions on biomedical circuits and systems, vol. 11, no. 2, pp. 245–254, 2017.

[30] A. H. M. Amin, N. M. Ahmad, and A. M. M. Ali, "Decentralized face recognition scheme for distributed video surveillance in iot-cloud infrastructure," in Region 10 Symposium (TENSYMP), 2016 IEEE. IEEE, 2016, pp. 119–124.

[31] J. Huang and Z.-N. Li, "Automatic detection of object of interest and tracking in active video," in Pacific-Rim Conference on Multimedia. Springer, 2009, pp. 368–380.

[32] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in 2009 IEEE 12th International Conference on Computer Vision. IEEE, 2009, pp. 1219–1225.

[33] G. Francini, S. Lepsøy, and M. Balestri, "Selection of local features for visual search," Signal Processing: Image Communication, vol. 28, no. 4, pp. 311–322, 2013.

[34] V. Bonato, E. Marques, and G. A. Constantinides, "A parallel hardware architecture for scale and rotation invariant feature detection," IEEE Trans. Circuits Syst. Video Technol., vol. 18, p. 1703, 2008.

[35] J. Qiu, T. Huang, and T. Ikenaga, A FPGA-based dual-pixel processing pipelined hardware accelerator for feature point detection part in SIFT. in Proc. 5th Int. Joint Conf. INC IMS IDC,, 2009.

[36] J. Jiang, X. Li, and G. Zhang, "Sift hardware implementation for real-time image feature extraction," IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 7, pp. 1209–1220, 2014.

[37] F.-C. Huang, S.-Y. Huang, J.-W. Ker, and Y.-C. Chen, "High-performance sift hardware accelerator for real-time image feature extraction," IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 3, pp. 340–351, 2012.

[38] V. Bonato, E. Marques, and G. A. Constantinides, "A parallel hardware architecture for scale and rotation invariant feature detection," IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, no. 12, pp. 1703–1712, 2008.

[39] B. Yu, L. Wang, and Z. Niu, "A novel algorithm in buildings/shadow detection based on harris detector," Optik-International Journal for Light and Electron Optics, vol. 125, no. 2, pp. 741–744, 2014.

[40] T. Suzuki and T. Ikenaga, "Spatio-temporal feature and mrf based keypoint of interest for cloud video recognition," IIEEJ Transaction on Image Electronics and Visual Computing, vol. 2, no. 2, pp. 150–158, 2014.

[41] T. Suzuki and T. Ikenaga, "Low complexity keypoint extraction based on sift descriptor and its hardware implementation for full-hd 60 fps video," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 96, no. 6, pp. 1376–1383, 2013.

[42] T. Suzuki and T. Ikenaga, "Full-hd 60fps fpga implementation of spatio-temporal keypoint extraction based on gradient histogram and parallelization of keypoint connectivity," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 99, no. 11, pp. 1937–1946, 2016.

[43] P. R. Beaudet, "Rotationally invariant image operators," in International Joint Conference on Pattern Recognition, vol. 579, 1978, p. 583.

[44] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu, "An optimal algorithm for approximate nearest neighbor searching,," Journal of the ACM, vol. 6, pp. 891–923, 1998.

[45] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," Int. J. of Computer Vision, vol. 65, no. 1/2, pp. 43–72, 2005.

[46] M. A. Fischler and R. C. Bolles, Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. Commun. Assoc. Comp., 1981.

[47] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," IEEE transaction on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pp. 1615–1630, 2005.

[48] C. Tomasi and T. Kanade, Detection and Tracking of Point Features,. Carnegie Mellon University Technical Report CMU-CS-91-132, apr 1991.

[49] B. D. Lucas and T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision. International Joint Conference on Artificial Intelligence, 1981.

[50] K. Tanaka, "Statistical-mechanical approach to image processing," Mathematical and General, vol. 35, no. 37, pp. R81–R150, 2002.

[51] A. S. Willsky, "Multiresolution markov models for signal and image processing," Proceedings of the IEEE, vol. 90, no. 8, pp. 1396–1458, 2002.

