Waseda University Doctor Dissertation

Temporal Mapping and Temporal Prediction based Ultra-low Delay Object Tracking Computing Architecture for Visual Feedback System

Tingting HU

Graduate School of Information, Production and Systems

Waseda University

June 2023

Acknowledgements

I would like to express my deepest gratitude to all thoese who gave me help and support during my master and doctor courses.

To my supervisor, Professor Takeshi Ikenaga, who always gave me suggestion and encouraged me to keep moving forward. It is he who directs me step by step on my research and provides me with all kinds of environment for research. Without his patient instruction, and insightful criticism, the completion of this thesis and my current achievements would not been possible.

To my advisor, Mr. Fuchikami, who led me into the world of ultra-low delay vision system and gave me many valuable suggestions for my research. It is a wonderful experience when discussing the feild of ultra-low delay vision system with him.

To the professors who guide me in improving and finishing this thesis, including Professor Shintaro Yamasaki, Professor Shoji Makino, Professor Hirofumi Shinohara, Professor Takaaki Kakitsuka, Professor Kiyoto Takahata, Professor Toshihiko Yoshimasu. They provided many valuable comments. Thanks for their help and patience.

To all the members of Ikenaga lab who gave me support and assistance to my research and daily life. I want to thank Dr. Suzuki who helped me a lot at the beginning of my research when I was quite confused to research directions, and Dr. Xina Cheng who encouraged me when I faced difficulties and tried to give up. I am grateful to senior members including Dr. Gaoxing Chen, Mr. Guojing Zhu, Ms. Shuyi Huang, Ms. Pei Liu, Mr. Hong Wu, Ms. Yuan Wang and Ms. Zihan Ma, who guided me not only in research but also in daily life. We had a memorable time at Kitakyushu. I would also show my thanks to lab colleagues including Dr. Songlin Du, Mr. Yawei Wang, Ms. Fanglu Xie, Ms. Yilin Hou, Ms. Ziwei Deng, Mr. Zhennan Wang, Mr. Yuan Hu, Mr. Yinkai Liu, Mr. Tianming Rao, Mr. Guannan Wu and Mr. Zijie Wang. We attended weekly seminars and joint seminars together, and the discussions inspired me through many challenges. Also, I would like to thank the current lab members, including Ms. Yuan Li, Mr. Yanchao Liu, Mr. Fengshan Zhao, Mr. Hang Zhang, Mr. Zhenhao Yang, Ms. Xiaoyu Shang, Mr. Tianyu Mao, Mr. Ziyue Wang, Mr. Junda Liao, Mr. Zhe Xu, Mr. Xiaomeng Xie, Mr. Yuhong Li, Mr. Zhihan Zhuang. Thanks for their accompanies and assistance.

To all other friends whose names are not listed here. They supported me in life and study in different ways, which were all so important for me. Sometimes, doing research can feel like walking through a dark tunnel with no end in sight. The support from my friends is like opening a window and lighting my heart in this dark tunnel. Thanks for having you there.

Finally, I would like to express appreciations to my beloved families for their support. It's them who gave me the courage to challenge and finish the doctor course in Waseda University.

Abstract

Visual feedback system, which combines virtual (computing) and real spaces by sensing the status of real space and feedbacking valued information to real space after visual processing in virtual space, draws increasing attention in fields such as robotics, factory automation (FA), and entertainment. Object tracking, which includes detection and tracking, is one of the most important visual processing techniques for real-time applications. While different detection and tracking methods exist depending on the application, the keypoint matching-based detection method, which detects rigid objects, and the differential-based tracking method, which tracks the object with high accuracy, are important technologies in FA and robotics. On the other hand, the change in real space continues while visual processing is running in virtual space, so the visual processing with a large delay will introduce large errors, making the ultra-low delay visual processing critical to ensuring the validity of the result outputted from virtual space. Among various visual processing architectures, FPGA (Field-Programmable Gate Array)-based stream architecture is the one that is possible to achieve both complex spatial processing and delay lower than 1 ms while suitable algorithms and architectures are needed. Therefore, the FPGA-based ultra-low delay object tracking algorithm and architecture are required to ensure the validity of the tracking

results when feedbacking the tracking results output from the object tracking process in the virtual space to the real space.

FPGA-based stream architecture achieves ultra-low delay by performing visual processing while transmitting the image data from the sensor to the processor, which also means the processing for each image pixel is fully pipelined. The complex spatial processing in the object tracking algorithms has always been a problem, preventing object tracking from being implemented with fully pipelined architecture. Here shows three types of representative complex spatial processing. Spatial multi-search processing, which explores multiple possibilities corresponding to a factor and selects the best result among them to achieve robustness to the factor, exists in keypoint matching-based detection. When implementing it in FPGA with a fully pipelined style, many resources are required due to the extensive processing performed in the spatial domain for wide-range parameter search. Dependent sequential processing, which sequentially performs two dependent processing, exists in differential-based tracking. The dependent relation for two processing performed in a local spatial area generates not only high resource costs but also high delay. Spatial iterative processing, which performs the same processing iteratively with the output of the previous processing as the input, exists in differential-based tracking. Iterative processing of a large amount of spatial processing not only consumes expensive resources but also causes a significant delay in the FPGA implementation. On the other hand, the object moves along with being affected by disturbance while the tracking processing is running, resulting in the difference between the output state from virtual space

and the true state in real space, where the difference remains albeit greatly reduced by ultra-low delay processing. Especially for tracking, the difference introduces tracking errors caused by delay.

To solve the above problems and achieve ultra-low delay object tracking architecture, temporal mapping and temporal prediction are proposed based on the high temporal resolution characteristics of the ultra-low delay system. Temporal mapping takes advantage of the small difference between successive frames of a sequence taken at the high temporal resolution, simplifying and reducing the amount of processing in the spatial domain by mapping a large amount of processing performed in the spatial domain to the temporal domain. Temporal prediction takes advantage of the detailed moving information captured at the high temporal resolution, compensating for changes due to movement by predicting changes in the temporal domain using a prediction model. Combining these two methods, the problem of spatial multi-search processing is solved by mapping the spatial search into the temporal domain based on a temporal prediction-based mapping rule, the problem of dependent sequential processing is solved by breaking the data dependency using the previous frame's data and temporal changes predicted by temporal prediction, the problem of spatial iterative processing is solved by mapping the iterative processing into the temporal domain, and the tracking error caused by delay is solved by predicting the motion change using temporal prediction.

The dissertation is organized as follows.

In Chapter 1, the background of the dissertation is described, including the widespread applications of visual feedback systems in modern life, the importance of object tracking in real-time applications, the importance of the ultra-low delay in the visual feedback system, the problems, and the concept of proposals. Furthermore, the target and organization of this dissertation are shown.

In Chapter 2, aiming at an ultra-low delay detection, temporal template prediction-based keypoint matching is proposed. Keypoint matching-based detection, which consists of feature detection and feature matching, is one of the popular detection algorithms. Handling scale change is one of the most challenging tasks for keypoint matching-based detection to be implemented into FPGA with ultra-low delay. The conventional method [IEICE 2013] proposed the multi-templates method, which handles size change by preparing multiple templates with various sizes for feature matching and searches for the template with the best matches. With a large number of spatial processing performed in the spatial domain, lots of resources are needed for FPGA implementation. Given that the scale change occurs continuously in the visual system with the high temporal resolution, it becomes possible to predict the size of the template for the next frame using the current existing matching results. The proposed method maps the feature-matching processing for various sizes of templates into the temporal domain based on the temporal template prediction rule, significantly reducing the complexity that exists in the feature matching. The proposed method is implemented as a practical system by integrating a high-speed camera (BASLER acA2000-340km) and an

FPGA (Xilinx XCZU7EV4). Hardware evaluation shows that the proposed method is capable of handling a wide range of scale changes (11 templates) at a resource cost of less than 80 % and achieves three times resource reduction in the matching module when compared with the conventional method. Additionally, it also shows that the designed detection system is capable of sensing and processing 1000 fps sequence (Resolution: 640×360) with a delay of 0.97 ms/frame.

In Chapter 3, to achieve ultra-low delay tracking with high real-time accuracy, temporal prediction-based parallel motion estimation and temporal iterative tracking are proposed. Differential-based tracking estimates the motion between two image patches based on derivatives of image intensities. Lucas Kanade method [CVPR, 1994] [CVIM 2003] is one of the representative differential methods. Methods that Accelerate LK with FPGA, such as [IJASS 2019], have been presented. However, they rarely achieve high accuracy and low delay tracking for rotated objects. Estimating both translation and rotation changes is crucial in practical applications. The conventional method [CVIM 2003] estimates the translation first, and rotation is estimated using the image patch that has been warped in accordance with the estimated translation. The data dependency-based sequential processing introduces intermediate processing, which is the image patch updating, leading to significant delays in addition to resource expenditures. The proposed method breaks the data dependency between translation and rotation estimations by using the translation predicted based on previously estimated translations for rotation estimation, allowing the translation and rotation estimations performed in a

parallel way. The independent processing style of rotation and translation estimations avoids the aforementioned problems caused by intermediate processing. High accuracy over the subpixel level is usually required for precise location. The conventional method [CVPR, 1994] applies Newton-Raphson iteration to direct a more accurate motion estimation. However, the large number of iterative processing performed on the spatial domain results in significant resource costs in addition to delay. On the other hand, the object moves along with being affected by disturbance while the tracking processing is running, making the tracking error depend not only on the image processing itself but also on the delay. The proposed method maps the spatial iterative motion estimation into the temporal domain to avoid a large number of spatial processing and predicts the motion that occurs during the tracking processing ongoing to reduce the tracking error caused by delay. The proposed methods are implemented as a practical system by integrating a high-speed camera (BASLER acA2000-340km) and an FPGA (Xilinx ZCZU9EG). Algorithm evaluation shows that the proposed method achieves subpixel-level real-time accuracy while the conventional method fails to track the target in case of tracking the object moving at a speed of 1 pixel/ms. Hardware evaluation shows that the designed tracking system supports sensing and processing 1000 fps sequence (Resolution: 640×360) with a delay of 0.94 ms/frame while costing resources less than 30 %.

In Chapter 4, the overall dissertation is summarized, and the future works are described. To realize ultra-low delay detection and tracking, temporal mapping and temporal prediction-based algorithms and architectures are pro-

posed. The proposed methods are implemented as a practical system by integrating a high-speed camera and an FPGA. Tracking and detection systems that have been proposed both achieve delays of less than 1 ms/frame when processing a 1000 fps sequence with a resolution of 640×360.

Contents

List of Figures

LIST OF FIGURES

List of Tables

Chapter 1

Introduction

1.1 Visual feedback systems

Recently, there has been a surge of interest in many fields for a visual feedback system that integrates real and virtual (computing) spaces through image information. As shown in Fig. 1.1, the visual feedback system senses the real space and feedbacks valuable information to real space after visual processing, playing a significant role in fields such as factory automation (FA), robotics, and entertainment. In the field of FA, it helps industrial robots bring components to the correct position, improving industrial productivity. In the field of robotics, it aids daily-life robots in picking up falling objects, supporting humans better. In the field of entertainment, it helps projectors map interesting content, providing a splendid show. Therefore, the visual feedback system is key to creating a smarter life in the future.

Since the camera serves as the machine's eye and the image sequence it records provides a wealth of information, various visual processing technologies have been developed. The visual processing technology required for a visual feedback system varies depending on the application's needs. Object tracking is required to locate the 2D position

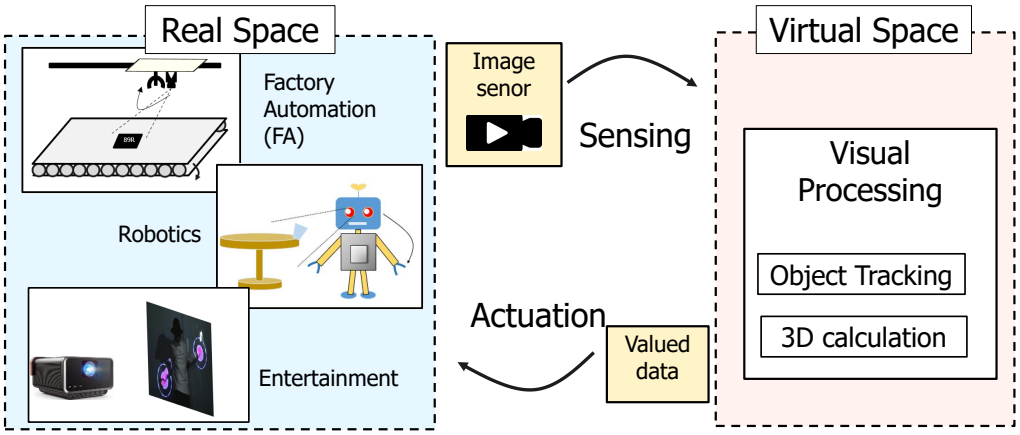

Figure 1.1: Visual feedback system.

of the target. The 3D calculation is required to calculate the distance between the camera and the target. To estimate the camera's location in an environment while creating the environment map, visual simultaneous localization and mapping (SLAM) are necessary.

1.2 Object tracking

Object tracking, which contains detection and tracking, is required in many visual feedback applications. Fig. 1.2 shows three scenes that require object tracking. The projector must recognize the hands and follow them throughout the projection period in the projection mapping scene where projectors project computer graphics onto the dancer's hands. When picking up components with an industrial robot arm, the robot must first identify the component and then track it throughout the picking-up process. The daily-life robot must first identify the falling cup before tracking it throughout the picking-up process in the scene of falling cup picking. As a result, object tracking is an important technology in visual feedback applications.

Various detection and tracking methods have been presented, depending on the type of

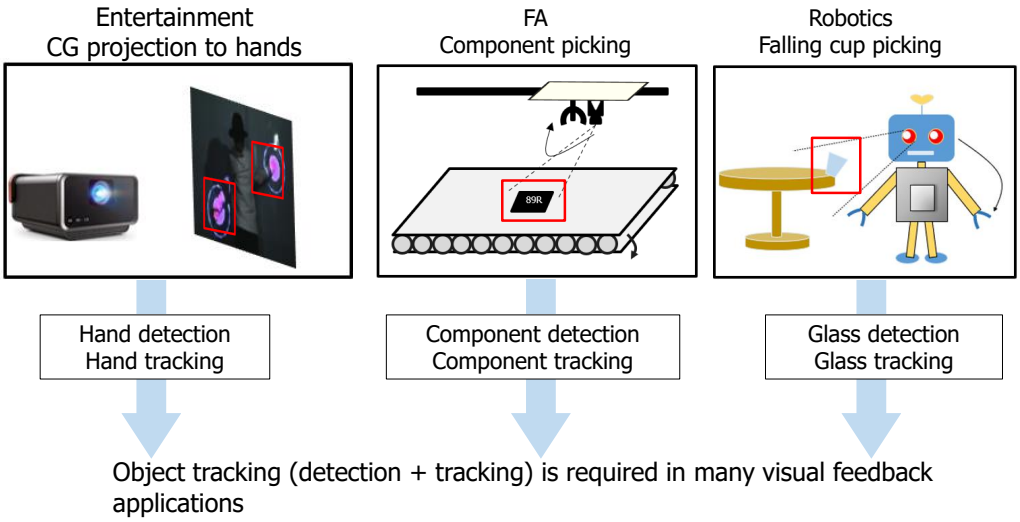

Figure 1.2: Object tracking in visual feedback applications.

target and applications. Fig. 1.3 shows several categories of detection and tracking methods. Detection methods are divided into categories based on whether the target can be defined by a rule or not. Rule-based methods, such as keypoint matching-based detection [1] and shape detection-based method [2], are used for the target that can be defined with rules. Keypoint matching-based methods aim at targets with texture features, while shape detection-based methods aim for targets with distinct contours. It's noteworthy that the keypoint matching-based method contains keypoint detection and feature matching where feature matching differs according to the movement of the target is rigid or deformable. For targets that are challenging to specify with rules, learning-based techniques such as CNN-based detection [3] are used. Tracking methods are divided into categories according to the application requirements. Some of them, such as differential-based iterative tracking [4], focus on achieving high accuracy over the subpixel level, whereas others, such as probability distribution-based tracking [5], focus on tackling complex situations. In the fields of FA and robotics, detecting texture object with rigid movement and tracking

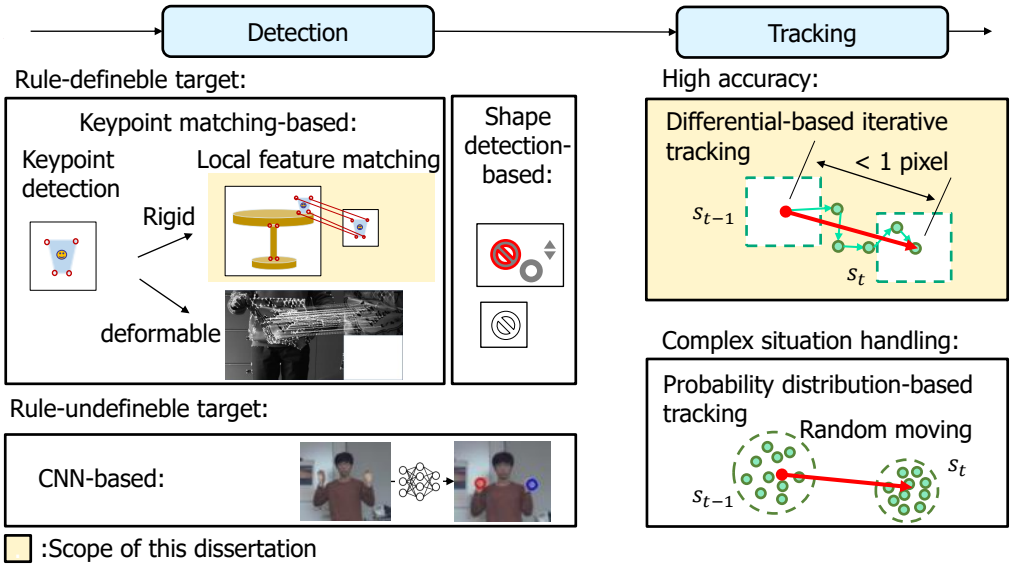

Figure 1.3: Technologies of object tracking.

it with high accuracy is one of the key issues.

1.3 Importance of ultra-low delay

High integration of real and virtual (computing) spaces enables seamless actuation of actuators, which also means that the actuators don't need to stop actuating during the processing period. The actuators' seamless actuation enables tasks that need an immediate reaction, bringing better support for people by completing challenging tasks such as picking up a falling glass, but also improving the productivity of the factory by effectively completing industrial tasks such as component picking. The magnitude of the delay from sensing to feedback determines how well real and virtual spaces can be integrated. Fig. 1.4 illustrates the significance of delay in the scene when the daily-life robot attempts to complete the difficult task of picking up the falling cup. It has been noted that changes in real space persist as visual processing is running in virtual space. When the processing

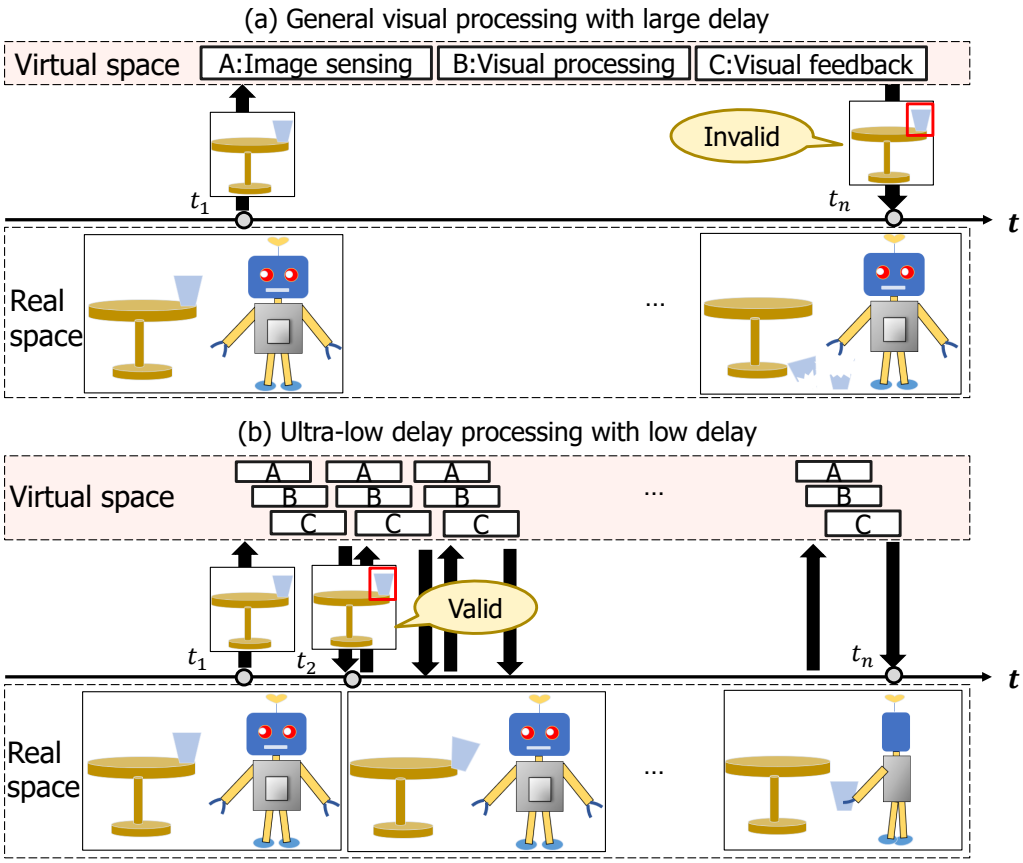

Figure 1.4: Importance of ultra-low delay in real-time scene.

in virtual space is performed in a large-delay style, a significant change may occur in real space during the processing period. With invalid information being fed back to the robot at a low frequency, it's difficult for the robot to actuate properly to handle changing situations in real space. In scenes where the change in real space is controlled by the actuator, the large-delay style is applied by stopping the actuator's action to prevent the change from occurring during the processing period. This, however, leads to a decrease in productivity. When the processing in virtual space is performed in the low-delay style, only minor changes occur in real space during the processing period, suppressing the validity of the virtual space's output from decreasing. With valid information being fed back to

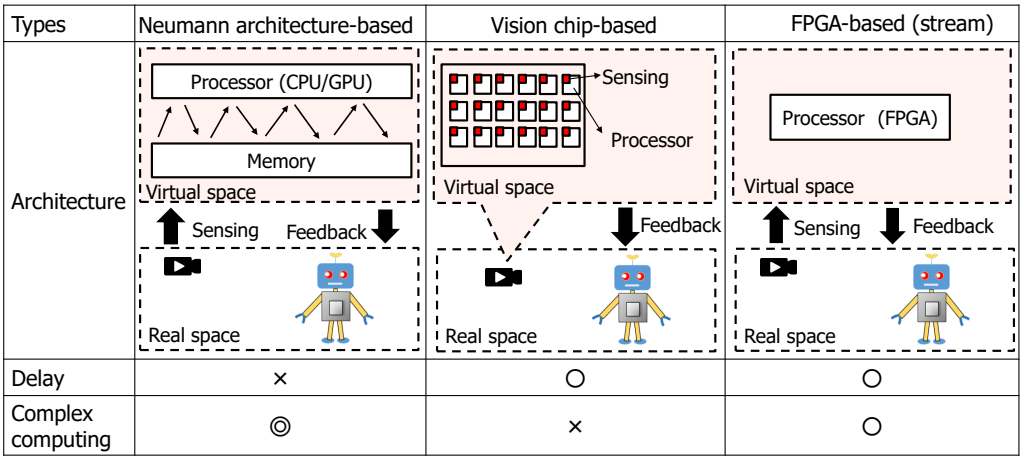

Figure 1.5: Visual processing architectures.

the robot at high frequency, the robot actuates seamlessly to tackle the changing situation in real space. Therefore, the ultra-low delay visual system, which realizes real-time computing, is one of the critical technologies for integrating real and virtual spaces.

In the real-time visual system, there's a balance between the temporal resolution and the spatial resolution due to the limitation of the camera. Here, the temporal resolution is known as the camera frame rate, while the spatial resolution is known as the camera resolution. In general, the temporal resolution decreases with the increase of spatial resolution. It's possible to get higher temporal resolution with smaller spatial resolution. Considering the balance between the temporal resolution and the spatial resolution, this dissertation takes 1 ms, which is the interval of two successive images in a sequence with a temporal resolution of 1000 fps, as a representative of ultra-low delay.

The existing visual processing architectures can be clustered into the general-purpose type and specific-purpose type. The general-purpose type is with Neumann architecture, implemented on processors such as CPU, and GPU. Thanks to the memory-based processing style of Neuman architecture, it is widely compatible with various I/O devices

and capable of implementing many complex visual algorithms, allowing it to realize a variety of visual systems with visual processing times ranging from 10 to 100 ms/frame [6, 7, 8]. Recently, with the development of GPU, some research [9] shows that it's possible to achieve visual processing time below 10 ms/frame when implementing algorithms with appropriate complexity. However, besides the visual processing time, a delay over two frames exists in the sensing and feedback procedure due to the memory-based connection between I/O and processor, which is also known as the I/O bottleneck. As a result, the general-purpose type doesn't satisfy our requirement due to the large delay. The specific-purpose type is implemented with specific architecture on processors such as FPGA, and other VLSI devices. Different from the general-purpose type, which has a fixed hardware architecture, the specific-purpose type has a relatively flexible architecture that the developer can control, making I/O bottleneck possible to be solved, which also means that the ultra-low delay visual system gets possible to be realized. So far, two types of ultra-low delay visual system, which targets sensing and processing image within milliseconds/frame level, have been presented. Vision chip-based low delay system, which integrates the photosensor and processing element, was presented by Ishikawa et.al [10]. Its special architecture solves the I/O bottleneck issue. Since then, various visual systems [11, 12] based on vision chips have been developed. M. Ishikawa [13] presents the development of the vision-chip-based low-delay system, including architectures, algorithms, and applications. However, its special architecture also limits the algorithm complexity, making it challenging to apply more practical algorithms with spatial complexity. FPGA-based ultra-low delay visual system solves the I/O bottleneck by a stream-based architecture that performs visual processing while transmitting the image data from the sensor to the processor. Furthermore, when compared to the vision chip, FPGA allows for more

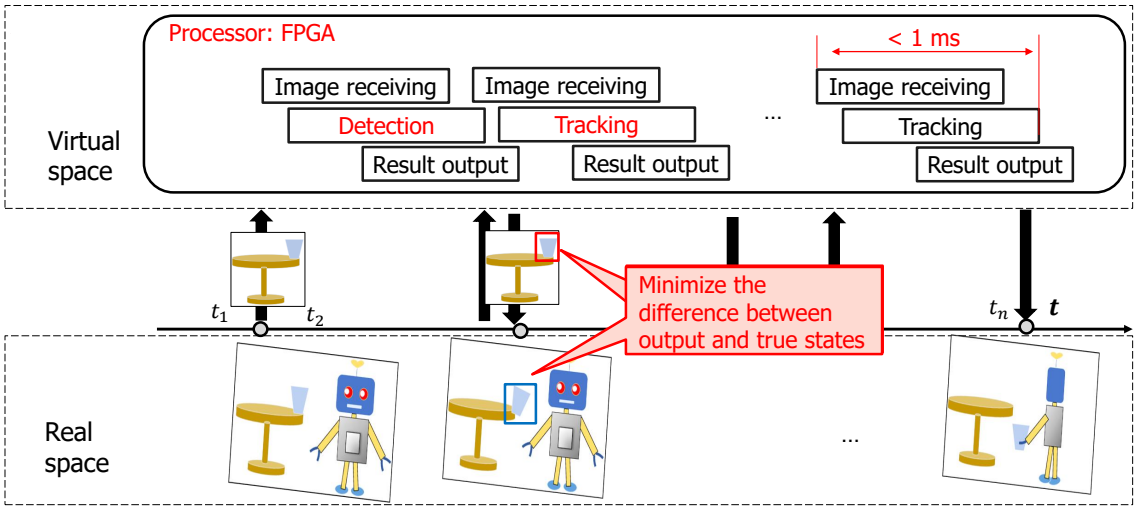

Figure 1.6: Targets of this dissertation.

complex algorithms to be implemented with a more flexible architecture. Because of the low delay and flexible architecture, an increasing number of FPGA-based ultra-low delay visual systems [14, 15, 16, 17, 18, 19, 20, 21] have been presented. The aforementioned three architectures are graphically depicted in Fig. 1.5.

1.4 Targets and problems

The goal of this dissertation is to develop object tracking that can adapt to changing situations as they happen in real space, in other words, minimize the difference between the object tracking's result and the real space's true state at the moment that the object tracking result is outputted from virtual space. FPGA-based 1-ms object tracking architectures and algorithms are essential for achieving it, as graphically shown in Fig. 1.6. Object tracking here includes both detection and tracking. The tracking method accurately tracks the target after the detection method has identified it. Keypoint matching-

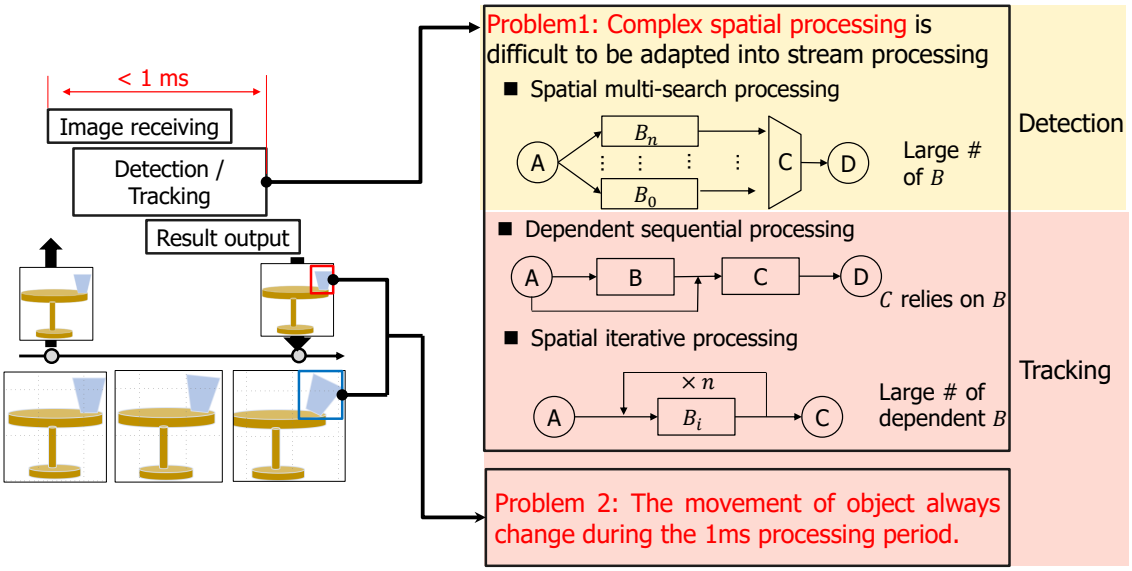

Figure 1.7: Problems in developing object tracking that can adapt changing situations.

based detection and differential-based iterative tracking methods are selected to recognize and track the textured object with high accuracy, which is one of the important tasks in the fields of FA and robotics. The detection and tracking are implemented individually in two FPGA boards as two different systems in the first trial.