[52] T. Suzuki and T. Ikenaga, "Sift-based low complexity keypoint extraction and its real-time hardware implementation for full-hd video," in Signal & Information Processing Association Annual Summit and Conference (APSIP -Pacific. IEEE, 2012, pp. 1–6.

[53] T. Suzuki and T. Ikenaga, "Keypoint of interest based on spatio-temporal feature considering mutual dependency and camera motion," in International Academy, Research and Industry Association, IARIA, 2014.

[54] R. Hess, An open-source SIFTLibrary. In Proceedings of the international, 2010.

Publications

Journal

◯ [1] **Takahiro Suzuki**, Takeshi Ikenaga, "Full-HD 60fps FPGA Implementation of Spatio-temporal Keypoint Extraction Based on Gradient Histogram and Parallelization of Keypoint Connectivity," IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E99-A, No. 11, pp. 1937–1946, Nov. 2016.

[2] Kubota Eijiro, **Takahiro Suzuki**, Honda Masaaki, Takeshi Ikenaga, "Action Detection of Volleyball Using Features Based on Clustering of Body Trajectories (in Japanese)," IIEEJ TRANSACTIONS on Image Electronics and Visual Computing, Vol. 45, No. 3, pp. 373–381, Jul. 2016.

[3] Sang-Yuep Kim, Noriko Iiyama, Jun-ichi Kani, Ryo Koma, **Takahiro Suzuki**, Ken-Ichi Suzuki, Akihiro Otaka, "Real-time demonstration of 30 Gbit/s hierarchical star 8-QAM passive optical network employing fractionally-spaced signed-error radius directed equalisation," Electronics Letters, Vol. 52, Issue 7, pp. 544 - 546 , Apr. 2016.

◯ [4] **Takahiro Suzuki**, Takeshi Ikenaga, "Spatio-Temporal Feature and

MRF Based Keypoint of Interest for Cloud Video Recognition," IIEEJ TRANSACTIONS on Image Electronics and Visual Computing, Vol. 2, No. 2, pp. 150–158, Dec. 2014.

○ [5] **Takahiro Suzuki**, Takeshi Ikenaga, "Low Complexity Keypoint Extraction Based on SIFT Descriptor and Its Hardware Implementation for Full-HD 60 fps Video," IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E96-A, No. 6, pp. 1376–1383, Jun. 2013.

International Conference with review

[1] **Takahiro Suzuki**, Sang-Yuep Kim, Jun-ichi Kani, Ken-Ichi Suzuki, Akihiro Otaka, " Real-time Demonstration of PHY Processing on CPU for Programmable Optical Access Systems ", Global Communications Conference (GLOBECOM 2016), Dec. 2016.

[2] Ryo Koma, Masamichi Fujiwara, Jun-ichi Kani, Sang-Yuep Kim, **Takahiro Suzuki**, Hideki Mori, Tomoyuki Wada, Ken-Ichi Suzuki, Akihiro Otaka, "22-dB Dynamic Range, Real-Time Burst-Mode Reception of Digital Coherent 20-Gb/s QPSK PON Upstream Signals," 42nd European Conference on Optical Communication (ECOC2016), Sep. 2016.

[3] **Takahiro Suzuki**, Sang-Yuep Kim, Jun-ichi Kani, Ken-Ichi Suzuki, Akihiro Otaka,"Parallelization of Cipher Algorithm on CPU/GPU for Real-time Software-Defined Access Network," APSIPA Annual Summit and

Conference (ASC 2015), Oct. 2015.

◯ [4] **Takahiro Suzuki**,Takeshi Ikenaga, "Keypoint of Interest Based on Spatio-temporal Feature Considering Mutual Dependency and Camera Motion," Sixth International Conferences on Advances in Multimedia (MMEDIA 2014), Feb.2014.

◯ [5] **Takahiro Suzuki**, Takeshi Ikenaga, "Keypoints of Interest Based on Spatio-Temporal Feature and MRF for Cloud Recognition System", APSIPA Annual Summit and Conference (ASC 2013), Oct. 2013.