Reducing the delay is one of the keys to developing object tracking that can adapt to changing circumstances. By using an FPGA-based stream architecture, it is possible to achieve a delay of less than 1 ms while having a relatively high level of computational complexity. However, simply implementing the detection and tracking algorithms into the FPGA is hard to achieve a delay of less than 1 ms. There is a demand for object-tracking algorithms and architectures that fit the stream-based architecture. Complex spatial processing in the conventional object tracking algorithms is the bottleneck for them to be implemented into stream-based architecture. Here introduces three representative com-

plex spatial processing in the object tracking algorithms.

- Spatial multi-search processing, which performs the same processing for different parameters and selects the result with the most suitable parameter, exists in keypoint matching-based detection. It's graphically illustrated in Fig. 1.7, where $B_0, ..., B_n$ are processing units of B for different parameters, and C is the selection unit that selects the results with the most suitable parameter. $B_0, ..., B_n$ can be processed parallelly in the spatial domain, however, a large number of paralleled heavy processing requires a significant amount of resources when it's implemented in FPGA with a fully pipelined style.

- Dependent sequential processing, which sequentially performs two dependent processing, exists in differential-based tracking. It's graphically illustrated in Fig. 1.7, where processing unit C relies on the processing result of processing unit B. Intermediate processing, which updates the information from A based on the result of B, is introduced due to the dependent relation between B and C. As a result, the intermediate processing results in a significant delay as well as high resource costs.

- Spatial iterative processing, which performs the same processing iteratively with the output of the previous processing as the input, exits in differential-based tracking. It's graphically illustrated in Fig. 1.7, where B_i is the n^{th} iterative processing for B. $B_0, ..., B_n$ are dependent, therefore they can only be processed sequentially. The large number of processing being performed sequentially in the spatial domain results in a significant delay as well as high resource usage.

On the other hand,the object moves along with being affected by disturbance during the period when visual processing is running in the virtual space. The difference between the

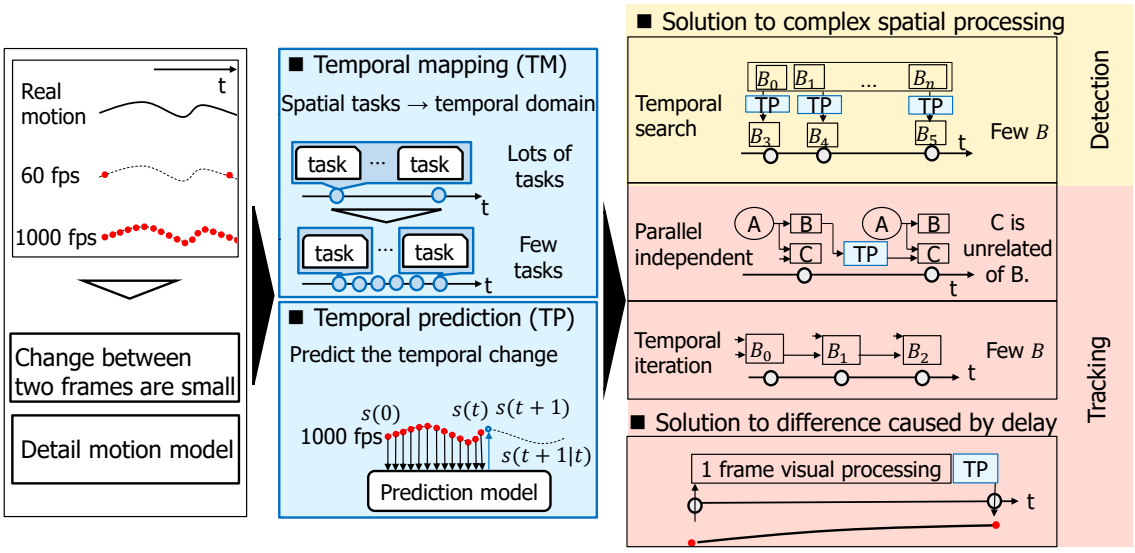

Figure 1.8: Proposed concepts.

output state from virtual space and the true state in the real space caused by delay remains, albeit greatly reduced by ultra-low delay processing, as shown in Fig. 1.7. The difference introduces the tracking errors caused by delay, particularly for tracking.

1.5 Proposed concepts

To solve the problems addressed in Section 1.4, this dissertation proposes temporal mapping and temporal prediction based on the characteristics of high temporal resolution in the ultra-low delay visual processing system. Fig. 1.8 gives an overview of the proposed concepts.

Compared with the general visual processing system which has an input of about 60 fps, the ultra-low delay visual processing system has an input with a higher frame rate of over 1000 fps. The high sampling rate brings the following merits. Firstly, the

change between two adjacent frames is quite small compared with the 60 fps sampling rate, allowing the heavy processing in one frame to be sampled into the adjacent frames. Secondly, the detailed motion is possible to be captured, allowing temporal change to be estimated with high accuracy. These advantages offer a fresh perspective for developing algorithms that consider not only the spatial domain but also the temporal domain. In addition, derivative information is possible to be captured with a high sampling rate, allowing the derivative-related information, such as force, acceleration, and so on, to be estimated at high accuracy. We believe this will increase the scope of the ultra-low delay visual processing system's applications.

Based on the aforementioned characteristics of high temporal capturing, this dissertation proposes:

- Temporal mapping, which maps a large number of spatial tasks into the temporal domain. The change between adjacent frames is small under the high framerate capture, so adjacent frames are assumed to contain the same data when the movement is slow. It's possible to map a large number of spatial tasks into the adjacent frames based on the above assumption. As a result, the number of tasks in one frame is significantly reduced.

- Temporal prediction, which predicts temporal change. With detailed motion captured by the high-framerate camera, the temporal motion change is estimated with high accuracy. Here a prediction model, which predicts temporal change based on the previously estimated results, is designed. The predicted change is applied to compensate for changes due to movement.

Combining the temporal prediction and temporal mapping, the problems addressed in Section 1.4 are solved. Regards the problem of spatial multi-search processing, the spatial searches are mapped into the temporal domain based on a temporal prediction-based mapping rule, significantly reducing the number of spatial processing. Regards the problem of dependent sequential processing, the data dependency between B and C is mapped into the temporal domain based on a temporal prediction, breaking the dependency relation between B and C in the spatial domain. Regards the problem of spatial iterative processing, the spatial iterations are mapped into the temporal domain, signicantly reducing the number of spatial processing and delays. Additionally, applying temporal prediction to predict the motion change significantly reduces the tracking error introduced by the movement that occurred during the visual processing period.

1.6 Organization of dissertation

The contents of this dissertation is organized as follows.

Chapter 1 introduces the background of this dissertation, including the introduction of visual feedback system, the importance of object tracking and ultra-low delay, the problems, and the concept of proposals. To achieve ultra-low delay object tracking, temporal template prediction-based keypoint matching for ultra-low delay detection, and temporal prediction-based temporal iterative tracking and parallel motion estimation for ultra-low delay tracking are proposed in Chapter 2, and Chapter 3, respectively. Chapter 2 is based on one paper [14] which describes the keypoint matching-based detection. Chapter 3 is based on one paper [19] which describes the differential-based tracking.

Chapter 2 describes the proposed temporal template prediction-based keypoint match-

ing for ultra-low delay detection. The proposed method maps the processing of spatial search for suitable template into the temporal domain based on a temporal template prediction mapping rule. Since less search processing is handled in the spatial domain, the resource cost is significantly reduced when implemented in an FPGA.

Chapter 3 presents the temporal prediction-based temporal iterative tracking and parallel motion estimation for ultra-low delay tracking. Firstly, the proposed method breaks the data dependency between the translation and rotation estimations by mapping it into the temporal domain based on temporal prediction. Since there's no spatial data dependency, translation and rotation estimations are carried out in parallel, solving the delay and resource costs issue caused by intermediate processing introduced by spatial data dependency. Secondly, the proposed maps the spatial iterative processing into the temporal domain, significantly reducing the resource cost and delay caused by spatial iterative processing. Meanwhile, a temporal prediction, which predicts the translation and rotation changes, is used to compensate for the motion change that occurs during the period of visual processing.

In chapter 4, the whole dissertation is concluded and future works are introduced.

Chapter 2

Temporal template prediction-based keypoint matching for ultra-low delay detection

2.1 Introduction

This chapter focuses on detection. Detection, which identifies and recognizes the object, is one of the essential technologies in the fields of FA and robotics. In the field of FA, it helps industrial robots such as orthogonal robots pick up the correct components by recognizing the target component. In the field of robotics, it aids daily-life robots in picking up the falling cup by identifying the falling cup. The ability to be robust to rotation and scale changes, which broadens the application range, is critical in the above applications.

Detection algorithms have been developed over the years as one of the key technologies in computer vision. Shape detecting-based methods, such as hough transform [2], vote in a four-dimensional accumulator array that includes x, y, rotation, and scale, making it possible to be robust to rotation and scale changes. However, voting in a four-

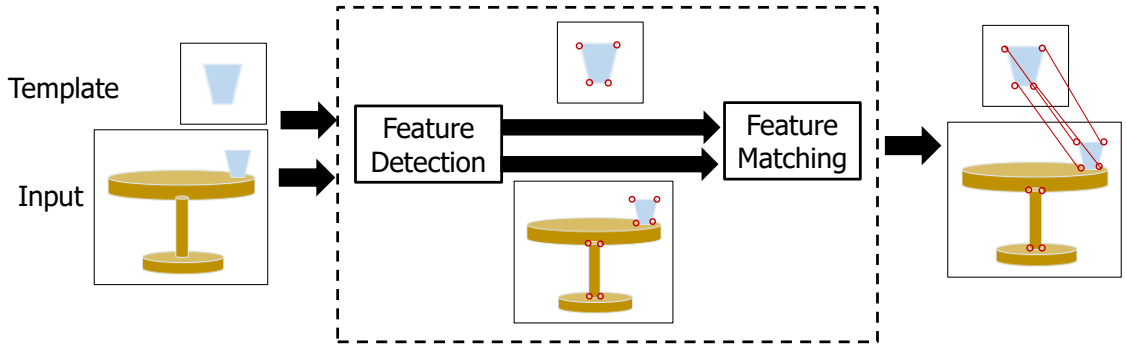

Figure 2.1: Keypoint matching-based detection.

dimensional accumulator array and searching the maximum in the array bring huge computation complexity, making the algorithm too complex to be implemented on a dedicated device. CNN-based methods [22] have a significant advantage in terms of robustness compared with many rule-based methods. However, dealing with complex situations always comes with complex network models, making the algorithm too complex to be implemented on a dedicated device. Meanwhile, the learning process requires lots of data, which are difficult to obtain in the field of FA. Keypoint matching-based methods, such as SIFT [1], SURF [30], and ORB [31], generate rotation-robust features and scale pyramid, making them possible to be robust to rotation and scale changes. As SIFT is not hardware-friendly because of high computation complexity, SURF and ORB are proposed to reduce the computation complexity. In comparison to the algorithms introduced above, the keypoint matching-based method is a good candidate that strikes a good balance between computation complexity and robustness.

Keypoint matching, which extracts the feature from a local region around a keypoint and finds the matches between keypoints with the extracted feature, plays an important role in many computer vision technologies, such as object detection [23], 3D reconstruc-

tion [24], and camera localization [25]. Fig. 2.1 illustrates the process for keypoint matching being applied to template-based detection. Generally, the keypoint matching consists of two parts: 1) feature detection; and 2) feature matching.

- Feature detection, which mainly consists of keypoint detection and keypoint description, detects and describes the feature in a local region. Firstly, intensities in the local region surrounding a pixel are used to determine if a pixel in the image is a keypoint or not. FAST [33] is commonly used for keypoint detection in real-time systems because of its fast computational ability. However, FAST does not provide a good measure of cornerness, which makes it not repeatable enough. Harris corner detector [34], which defines a mathematical form to find the corners according to the nature of the corner, is not fast as FAST, but provides a good measure of cornerness and is more repeatable. The results in Heinly's work [35] also shows that Harris does quite well when coupled with many descriptors. Secondly, intensities in the local region surrounding the keypoint are used to describe the feature of the keypoint. Scale-invariant feature transform (SIFT) [1] and speeded-up robust features (SURF) [30] are commonly used for keypoint description because of their good performance. However, the high computational complexity and the large dimensionality of the generated descriptor make it difficult to implement on embedded systems in real time. In recent years, binary descriptors such as binary robust independent elementary features (BRIEF) [36] and oriented FAST and rotated BRIEF (ORB) [31], can directly build short descriptors with low computational complexity by comparing the intensities of pixel pairs in the surrounding area of the keypoint. Using comparison as the mostly basic operation and with the

high parallel computational ability, BRIEF and ORB are much more suitable for the FPGA implementation.

- Keypoint matching, which determines whether two keypoints are matched according to the feature similarity, finds a set of matches between the query image and the train image. With the development of feature matching, feature matchers such as Brute force-based matcher, FLANN-based matcher, and ANNOY-based matcher have been presented. The brute force matcher, which finds one best match for each query descriptor, is commonly used in the implementation of embedded systems because of its simple structure.

In recent years, several works [40]–[43] that try to accelerate the keypoint matching by FPGA have been presented. So far, no work targets on the high frame rate and ultra-low delay matching system is found. Suzuki's work [40] and Rao's work [41] support processing 60 fps videos within 16 ms/frame. The processing time of Heo's work [43] reaches 18 ms/frame, which is far away from the need in a high frame rate and ultra-low delay system. As a result, the current existing works do not support processing 1000 fps videos within 1 ms/frame.

Directly implementing an algorithm on FPGA is difficult to achieve a delay lower than 1 ms. The algorithm and architecture that adapt the FPGA-based stream processing style are necessary. Complex spatial processing is the main problem preventing the conventional methods from adapting the FPGA-based stream processing style. In order to design a keypoint matching-based detection system, which senses and processes 1000 fps sequences within 1 ms, the following two problems in the conventional methods need to be solved.

- A large number of spatial processing in the robust solution for scale change. In general, in order to achieve scale change robustness, a scale pyramid is made for keypoint detection and keypoint description to be applied. The generation of the scale pyramid, however, is frame-level-dependent processing, resulting in frame-level delay. Meanwhile, the heavy processing in scale pyramid generation makes FPGA implementation challenging. In Suzuki's work [40], instead of making a scale pyramid, three templates of varying sizes are prepared for matching, avoiding the significant delay caused by scale-pyramid generation. When dealing with a wide-scale change, a large number of templates are prepared for feature matching. This process is a spatial search for a suitable template. A large number of spatial processing brings a significant resource consumption when it's implemented on FPGA.

- Complex operations in the rotation-robust feature detection. A keypoint orientation operator is needed in the descriptor description part when considering the rotation robustness of the matching system. Intensity centroid, first presented in Rosin's work [44], is a fast and accurate orientation operator. ORB [31] uses this method to build a rotation-robust descriptor that is capable of real-time performance. However, several operations which are not friendly for FPGA implementation are included in this method, such as multiplication, division, and inverse trigonometric function. The conventional works [41], [42] try to implement the intensity centroid into FPGA in a hardware-friendly way. However, they still cannot be capable of the high frame rate and ultra-low delay performance. Symmetry is also an orientation operator presented in Rosin's work [44], which decides the keypoint orientation

based on the corner symmetry. With high parallel computation ability and without using operations such as division and inverse trigonometric function, symmetry is more suitable for FPGA design. However, the orientation calculated by this method is with a range of $(0°, 180°)$, which is not enough for the improvement of the rotation robustness in the matching system.

This chapter proposes an ultra-low delay keypoint matching-based detection system and its real-time hardware implementation on FPGA. The main contributions of this work are summarized as follows.

- A temporal template prediction-based feature matching method is proposed to reduce the number of spatial processing in spatial template search. The proposed method maps the spatial template search into the temporal domain based on a template-prediction rule.
- An intensity-directed symmetry method is proposed to calculate the keypoint orientation in a hardware-friendly way from the algorithm level. Assumes that the keypoint orientation will maximize the corner symmetry and it will point to a region with high intensity, intensity-directed symmetry uses the operations with less computational complexity and can calculate keypoint orientation with a range of $(0°, 360°)$.
- An FPGA-based system-level architecture, based on one chip-level implementation of the proposed algorithm, is proposed for real-time applications with sensing and processing delays of less than 1 ms/frame. Extensive experiments on both algorithm and hardware have been conducted to thoroughly validate the proposed algorithm and system.

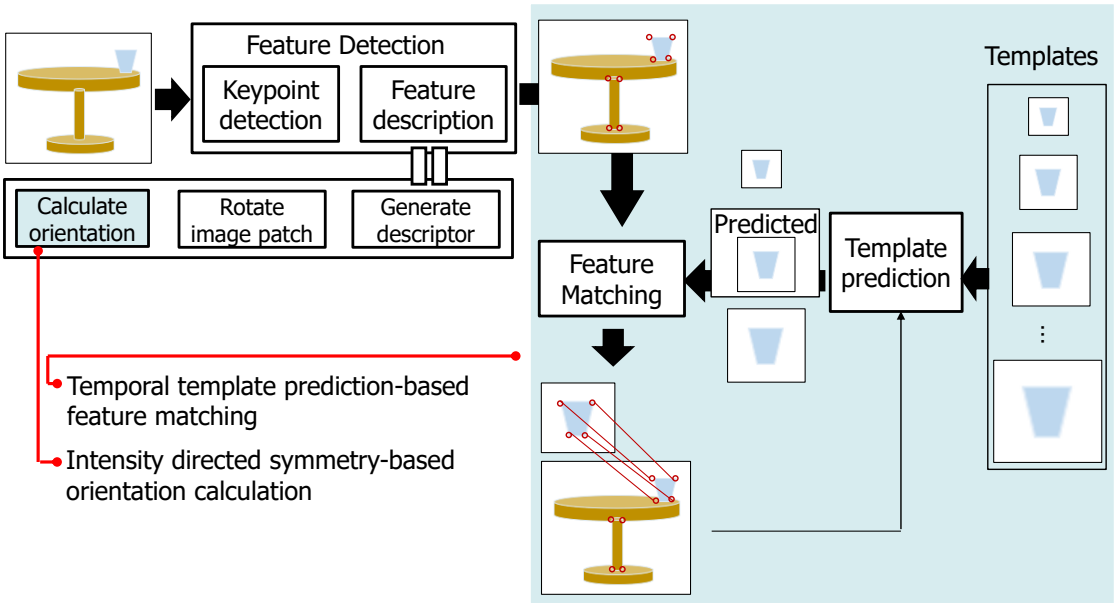

Figure 2.2: Framework of the proposed keypoint matching-based detection.

The remainder of this chapter is organized as follows. Section 2.2 describes the proposed keypoint matching-based detection method. Section 2.3 describes the hardware architecture and system implementation. Section 2.4 presents the algorithm evaluation results and analysis. Section 2.5 presents the hardware evaluation results and analysis, followed by the conclusion in Section 2.6.

2.2 Proposed keypoint matching-based detection method

2.2.1 Framework of the proposed keypoint matching-based detection

The framework of the whole detection method is shown in Fig. 2.2. In the keypoint detection part, Harris [34] is used to detect the corner as the keypoint. And the local

maximum neighboring check presented in work [45] is applied to reduce the large delay in Harris [34] caused by the global sorting. In the feature description part, the rotation-robust binary descriptor of each keypoint is generated based on ORB [31]. And there are mainly three important steps in the feature description part for generating the rotation-robust descriptor. First, the orientation of the keypoint is calculated. Next, the patch used to generate the descriptor is rotated according to the orientation. At last, the descriptor generated in the new patch is robust to rotation. In the feature matching part, the matched keypoint for each keypoint in the current frame of a video stream is found from the template based on brute force matching. For each detected keypoint in the current frame, a candidate match with the minimum distance in the object template can be found according to the similarity of the keypoint descriptors, where the similarity can be measured by the hamming distance. The candidate match is considered a match only when the corresponding minimum distance is less than a certain threshold. Regarding the templates, templates with different sizes are prepared to deal with size changes. Template predicted through template prediction and two of its derivate templates are used for feature matching.

The positions of the proposed two methods are also shown in Fig. 2.2. Temporal template prediction-based feature matching is proposed to deal with the size-change problem with a reasonable number of spatial procesing. Intensity-directed symmetry is proposed to calculate the orientation of the keypoint in a hardware-friendly way from the algorithm level. These two proposals will be introduced in detail in this section.

2.2.2 Temporal template prediction-based feature matching

Enabling the keypoint matching-based detection system to be robust to scale change is a challenging task. In the software algorithm, a scale pyramid is generally used to

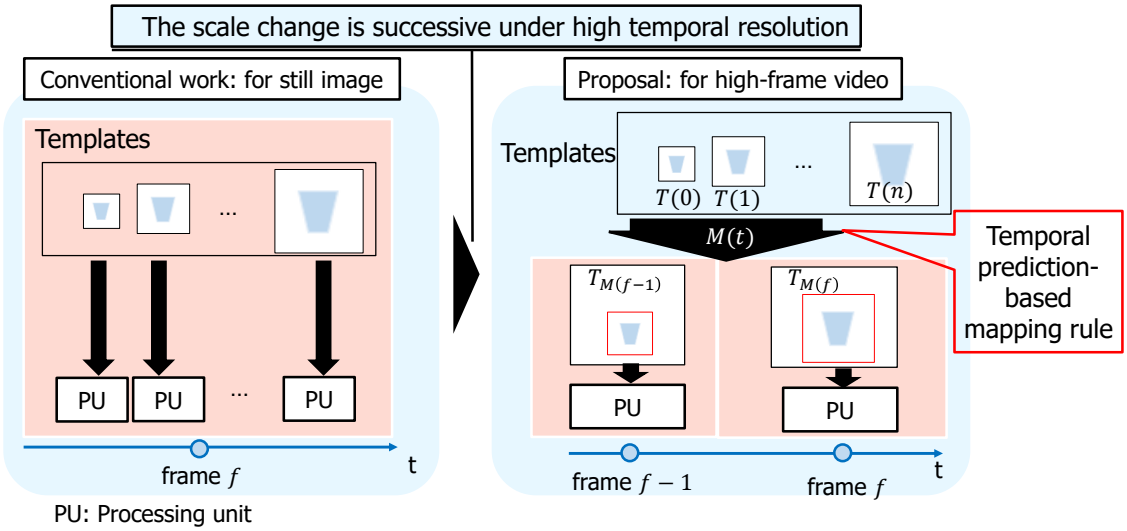

Figure 2.3: Conceptual difference in feature matching.

make scale change robust descriptor by applying feature detection and description on it. However, the complex procedure makes it difficult to adapt the FPGA-based stream architecture. In order to handle large-scale changes in a hardware-friendly way, temporal template prediction-based feature matching is proposed. This method makes use of the advantages of high-framerate capture.

Fig. 2.3 shows the conceptual difference between the conventional multi-template input-based method (MT) [40] and the proposed temporal template prediction-based method. Multi-template input, which achieves robustness of scale change by performing feature matching with multiple templates of different sizes, is presented for still image or low-framerate sequences. With lots of processing performed in the spatial domain for template search, huge resources are required to cover a wide range of scale changes. The scale change is found to be successive under high temporal resolution, indicating that the scale change is predictable using the temporal data. A temporal template prediction-based map-

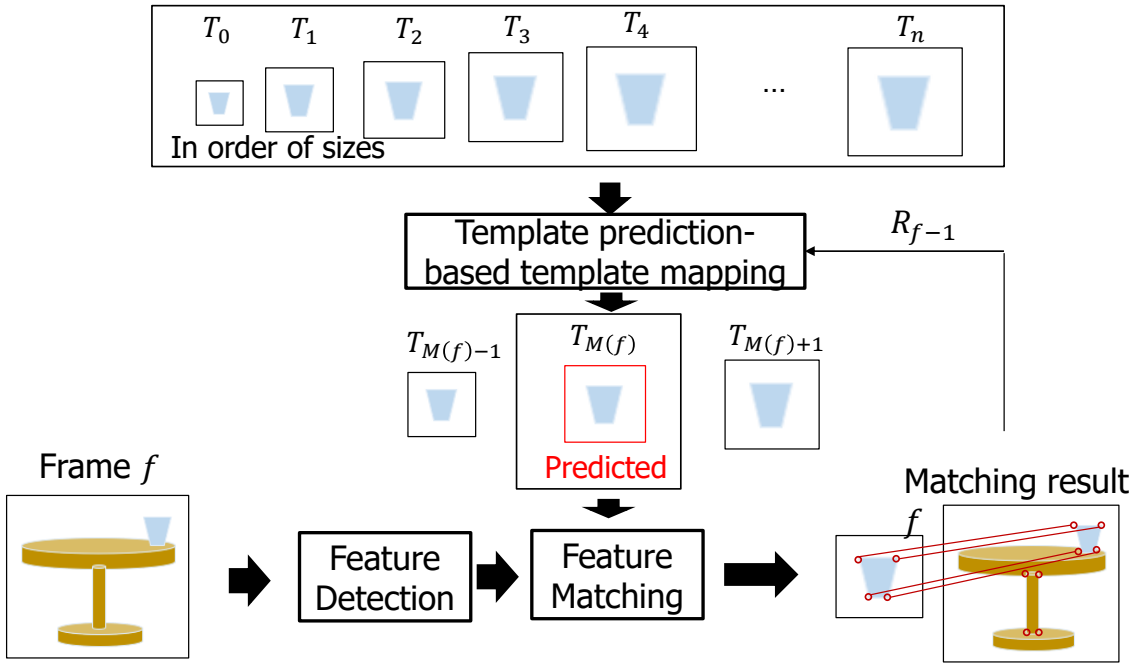

Figure 2.4: Temporal template prediction-based feature matching.

ping rule is made using the aforementioned characteristic, allowing the spatial template processing to be mapped into the temporal domain. With a small amount of processing performed in the spatial domain, the proposed method handles a wide-scale change at a small resource cost.

The detail of temporal template prediction-based feature matching is illustrated in Fig. 2.4. Templates $T_0, ..., T_n$, which have different sizes, are prepared in order of size. Instead of performing feature matching for all templates, the template $T_{M(f)}$ that best fitted the scene in frame f is predicted according to the template prediction-based mapping. To achieve a stable template prediction, $T_{M(f)-1}$ that with a smaller size and $T_{M(f)+1}$ with a larger size are also selected for feature matching. Temporal template prediction for frame f is based on the matching results of frame $f-1$, as shown in Fig. 2.5. The predicted

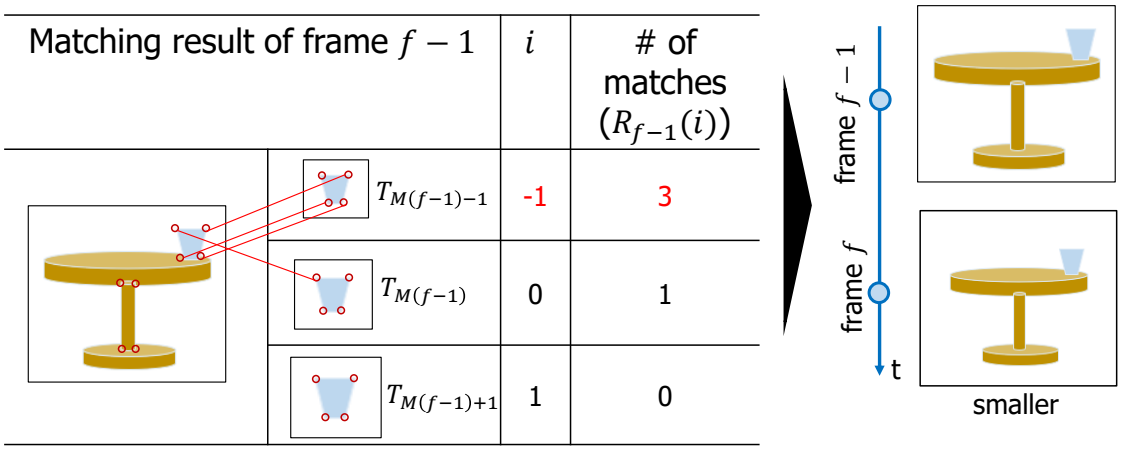

Figure 2.5: Example of temporal template prediction.

template number $M(f)$ of frame i is described as:

$$M(f) = M(f-1) + \arg\max_{i}(R_{f-1}(i)) \tag{2.1}$$

where i is a flag including values of -1, 0, $+1$, and$i = -1$, $i = 0$, $i = -1$ are assigned to template $T_{M(f)-1}$, $T_{M(f)}$, and $T_{M(f)+1}$ respectively. $R_{f-1}(i)$ represents the number of matches between frame $f-1$ and template which corresponding with flag i. For the example shown in Fig. 2.5, template $T_{M(f-1)-1}$ $(i = -1)$ gets the maximum of matches, indicating that the captured object is going to be smaller. Therefore, $M(f) = M(f-1) - 1$, and the template with a smaller size is predicted as the template that is best fitted for frame i.

2.2.3 Intensity-directed symmetry

Fig. 2.6 gives the conceptual comparison between conventional methods and intensity-directed symmetry. Conventional method Intensity centroid defines the vector points from the keypoint to the intensity centroid as the keypoint orientation. Conventional method symmetry assumes that the keypoint orientation maximizes the corner symmetry, and

finds the line which bisects the patch around the keypoint best as the keypoint orientation. The proposed method assumes that the keypoint orientation will bisect the patch best and will point to the region with high intensity. Firstly, the line which bisects the patch is found based on the orientation definition that the keypoint orientation will bisect the patch best. Next, the direction of this line is determined based on the orientation definition that the keypoint orientation will point to the region with high intensity. The orientation consists of the line and its direction is defined as the keypoint orientation.

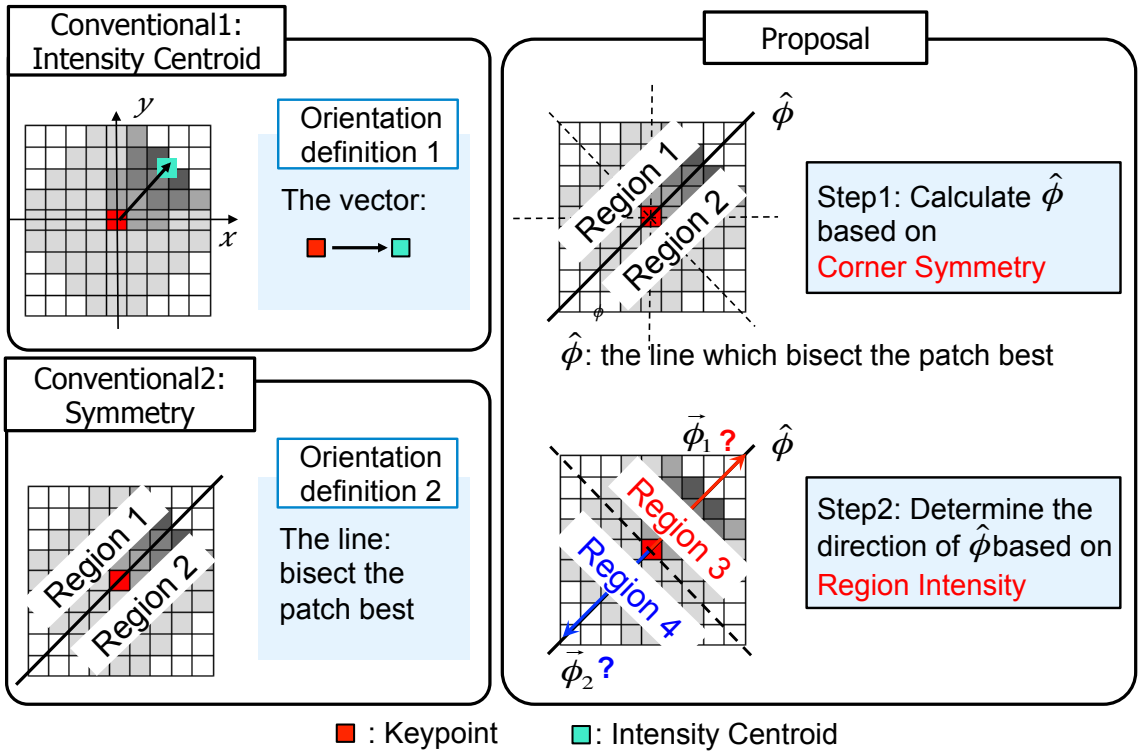

Figure 2.6: Conceptual difference of orientation calculation.

The detailed process of the intensity-directed symmetry is as the following:

Step 1: Calculate the pre-orientation $\hat{\phi}$. The pre-orientation $\hat{\phi}$ is defined as the line that

bisects the region surrounding the keypoint best. In order to find the pre-orientation, the symmetry value SV_{ϕ_i} for each hypothesized orientation ϕ_i is calculated. In the conventional method symmetry [44], Rosin rotates the image window by the hypothesized orientation, and the difference of the intensity sum between the two regions that are separated by the x-axis is used to measure the corner symmetry. The pixel intensity in the rotated patch is obtained by the bilinear interpolation, which is too complex for a high frame rate and ultra-low delay system. Here, instead of rotating the image window and obtaining the pixel intensity by bilinear interpolation, the pixel intensities in two regions (region1 and region2) that are separated by the hypothesized orientation ϕ_i in the patch around the keypoint (corner) are directly used to measure the corner symmetry. The symmetry value of a patch for each hypothesized orientation is defined as:

$$SV_{\phi_i} = |\sum I_{region1_{\phi_i}} - \sum I_{region2_{\phi_i}}| \tag{2.2}$$

where $\sum I_{region1_{\phi_i}}$ is the sum of intensities in region1 when hypothesized orientation is ϕ_i, $\sum I_{region2_{\phi_i}}$ is the sum of intensities in region2 when hypothesized orientation is ϕ_i. With these symmetry values, the pre-orientation $\hat{\phi}$ can be found as which minimizes the symmetry value in all the hypothesized orientations:

$$\hat{\phi} = \min_{\phi_i} SV_{\phi_i} \tag{2.3}$$

It's necessary to mention that 32 hypothesized orientations are prepared to calculate the pre-orientation in the current system. The accuracy of the calculated orientation is proportional to the number of the hypothesized orientations. It's possible to select a different number of hypothesis orientations according to the needs of the target.

Step 2: Determine the direction of the pre-orientation $\hat{\phi}$. After getting the pre-orientation

$\hat{\phi}$, it's needed to make sure which direction the final orientation $\vec{\phi}$ points to along the pre-orientation $\hat{\phi}$. The line which is perpendicular to the pre-orientation $\hat{\phi}$ divides the patch into two regions (region3 and region4). The final orientation $\vec{\phi}$ is determined by:

$$\vec{\phi} = \begin{cases} \vec{\phi_1}, & \sum I_{region3_{\vec{\phi_1}}} > \sum I_{region4_{\vec{\phi_2}}} \\ \vec{\phi_2}, & \sum I_{region3_{\vec{\phi_1}}} \leq \sum I_{region4_{\vec{\phi_2}}} \end{cases} \tag{2.4}$$

where $\sum I_{region3_{\vec{\phi_1}}}$ is the sum of intensities in region3 pointed by direction $\vec{\phi_1}$, $\sum I_{region4_{\vec{\phi_2}}}$ is the sum of intensities in region4 pointed by direction $\vec{\phi_2}$. It's worth mentioning that $\sum I_{region3_{\vec{\phi_1}}}$ and $\sum I_{region4_{\vec{\phi_2}}}$ have been calculated in step 1, which makes the implementation of this step quite simple.

In the implementation of intensity-directed symmetry, firstly, the sum of intensity in region1 $\sum I(x,y)_{region1_{\phi_i}}$ and the sum of intensity in region2 $\sum I(x,y)_{region2_{\phi_i}}$ for each hypothesized orientation ϕ_i are calculated. This process is called the region sum (RS) calculation. In order to balance the processing speed and the resource cost, an asymptotic structure is put forward to calculate the region sum (RS).

Fig. 2.7 shows the conceptual comparison between a general method and the proposed method in the implementation of the region sum calculation part:

Fully parallel region sum calculation: With a high parallel computational capability, it's general to consider a fully parallel structure as shown in Fig. 2.7 (a). The region sum (RS) for each hypothesized orientation ϕ_i is calculated by:

$$RS_{region1_{\phi_i}} = \sum I_{region1_{\phi_i}} \tag{2.5}$$

$RS_{region2_{\phi_i}}$ is calculated in the same way with $RS_{region1_{\phi_i}}$. High process speed can be achieved with this method while the problem of resource cost comes into consideration. Firstly, the resource cost of RS_PU itself is heavy when the patch used to calculate the

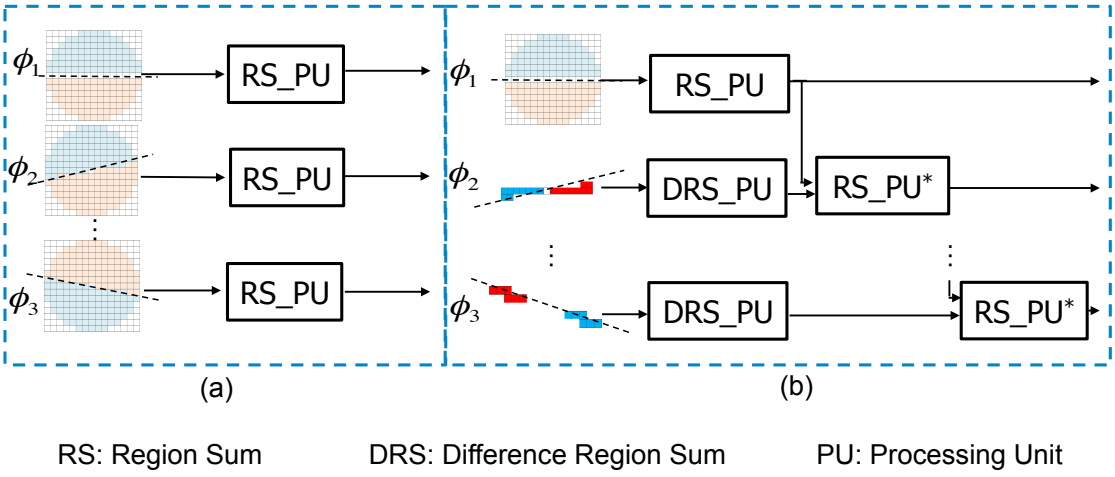

Figure 2.7: Conceptual comparison of the region sum computation. (a) Fully paralleled region sum calculation; (b) Asymptotic region sum calculation.

orientation is large. The resource cost in region sum calculation will become quite large when all the RS_PU are paralleled. Secondly, the resource cost is proportional to the number of the hypothesized orientation, which leads to large increments in resource consumption when the number of hypothesized orientations increases.

Asymptotic region sum calculation: Because of the small change in the number of processing data between two adjacent hypothesis orientations as shown in Fig. 2.8, an asymptotic structure is put forward to calculate the region sum according to:

$$RS_{region1_{\phi_i}} = RS_{region1_{\phi_{i-1}}} + DRS_{region1_{\phi_i}} \tag{2.6}$$

where $DRS_{region1_{\phi_i}}$ is the difference region sum in region1 between ϕ_i and ϕ_{i-1}. For the first hypothesis orientation ϕ_1, $RS_{region1_{\phi_1}}$ is calculated according to Eq. 2.5. $RS_{region2_{\phi_i}}$ is calculated in the same way with $RS_{region1_{\phi_i}}$. With the asymptotic region sum calculation, the region sum can be calculated with less resource utilization. Furthermore, there will be

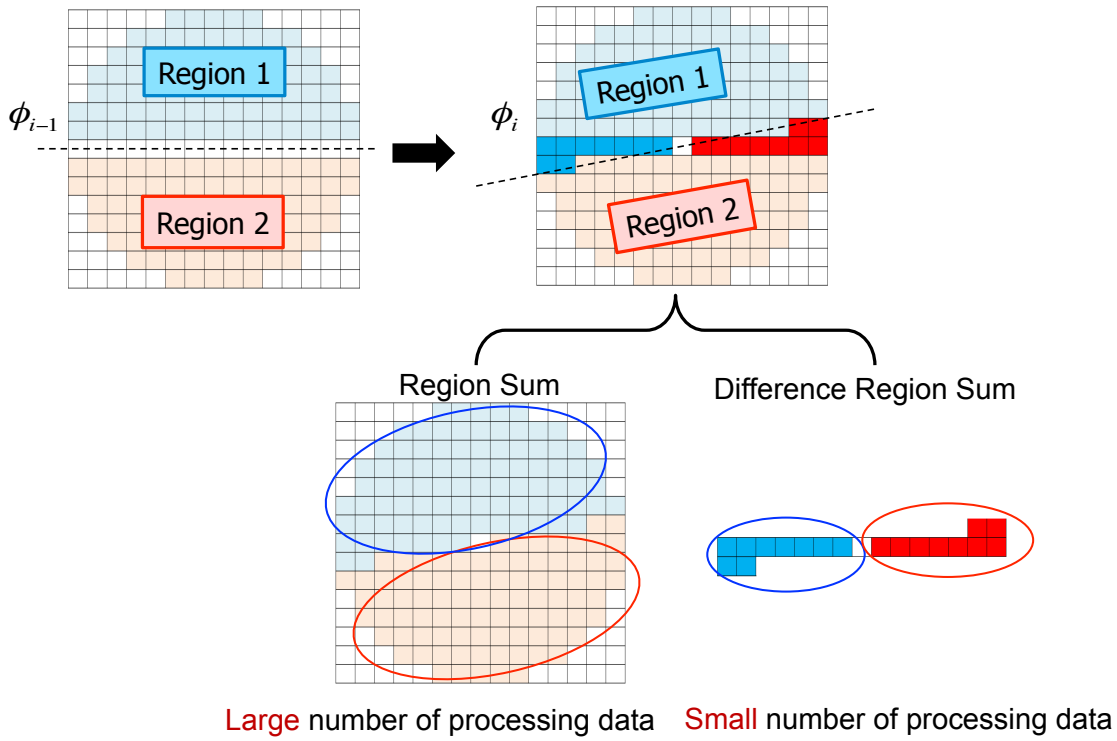

Figure 2.8: The motivation of asymptotic region sum calculation.

a small change in resource consumption when the number of the hypothesis orientations changes.

2.3 Hardware architecture and system implementations

2.3.1 Hardware environment

Fig. 2.9 shows the demonstration environment of the proposed ultra-low delay tracking system. A Camera ⇒ PL (FPGA board) ⇔ PS (FPGA board) system is designed with a high-speed camera and system-on-chip FPGA board, where PS is the processing system with the Debian operation system, and PL is the programmable logic.

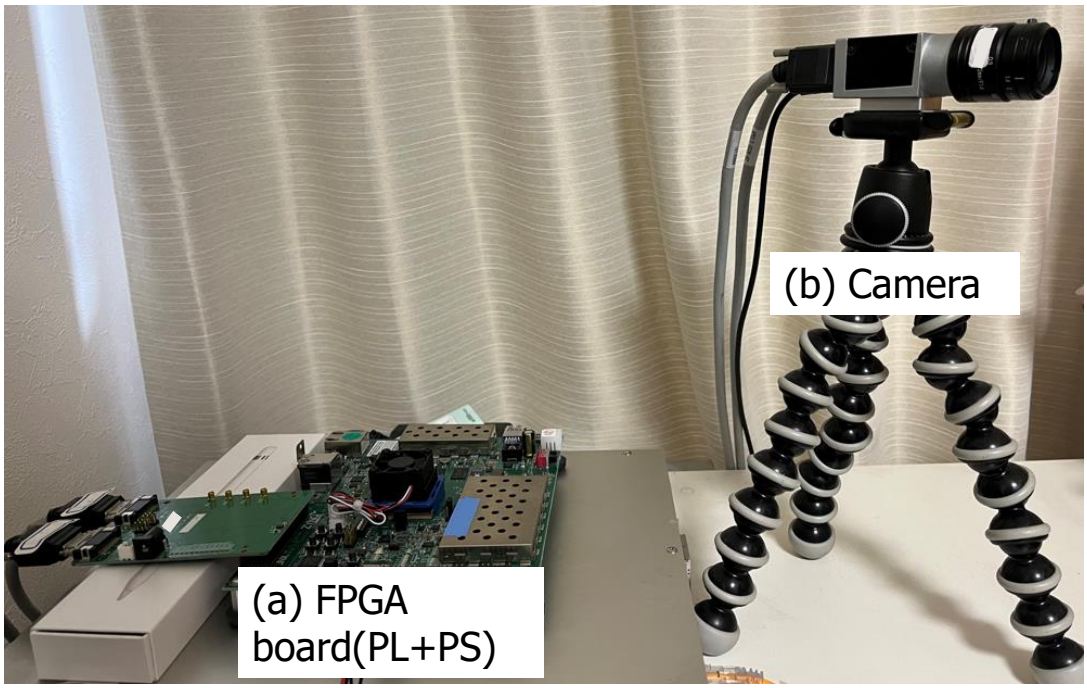

Figure 2.9: Demonstration of the tracking system. (a) Camera: Basler acA2000-340km; (b) FPGA: Zynq UltraScale+ MPSoC ZCU104 (ZU7EV).

2.3.2 Overall hardware architecture

The system-level framework of the proposed detection system is shown in Fig. 2.10. The tracking processing is completed on one chip, where PL serves as the main processing unit of the tracking system, and PS serves as the initializer and postprocessor. The PL receives a pixel stream from a high-speed camera, which is grayscale data fed in 10 pixels at a time. The data is transformed into four paralleled pixels in the camera link receiver, serving as the image process core's input. The primary means by which PS communicates with PL is via a register-access bus. It carries out the two critical data communications listed below. a) Transmit the template data from PS to PL when PS

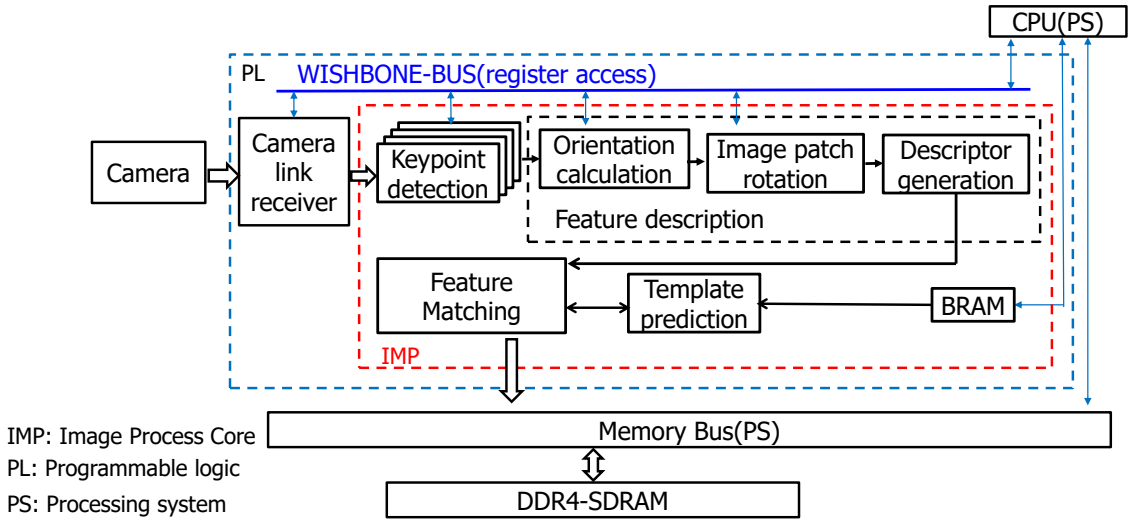

Figure 2.10: System-level framework of the proposed tracking system.

serves as the initializer; b) Transmit the predicted position and angle from PL to PS for further post-processing when PS serves as the postprocessor. Besides, an external memory DDR4-SDRAM, which allows a large amount of data communication between PL and PS, is provided for simulation and post-analysis.

In the image process core, The data goes through keypoint detection, orientation calculation, image patch rotation, descriptor generation, and feature matching modules. Finally, 32-bit data outputs from the image process module. 16-bit data of the output includes image information, feature information, and matching information. The extra 16 bits can store additional information for further use.

2.3.3 Pixel selection-based 4-1-4 thread transformation

In the hardware design, it's challenging to balance the process speed and resource cost with the parallel structure. As shown in Fig. 2.11, if a system processes one data at one

time, the resource cost of the system is small, but the processing speed is slow. If the system processes four data (more than one) at one time, the processing speed becomes four times faster while the resource cost becomes four times larger. However, in the matching system, it's found that the feature description and feature matching are only needed for the pixel which is a keypoint. Therefore, the idea that making sure there is only one keypoint in the simultaneously processed four pixels and selecting the pixel which is a keypoint from these four pixels to do the main process comes into consideration.

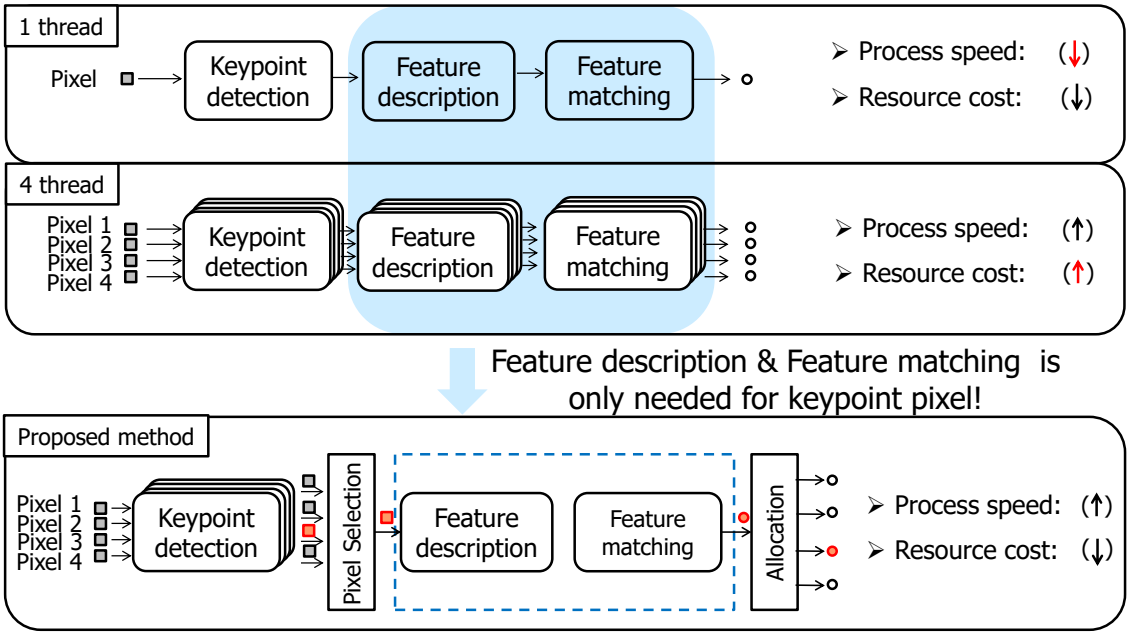

Figure 2.11: Motivation of pixel selection based 4-1-4 thread transformation.

Pixel selection based 4-1-4 structure makes it possible to use one-thread resource consumption to do the four-thread processing. In the proposed 4-1-4 structure, the overall matching system is paralleled into four threads, allowing for the processing of four pixels at the same time. There are mainly three parts in this method: 1) local maximum

neighboring check; 2) pixel selection; and 3) allocation.

1) Local maximum neighboring check: In the keypoint detection part, the Local maximum neighboring check is utilized to reduce the large delay caused by the global sorting in the Harris [34]. Here, it has another function. The block size of the local maximum neighboring check is set as 7×7 to make sure that there is only one keypoint in the simultaneously processed four pixels.

2) Pixel selection: Before feature description and feature matching, each pixel of the four simultaneously processed pixels is distinguished whether it is a keypoint or not according to the flag made for each pixel in the key point detection part, and only the pixel which is a keypoint is selected to get its description in the feature description module and find its match from the template in the feature matching module.

3) Allocation: Following feature description and matching, the corresponding results are allocated to these four simultaneously processed pixels. For the pixel that is a keypoint, the result after the processing of the feature description and feature matching is directly allocated to it. For the pixel that is not a keypoint, the result which represents not matched is allocated to it. Although there is only one pixel processed in the feature description module and feature matching module, the throughput of the system is four pixels/clock.

2.3.4 Register storage-based parallel matching

In the feature matching part, a parallel matching based on the register storage is put forward to accelerate the process of the descriptor matching. The conventional method stores all the descriptors of the template in a RAM. Data stored in RAM can only be accessed one by one, which leads to sequential processing with a large delay when finding

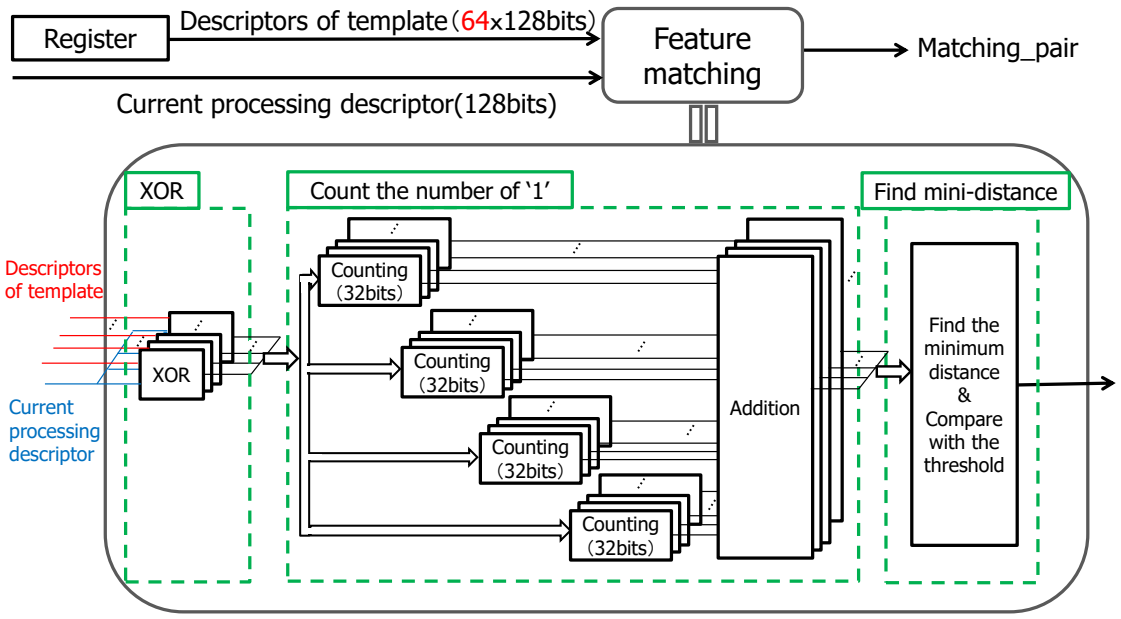

Figure 2.12: Process flow of register storage based parallel matching.

the match from the template. It's reasonable to store the data in the RAM when the data length is large, such as Suzuki's work [40] which implements SIFT (128 bytes for each keypoint) based matching system on FPGA. However, the data length of the binary descriptor is short, and a 16-byte binary descriptor is generated in the high frame rate and ultra-low delay matching system, which makes it possible to store all the descriptor data in a register. Register storage has the property that all the data stored in it can be accessed at one time, which makes it possible to design a parallel process.

Fig. 2.12 shows the process flow of the register storage-based parallel matching. 64 descriptors selected from the template are stored in a template register. There are mainly three parts in the matching process. Firstly, 64 128-bit xor data between the currently processed descriptor and 64 descriptors in the template register can be obtained simul-

taneously through the "XOR" operation. Secondly, the number of '1' for each xor data which represents the hamming distance between the currently processed descriptor and each descriptor in the template is counted. In order to achieve a high maximum frequency, 64 128-bit xor data is divided into 4 parts, and every 64 32-bit xor data are processed simultaneously. At last, the minimum distance can be found after obtaining all the distance between the currently processed descriptor and all the descriptors in the template, and a distance threshold is prepared to eliminate the outmatching pairs. The whole matching process is made fully paralleled and pipelined. In this way, a feature-matching module with high process speed and low delay is designed.

2.4 Algorithm evaluation

Algorithm evaluation is utilized to evaluate the validity of the intensity-directed symmetry in the matching system. The development environment for the software is Visual Studio C++ 2013. Opencv 2.4.11 is utilized to finish the implementation.

2.4.1 Evaluation datasets introduction

In order to evaluate the validity of the intensity-directed symmetry in rotation robustness, two sequences are made by the two images shown in Fig. 3.1. Fig. 3.1 (a) is from the USC-SIPI image database[1], and Fig. 3.1 (b) is from the Oxford dataset [46], [47]. The test sequences are shown in Fig. 3.2. For each test image, eight rotated images can be got from rotating the original test image every 11.25°. These 8 rotated images and the original image together constitute a test sequence. The original image and every rotated

[1] http://sipi.usc.edu/database/

image can constitute an image pair. Each test sequence includes 8 image pairs, and a total of 16 image pairs are used to evaluate the rotation robustness of the proposed algorithm.

(a) (b)

Figure 2.13: Images used for sequence making. (a) Lena: 512×512 pixels; (b) Boat: 800×640 pixels.

Five image pairs are used to test if the proposed algorithm has an influence on other image conditions except for the rotation, such as blur changes, illumination changes, JPEG compression, and viewpoint changes. These image pairs are from the Oxford dataset [46], [47], which is widely used and publicly available.

2.4.2 Matching performance definition

There are two images in each image pair, which are called image (1) and image (2) respectively. Given an image pair, for each keypoint in image (1), a match is succeeded when the nearest neighbor of this keypoint in the image (2) is found. To evaluate the matching performance, ”matching score” is used as a metric, which is recommended in the Oxford benchmark [46], [47]. The ”matching score” is defined as:

$$matching\ score = \frac{\#\ correct\ matches}{\min\ (\#\ kpts^{(1)}, \#\ kpts^{(2)})} \times 100\% \tag{2.7}$$

Figure 2.14: Sequences used for evaluation of rotation robustness.

where "# correct matches" represents the number of correctly matched keypoints between 2 images, "# $kpts^{(1)}$" is the number of keypoints detected in the image (1), # $kpts^{(1)}$ is the number of keypoints detected in the image (2).

$kpts^{(1)}$ = # $kpts^{(2)}$ = 500 is set in this evaluation. A good performance always goes along with a high "matching score".

2.4.3 Experiment results and discussions

In order to check the validity of the proposed keypoint orientation calculation method, the matching performance of the conventional method (intensity centroid [44]) and the proposed method (intensity directed symmetry) are evaluated in the same matching sys-

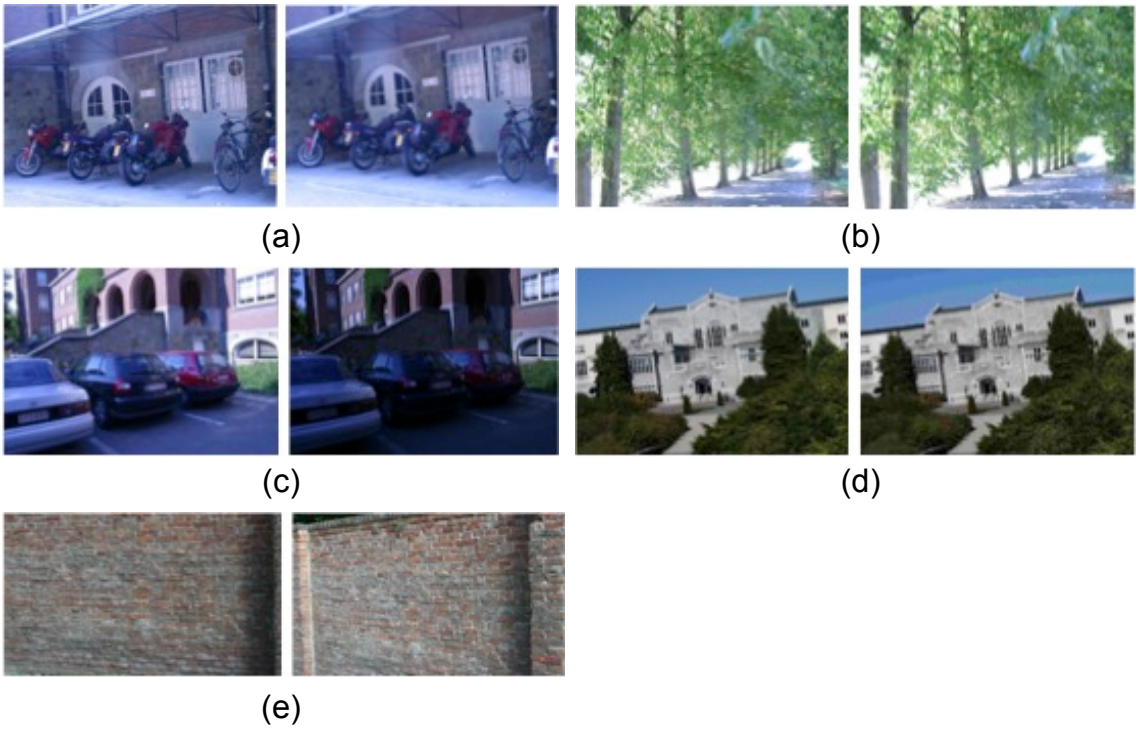

Figure 2.15: Datasets used for evaluation of other image conditions. (a) Bikes: 1000×700 pixels, blur change; (b) Trees: 1000×700 pixels, blur change; (c) Leuven: 921×614 pixels, illumination change; (d) UBC: 800×640 pixels, JPEG compression change; (e) Wall: 1000×700 pixels, viewpoint change.

tem (except the orientation calculation part).

Table 2.1 shows the matching performance in rotation. Column 3 shows the matching performance of the proposed method. The matching scores of the proposed method for all the image pairs are all more than 70 %, which verifies the validity of the proposed method in rotation change. Column 4 shows the difference in the matching score between the conventional method and the proposed method. The difference in the matching score between these two methods is quite small (within 2.2 %), which implies the proposed

Table 2.1: Matching performance in rotation on matching score (%).

Sequence		IC(Conventional)	IDS(Proposed)	Difference
Lena	11.25°	78.8	79.2	+0.4
	22.5°	80.4	79.2	-1.2
	33.75°	78.8	76.8	-2.0
	45°	76.8	75.8	-1.0
	56.25°	75.4	73.8	-1.6
	67.5°	77.8	77.6	-0.2
	78.75°	80.5	79.2	-1.3
	90°	99.6	99.6	0.0
Boat	11.25°	84.8	84.6	-0.2
	22.5°	83.6	82.0	-1.6
	33.75°	82.0	80.4	-1.6
	45°	78.4	77.2	-1.2
	56.25°	79.4	77.2	-2.2
	67.5°	83.2	83.2	0.0
	78.75°	85.8	86.2	+0.4
	90°	100	100	0.0

IC: Intensity centroid

IDS: Intensity directed symmetry

Difference = IDS - IC (‘+’: IDS is better; ‘-’: IC is better)

method can achieve almost the same performance as the state-of-art orientation method - intensity centroid in the case of rotation change.