◯ [6] **Takahiro Suzuki**, Takeshi Ikenaga, "SIFT-Based Low Complexity Keypoint Extraction and Its Real-Time Hardware Implementation for Full-HD Video", APSIPA Annual Summit and Conference (ASC 2012), Dec. 2012.

◯ [7] **Takahiro Suzuki**, Takeshi Ikenaga, "Corner Detection Based Low Complexity SIFT Algorithm for Real-time Keypoints Matching", RISP International Workshop on Nonlinear Circuits, Communications and Signal Processing (NCSP 2012), Mar. 2012.

Invited Talk

[1] Takeshi Ikenaga, **Takahiro Suzuki**, "Smart Feature Detection Device for Cloud based Video Recognition System," 2014 International Symposium on VLSI Design, Automation and Test (2014 VLSI-DAT), Apr. 2014.

National Conference with review

○ [1] **Takahiro Suzuki**, Takeshi Ikenaga, "Real-time Implementation of Corner Detection Based Low-complexity SIFT (in Japanese) ," The 18th Symposium on Sensing via Image Information, Jun. 2012.

Awards

[1] Best paper award, IEEE Communications Society's Transmission, Access, and Optical Systems Committee (TAOS): **Takahiro Suzuki**, Sang-Yuep Kim, Jun-ichi Kani, Ken-Ichi Suzuki, Akihiro Otaka, " Real-time Demonstration of PHY Processing on CPU for Programmable Optical Access Systems," Global Communications Conference (GLOBECOM 2016), Dec. 2016.

○ [2] Best paper award: **Takahiro Suzuki**, Takeshi Ikenaga, "Keypoint of Interest Based on Spatio-temporal Feature Considering Mutual Dependency and Camera Motion," Sixth International Conferences on Advances in Multimedia (MMEDIA 2014), Feb.2014.

Other Presentations

[1] **鈴木貴大**, キムサンヨプ, 可児淳一, 鈴木謙一, 大高明浩”光アクセス装置プログラマブル化に向けたCPU/GPUによる暗号化の並列実装の検討”, 通信方式研究会，Apr. 2016.

[2] **鈴木貴大**, キムサンヨプ, 可児淳一, 鈴木謙一, 大高明浩”光アクセス装置プログラマブル化に向けたデータ並列に基づく暗号化のCPU/GPU 実装”, 2016 年電子情報通信学会総合大会，Mar. 2016.

[3] **鈴木貴大**, キムサンヨプ, 可児淳一, 鈴木謙一, 大高明浩”将来アクセスネットワークに向けたプログラマブル OLT の提案”, 2015 年電子情報通信学会ソサイエティ大会，Sep. 2015.

[4] **鈴木貴大**, キムサンヨプ, 飯山法子, 可児淳一, 鈴木謙一,”階層変調によるディジタルコヒーレントシステムと 10G-EPON の共存実証 ”, 2015 年電子情報通信学会総合大会，Mar. 2015.

○ [5] **鈴木貴大**, 池永剛, ”時空間特徴と MRF に基づく実時間 Keypoint of Interest 抽出”, 2013 年電子情報通信学会ソサイエティ大会，Sep. 2013.

○ [6] **鈴木貴大**, 池永剛, ”勾配ヒストグラムを利用した特徴点検出と特徴量 記述の並列ハードウェア実装”, 映像メディア処理シンポジウム (IMPS 2012), Oct. 2012.

○ [7] **鈴木貴大**, 池永剛, ”コーナー検出に基づく低演算量SIFT アルゴリ

ズムの実時間ハードウェア実装”, 2012 年電子情報通信学会総合大会，Mar. 2012.

○ [8] **鈴木貴大**, 池永剛, ”積分画像によるコーナー検出と SIFT 特徴量を組み合わせた実時間マッチング”, 映像メディア処理シンポジウム (IMPS 2011), Oct. 2011.