Table 2.2 shows the matching performance in other image conditions, such as blur change, illumination change, JPEG compression, and viewpoint change. Column 4 shows the difference in the matching performance between the conventional method and the proposed method. The difference in the matching score between these two methods is quite small (within 1.0 %), which implies that the proposed method doesn’t cause a large

Table 2.2: Matching performance in other image conditions on matching score (%).

Image pair	IC(Conventional)	IDS(Proposed)	Difference
bikes	47.0	46.0	-1.0
trees	16.8	16.8	0.0
leuven	56.0	55.4	-0.6
ubc	82.0	82.2	+0.2
wall	20.0	20.0	0.0

IC: Intensity centroid
IDS: Intensity directed symmetry
Difference = IDS - IC ('+': IDS is better; '-': IC is better)

influence in other image conditions, and the proposed method can achieve almost the same performance as the conventional method.

2.5 Hardware evaluation

In the hardware evaluation, ZCU104 (ZU7EV) as FPGA offered by Xilinx, is used for hardware evaluation. The logic synthesis on FPGA is performed by Vivado 2019.1.

Table 2.3: Hardware performance of the image process core.

Item		Image process core
Logic Utilization	# of LUT	167,938(72.89%)
	# of FF	139,486(30.27%)
	# of BRAM	45(14.42%)
	# DSP	36(2.08%)
Performance	Frequency	100MHz
	Process time(1 frame)	0.97(<1ms)

Table 2.3 shows the hardware performance of the designed image process core. Row

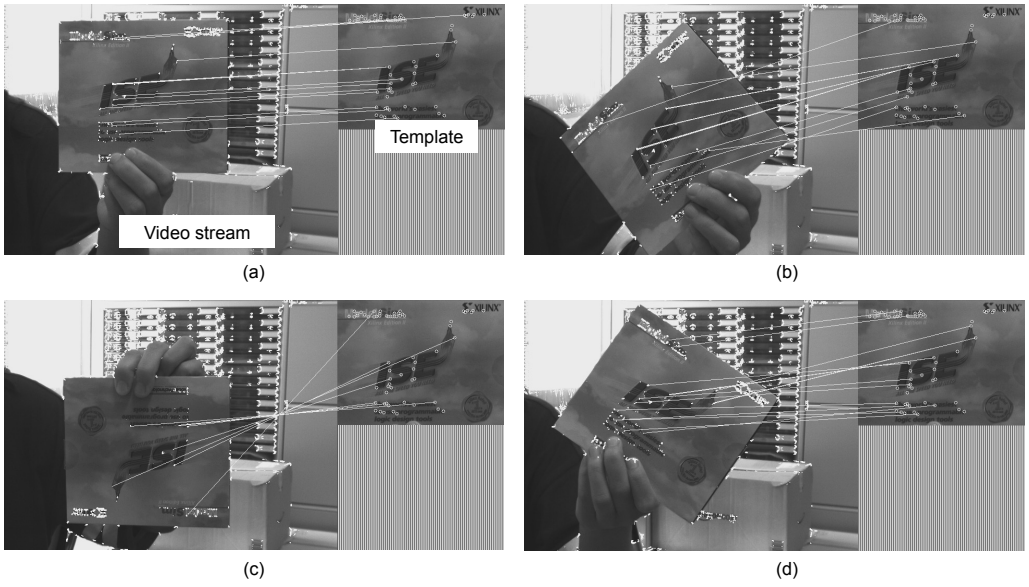

Figure 2.16: Examples of the matching result using the image process core.

2–5 shows the resource utilization of the image process core. Row 6–7 shows the performance of the image process core based on the input clock frequency of 100 MHz. The whole system works at 100 MHz, and the processing time to perform one frame tracking is around A (time to transmit one frame data) + B (4 lines' delay), which is less than 1 ms/frame. Currently, the image process core is connected to the high-speed camera (acA2000-340km) which captures 640 × 360 video at a frame rate of 1000 fps. Fig. 3.4 shows examples of the matching result using the designed image process core. As can be seen that it's successful to match the template object from the video stream in the case of rotation.

The validity of the proposed temporal-template prediction-based feature matching is evaluated by comparing its resource cost in feature matching unit with the conventional method Multi template as shown in Table 2.4. Colum 2 shows the resource cost of the

Table 2.4: Resource utilization of feature matching in case of matching with 11 templates.

Item	Multi template (conventional method)	Proposed method
# of LUT	68,222(29.61%)	45,713(9.92%)
# FF	250,054(108.53%)	167,441(36.34%)

conventional method with templates of different sizes. As can be seen in Table 2.4, the utilization of LUT exceeds the current FPGA board's available LUT. Colum 3 shows the resource cost of the proposed method with 11 templates. The results show that the proposed method allows feature matching with more than 11 templates at a small resource cost.

2.6 Conclusion

In this chapter, an ultra-low delay keypoint matching-based detection method, named temporal template prediction-based keypoint matching, is proposed. Temporal template prediction-based feature matching is proposed to make the keypoint matching robust to a wide range of scale changes with a certain small resource cost. Intensity-directed symmetry is proposed to calculate rotation-robust descriptor, making it possible to calculate keypoint orientation in a hardware-friendly way. The proposed method is implemented as a practical system by integrating a high-speed camera and an FPGA. Hardware evaluation shows that the proposed method is capable of handling a wide range of scale changes (11 templates) at a low resource cost (<80%) and achieves three times resource reduction in the matching module when compared with the conventional method. Additionally, it also shows that the designed detection system is capable of sensing and processing 1000 fps

sequence (Resolution: 640 $\times$ 360) with a delay of 0.97 ms/frame.

Our future research will dive more into the following aspects. In the technical aspect, descriptor pattern which is more compact for FPGA, is added into consideration for decreasing the resource cost, so as to make it possible to combine it with other image processing techniques, such as outline deleting for a more accurate matching result. In the application aspect, we consider applying the detection on the FA sensor and robotics sensor. We believe the combination of high-speed systems and actuators will bring new value to the development of robotics.

Chapter 3

Temporal prediction-based temporal iterative tracking and prarallel motion estimation for ultra-low delay tracking

3.1 Introduction

This chapter focuses on tracking. Tracking, which locates moving objects over time, is one of the most important technologies in the fields of FA and robotics. In the field of FA, it helps industrial robots such as orthogonal robots bring components to the correct position by tracking the marker that indicates the target location. In the field of robotics, it helps daily-life robots pick up a falling object by locating the falling target during the picking process. Tracking accuracy and rotational robustness are critical in the above applications. The ability to be robust to rotation change broadens the application range, and high tracking accuracy allows for precise location. Meanwhile, in the real-time scene introduced above, the sensor or tracking target is still moving during the period when the object tracking is performing in the visual system, making the tracking accuracy dependent not only on the image processing error but also on the error caused by the delay.

Therefore, a rotation-robust tracking system with high real-time accuracy becomes important.

One of the ways to reduce the error caused by delay is to reduce the delay. In recent years, FPGA-based ultra-low delay tracking systems [18, 19, 20] have been developed. The low delay of less than 1 ms/frame significantly reduces delay error. However, none of them succeeds in achieving both rotation robustness and high accuracy at the image processing level. Some of them [19, 20] achieve high image processing level accuracy over subpixel level while they are not robust to rotation change. Besides, it is noteworthy that the delay error still exists, albeit greatly reduced due to the low delay.

Implementing a tracking algorithm that has both rotation robustness and high image processing accuracy in an FPGA-based ultra-low delay system, is the key to suppressing the delay error by reducing the delay. As a result, it becomes crucial to have a rotation-robust tracking algorithm with high image processing accuracy and appropriate computation complexity. Tracking algorithms have been developed over the years as one of the key technologies in computer vision. Area-based methods, including correlation-like methods (such as NCC [48]) and Fourier transform-based methods (such as Fourier phase-only correlation [49]), have a great advantage over subpixel-level motion estimation. However, the motion estimation process is associated with high computational complexity, and they are generally suitable for high-contrast images with translation changes. B. S. Reddy et al. [50] extends the phase-correlation method to rotation and scale-invariant while much more computation complexity is introduced. Probability-based methods, such as particle filter [51], have an advantage in handling complex situations by estimating the post-probability. However, they rarely achieve image processing accuracy over the subpixel level, and computing complexity rises sharply as situation complexity rises. Differential

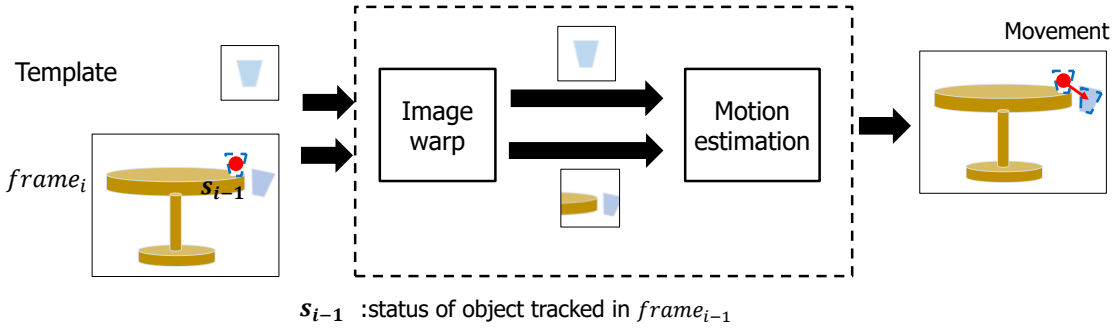

Figure 3.1: LK-based tracking.

methods, such as Lucas Kanade (LK) [4], and Horn and Schunck algorithm [52], estimates the motion between two image patches through the gradient information. The LK algorithm, according to the performance evaluation [53], provides the best accuracy and lowest computational complexity among many differential methods. The original LK is proposed to solve the image registration problem by iteratively estimating translation motion. J. Shi et al. [54] extends the motion model of LK into the affine model, allowing it to estimate affine motion. However, attempting to solve a complex affine motion problem with six parameters to estimate in a small local region yields poor motion estimation. T. Fukao et al. [55] proposed a two-step strategy to attain more accurate estimations for translation, rotation, and scale change in a simple way. Compared with the algorithms introduced above, the LK algorithm is a good candidate that strikes a good balance between tracking accuracy, rotation robustness, and computation complexity.

As a key motion estimation algorithm, the LK algorithm makes significant contributions to real-time vision technologies such as optical flow, feature tracking, object tracking, active vision, and video stabilization. Figure 3.1 illustrates the process for LK being applied to template-based tracking. Generally, LK-based tracking consists of two parts:

1) image warp, which warps the template image or the input frame according to the previously estimated motion; and 2) motion estimation, which estimates the transformation between the two image patches obtained from the image warp.

A number of works [19, 56, 57, 58, 59, 60, 61, 62, 63, 64] that attempt to accelerate LK algorithm using FPGA have been presented. Most of them [56, 57, 59, 61, 62, 63] are far away from ultra-low delay processing. Z. Chai et al. [62] simplified LK-based feature tracking by oversampling video sequences, which supports to process of 182 fps sequence. F. Barranco et al. [58] achieves 270 fps for 640×480 resolution. H. Seong et al. [60] achieves 170 fps for 800×600 resolution. Even though, the delay of these systems is not low enough for our target. As a result, the current existing works do not support reading and processing 1000 fps sequence within 1 ms/frame. Besides, they are all intended for translation estimation, and no work estimating complex motions such as rotation and affine change has been found in our investigation. In addition, the existing works don't take the error caused by delay into consideration.

There are mainly two challenges to designing an LK-based rotation-robust tracking system, which supports sensing and processing 1000 fps sequences within 1 ms and achieves high image processing accuracy. a) A large number of spatial iterative processing. In the LK algorithm, Newton-Raphson iteration is used to direct a more accurate motion estimation. In general, the more iterative processing, the more accurate the motion is estimated. However, the large number of spatial iterative processing not only cause large resource cost but also cause large delay. Therefore, how to balance accuracy and hardware performance becomes an important point in FPGA implementation. [62] limits the number of spatial iterative processing to four, and implements iterative processing by accessing two memory sessions that contain the gradient information of the reference

frame and current frame. The memory access generates the I/O bottleneck, which makes it hard to achieve a lower delay. [18] proposed the local search-based block matching to replace spatial iterative tracking, making the feature tracking able to be implemented in a full stream type. Tracking under 1 ms/frame is achieved by this method, however, the accuracy of this method is limited. b) Sequential and dependent estimation of translation and rotation. T. Fukao et al. [55] proposed a two-step strategy that allows for accurate translation and rotation estimation. The translation is estimated first in the two-step strategy, and rotation is estimated based on the image that has been warped according to the estimated translation in the first step. The image-warping-based sequential structure not only consumes a significant amount of computational resources but also causes a significant delay, making the whole system quite complex.

In summary, since the two-step LK algorithm strikes a good balance between tracking accuracy, rotation robustness, and computation complexity, it is therefore chosen as a baseline for this work. However, as reviewed above, the following problems need to be solved to achieve rotation-robust tracking with high real-time accuracy.

- The delay error still exists although ultra-low delay processing greatly reduces delay error by reducing the delay.
- A large amount of spatial iterative processing consumes a large number of computational resources and introduces a significant delay, making the hardware system quite complex.
- Sequential and dependent two-step estimation for translation and rotation consumes a large number of computational resources and introduces a significant delay, making the hardware system quite complex.

3. TEMPORAL PREDICTION-BASED TEMPORAL ITERATIVE TRACKING AND PRARALLEL MOTION ESTIMATION FOR ULTRA-LOW DELAY TRACKING

The goal of this work is to achieve high real-time accuracy by suppressing both image processing errors and delay errors. In terms of delay error, this work attempts to reduce it not only by reducing the sensing and processing delay to 1 ms/frame but also by reducing the existing delay error caused by 1 ms processing. The main contributions of this article are summarized as follows.

- A temporal prediction-based temporal iterative tracking method is proposed to reduce the delay error caused by 1 ms processing and avoid the iterative processing in the spatial domain.

- A temporal prediction-based parallel motion estimation method is proposed to solve the resource consumption and delay problem caused by sequential and dependent motion estimation.

- An FPGA-based system-level architecture, based on one chip-level implementation of the proposed algorithm, is proposed for real-time applications with sensing and processing delays lower than 1 ms/frame. Extensive experiments on both algorithm and hardware have been conducted to deeply validate the proposed algorithm and system.

The remainder of this chapter is organized as follows. Section 3.2 presents the related works. Section 3.3 describes the proposed LK-based ultra-low delay tracking method. Section 3.4 describes the hardware architecture and system implementation. Section 3.5 presents the algorithm evaluation results and analysis. Section 3.6 presents the hardware evaluation results and analysis, followed by the conclusion in Section 3.7.

3.2 Related work

3.2.1 Two-step LK algorithm

Two-step LK [55] estimates the translation and rotation change in two steps. Firstly, only the translation is estimated in a small area. Secondly, changes other than the translation such as rotation and scale change are compensated in a larger area.

3.2.1.1 Translation estimation

The displacement $\mathbf{d_a}$ between two image patches, which are image $I(\mathbf{x})$ and image $J(\mathbf{x})$, can be estimated by solving the following 2×2 linear system.

$$\mathbf{G_a d_a} = \mathbf{e_a} \tag{3.1}$$

$$\mathbf{x} = [x, y]^T,\ \mathbf{d_a} = [d_x, d_y]^T$$

error vector $\mathbf{e_a}$, which depends on the difference between the two image patches, is written as:

$$\mathbf{e_a} = \begin{bmatrix} E_{a1} \\ E_{a2} \end{bmatrix} = \iint_{W_a} \begin{bmatrix} g_t g_x \\ g_t g_y \end{bmatrix} d\mathbf{x} \tag{3.2}$$

$$g_x = \frac{\partial J}{\partial x},\ g_y = \frac{\partial J}{\partial y},\ g_t = I - J$$

and matrix $\mathbf{G_a}$ can be written as:

$$\mathbf{G_a} = \begin{bmatrix} G_{a11} & G_{a12} \\ G_{a12} & G_{a22} \end{bmatrix} = \iint_{W_a} \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} d\mathbf{x} \tag{3.3}$$

As a result, $\mathbf{d_a}$ can be decribed as:

$$\mathbf{d_a} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} \frac{G_{a12}E_{a2} - G_{a22}E_{a1}}{\delta_a} \\ \frac{G_{a12}E_{a1} - G_{a11}E_{a2}}{\delta_a} \end{bmatrix} \tag{3.4}$$

$$\delta_a = G_{a12}G_{a12} - G_{a11}G_{a22}$$

3.2.1.2 Rotation and scale change estimation

Having obtained the estimated translation motion $\mathbf{d_a}$, image patch in image $J(\mathbf{x})$ is updated to $J(\mathbf{x} + \mathbf{d_a})$. The rotation d_θ and scale change d_s can be estimated by solving the following 2×2 linear system.

$$\mathbf{G_b d_b} = \mathbf{e_b} \tag{3.5}$$

$$\mathbf{d_b} = [d_\theta, d_s]^T$$

error vector $\mathbf{e_b}$ is written as:

$$\mathbf{e_b} = \begin{bmatrix} E_{b1} \\ E_{b2} \end{bmatrix} = \iint_{W_b} \begin{bmatrix} g_t(-g_x y + g_y x) \\ g_t(g_x x + g_y y) \end{bmatrix} d\mathbf{x} \tag{3.6}$$

and matrix $\mathbf{G_b}$ can be written as:

$$\begin{aligned} \mathbf{G_b} &= \begin{bmatrix} G_{b11} & G_{b12} \\ G_{b12} & G_{b22} \end{bmatrix} \\ &= \iint_{W_b} \begin{bmatrix} (-g_x y + g_y x)^2 & (-g_x y + g_y x)(g_x x + g_y y) \\ (-g_x y + g_y x)(g_x x + g_y y) & (g_x x + g_y y)^2 \end{bmatrix} d\mathbf{x} \end{aligned} \tag{3.7}$$

As a result, $\mathbf{d_b}$ can be decribed as:

$$\mathbf{d_b} = \begin{bmatrix} d_\theta \\ d_s \end{bmatrix} = \begin{bmatrix} \frac{G_{b12}E_{b2} - G_{b22}E_{b1}}{\delta_b} \\ \frac{G_{b12}E_{b1} - G_{b11}E_{b2}}{\delta_b} \end{bmatrix} \tag{3.8}$$

$$\delta_b = G_{b12}G_{b12} - G_{b11}G_{b22}$$

and the transformation matrix A can be written as:

$$A = \begin{bmatrix} 1 + d_s & 0 \\ 0 & 1 + d_s \end{bmatrix} \begin{bmatrix} cosd_\theta & -sind_\theta \\ sind_\theta & cosd_\theta \end{bmatrix} \tag{3.9}$$

Having obtained the estimated motion $\mathbf{d_a}$ and $\mathbf{d_b}(A)$, image patch in image $J(\mathbf{x})$ is updated to $J(A\mathbf{x} + \mathbf{d_a})$. A more accurate motion vector $\mathbf{d} = [\mathbf{d_a}, \mathbf{d_b}]^T$ can be found by repeating the above procedure in a Newton-Raphson style.

3.3 Ultra-low delay rotation-robust tracking method

3.3.1 Framework

The framework of the proposed ultra-low delay tracking method is shown in Fig. 3.2. It mainly consists of five phases. In the first phase, an image patch J_w surrounding the current known position in the input image is extracted by image warp 1, and a rotated image patch I_w is extracted in image warp 2 by rotating the template image according to the current known angle. The current known position and angle are prediction results at time $t-1$ ($\mathbf{p}(t|t-1)$, $\theta(t|t-1)$) or the initial values ($\mathbf{p}_{init}$, θ_{init}). Furthermore, rotating the template image rather than the input image patch is better suited for hardware implementation. In the second phase, spatial gradients g_x and g_y are calculated by differentiating the image patch J, and g_t is calculated by differencing image patches I_w and J_w. In the third phase, position $\mathbf{p}(t)$ and angle $\theta(t)$ at time t are estimated by the translation estimation and rotation estimation respectively, where $\mathbf{p}(t) = [x(t), y(t)]^T$. In the fourth phase, the position $\mathbf{p}(t+1|t)$ and angle $\theta(t+1|t)$ at time $t+1$ are predicted based on the previously estimated states. Finally, $\mathbf{p}(t+1|t)$ and $\theta(t+1|t)$ are fed back to next temporal iterative tracking. State prediction not only helps the temporal iterative tracking reduce the existing delay error and improve state estimation accuracy in high-speed cases but also enables translation and rotation estimation to run parallelly without significant accuracy loss. A video demonstration is available at https://wcms.waseda.jp/em/6422257c9c402.

3. TEMPORAL PREDICTION-BASED TEMPORAL ITERATIVE TRACKING AND PRARALLEL MOTION ESTIMATION FOR ULTRA-LOW DELAY TRACKING

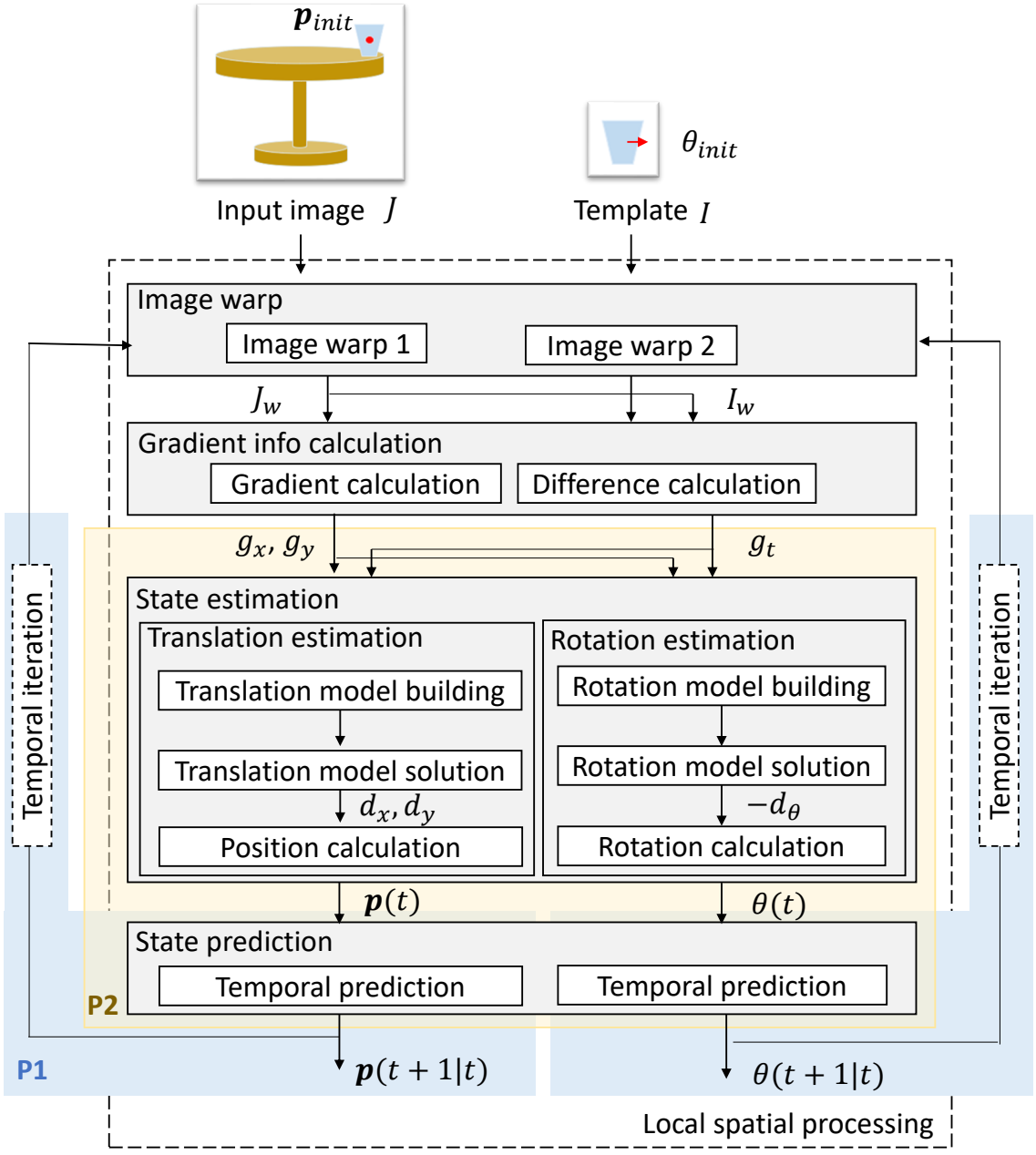

Figure 3.2: Framework of the proposed ultra-low delay tracking method.

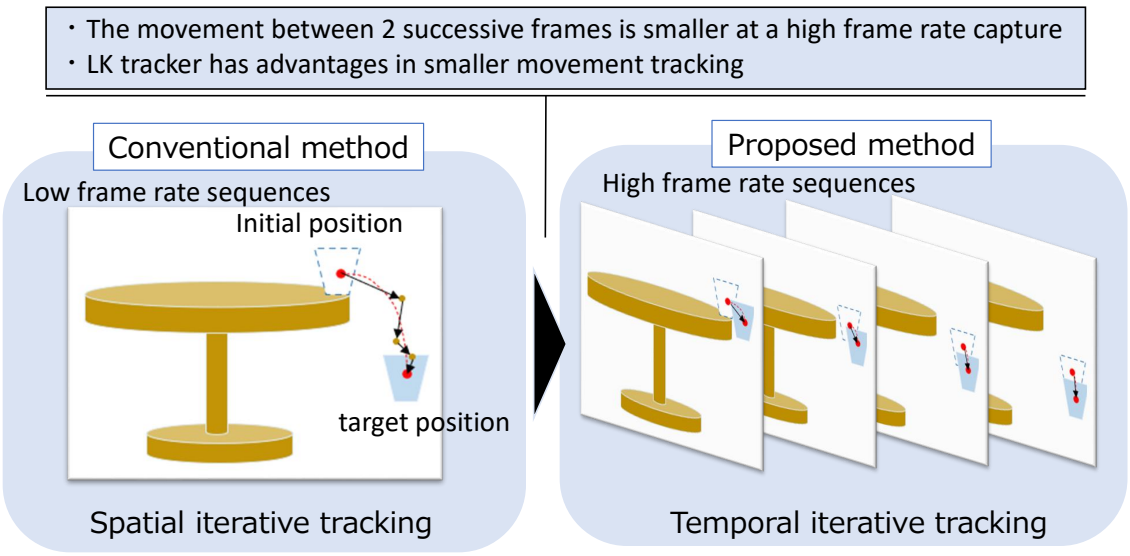

Figure 3.3: Conceptual difference of the iterative tracking.

The positions of proposed methods are also shown in Fig. 3.2. In Sections 3.3.2 and 3.3.3, we describe the detail of the temporal prediction method and how it works in temporal iterative tracking and state estimation.

3.3.2 Temporal prediction-based temporal iterative tracking

3.3.2.1 Temporal iterative tracking

The iterative procedure is an important part of the LK algorithm. It directs the displacement to a more accurate one so as to achieve sub-pixel level accuracy. In general, the iterative processing is performed in the spatial domain. However, this will lead to large resource cost and delay in FPGA implementation. In this paper, high temporal resolution-based temporal iterative tracking, a hardware-friendly algorithm, is proposed to avoid iterative processing in the spatial domain. The proposed method mainly has two advantages: 1) Lower delay is achieved since there's no iterative processing in the spatial

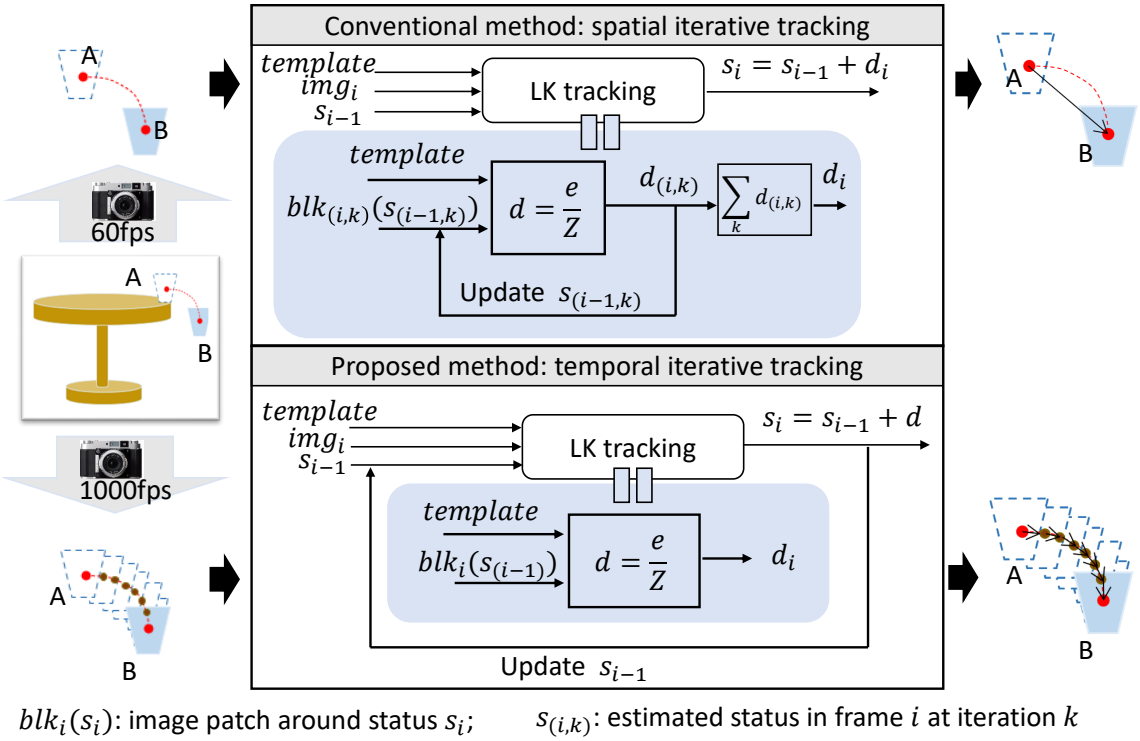

Figure 3.4: Frameworks of iterative tracking.

domain. 2) Dynamic tracking, which tracks faster-moving objects with coarse accuracy and achieves higher accuracy when it slows down, becomes possible. Dynamic tracking will be described in detail in Section 3.5.2.3.

Figure 3.3 shows the conceptual comparison between conventional spatial iterative tracking and the proposed temporal iterative tracking. In the conventional tracking system, the movement between two successive frames is large due to the low frame rate. In order to track the moving object with high accuracy, iterative processing is applied to the updated position in the same image. This is called spatial iterative tracking. According to the experiment result shown in Section 3.5.2.3, we found that the LK tracker with one local spatial processing has accurate tracking capabilities when the movement is small.

Therefore, the idea that splitting the movement into several smaller ones by higher frame rate capture and applying the iterations into the temporal domain, comes into consideration.

In practice, the general real-time vision system works at 60 fps, while the high frame rate and ultra-low delay vision system aims at working over 1000 fps. Compared to the general spatial iterative tracking system working at 60 fps, the high frame rate and ultra-low delay tracking system working at 1000 fps splits the movement into about 17 smaller movements and applies 17 temporal iterative processing into the temporal domain.

The detail of the two kinds of iterative tracking is shown in Figure 3.4. Takes the scene that glass moves from A to B as an example. Given the template image (*template*), input sequence (img_i), and the initial position of A in the first frame of the input sequence ($\mathbf{p_0} = \mathbf{p_A}$), the motion estimation for movement from A to B with different visual tracking systems is shown as follows:

1) General real-time tracking system (spatial iterative tracking): Assumes that the motion from A to B is recorded by two frames when it's captured by a 60 fps camera, the displacement from A to B that is estimated by a spatial iteration, is written as:

$$\mathbf{d_S} = \mathbf{d_i} = \sum_{k=1}^{n} \mathbf{d_{(i,k)}} \tag{3.10}$$

where n is number of spatial iteration; $\mathbf{d_{(i,k)}}$ is the estimated motion vector at the k^{th} iteration in frame i; $\mathbf{d_i}$ is the estimated motion vector in frame i. In this case, $\mathbf{s_1} = \mathbf{s_B}$. The final estimated status is described as:

$$\mathbf{s_B} = \mathbf{s_A} + \mathbf{d_S} \tag{3.11}$$

2) High frame rate and ultra-low delay tracking system (temporal iterative tracking): Assumes that the motion from A to B is recorded by 18 frames when it's captured by a 1000

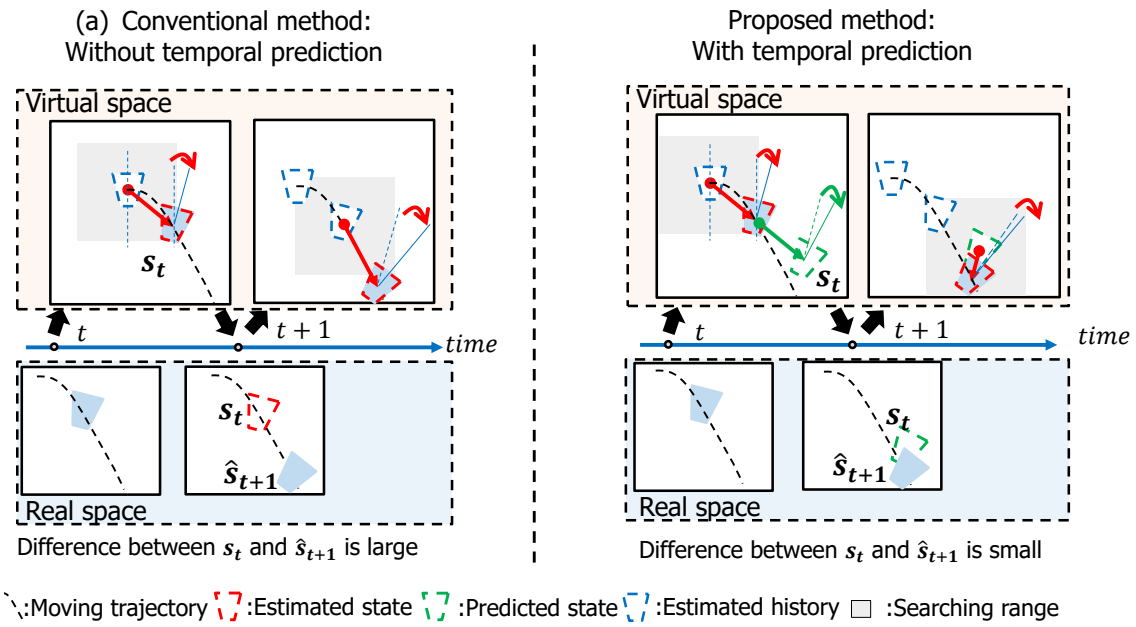

Figure 3.5: Conceptual difference of temporal prediction.

fps camera, the displacement from A to B that is estimated by a temporal iteration, is written as:

$$\mathbf{d_T} = \sum_{i=1}^{m} \mathbf{d_i} = \sum_{i=1}^{m} \mathbf{d_{(i,1)}} \tag{3.12}$$

where m is the number of frames that are applied to the temporal iterative tracking. In this case, $m = 17$, and $\mathbf{s_{17}} = \mathbf{s_B}$. The final estimated status is described as:

$$\mathbf{s_B} = \mathbf{s_A} + \mathbf{d_T} \tag{3.13}$$

Compared with the general real-time tracking system with spatial iterative tracking, the proposed tracking system performs tracking with only one local spatial processing, effectively reducing the delay and resource cost caused by spatial iterative processing.

3.3.2.2 Temporal prediction of target status

Temporal iterative tracking maps the spatial iterative processing into the temporal domain, making FPGA implementation of iterative processing possible. Eq. (3.12) describes a motion tracked by temporal iterative tracking. The outcome of one temporal iteration is described as:

$$\mathbf{s_t} = \mathbf{s_{t-1}} + \mathbf{d_t} \tag{3.14}$$

where $\mathbf{s_t}$ represents the output state vector of tracking for frame captured at time t; $\mathbf{d_t}$ represents the motion vector obtained in tracking for frame captured at time t. In the conventional method [19], $\mathbf{d_t}$ is the estimated motion vector $\mathbf{d}(t)$ calculated according to Eq. (3.4) (3.8), and $\mathbf{s_t}$ is the estimated state $\mathbf{s}(t)$, where $\mathbf{d}(t) = [d_x(t), d_y(t), d_\theta(t)]^T$ and $\mathbf{s}(t) = [x(t), y(t), \theta(t)]^T$. As shown in Fig. 3.5(a), the movement continues in the real space when visual tracking is performed in the virtual space, resulting in a large difference between output state $\mathbf{s_t}$ and the true state $\hat{\mathbf{s}}_{\mathbf{t+1}}$. The difference, which is primarily caused by the tracking system's delay when the tracking system's estimation is accurate, persists as long as the change occurs in real space. The difference, on the other hand, is required to be estimated in the next temporal iteration, and it grows larger in high-speed moving cases, resulting in a poorer estimation of the next temporal iterative tracking. As a result, the tracking system's output and the true state of the object differ significantly due to the delay error and image processing error. To solve this problem, a temporal prediction method is proposed to compensate for temporal change that occurs during the processing period.

A predicted motion vector is introduced to compensate for the temporal change. The predicted motion vector $\mathbf{d}(t+1|t)$ and the estimated motion vector $\mathbf{d}(t)$ combine to form

the motion vector $\mathbf{d_t}$, which is defined as:

$$\mathbf{d_t} = \mathbf{d}(t) + \mathbf{d}(t+1|t) \tag{3.15}$$

$\mathbf{d}(t+1|t)$ is predicted based on previous estimated states, described as:

$$\mathbf{d}(t+1|t) = \mathcal{P}(\mathbf{s}(t), \mathbf{s}(t-1), ..., \mathbf{s}(0)) \tag{3.16}$$

$$\mathbf{s}(t) = \mathbf{s_{t-1}} + \mathbf{d}(t)$$

where $\mathcal{P}$ is the prediction function. In this paper, the prediction model, defined as the velocity at time t, is formulated as:

$$\mathcal{P}(\mathbf{s}(t), \mathbf{s}(t-1), ..., \mathbf{s}(0)) = \mathbf{s}(t) - \mathbf{s}(t-1) \tag{3.17}$$

It is worth noting that velocity estimation in an ultra-low delay visual system with a sampling rate of more than 1000 Hz is much more accurate than in a general visual system with a sampling rate of around 60 Hz, allowing for a more reliable prediction of $\mathbf{d}(t+1|t)$. It's possible to use a more practical model for complex cases. Having obtained the predicted motion vector $\mathbf{d}(t+1|t)$ and the estimated motion vector $\mathbf{d}(t)$, the state at time $t+1$ is predicted as:

$$\mathbf{s}(t+1|t) = \mathbf{s_{t-1}} + \mathbf{d}(t) + \mathbf{d}(t+1|t) \tag{3.18}$$

The predicted state $\mathbf{s}(t+1|t)$ is taken as the output state $\mathbf{s_t}$ of the tracking system, output at time $t+1$, and fed back as the initial state to the next temporal iterative tracking. As shown in Fig. 3.5(b), the difference between the output state and the true state is significantly reduced by the introduced prediction motion vector, bringing two merits: a) The error introduced by the tracking system's delay is reduced; b) The motion estimation of next

temporal iterative tracking gets possible to be performed accurately even in high-speed moving cases. As a result, the proposed method significantly improves tracking accuracy from the perspective of real-time processing.

3.3.2.3 Temporal prediction-based temporal iterative tracking

Combining temporal prediction with temporal iterative tracking, temporal prediction-based temporal iterative tracking, which has high real-time accuracy with a low spatial computation complexity, is proposed.

Figure 3.6 graphically illustrates the conceptual difference between the conventional spatial iterative tracking and the proposed temporal prediction-based temporal iterative tracking method. In the conventional method, image warp and motion estimation are performed iteratively in the spatial domain to achieve high image processing accuracy. However, this iterative processing takes time, resulting in a significant delay. The object in the real space continues to move during this delay, leading to a large difference between the output state and the true state at t_{16} even if the output state is quite close to the true state at t_0. The proposed method maps the spatial iterative processing of image warp and motion estimation into the temporal domain. Besides, temporal prediction is applied in each spatial processing to compensate for the change that happened during the spatial processing. As sensing and feedback are performed at a higher frequency, temporal prediction predicts the change with higher accuracy. As a result, the proposed method largely reduces the difference between the output state and true state by the temporal prediction, while reducing the amount of spatial processing through temporal iterative tracking.

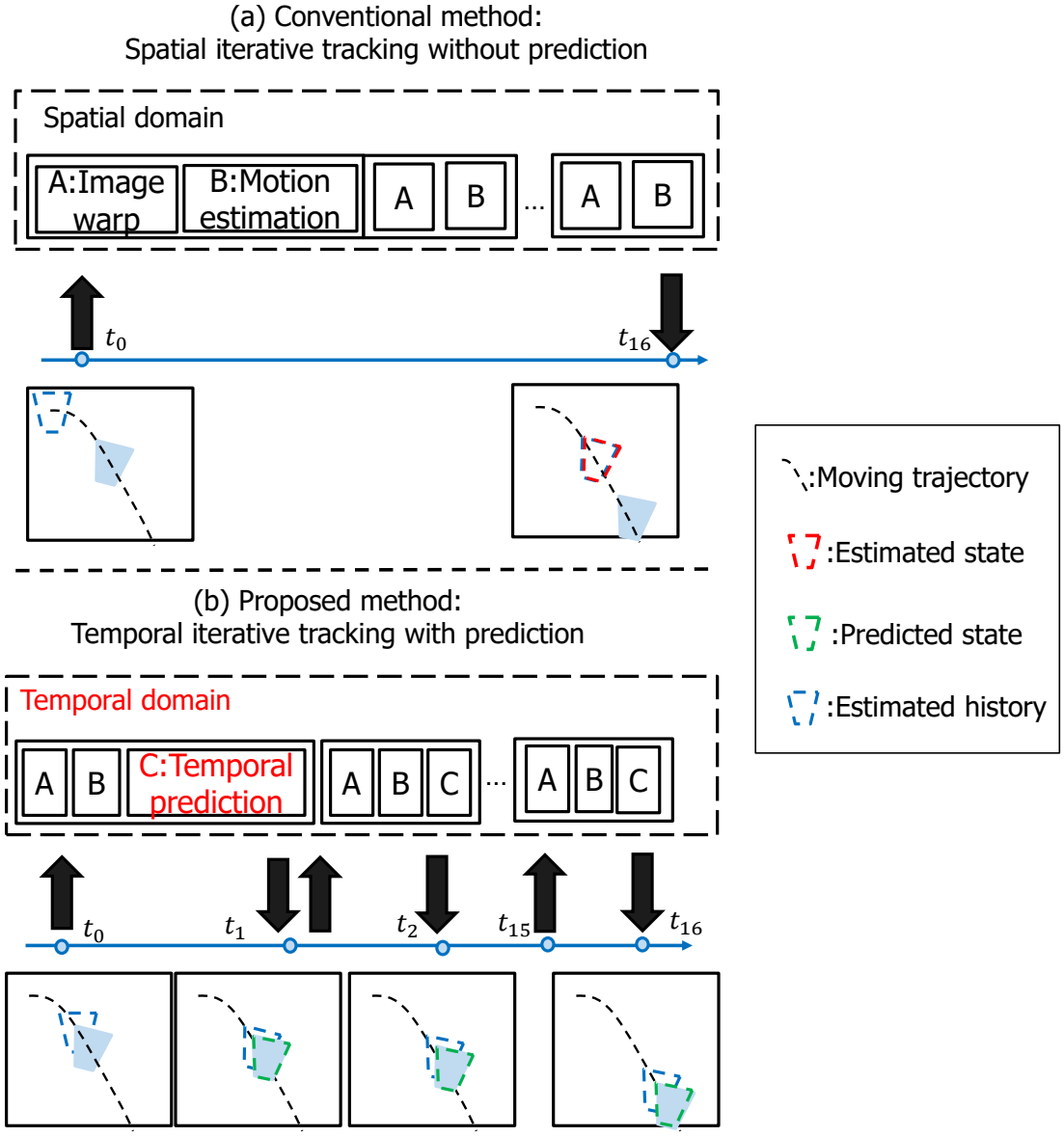

Figure 3.6: Conceptual difference between spatial iterative tracking and temporal prediction-based temporal iterative tracking.

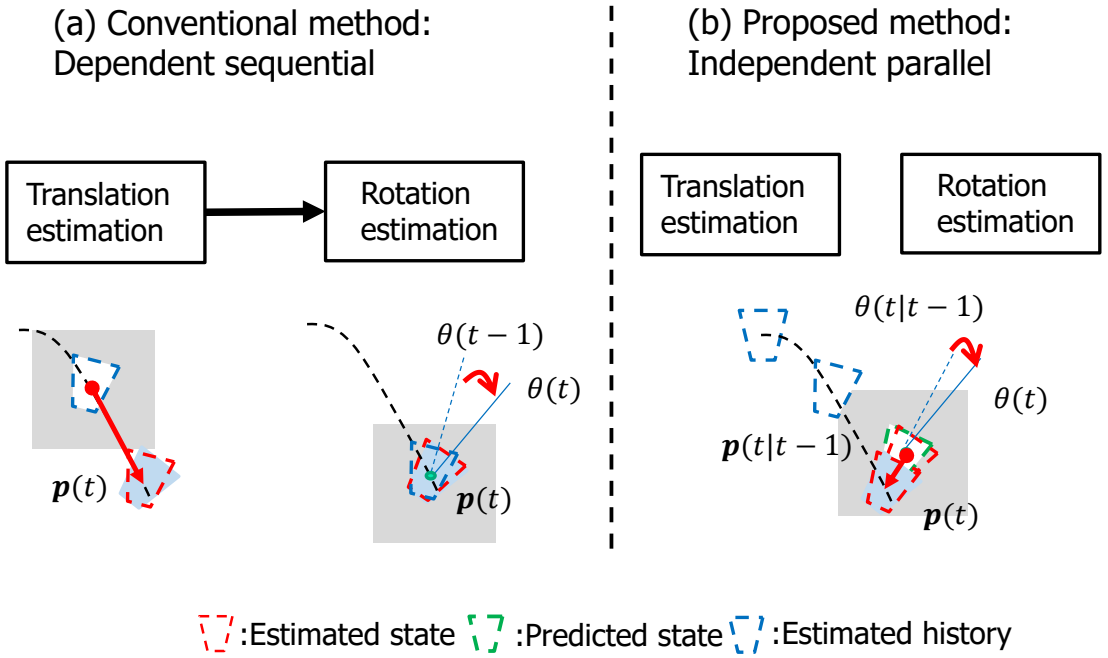

Figure 3.7: Conceptual difference in motion estimation.

3.3.3 Temporal prediction-based parallel motion estimation

Motion estimation, which is part of state estimation shown in Fig. 3.2, estimates the motion vector $\mathbf{d}(t)$ based on the difference of the corresponding image patches between the previous output state $\mathbf{s_{t-1}}$ and the current true state $\hat{\mathbf{s}}_\mathbf{t}$. It includes the estimations for translation and rotation. In the translation estimation, the translation change $\mathbf{d_a}(t)$ is estimated, where $\mathbf{d_a}(t) = [d_x(t), d_y(t)]^T$. In the rotation estimation, the rotation change $d_\theta(t)$ is estimated. Translation and rotation estimations are performed sequentially in the conventional method [55], formulated as:

$$\mathbf{d_a}(t) = \mathcal{T}(J(t, \mathbf{p}(t-1)), I(t, \theta(t-1))) \tag{3.19}$$

$$d_\theta(t) = \mathcal{R}(J(t, \mathbf{p}(t)), I(t, \theta(t-1))) \tag{3.20}$$

$$\mathbf{p}(t) = \mathbf{p}(t-1) + \mathbf{d}_a(t)$$

where $\mathcal{T}$ is the translation estimation function, which refers from Eq. (3.2)–(3.4); $\mathcal{R}$ is the rotation estimation function, which refers from Eq. (3.6)–(3.8); $J(t, \mathbf{p}(t))$ is the image patch surrounding position $\mathbf{p}(t)$ in the input image captured at time t; $I(t, \theta(t))$ is the image patch obtained by rotating image I according to angle $\theta(t)$. To ensure accurate rotation estimation, $d\theta(t)$ is estimated based on the estimated position state $\mathbf{p}(t)$ obtained after estimating the translation change vector $\mathbf{d_a}(t)$, as shown in Fig. 3.7(a). This, however, results in a data dependency between the estimation of $\mathbf{d_a}(t)$ and $d\theta(t)$, making the estimation of $\mathbf{d_a}(t)$ and $d\theta(t)$ performed in sequential. Sequential processing causes not only significant delays but also significant resource costs in FPGA implementation. To solve this problem, a temporal prediction-based motion estimation method, which breaks the data dependency by temporal prediction, is proposed to make the translation and rotation estimation performed in parallel.

To break the data dependency between the translation and rotation estimations, predicted position $\mathbf{p}(t|t-1)$ and angle $\theta(t|t-1)$ are introduced. The estimation of translation change $\mathbf{d_a}(t)$ and rotation change $d\theta(t)$ are performed independently based on the previously predicted states $\mathbf{p}(t|t-1)$ and $\theta(t|t-1)$, allowing translation and rotation estimations to be processed parallelly, as shown in Fig. 3.7(b), formulated as:

$$\mathbf{d_a}(t) = \mathcal{T}(J(t, \mathbf{p}(t|t-1)), I(t, \theta(t|t-1))) \tag{3.21}$$

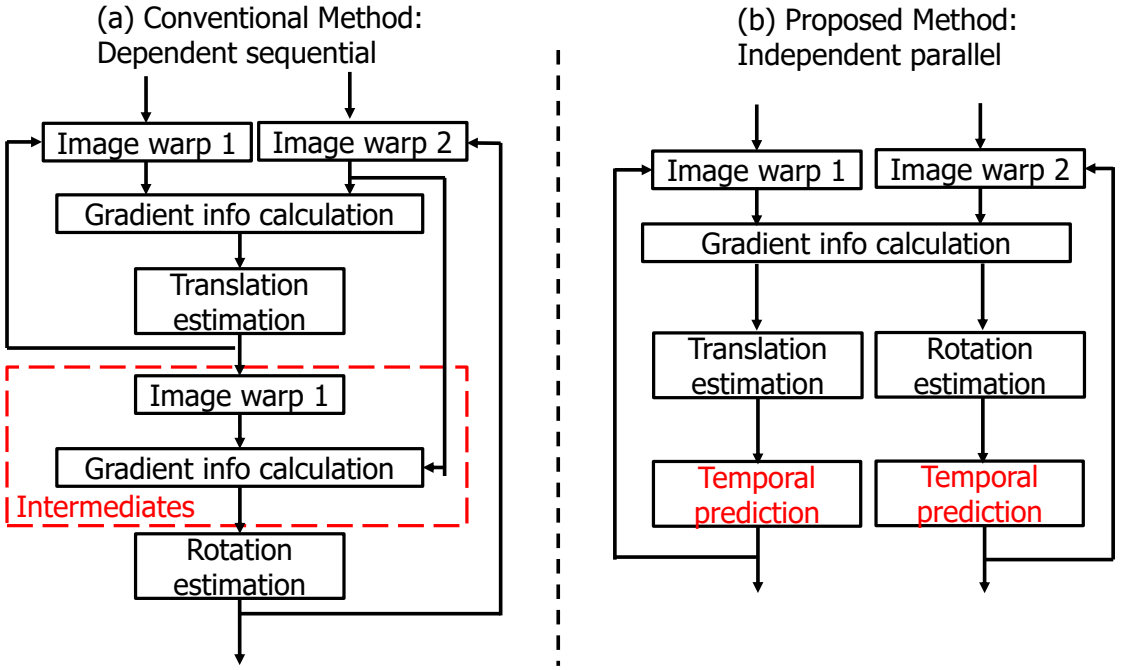

Figure 3.8: Comparison of motion estimation architectures.

$$d_\theta(t) = \mathcal{R}(J(t, \mathbf{p}(t|t-1)), I(t, \theta(t|t-1))) \tag{3.22}$$

A good prediction provides a displacement vector $\mathbf{d_a}(t|t-1)$, which is a good substitute of $\mathbf{d_a}(t)$ that compensates for the translation motion, allowing the translation and rotation estimations to be performed in parallel while keeping the estimation accuracy.

Parallel motion estimation is greatly beneficial for hardware implementation. In the conventional method, The rotation estimation and translation estimation are indirectly dependent as shown in Fig. 3.8(a). Image warp 1 and gradient info calculation modules are intermediates that help update the gradient information used in rotation estimation based on the translation estimated in the translation module. The pixel stream used for rotation estimation, however, must wait until translation estimation is complete before being

processed in intermediate modules, resulting in a significant delay. Assumes tracking a $M \times M$ template, where translation estimation is performed in a $N \times N$ ($N < M$) local area and rotation estimation is performed in a $M \times M$ local area, the waiting process's delay demands are around $\frac{M+N}{2}$ lines. This waiting process generates not only significant delays but also significant memory costs. Besides, the processing of intermediate modules necessitates additional delays and resource costs. As a result of the indirectly dependent processing style, a large number of resources and delays are required. The proposed method overcomes these issues through the parallel architecture depicted in Fig. 3.8(b). The independent processing style of rotation and translation estimations avoids the aforementioned problems caused by intermediate modules.

3.4 Hardware architecture and system implementations

3.4.1 Hardware environment

Fig. 3.9 shows the demonstration environment of the proposed ultra-low delay tracking system. A Camera $\Rightarrow$ PL (FPGA board) $\Leftrightarrow$ PS (FPGA board) system is designed with a high-speed camera and system-on-chip FPGA board, where PS is the processing system with the Debian operation system, and PL is the programmable logic.

3.4.2 Overall hardware architecture

The system-level framework of the proposed tracking system is shown in Fig. 3.10. The tracking processing is completed on one chip, where PL serves as the main processing unit of the tracking system, and PS serves as the initializer and postprocessor. The PL receives a pixel stream from a high-speed camera, which is grayscale data fed in 10

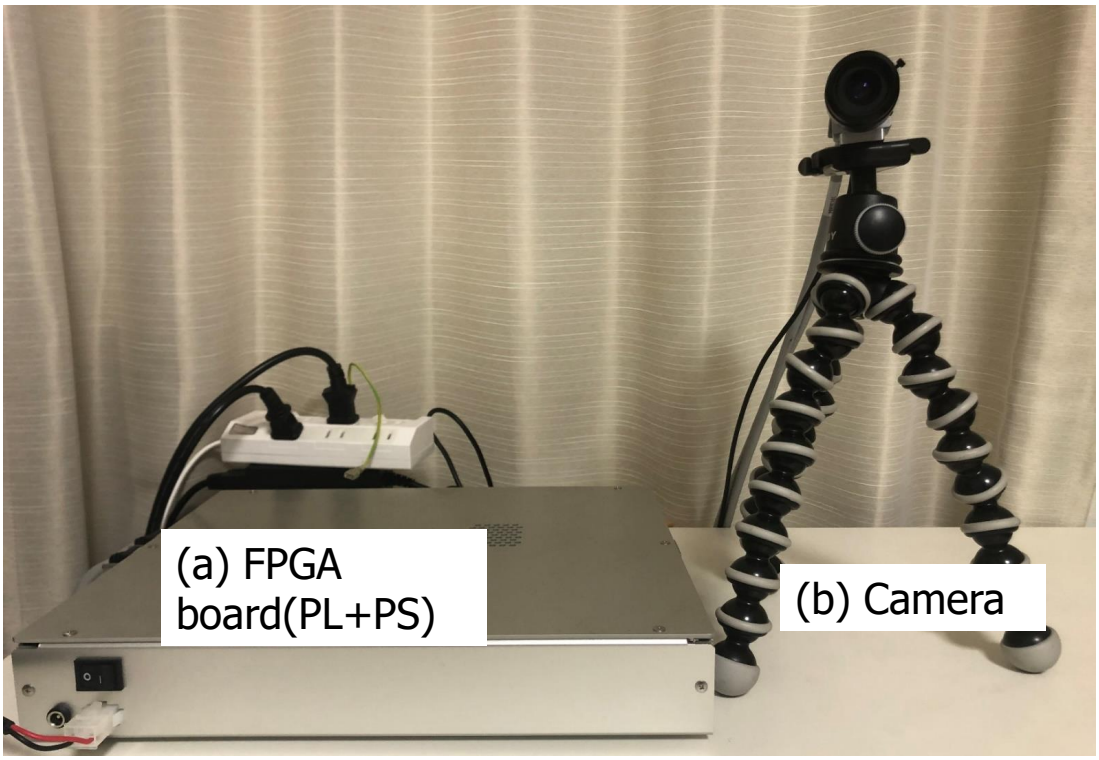

Figure 3.9: Demonstration of the tracking system. (a) Camera: Basler acA2000-340km; (b) FPGA: Zynq UltraScale+ MPSoC ZCU102 (ZU9EG).

pixels at a time. The data is transformed into four paralleled pixels in the camera link receiver, serving as the image process core's input. The primary means by which PS communicates with PL is via a register-access bus. It carries out the two critical data communications listed below. a) Transmit the template data from PS to PL when PS serves as the initializer; b) Transmit the predicted position and angle from PL to PS for further post-processing when PS serves as the postprocessor. Besides, an external memory DDR4-SDRAM, which allows a large amount of data communication between PL and PS, is provided for simulation and post-analysis.

In the image process core, the data stream flows from Pre-process to Prediction. The

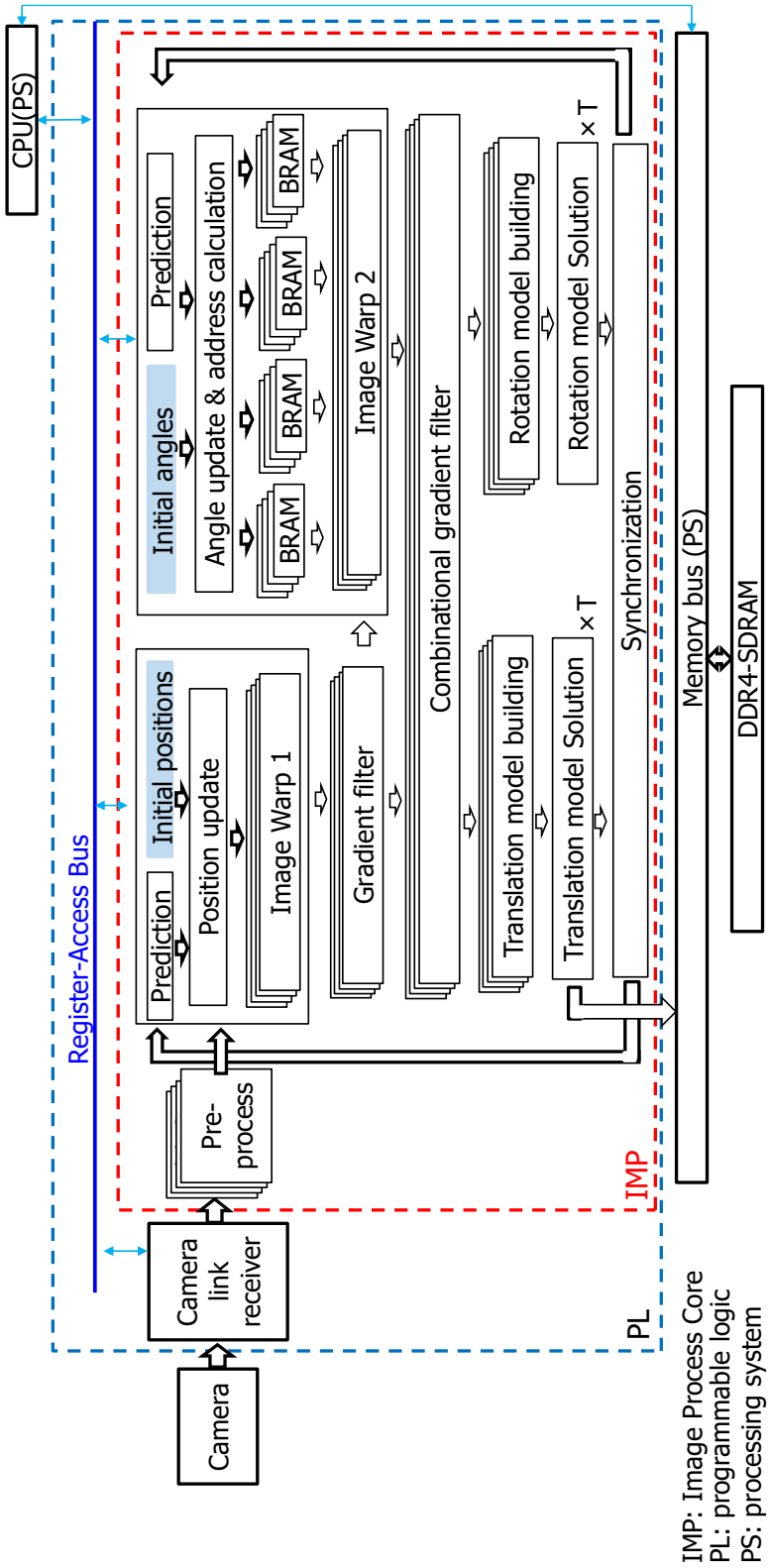

Figure 3.10: System-level framework of the proposed tracking system

predicted position is fed back to the Position update module, updating the position used for input image warping. The predicted angle is fed back to the Angle update & address calculation module, updating the angle used for template image warping. To keep the fully pipelined stream data flow, four-by-four sets of template data are read from corresponding Block RAMs (BRAM) that contain intensity and gradient information of the template. Translation and rotation estimations are performed separately on different-sized local areas, resulting in estimated position and angle output at different times. A synchronization module is leveraged to synchronize them. The calculation in the image process core is based on the 18-bit signed fixed point. Last but not least, a label scanner-based multi-stream spatial processing style is applied to handle multi-template tracking with low resource cost.

3.4.3 Label scanner-based multi-stream spatial processing

In the LK method, the motion between the input image patch and the template is estimated through local spatial processing, which is shown in Figure 3.2. However, in the FPGA-based high frame rate and ultra-low delay vision system, spatial processing influences both the resource cost and the delay. Therefore, how to handle heavy spatial processing properly becomes quite important. In this paper, label scanner-based multi-stream spatial processing, a template tracking orientated hardware structure, is proposed to handle this issue. It mainly has two advantages: 1) Small resource costs; 2) Handling templates with different sizes.

Figure 3.11 shows the comparison of the hardware architecture of local spatial processing between the conventional method and the proposed method. Sliding window-based single stream, which accumulates all the data in the image patch by line buffers

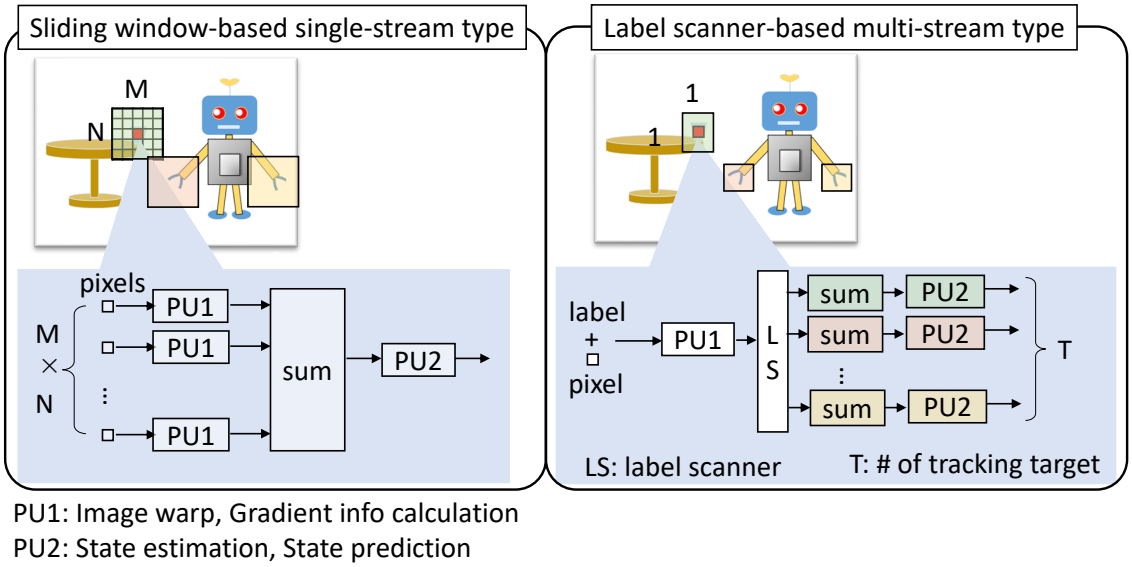

Figure 3.11: Hardware architectures of spatial processing.

and then applies PU1 to all of them in parallel, is commonly used in the FPGA-based high-speed vision system, such as [18]. It works well with small image patches. However, in template tracking, applying motion measurement on large image patches becomes necessary. This makes the resource cost caused by the parallelism of PU1 become a big problem, where PU1 is a heavy processing unit to calculate the derivatives of each data in the image patch. The proposed method - label scanner-based multi-stream spatial processing, which avoids the parallelism of PU1 via label scanner and achieves multi-template tracking via multi streams, is an architecture specialized for large-size template tracking.

The details of the label scanner-based multi-stream spatial processing when tracking three templates are shown in Figure 3.12. A label map, which contains the information that represents which template the pixel belongs to, is generated according to the positions of templates and their sizes. In this way, the input to the local spatial processing becomes a data stream that contains the current input pixel and its corresponding label. Instead of

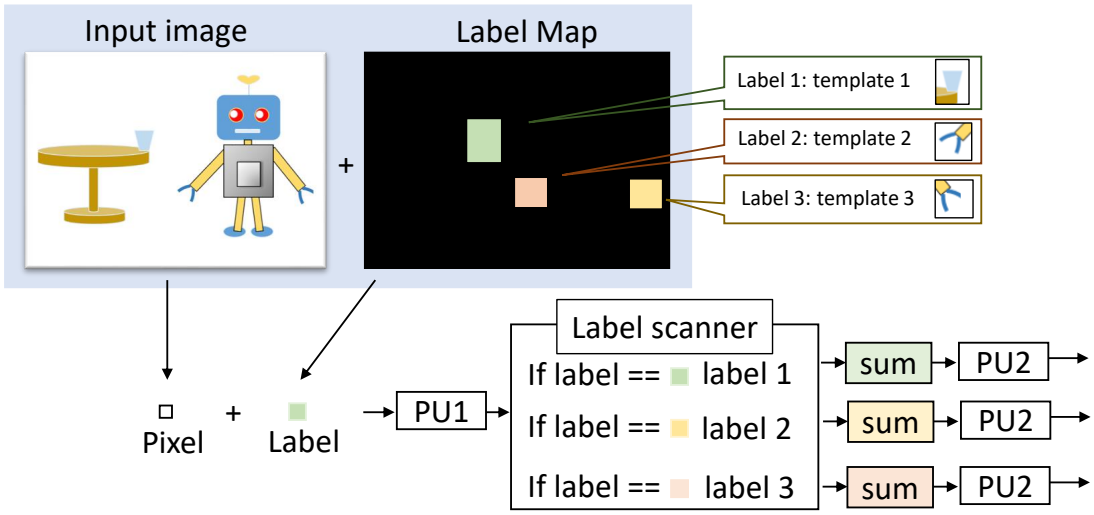

Figure 3.12: Label map-based label scanner.

accumulating all the data in the image patch before applying PU1, the proposed method maps the local spatial processing into the temporal input data stream and performs PU1 to each coming labeled data. Having obtained the derivatives of each labeled data, the label scanner determines which PU2 stream the derivatives belong to according to the label, and distributes them into the corresponding PU2 stream. In this process, the spatial data of the image patch are aggregated into its corresponding stream for displacement estimation.

The proposed method is a template-tracking-oriented architecture. On the one hand, the proposed method significantly reduces the resource cost caused by local spatial processing in large image patches. Assume tracking T templates with the size of $M \times N$, the resource cost of the conventional and the proposed method are $M \times N \times R + Q$ and $T \times Q + R$ respectively, where R is the resource cost of PU1, Q is the resource cost of PU2. The proposed method requires much fewer resources when tracking several tem-

plates with large sizes. On the other hand, the hardware performance of the proposed is not influenced by the template size, making it possible to handle multiple templates with different sizes. Compared with the conventional method that only handles templates of the same size, the proposed method has a better tracking ability by giving templates more suitable sizes according to the shape of the tracking targets.

3.5 Algorithm evaluation

Algorithm evaluation is utilized to assess the validity of the proposed methods, including temporal iterative tracking, temporal prediction-based temporal iterative tracking, and temporal prediction-based parallel motion estimation.

3.5.1 Evaluation criterion

3.5.1.1 Tracking error

Tracking error (E), which represents the average of error of tracking targets are tracked successfully, is described as:

$$E_m = \frac{\sum_{i=1}^{i=n}(1-l_i)E_{(i,m)}}{\sum_{i=1}^{i=n}(1-l_i)} \tag{3.23}$$

where n is the total number of tracking targets. l_i, which represents whether the ith target fails to be tracked or not, is described in detail in Sec. 3.5.1.2. The error for each tracking target can be the mean absolute error, which is written as:

$$E_{(m,i)} = MAE_{(i,m)} = \frac{1}{T}\sum_{t=1}^{t=T}(|m_{(i,t)} - \hat{m}_{(i,t)}|) \tag{3.24}$$

The error for each tracking target can be the root mean square error, which is written as:

$$E_{(m,i)} = RMSE_{(i,m)} = \sqrt{\frac{1}{T}\sum_{t=1}^{t=T}(m_{(i,t)} - \hat{m}_{(i,t)})^2} \tag{3.25}$$

When it comes to feeding the tracking result back to the actuators, the accuracy of each axis is critical. Therefore, tracking errors for x, y, and θ are evaluated separately in this evaluation. $m_{(i,t)} = x_{(i,t)}$ is set when evaluating tracking error for x, where $x_{(i,t)}$ is the output x value of tracking target i in frame t. $m_{(i,t)} = y_{(i,t)}$ is set when evaluating tracking error for y. $m_{(i,t)} = \theta_{(i,t)}$ is set when evaluating tracking error for θ. $\hat{m}_{(i,t)}$ is the corresponding ground truth for $m_{(i,t)}$; When evaluating from the perspective of image processing, $\hat{m}_{(i,t)}$ represents the state of the object captured at frame t. When evaluating from the perspective of real-time processing, $\hat{m}_{(i,t)}$ represents the state of the object captured at frame $t + t_0$, where t_0 is the delay of the tracking system. Accurate tracking always goes along with a low tracking error.

3.5.1.2 Tracking lost rate

Tracking lost rate (LR), which represents the rate of tracking targets that failed to be tracked, is described as:

$$LR = \frac{1}{n} \sum_{i=1}^{n} l_i \tag{3.26}$$

whether the target fails to be tracked is determined by:

$$l_i = \begin{cases} 1, & E_{(i,x)} > h \parallel E_{(i,y)} > h \parallel E_{(i,\theta)} > h \\ 0, & otherwise \end{cases} \tag{3.27}$$

where h is the threshold set for tracking loss. A stable tracking performance always goes along with a low tracking lost rate.

3.5.1.3 Maximum tracking distance

Maximum tracking distance is the tracking target's maximum moving distance that can be tracked by the tracking algorithm. Given D_i as the maximum tracking distance

when applying the tracking algorithm with i local spatial processings to two consecutive frames, the maximum tracking distance of spatial iterative tracking under I iterations can be described as:

$$DS_I = 1 \times D_I \tag{3.28}$$

The maximum tracking distance of the temporal iterative tracking under I iterations can be described as:

$$DT_I = I \times D_1 \tag{3.29}$$

In order to get an effective maximum tracking distance, the maximum distance that can be tracked by 80% of templates is regarded as the maximum tracking distance. A good tracking capability always goes along with a large maximum tracking distance.

3.5.2 Evaluation of temporal iterative tracking

This evaluation focuses on evaluating temporal iteration tracking using a translation estimation model. The translation model consists of image warp, gradient info calculation, and state estimation which only contains the translation estimation.

3.5.2.1 Datasets

The following two kinds of datasets are used for evaluation.

1) Synthetic datasets: three sequences are made by the image shown in Figure 3.13. Figure 3.13 is an 8K image, provided by ITE [1]. Sequences, which contain the subpixel information made by shrinking the 8K image (1/10), are shown in Figure 3.14. These three sequences record the uniform linear motions along with the directions of 0°, 45°, and 90°, respectively. Each of the sequences generates two kinds of datasets. (a) Boat

[1] https://www.ite.or.jp/content/chart/uhdtv/

Figure 3.13: Image used for sequence generation: 7680×4320 pixels.

dataset1, records uniform linear motions moving at the speed of $n \times v$ pixel/frame, where v is the speed of the above basic sequence. (b) Boat dataset2, records uniform linear motions moving at the $n \times v$ pixel/ms speed and captured at 30 fps, 62 fps, and 1000 fps separately. In the evaluation using synthetic datasets, 20 image patches from the first frame are selected as templates for tracking. Figure 3.15 shows several examples of them.

2) Real datasets: two sequences captured by the high-speed camera at 1000 fps are shown in Figure 3.16. These two sequences record the movement of the object moving mainly along the direction of 0° and 90° respectively. Besides the movement in the main direction, movement in other directions near the main direction is also included in the real sequence. The object moves at a random speed. Each of the sequences generates a dataset named people dataset, which records the motion that is captured at 30 fps, 62 fps, and 1000 fps separately. In the evaluation using real datasets, 50 image patches from the

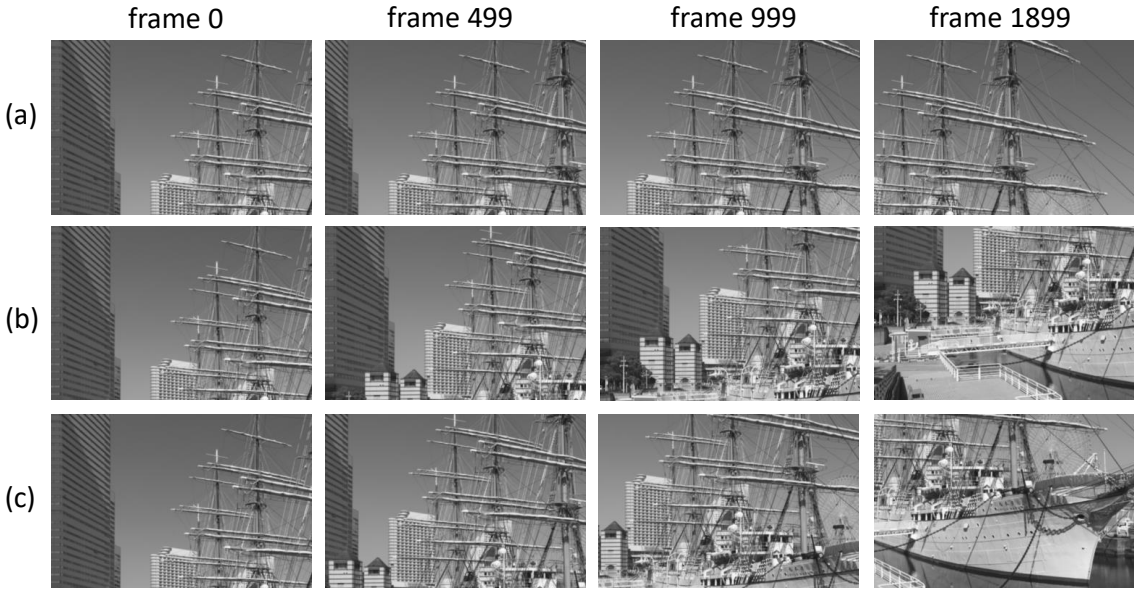

Figure 3.14: Synthetic sequences. (a) Moving along with the direction of 0°: 360 × 240 pixels, 1900 frames, v = 0.1 pixel/frame; (b) Moving along with the direction of 90°: 360 × 240 pixels, 1900 frames, v = 0.1 pixel/frame; (c) Moving along with the direction of 45°: 360 × 240 pixels, 1900 frames, v = 0.141 pixel/frame.

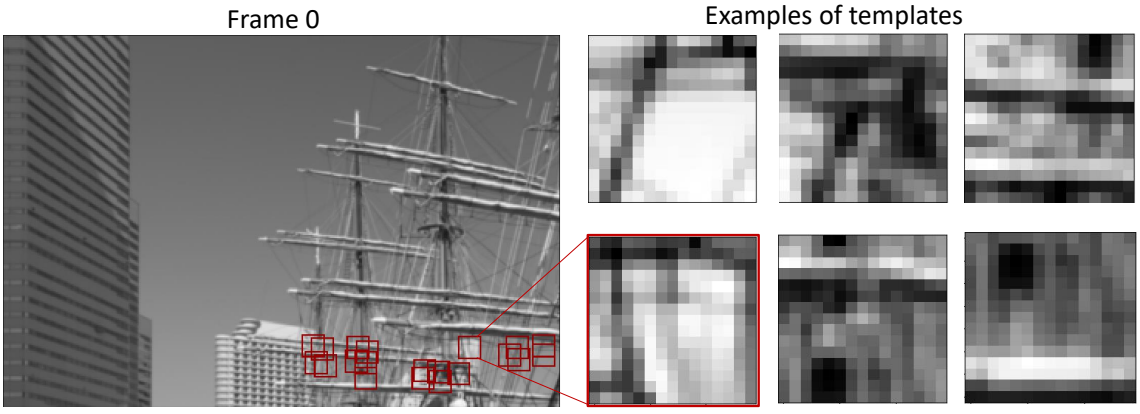

Figure 3.15: Examples of templates in synthetic sequences. Template size: 15×15 pixels.

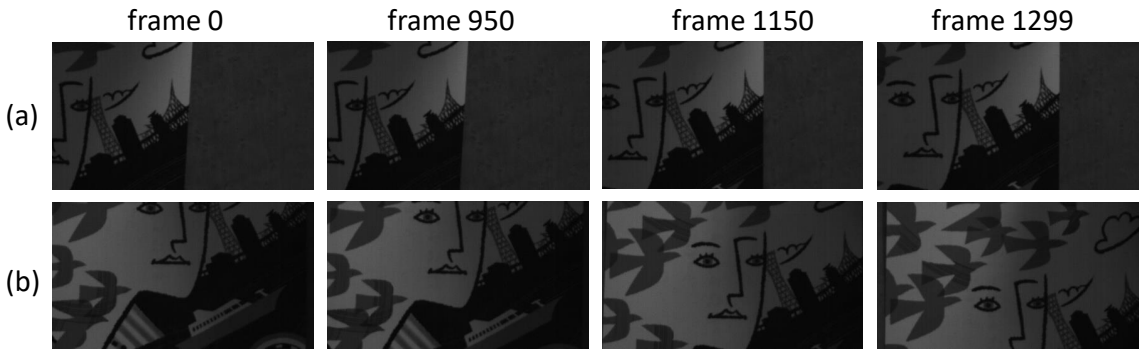

Figure 3.16: Real 1000 fps sequences. (a) Moving mainly along with the direction of 0°: 640 × 360 pixels, 1860 frames, v =0.002~0.783 pixel/ms; (b) Moving mainly along with the direction of 90°: 640 × 360 pixels, 1300 frames, v = 0.003~0.875 pixel/ms.

first frame are selected as templates. Figure 3.17 shows several examples of them.

3.5.2.2 Evaluation on maximum tracking distance

In this evaluation, the maximum tracking distance of the conventional spatial iterative tracking and the proposed temporal iterative tracking under the same iteration number is evaluated based on the boat dataset1. With the same iteration number, the computation complexity of spatial iterative tracking and temporal iterative tracking is the same.

Figure 3.18 shows the evaluation results of maximum tracking distance for spatial iterative tracking and temporal iterative tracking performed on boat dataset1. With the increase of the iteration number, the maximum tracking distance of the temporal iterative tracking is proportional to the iteration number, while the maximum tracking distance of the spatial iterative tracking is converged to a certain value. Therefore, under the same computational complexity, temporal iterative tracking outperforms spatial iterative tracking in terms of whether or not the object can be tracked.

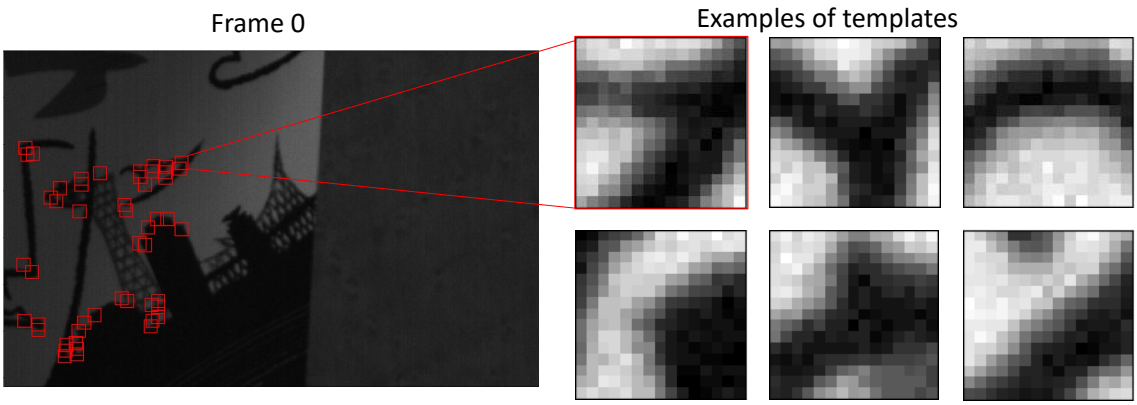

Figure 3.17: Examples of templates in real sequences. Template size: 15×15 pixels.

3.5.2.3 Evaluation on tracking error and tracking lost rate from the perspective of image processing

In this evaluation, tracking lost rate and tracking error of the following three methods are evaluated based on boat dataset2 and people dataset from the perspective of image processing: (Method 1) spatial iterative tracking with 33 spatial iterations performed on 30 fps sequences; (Method 2) spatial iterative tracking with 16 spatial iterations performed on 62 fps sequences; (Method 3) temporal iterative tracking with one spatial iteration performed on 1000 fps sequences. These three methods have the same computational complexity for tracking the same scene due to the balanced framerate and the spatial iteration numbers.

Figure 3.19 shows the evaluation results on the boat dataset2. In terms of tracking error, temporal iterative tracking gradually approaches spatial iterative tracking as the speed of the object decreases. As can be seen from tracking errors in the x-axis and y-axis, temporal iterative tracking can achieve a relatively low tracking error of fewer than 0.1 pixels when tracking objects moving at a low speed. In terms of tracking lost rate,

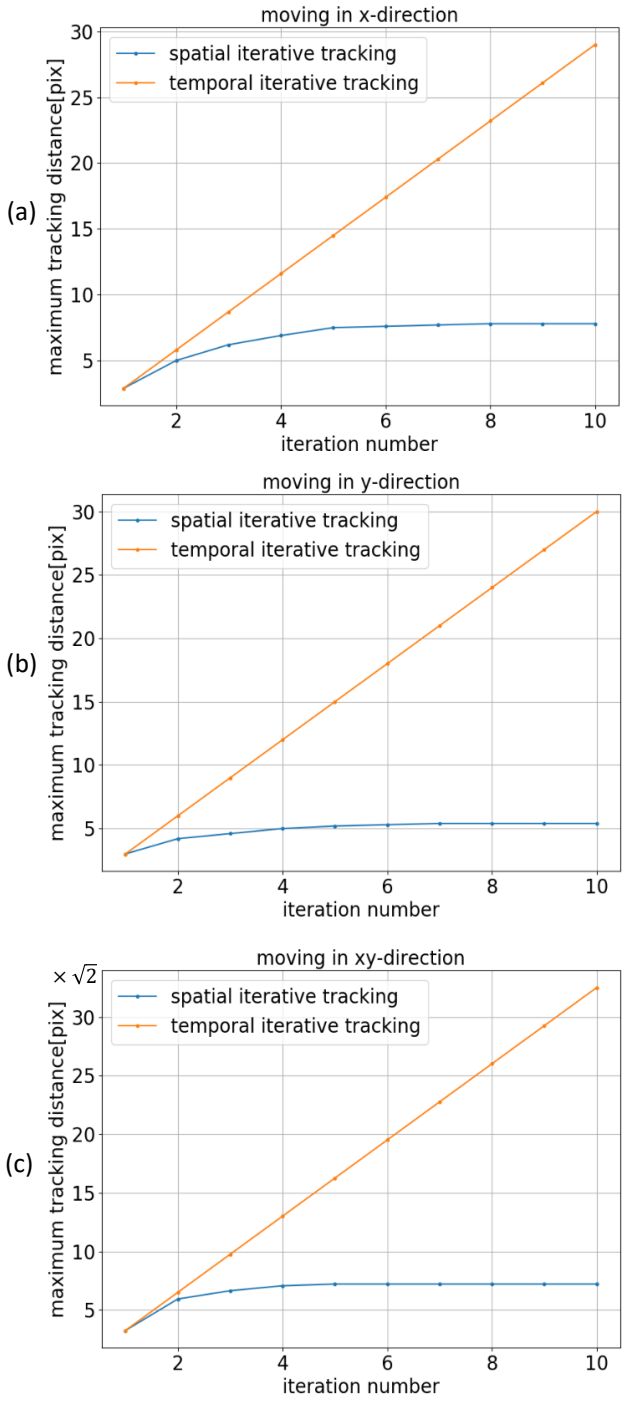

Figure 3.18: Evaluation results of maximum tracking distance. The evaluation is performed on boat dataset1 that records boat moving along with the direction of (a) 0°; (b) 90°; (c) 45°.

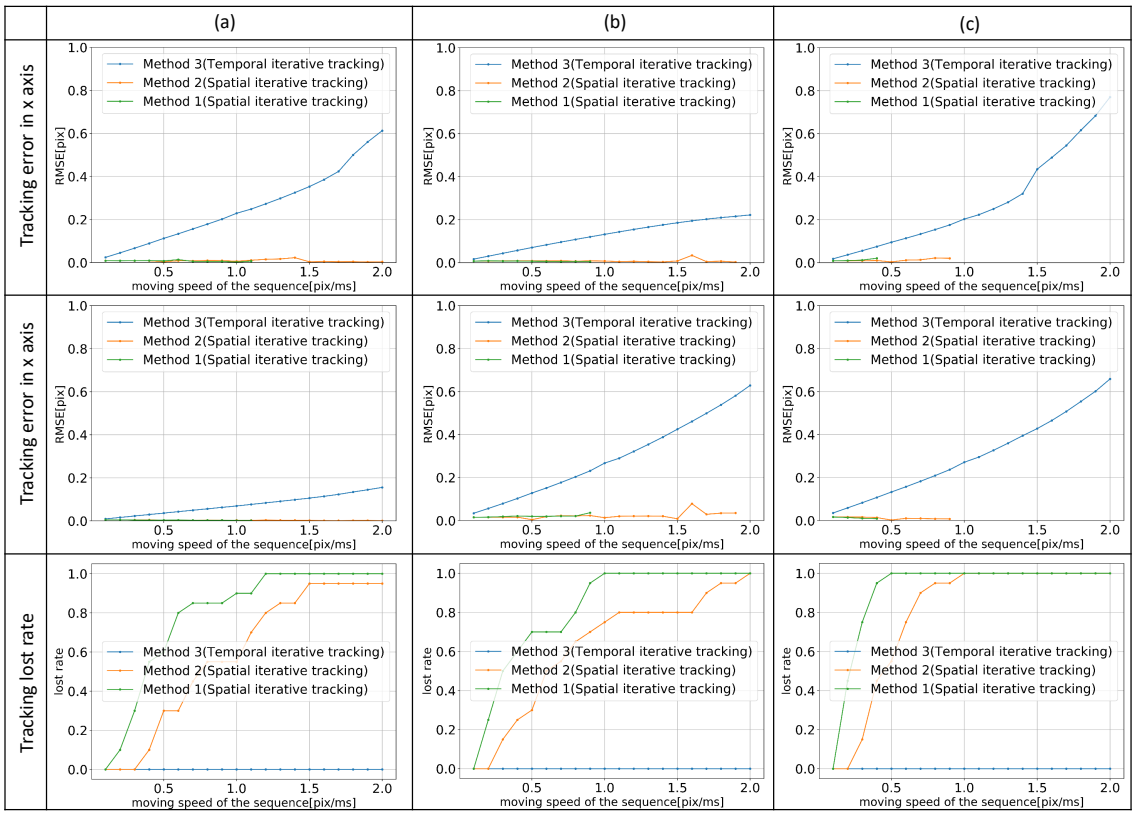

Figure 3.19: Evaluation results of tracking error and tracking lost rate from the perspective of image processing. The evaluation is performed on boat dataset2 that records boat moving along with the direction of (a) 0°; (b) 90°; (c) 45°.

temporal iterative tracking maintains lower tracking lost rate even when the moving speed of the objects increases, which performs better than spatial iterative tracking methods. As a result, temporal iterative tracking has the characteristic of tracking object with coarse accuracy when it moves at high speed and obtaining higher accuracy when it slows down. This characteristic is called dynamic tracking, which is preferred in robotics applications.

Table 3.1 and 3.2 show the evaluation results on the people dataset. The same as the results on boat dataset2, the temporal iterative tracking method (Method 3) performs

better than the spatial iterative tracking methods (Method 1, 2) in tracking lost rate. And the spatial iterative tracking methods perform better than the temporal iterative tracking method in tracking error.

Table 3.1: Tracking lost rate on people dataset.

Sequence	Method 1	Method 2	Method 3
0°	0.82	0.40	0
90°	0.84	0.52	0

Table 3.2: Tracking error on people dataset.

Sequence	Method 1		Method 2		Mthod 3	
	x	y	x	y	x	y
0°	0	0	0.0003	0.0001	0.1559	0.0509
90°	0.0067	0.009	0.0006	0.0003	0.0689	0.1698

3.5.2.4 Evaluation on tracking error from the perspective of real-time system

From the perspective of image processing, spatial iterative tracking gets more accurate estimated positions than temporal iterative tracking. However, due to large delays, the positions estimated by spatial iterative tracking can hardly reflect the actual position of the object in a real-time system. In the real-time system, the object will continue moving during the time used to calculate the estimated position. Therefore, the object may move to a new position after we get the estimated position. This leads to a large gap between the estimated position and the actual position of the object. The tracking error in the real-time

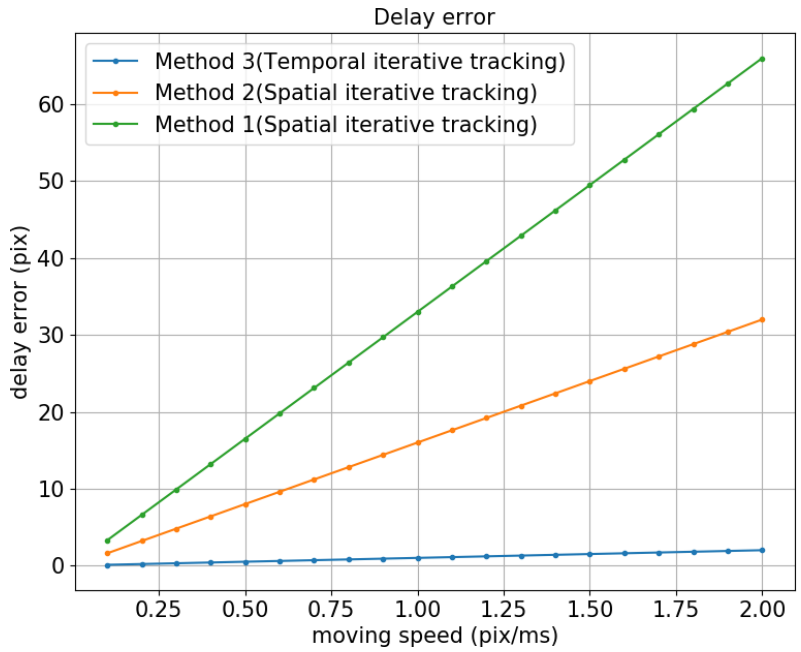

Figure 3.20: Evaluation results of delay error.

system can be expressed as:

$$RMSE_R = RMSE_I + E_D \tag{3.30}$$

where $RMSE_R$ is the tracking error of the real-time vision system; $RMSE_I$ is the error caused by image processing; E_D is the error caused by delay. In this evaluation, $RMSE_R$ of the three methods introduced in Section 3.5.2.3 is evaluated based on boat dataset2.

The delays of Method 1-3 are 30 ms, 16 ms, and 1 ms respectively when they are implemented in real-time with architecture that has no I/O bottleneck. Figure 3.20 shows the delay error (E_D) of Method 1–3 at various moving speeds. It can be seen that the proposed temporal iterative tracking outperforms the conventional spatial iterative tracking methods due to the lower delay.

Combining the error caused by the delay (E_D) and the error caused by the image processing ($RMSE_I$), the tracking error of the real-time system ($RMSE_R$) is shown as

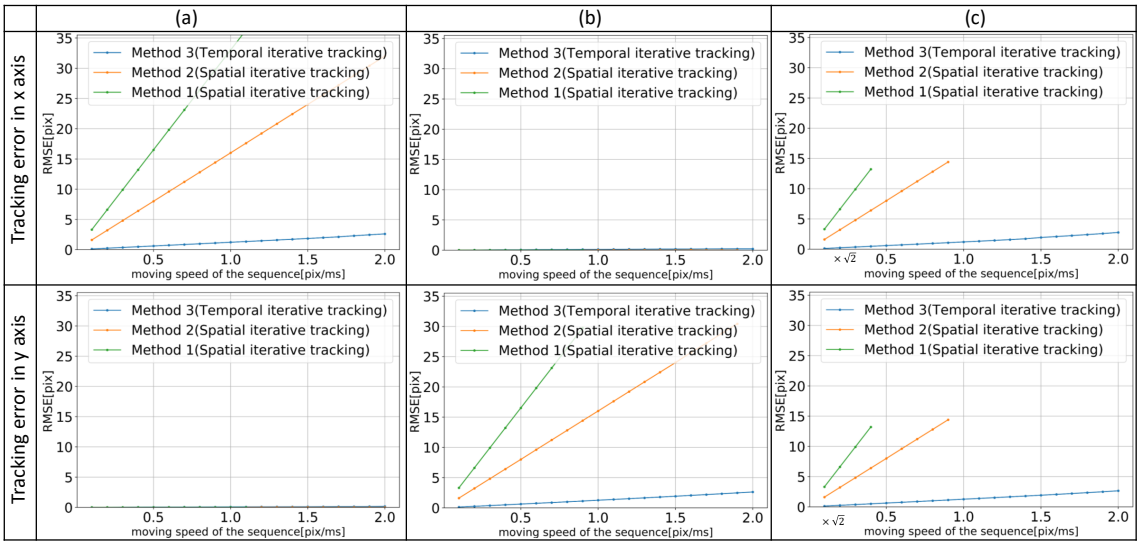

Figure 3.21: Evaluation results of tracking error from the perspective of real-time system. The evaluation is performed on boat dataset2 that records boat moving along with the direction of (a) 0°; (b) 90°; (c) 45°. The results are plotted in the same scale range for comparison.

Figure 3.21. Compared with the $RMSE_I$ shown in Figure 3.19, it can be known that the delay error affects the performance of the real-time system a lot, and it mainly happens in the direction of the movement. As a result, the proposed temporal iterative tracking, which has a smaller delay (1 ms), effectively suppresses the delay error, making the real-time system achieves better performance.

3.5.3 Evaluation of temporal prediction-based temporal iterative tracking and prallel motion estimation

This evaluation focuses on evaluating temporal prediction-based temporal iteration tracking and parallel motion estimation using a translation and rotation estimation model. The translation and rotation model consists of image warp, gradient info calculation, state

estimation, and state prediction, where state estimation and state prediction contain estimation and prediction for both translation and rotation.

3.5.3.1 Datasets

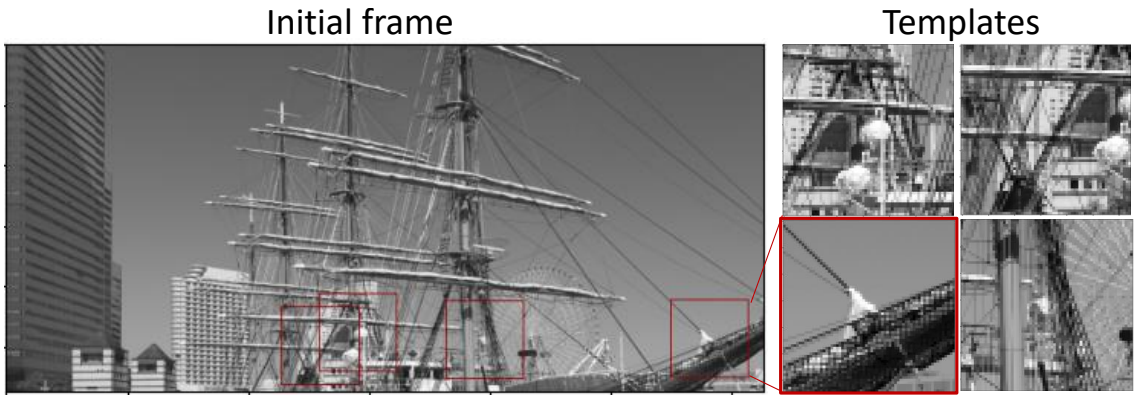

Figure 3.22: Templates in synthetic sequences. Template size: 65×65 pixels.

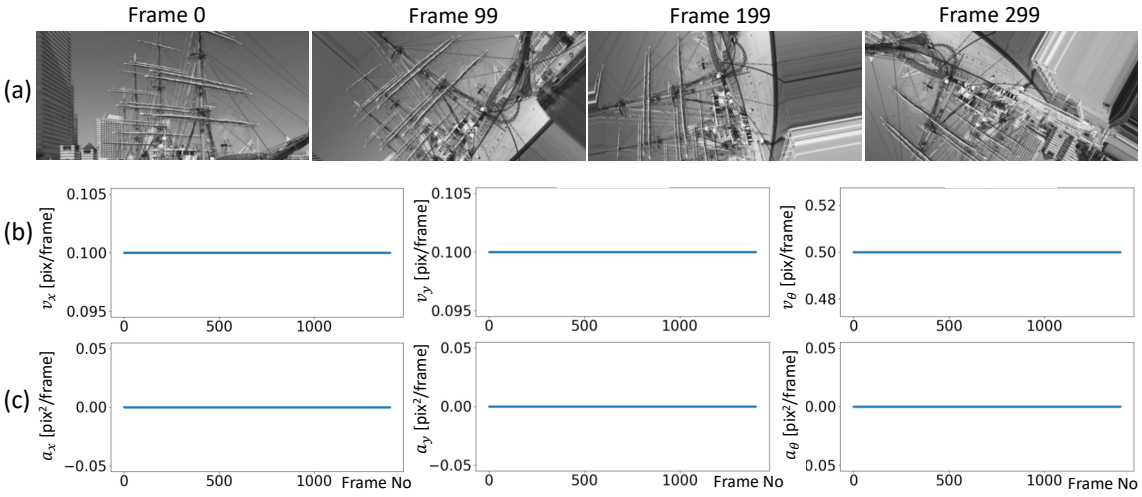

Figure 3.23: Templates in synthetic sequences. Template size: 65×65 pixels.

In FA scenes, such as object picking with orthogonal robots and position location with an alignment system, the relative movement between the object and camera mainly

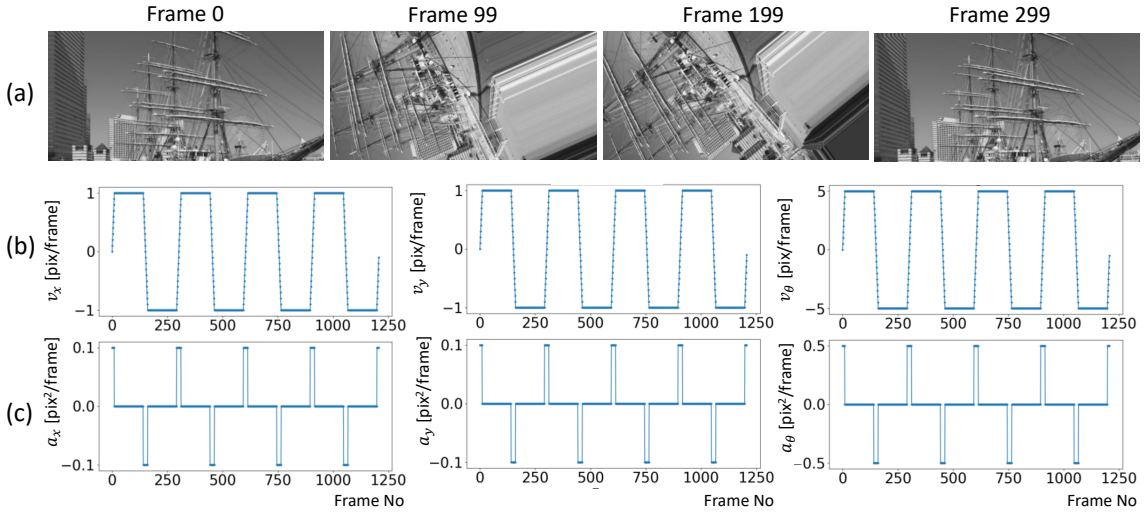

Figure 3.24: Sequence and motion model in dataset B. (a) Basic sequence: 627×287 pixels, 1208 frames; (b) Velocity of motion model [pixel/frame]: $vmax_x = vmax_y = 1$, $vmax_\theta = 5$, $vmin_x = vmin_y = -1$, $vmin_\theta = -5$; (c) Acceleration of motion model [pixel^2/frame]: $amax_x = amax_y = 0.1$, $amax_{angle} = 0.5$, $amin_x = amin_y = -0.1$, $amax_\theta = -0.5$.

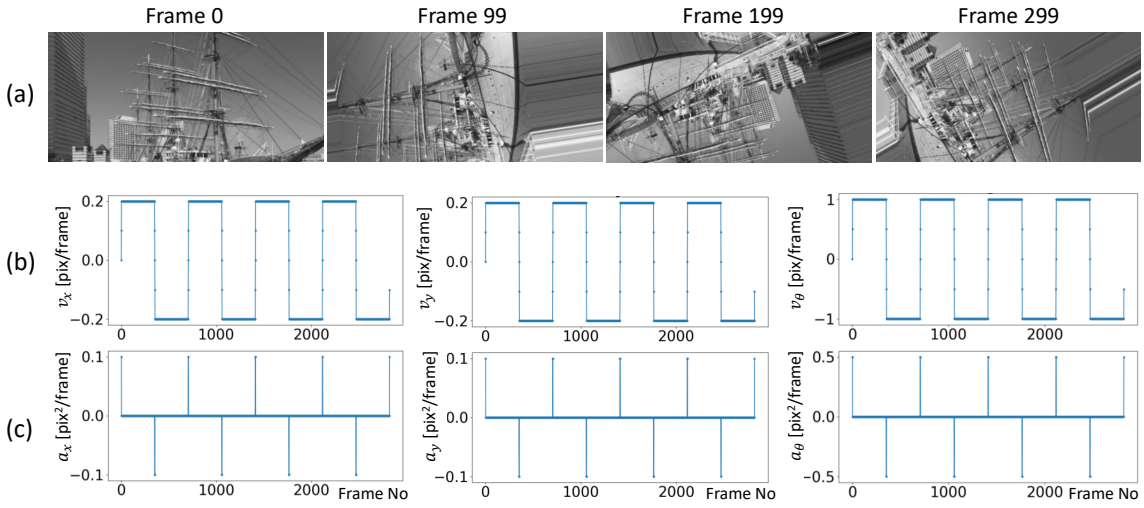

Figure 3.25: Sequence and motion model in dataset C. (a) Basic sequence: 627×287 pixels, 2824 frames; (b) Velocity of motion model [pixel/frame]: $vmax_x = vmax_y = 0.2$, $vmax_\theta = 1$, $vmin_x = vmin_y = -0.2$, $vmin_\theta = -1$; (c) Acceleration of motion model [pixel^2/frame]: $amax_x = amax_y = 0.1$, $amax_{angle} = 0.5$, $amin_x = amin_y = -0.1$, $amin_\theta = -0.5$.

depends on the movement of actuators. With an actuator that moves at a maximum speed of 480 mm/s, the target moves at about 1 pixel/ms when it ' s captured by the camera with a specification of 0.46 mm/pixel. Synthetic datasets, which take this into account, are made for evaluation.

Synthetic datasets are made by the image shown in Fig. 3.13. Sequences with subpixel-level information are made by shrinking the 8K image into $\frac{1}{10}$. Meanwhile, the sequences are created in grayscale because color information is not a necessity in the Lucas Kanade algorithm that estimates motion using gradient information. Four image patches with high-intensity variance, shown in Fig. 3.22, are selected as the tracking templates. In this evaluation, the template size is set as 65×65 pixels, which is sufficient for tracking precision components with small sizes in the field of FA. When tracking targets with large sizes, it is also a good idea to use several interesting areas with small sizes in the object as tracking templates rather than the entire object area. Two types of datasets with different motion models are made for each template. a) Dataset A, which records uniform linear motions, is used to evaluate the tracking performance under uniform moving at maximum speed. Fig. 3.23 shows the basic sequence of one template and its motion model. It can be extended to sequences captured at 62 fps and 1000 fps with the moving speed of $n \times v$ pixel/ms, where v pixel/frame is the speed of the above basic sequence. In this evaluation, dataset A contains sequences with speeds ranging from 0.1 pixel/ms to 1.9 pixel/ms captured at both 62 fps and 1000 fps. And the maximum moving speed of the above-illustrated FA scene (1 pixel/ms) is included in this dataset. b) Dataset B and dataset C, which record non-uniform motion moving, are utilized to evaluate the tracking performance in FA scenes. FA tasks, such as object picking and alignment, involve primarily three types of motions: acceleration motion, uniform motion, and deceleration motion.

These three motions are contained in datasets B and C. Fig. 3.24 and Fig. 3.25 show the sequence of one template and its motion model with maximum speeds of 1 pixel/frame and 0.2 pixel/frame respectively. They can be extended to sequences captured at 62 fps and 1000 fps. In this evaluation, datasets B and C contain sequences captured at both 62 fps and 1000 fps with maximum speeds of 1 pixel/ms and 0.2 pixel/ms respectively. In dataset B, the moving speed is accelerated to the maximum moving speed of the above-illustrated FA scene (1 pixel/ms), and a deceleration motion starts to reduce the moving speed to 0 pixel/ms after a uniform moving. The repetition of this series of motions constitutes the reciprocating motion. Dataset C records the reciprocating motion with a lower maximum speed of 0.2 pixel/ms.

3.5.3.2 Related methods

Related methods, as shown in Table 3.6, are used for comparative evaluation. Among them, Method 4 is the proposed method. They are classified according to model type, prediction type, and tracking type. The model type, which can be sequential or parallel, indicates the model used for translation and rotation motion estimation. The prediction type, which can be on or off, indicates the existence of temporal prediction. The tracking type, which can be temporal iterative tracking (TIT) or spatial iterative tracking (SIT), indicates the iterative tracking type. In this evaluation, TIT is the temporal iterative tracking with one spatial iteration performed on 1000 fps sequences, and SIT is the spatial iterative tracking with 16 spatial iterations performed on 62 fps sequences. Based on the above settings, the computational complexity of SIT and TIT methods is comparable.

Table 3.3: Related methods.

Methods	Description		
	Model type	Prediction type	Tracking type
Method 1	Sequential	Off	TIT
Method 2	Sequential	Off	SIT
Method 3	Parallel	Off	TIT
Method 4(∗)	Parallel	On	TIT
Method 5	Parallel	On	SIT
Method 6	Parallel	Off	SIT

TIT: Temporal iterative tracking (1000 fps)

SIT: Spatial iterative tracking (62 fps)

3.5.3.3 Evaluation from the perspective of image processing

In this evaluation, tracking error and tracking lost rate of methods introduced in Sec. 3.5.3.2 are evaluated based on dataset A–C from the perspective of image processing.

The evaluation results for dataset A which records the uniform motions are shown in Fig. 3.26. In terms of tracking error, spatial iterative tracking methods (Method 2, 5, 6) outperform all others, while the proposed method (Method 4) outperforms all other temporal iterative tracking methods (Method 1,3). As shown by the tracking error of temporal iterative methods (Method 1, 3, 4): a) Directly parallel rotation and translation estimation (Method 3) degrades tracking performance when compared to the sequential model (Method 1); b) The tracking errors of temporal iterative methods without temporal prediction (Method 1,3) become more diverse as moving speed increases; c) The proposed method (Method 4) significantly mitigates the negative effects as described in a) b). In terms of tracking lost rate, the proposed method (Method 4) outperforms all others, while the spatial iterative tracking methods have relatively poor tracking ability for high-

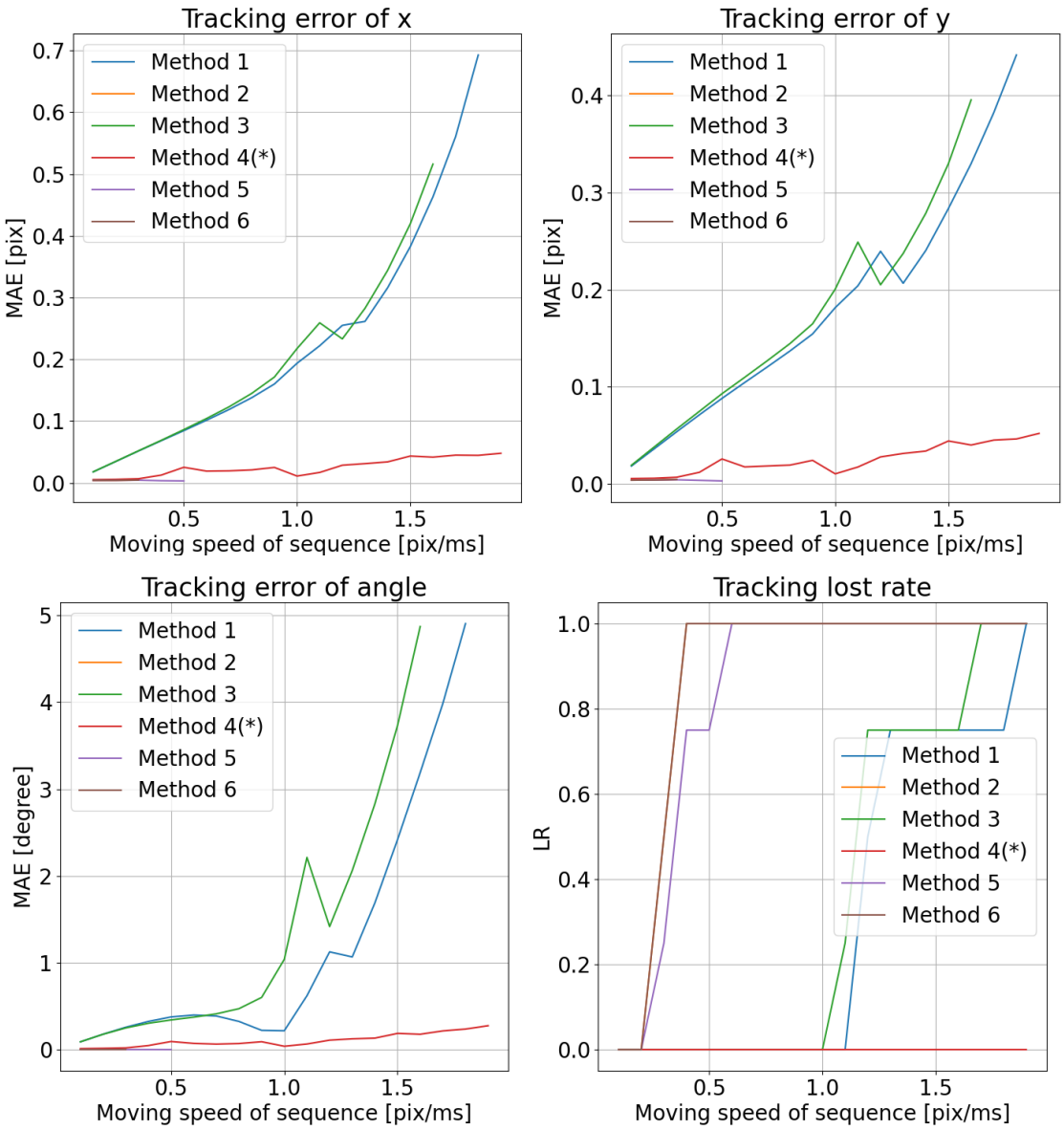

Figure 3.26: Evaluation results from the perspective of image processing based on dataset A.

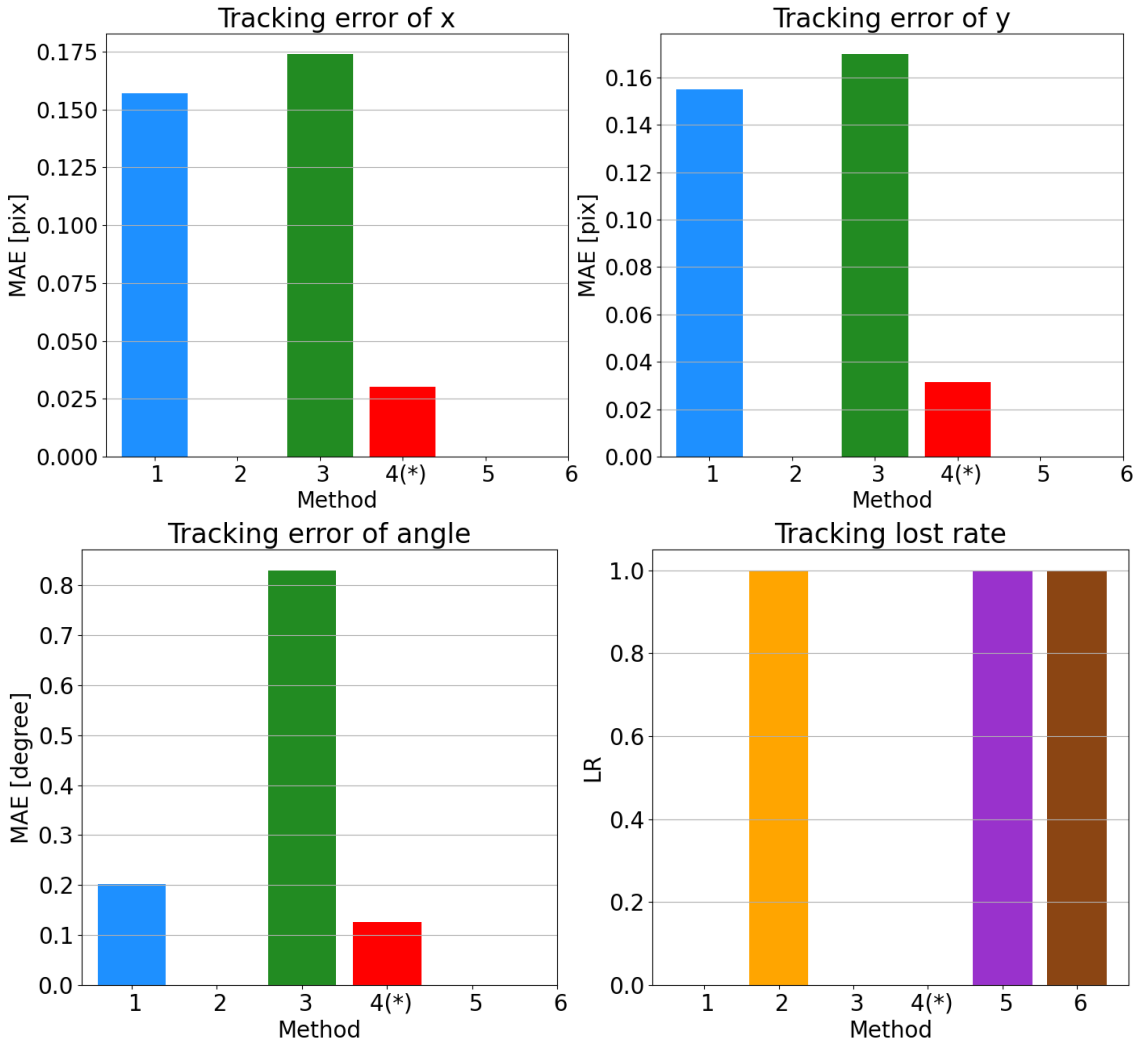

Figure 3.27: Evaluation results from the perspective of image processing based on dataset B.

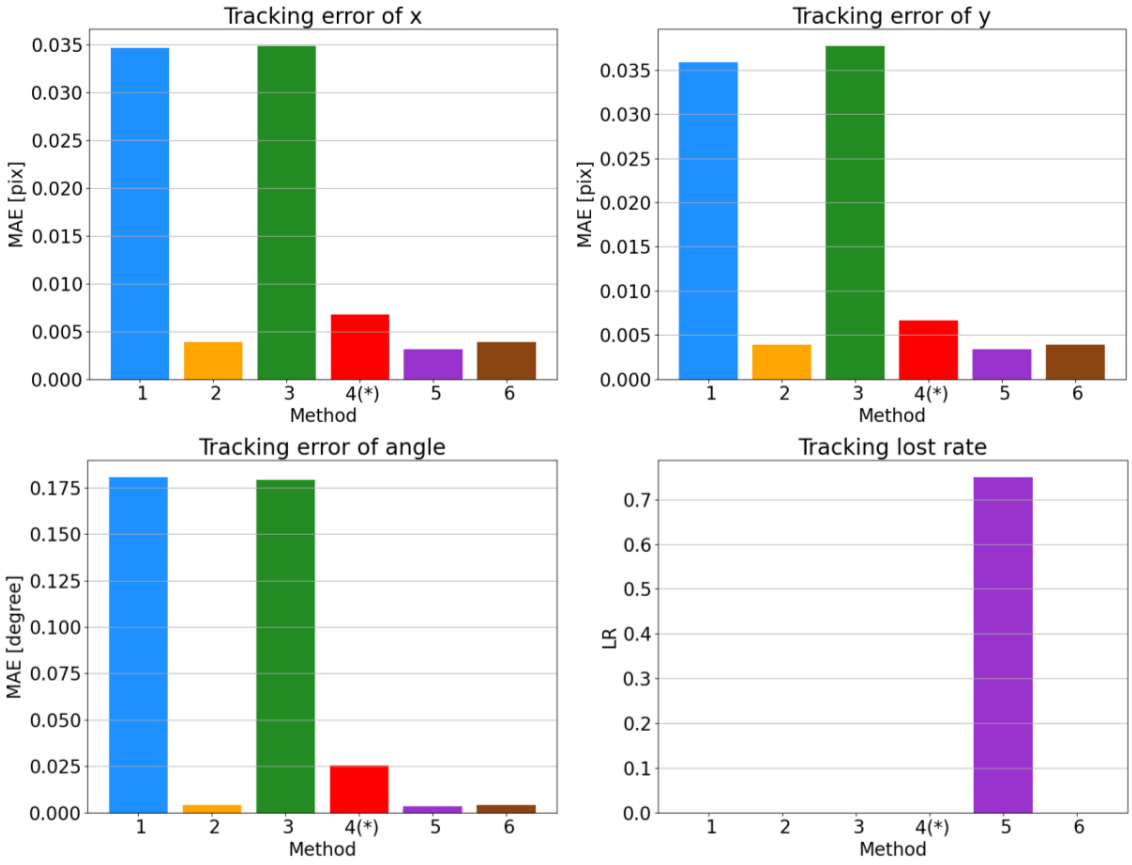

Figure 3.28: Evaluation results from the perspective of image processing based on dataset C.

speed moving cases. As a result, the proposed method (Method 4) strikes a good balance between tracking error and tracking lost rate performance.

Fig. 3.27 and Fig. 3.28 show the evaluation results for datasets B and C, respectively. Similar results are observed in non-uniform motions. In the case of high-speed moving ranges (Fig. 3.27), the proposed method (Method 4) outperforms all others, while the spatial iterative tracking methods (Method 2, 5, 6) fail tracking. In the case of low-speed moving ranges (Fig. 3.28), spatial iterative tracking methods (Method 2, 5, 6) outperform all others without tracking loss, while the proposed method (Method 4) outperforms all other temporal iterative tracking methods (Method 1, 3).

3.5.3.4 Evaluation from the perspective of real-time processing

In this evaluation, tracking errors of methods introduced in Sec. 3.5.3.2 are evaluated based on dataset A–C from the perspective of real-time processing, where real-time tracking error denotes the error between the virtual space ' s output and the true state in the real space.

The evaluation results for dataset A which records the uniform motions with speeds ranging from 0.1 pixel/ms to 1.9 pixel/ms are shown in Fig. 3.29. In terms of tracking error, the proposed method (Method 4) outperforms all others. In other words, the proposed method successfully minimizes the error between the virtual space's output and the true state in real space. In the real-time scene, the object continues to move during the processing period. As a result, the error caused by delay significantly worsens the tracking error of methods without temporal prediction (Method 1, 2, 3, 6), particularly spatial iterative tracking methods (Method 2, 6) with significant delays over 16 ms, despite achieving the highest tracking accuracy when evaluated from the perspective of image processing. As

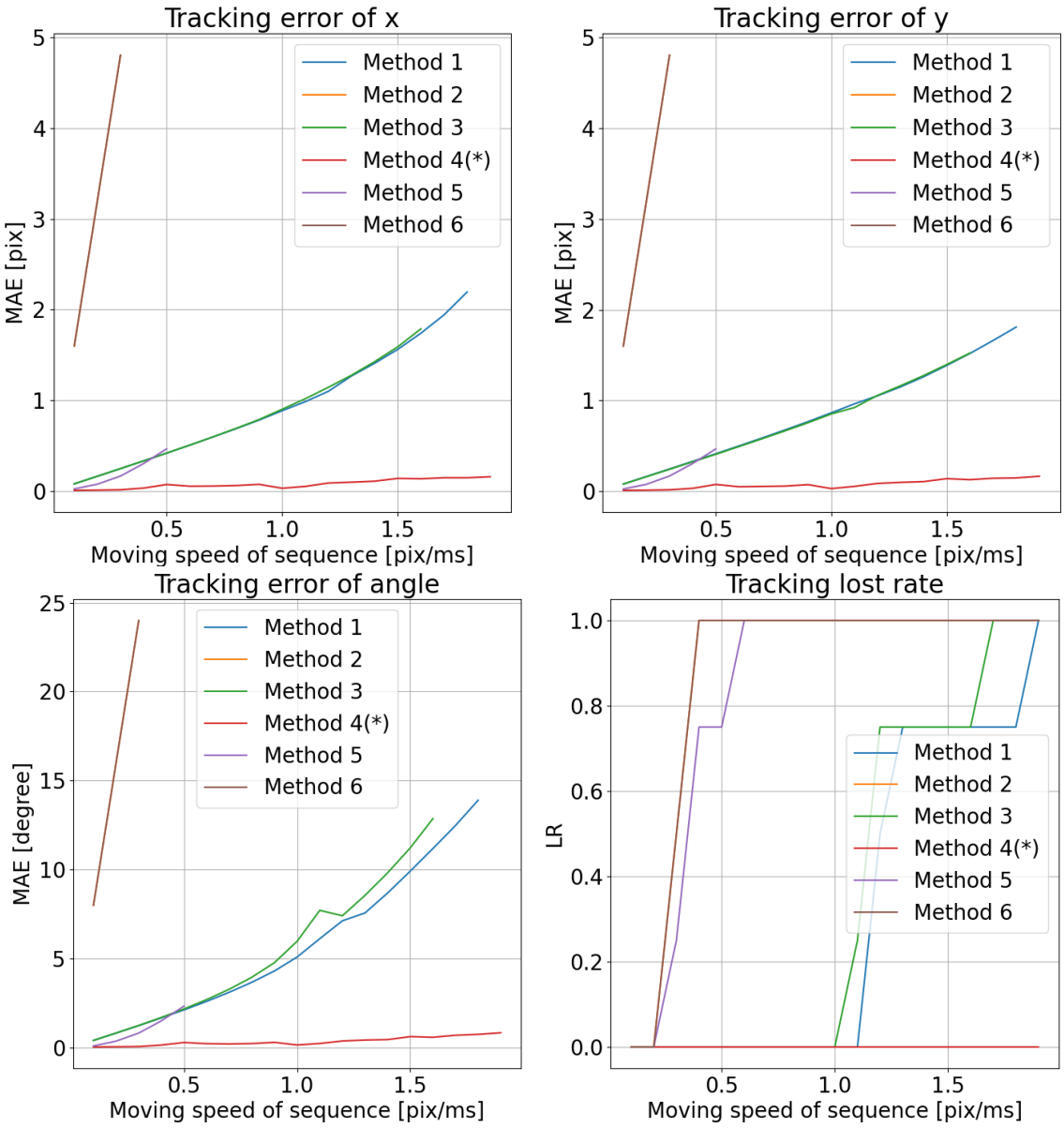

Figure 3.29: Evaluation results from the perspective of real-time processing based on dataset A.

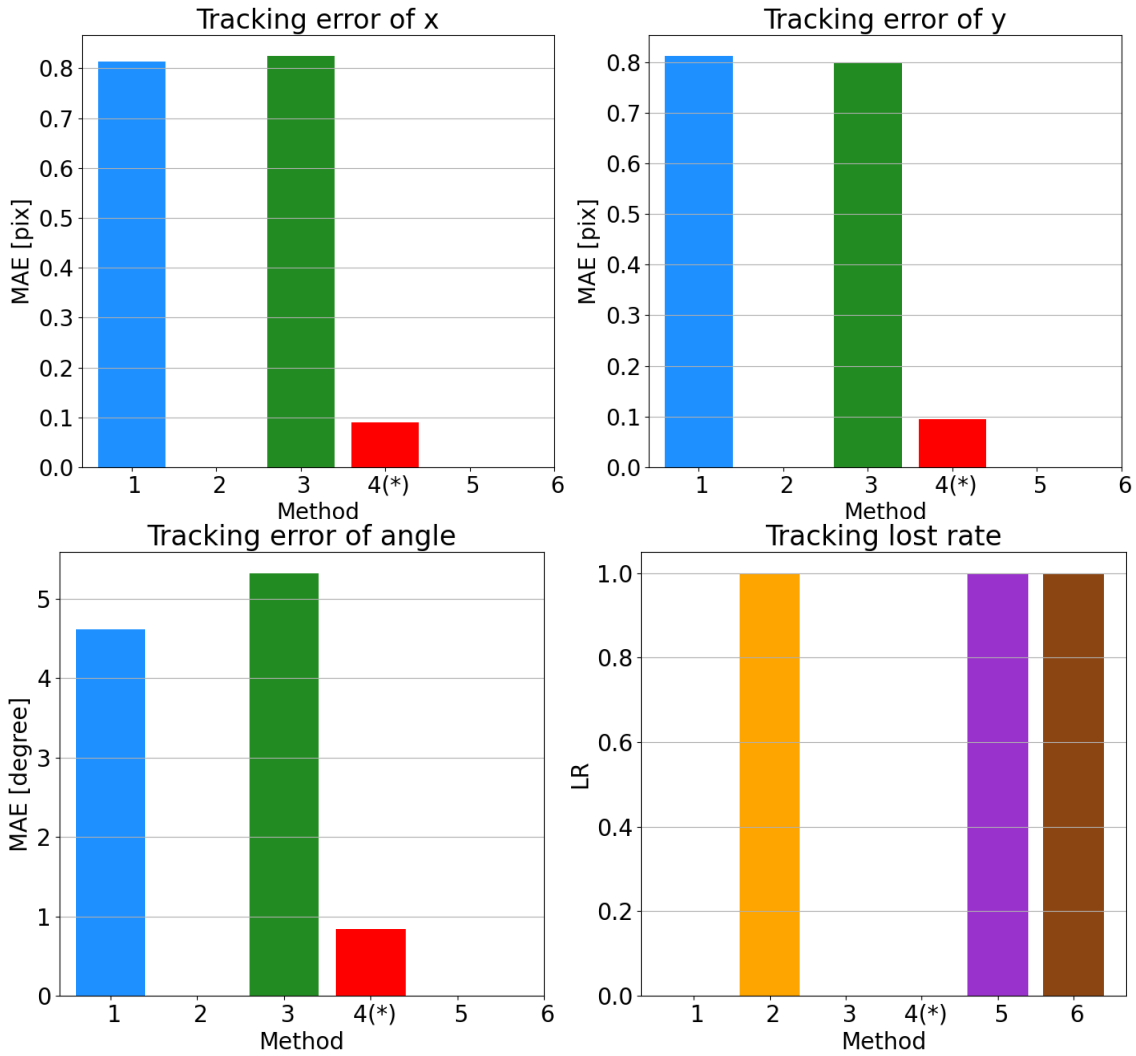

Figure 3.30: Evaluation results from the perspective of real-time processing based on dataset B.

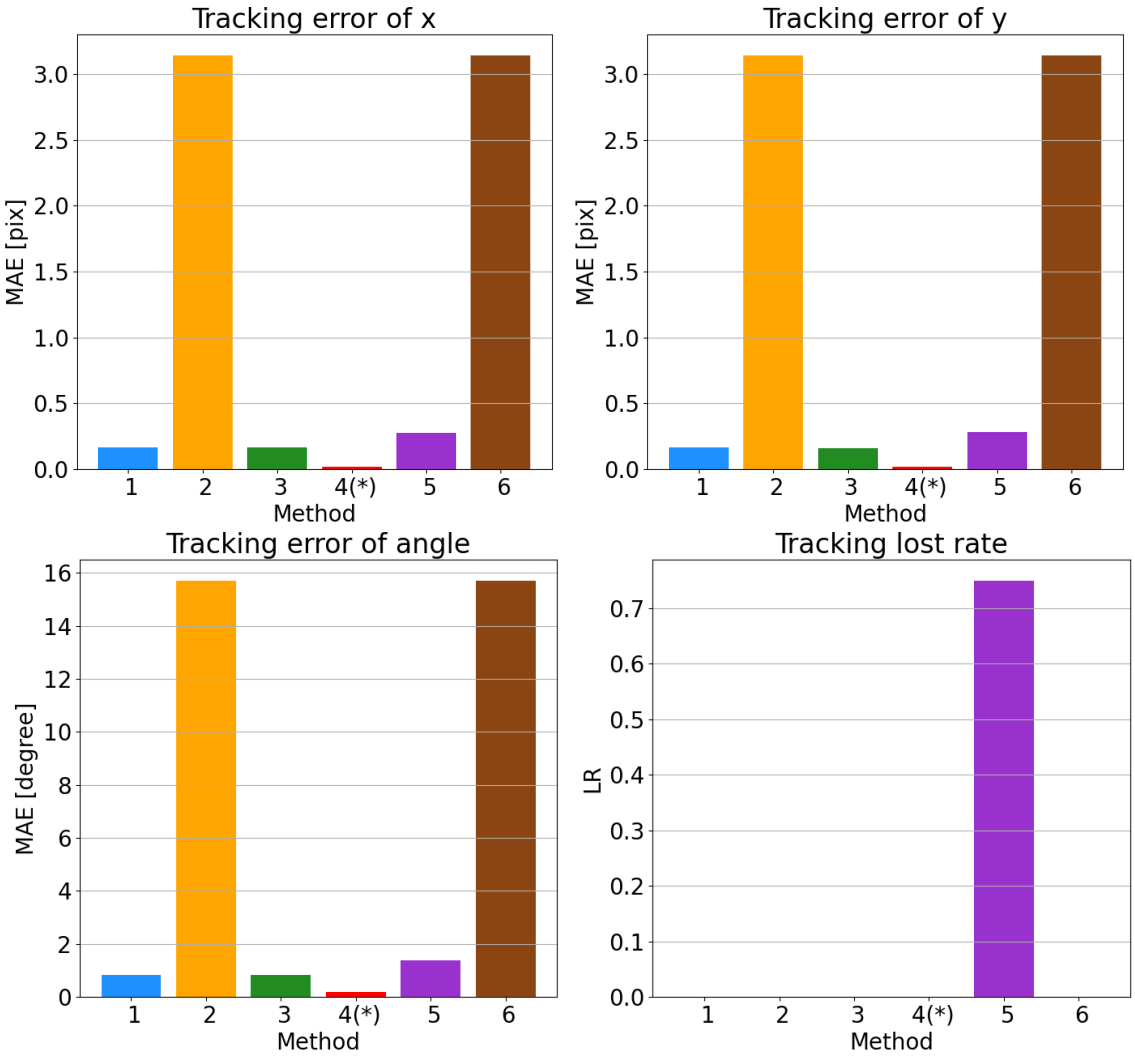

Figure 3.31: Evaluation results from the perspective of real-time processing based on dataset C.

shown by the tracking error of temporal iterative methods (Method 1, 3, 4), temporal prediction (Method 4) suppresses the delay error a lot for temporal iterative tracking. The tracking error of spatial iterative methods (Method 2, 5, 6) shows that temporal prediction (Method 5) helps suppress the delay error, but a large error exists when compared to the temporal prediction-based temporal iterative tracking method, and the error increases with moving speed. In spatial iterative tracking, the large motion changes between two successive frames cause poor speed prediction, resulting in insufficient suppression for delay error. As a result, the proposed method (Method 4) outperforms all other related methods in terms of not only tracking lost rate but also tracking error. It is noteworthy that the proposed method achieves high accuracy over the subpixel level in the evaluated speed range.

Fig. 3.30 and Fig. 3.31 show the evaluation results for datasets B and C, which records non-uniform motion with a maximum speed of 1 pixel/ms and 0.2 pixel/ms respectively. On the one hand, similar results are observed in non-uniform motions. The proposed method outperforms all others in cases of both high-speed moving ranges (Fig. 3.30) and low-speed moving ranges (Fig. 3.31). On the other hand, as illustrated by the tracking lost rate of Fig. 3.31, the temporal prediction degrades the tracking ability of the spatial iterative tracking method (Method 5) when compared to other spatial iterative methods that do not apply temporal prediction (Method 2, 6). The speed dramatically changes when the non-uniform motion is captured by the 60 fps visual system, whereas the current instantaneous speed-based temporal prediction model is incapable of predicting the temporal change in case of dramatic speed change, resulting in a high tracking lost rate.

In the object picking and alignment scenes, where the target moves at about 1 pixel/ms. According to the evaluation result of uniform motion that is shown in Fig. 3.29, the pro-

posed method (Method 4) achieves high accuracy over the subpixel level when the target moves at speed of 1 pixel/ms, which is also the maximum moving speed of the above scene. In the non-uniform moving case, which contains the three motion types (acceleration motion, uniform motion, and deceleration motion) in the processing of object picking and alignment, the proposed method (Method 4) still achieves high tracking accuracy over the subpixel level in the case with a maximum speed of 1 pixel/ms, as shown in Fig. 3.30. As a result, the proposed method (Method 4) is feasible in the specific FA scenes.

3.6 Hardware evaluation

In the hardware evaluation, ZCU102 (ZU9EG), an FPGA board from Xilinx, is used for evaluation. The logic synthesis on FPGA is performed by Vivado 2019.2.

3.6.1 Overall hardware performance

Table 3.4 shows the hardware performance of the proposed tracking system. Row 2 – 6 shows the resource utilization of the whole system. The resource utilization rate of the proposed tracking system is lower than 30% of the entire FPGA resources. Row 7 – 8 shows the speed performance of the proposed tracking system. The whole system works at 200 MHz, and the processing time to perform one frame tracking is around A (time to transmit one frame data) + B (4 lines' delay), which is less than 1 ms/frame. Currently, the tracking system is connected to the high-speed camera, which captures 640 $\times$ 360 sequences at a frame rate of 1000 fps.

Table 3.5 shows the comparison of resource utilization between the translation estimation model [19] and the translation and rotation estimation model. The translation and rotation estimation model significantly increases BRAM and DSP costs in comparison to

Table 3.4: Hardware performance of the proposed ultra-low delay tracking system.

Item		Proposed tracking system
Logic Utilization	# of LUT	54,595(19.92%)
	# LUTRAM	9,561(6.64%)
	# FF	78,493(14.32%)
	# BRAM	247(27.08%)
	# DSP	373(14.80%)
Speed	Frequency	200MHz
	Process time	0.94 (<1ms/frame)

about a 2% increase in resource costs for LUT and FF. When compared to the solution for translation estimation shown in Eq. (3.2) (3.3), the solution for rotation estimation shown in Eq. (3.6) (3.7) contains more complex computation, resulting in a significant increase in DSP. Meanwhile, as shown in Fig. 3.10, four-by-four BRAM sets are used to obtain the addresses of rotated template data while maintaining throughput, resulting in a significant increase in BRAM. BRAM utilization can be reduced through more efficient BRAM access methods. Overall, the translation and rotation estimation model has reasonable resource utilization.

The detailed timing flow is shown in Fig. 3.32. The proposed system achieves ultra-low delay by performing visual processing while receiving the image data from the camera. To maintain this stream-based processing style, the overall processing is designed to be fully pipelined. The processing time to perform one frame tracking is around A (time to transmit one frame data, which is 925.2 us) + B (delay of the image processing modules, which is 10.99 us), which is less than 1 ms/frame. Among these image processing modules, the gradient filter, which consumes over a three-line delay, takes up a huge por-

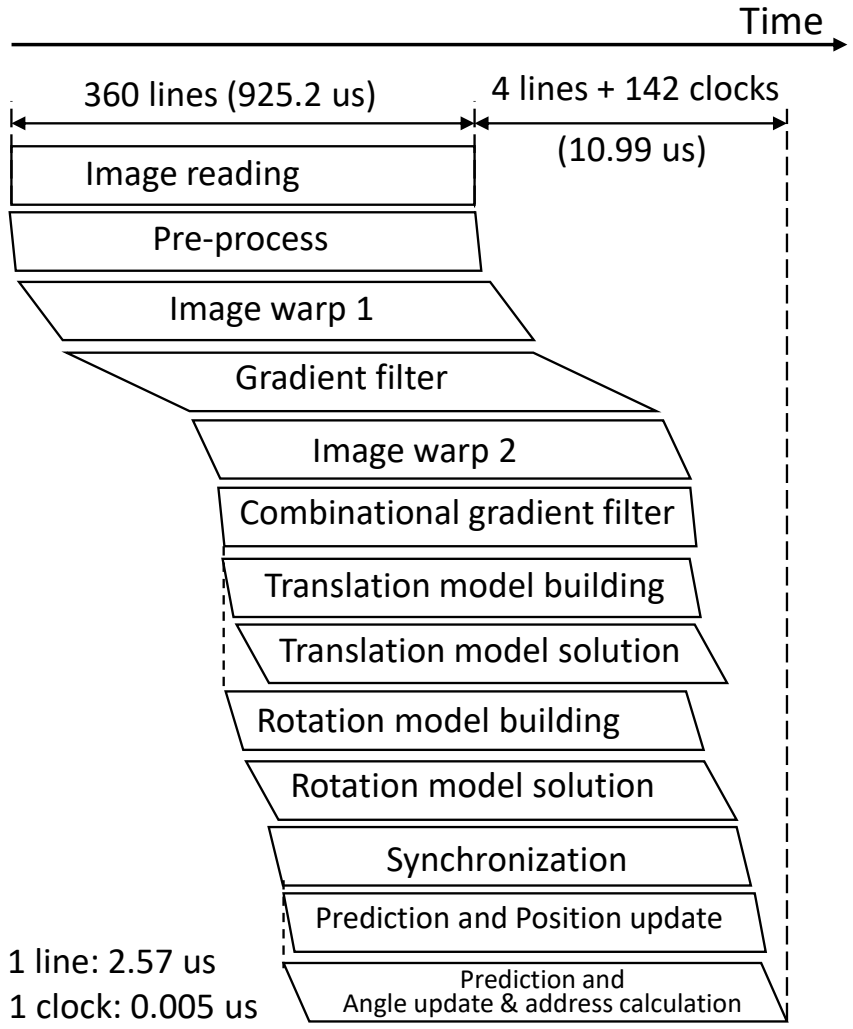

Figure 3.32: Detailed timing flow of the whole proposed system.

tion of the overall processing time. Translation estimation, which consists of translation model building and translation model solution, and rotation estimation, which consists of rotation model building and rotation model solution, are performed in parallel in the proposed system with a delay of 51 clocks (0.255 us). When compared to the conventional sequential processing method, which takes a delay over 40 lines (102.8 us) in the case of rotation estimation in a 65 $\times$ 65 local area and translation estimation in a 15 $\times$ 15 local

area, the proposed temporal prediction-based parallel motion estimation significantly reduces the delay caused by the dependent relationship between the rotation and translation estimations.

Table 3.5: Resource utilization comparision.

Item	Translation estimation model [19]	Translation and rotation estimation model
# of LUT	49,188(17.95%)	54,595(19.92%)
# FF	67,422(12.30%)	78,493(14.32%)
# BRAM	90.5(9.92%)	247(27.08%)
# DSP	103(4.09%)	373(14.80%)

3.6.2 Evaluation of label scanner-based multi-stream spatial processing

Table 3.6: Hardware performance comparision.

Item		Sliding window type[18]	Label scanner type
Parameter	Template size	5×5	free size
Logic Utilization	# of LUT	177,172	49,188
	# FF	106,800	67,422

Table 3.6 shows the comparison of hardware performance between the sliding window-based architecture [18] and the proposed label scanner-based architecture based on the translation estimation model. The label scanner type maintains a certain resource consumption no matter how the size of the template changes, while the resource utilization of

the sliding window type will increase a lot with the increase of the template size. Row 3 – 4 shows the comparison of resource utilization. As it can be seen that the sliding window type requires much more resources than the label scanner type even when tracking a small template with a size of 5×5.

3.7 Conclusion

An ultra-low delay rotation-robust tracking system with high real-time tracking performance has been proposed in this chapter. Two methods are proposed to implement this system. Firstly, to reduce complex computation caused by spatial iterative tracking and error caused by delay, temporal prediction-based temporal iterative tracking is proposed for mapping the spatial iterative tracking into the temporal domain and reducing the difference between the system output state and true state. Secondly, to reduce significant delay and resource consumption caused by sequential estimation of translation and rotation, temporal prediction-based parallel motion estimation is proposed for paralleling estimation of translation and rotation. The proposed methods are implemented on FPGA with a stream-based architecture. It is prototyped as a practical system by combining a high-speed camera and an FPGA. Algorithm evaluation shows that the proposed method achieves subpixel-level real-time accuracy while the conventional method fails to track the target in case of tracking the object moving at a speed of 1 pixel/ms. Hardware evaluation shows that the designed tracking system supports sensing and processing 1000 fps sequence (Resolution: 640×360) with a delay of less than 0.94 ms/frame, and costing resources less than 30%.

Our future research will dive more into the following aspects. In the technical as-

pect, a more practical prediction model that is adapted for more complex cases is added into consideration. In the application aspect, a combination of ultra-low delay tracking systems and actuators is added into consideration.

Chapter 4

Conclusion and future work

4.1 Summary

This dissertation targets minimizing the difference between the object tracking system's output and the real space's true state through ultra-low delay processing. Temporal mapping and temporal prediction are proposed and combined to achieve detection and tracking with a delay of less than 1 ms/frame.

The proposed temporal mapping and temporal prediction make significant contributions to real-time computing. Firstly, the combination of temporal mapping and temporal prediction contributes significantly to the design of the 1-ms algorithm and architecture with an input of 1000 fps sequence (Resolution 640×360). Temporal mapping maps the complex spatial processing into the temporal domain, providing a solution for complex processing to adapt the ultra-low delay processing architecture. Temporal prediction predicts the change during the processing period, enabling the temporal mapping to be applied without large accuracy loss. Secondly, temporal prediction compensates for the change during the processing period, significantly reducing the difference between the computing space's output and real space's true state, thus enabling high real-time tracking accuracy over the subpixel level.

Real-time computing with ultra-low delay enables the high integration of virtual (computing) and real spaces, making it possible to handle changing situations. In the field of FA, ultra-low delay real-time computing allows industrial robots to actuate without halting, significantly improving productivity. In the field of robotics, it allows daily-life robots to complete the challenging task that needs immediate reactions.

4.2 Future work

Among various visual processing techniques, this dissertation targets 2D object tracking, which is sufficient for FA applications that have a fixed depth. However, for applications that don't have a fixed depth and need to estimate depth in real-time, such as robotics, 3D calculation technology becomes crucial. Therefore, future work will dive more into the ultra-low delay 3D calculation.

Bibliography

[1] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Key Points,” *Internatinal Journal of Computer Vision*, vol.60, pp.91–110, 2004.

[2] D. H. Ballard, “Generalizing the Hough transform to detect arbitrary shapes,” *Pattern recognition*, vol.13, no. 2, pp.111–122, 1981.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *IEEE conference on computer vision and pattern recognition*, pp.779–788, 2016.

[4] B. D. Lucas, and T. Kanade, “An iterative image registration technique with an application to stereo vision,” *International joint conference on Artificial intelligence*, vol.2, pp.674–679, 1981.

[5] K. Nummiaro, E. Koller-Meier, and L. Van Gool, “Object tracking with an adaptive color-based particle filter,” *Joint Pattern Recognition Symposium*, pp.353–360, 2002.

[6] K. A. Acharya, R. Venkatesh Babu, and S. S. Vadhiyar,“A real-time implementation of SIFT using GPU,” *Journal of Real-Time Image Processing*, vol. 14, no. 2, pp. 267-277, 2018.

[7] R. Huang, J. Pedoeem and C. Chen,"YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers," *IEEE International Conference on Big Data*, pp. 2503-2510, 2018.

[8] Z. Ouyang, J. Niu, Y. Liu, and M. Guizani,"Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles," *IEEE Transactions on Mobile Computing*, vol. 19, no. 2, pp. 300-313, 2020.

[9] I. Gurcan, and A. Temizel,"Heterogeneous CPU-GPU tracking-learning-detection (H-TLD) for real-time object tracking," *Journal of Real-Time Image Processing*, vol. 16, no. 2, pp. 339-353, 2019.

[10] M. Ishikawa, A. Morita, N. Takayanagi, "High Speed Vision System Using Massively Parallel Processing," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.373–377, 1992.

[11] I. Ishii, Y. Nakabo, M. Ishikawa, "Target tracking algorithm for 1 ms visual feedback system using massively parallel processing," *IEEE International Conference on Robotics and Automation*, vol.3, pp.2309–2314, 1996.

[12] Y. Watanabe, T. Komuro, S. Kagami, and M. Ishikawa, "Multi-target tracking using a vision chip and its applications to real-time visual measurement," *Journal of Robotics and Mechatronics*, vol.17, no.2, pp.121–129, 2005.

[13] M. Ishikawa, "High-Speed Vision and its Applications Toward High-Speed Intelligent Systems," *Journal of Robotics and Mechatronics*, vol.34, no.5, pp.912–935, 2022.

[14] T. Hu, and T. Ikenaga, "Pixel selection and intensity directed symmetry for high frame rate and ultra-low delay matching system," *IEICE Transactions on Information and Systems*, vol.E101, no.5, pp.1260–1269, 2018.

[15] P. Zhang, T. Hu, D. Luo, S. Du, and T. Ikenaga, "Highly-Parallel Hardwired Deep Convolutional Neural Network for 1-ms Dual-Hand Tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[16] Y. Xu, T. Hu, S. Du, and T. Ikenaga, "Partial Descriptor Update and Isolated Point Avoidance Based Template Update for High Frame Rate and Ultra-Low Delay Deformation Matching," *International Conference on Pattern Recognition*, pp.1827–1832, 2018.

[17] S. Du, Y. Li, and T. Ikenaga, "Temporally Forward Nonlinear Scale Space for High Frame Rate and Ultra-Low Delay A-KAZE Matching System," *IEICE Transactions on Information and Systems*, vol.E103-D, No.6, pp.1226–1235, 2020.

[18] T. Hu, H. Wu, and T. Ikenaga, "FPGA Implementation of High Frame Rate and Ultra-Low Delay Tracking with Local-Search Based Block Matching," *International Conference on Machine Vision and Information Technology*, pp.93–98, 2017.

[19] T. Hu, R. Fuchikami, and T. Ikenaga, "High temporal resolution-based temporal iterative tracking for high framerate and ultra-low delay dynamic tracking system," *IEICE Transactions on Information and Systems*, vol.E105-D, No.5, pp.1064–1074, 2022.

[20] S. Du, K. Gu, and T. Ikenaga, "Subpixel Displacement Measurement at 784 FPS: From Algorithm to Hardware System," *IEEE Transactions on Instrumentation and Measurement*, vol.71, pp.1–10, 2022.

[21] S. Du, Z. Dong, Y. Li, and T. Ikenaga, "Straight-Line Detection within 1 Millisecond per Frame for Ultra-High-Speed Industrial Automation," *IEEE Transactions on Industrial Informatics*, published online, 2022.

[22] Z. Dong, and B. Lin, "Learning a robust CNN-based rotation insensitive model for ship detection in VHR remote sensing images," *International Journal of Remote Sensing*, vol.41, no.9, pp.3614–3626, 2020.

[23] P. Piccinini, A. Prati, and R. Cucchiara, "Real-time object detection and localization with SIFT-based clustering," *Image and Vision Computing*, vol. 30, No. 8, pp.573–587, 2012.

[24] Y. Guo, H. Zeng, Z.-C. Mu, and F. Zhang, "Rotation-invariant DAISY descriptor for keypoint matching and its application in 3D reconstruction," *International Conference on Signal Processing*, pp.1198–1201, 2010.

[25] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol.31, no.5, pp.1147–1163, 2015.

[26] E. Rosten, R. Porter, and T. Drummond, "Faster and better: a machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.32, no.1, pp.105–119, 2010.

[27] C. Harris, and M. Stephens, "A combined corner and edge detector," *Alvey Vision Conference*, pp.147–151, 1988.

[28] J. Heinly, E. Dunn, and J. M. Frahm, "Comparative evaluation of binary features," *European Conference on Computer Vision*, pp.759–773, 2012.

[29] Y. Watanabe, G. Narita, S. Tatsuno, T. Yuasa, K. Sumino, M. Ishikawa, "High-speed 8-bit Image Projector at 1000 fps with 3 ms Delay," *The International Display Workshops*, 2015.

[30] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol.110, pp.346–359, 2008.

[31] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," *IEEE International Conference on Computer Vision*, 2011.

[32] S. Leutenegger, M. Chli, R. Y. Siegwart, "BRISK: Binary Robust Invariant Scalable Keypoints," *IEEE International Conference on Computer Vision*, 2011.

[33] E. Rosten, R. Porter, T. Drummond, "FASTER and Better: A Machine Learning Approach to Corner Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.32, pp.105–119, 2010.

[34] C. Harris, M. Stephens, "A Combined Corner and Edge Detector," *Alvey Vision Conference*, 1988.

[35] J. Heinly, E. Dunn, J. M. Frahm, "Comparative Evaluation of Binary Features," *European Conference on Computer Vision*, 2012.

[36] M. Calonder, V. Lepetit, C. Strecha, P. Fua, "BRIEF: Binary Robust Independent Elementary Features," *European Conference on Computer Vision*, 2010.

[37] Y. Ke, R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," *International Conference on Computer Vision and Pattern Recognition*, 2004.

[38] Y. Ke, R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," *International Conference on Computer Vision and Pattern Recognition*, 2004.

[39] A. Alahi, R. Ortiz, P. Vandergheynst, "FREAK: Fast Retina Key-Point," *International Conference on Computer Vision and Pattern Recognition*, 2012.

[40] T. Suzuki, T. Ikenaga, "Low complexity keypoint extraction based on SIFT descriptor and its hardware implementation for Full-HD 60fps video," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol.96, pp.1376–1383, 2013.

[41] T. Rao, T. Ikenaga, "Quadrant segmentation and ring-like searching based FPGA implementation of ORB matching system for Full-HD video," *IAPR International Conference on Machine Vision Applications*, pp.76–79, 2017.

[42] J. Weberrus, L. Kleeman, T. Drummond, "ORB feature extraction and matching in hardware," *Australasian Conference on Robotics and Automation*, pp.2–4, 2015.

[43] H. Heo, J. Lee, K. Lee, C. Lee, "FPGA based implementation of FAST and BRIEF

algorithm for object recognition," *IEEE International Conference of IEEE Region 10*, pp.1–4, 2013.

[44] P. L. Rosin, "Measuing corner properties," *Computer Vision and Image Understanding*, vol.73, no.2, pp. 291–307, 1999.

[45] T. Hu, H. Wu, T. Ikenaga, "FPGA implementation of high frame rate and ultra-low delay tracking with local-search based block matching," *International Conference on Machine Vision and Inforamation Technology*, pp.93–98, 2017.

[46] K. Mikolajczyk, T. Tuytelaars, C. Schimid, "A comparison of affine region detectors," *International Journal of Computer Vision*, vol.65, pp.43–72, 2005.

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

[48] K. Briechle, and U. D. Hanebeck, "Template matching using fast normalized cross correlation," *Optical Pattern Recognition XII*, vol.4387, pp.95–102, 2001.

[49] A. Averbuch, and Y. Keller, "FFT based image registration," *IEEE International Conference on Acoustics, Speech,and Signal Processing*, vol.4, pp.IV–3608, 2002.

[50] B. S. Reddy, and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," *IEEE transactions on image processing*, vol.5, no.8, pp.1266–1271, 1996.

[51] M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, "A tutorial on particle filters for online nonlinear / non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol.50, no.2, pp.174–188, 2002.

[52] B. K. Horn, and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol.17, no.1, pp.185–203, 1981.

[53] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International journal of computer vision*, vol.12, no.1, pp.43–77, 1994.

[54] J. Shi, and C. Tomasi, "Good features to track," *IEEE conference on computer vision and pattern recognition*, pp.593–600, 1994.

[55] T. Fukao, and T. Kanade, "Two step algorithm for point feature tracking," *IPSJ SIG Notes. CVIM*, vol.2003, no.109, pp.103–110, 2003.

[56] R. Allaoui, H. H. Mouane, Z. Asrih, S. Mars, and I. EI Hajjouji, "FPGA-based implementation of optical flow algorithm," *International Conference on Electrical and Information Technologies*, pp.1–5, 2017.

[57] V. Mahalingam, K. Bhattacharya, N. Ranganathan, H. Chakravarthula, R. R. Murphy, and K. S. Pratt, "A VLSI Architecture and Algorithm for Lucas-Kanade-Based Optical Flow Computation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.18, no.1, pp.29–38, 2010.

[58] F. Barranco, M. Tomasi, J. Diaz, M. Vanegas, and E. Ros, "Parallel architecture for hierarchical optical flow estimation based on FPGA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.20, no.6, pp.1058–1067, 2012.

[59] J. Díaz, E. Ros, F. Pelayo, E. M. Ortigosa, and S. Mota, "FPGA-based real-time optical-flow system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.16, no.2, pp.274–279, 2006.

[60] H. -S. Seong, C. E. Rhee and H. -J. Lee, "A Novel Hardware Architecture of the Lucas-Kanade Optical Flow for Reduced Frame Memory Access," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.26, no.6, pp.1187–1199, 2016.

[61] A. Arif., F. A. Barrigon, F. Gregoretti, J. Iqbal, L. Lavagno, M. T. Lazarescu, L. Ma, M. Palomino, and J. L. L. Segura, "Performance and energy-efficient implementation of a smart city application on FPGAs," *Journal of Real-Time Image Processing*, vol.17, no.3, pp.729–743, 2020.

[62] Z. Chai, and J. Shi: "Improving KLT in Embedded Systems by Processing Oversampling Video Sequence in Real-Time," *International Conference on Reconfigurable Computing and FPGAs*, pp.297–302, 2011.

[63] T. Nam., S. Kim, and D. Jung, "Hardware implementation of KLT tracker for real-time intruder detection and tracking using on-board camera," *International Journal of Aeronautical and Space Sciences*, vol.20, pp.300–314, 2019.

[64] Q. He, W. Chen., D. Zou., and Z. Chai, "A novel framework for UAV returning based on FPGA," *The Journal of Supercomputing*, vol.77, pp.4294–4316, 2021.

Publications

Journals

[1] **T. Hu**, and T. Ikenaga, "Pixel selection and intensity directed symmetry for high frame rate and ultra-low delay matching system," *IEICE TRANSACTIONS on Information and Systems*, vol. 101, no. 5, pp. 1260–1269, 2018.

[2] S. Du, **T. Hu**, and T. Ikenaga, "Adaptive-Partial Template Update with Center-Shifting Recovery for High Frame Rate and Ultra-Low Delay Deformation Matching," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 102, no. 12, pp. 1872–1881, 2019.

[3] P. Zhang, **T. Hu**, D. Luo, S. Du, and T. Ikenaga, "Highly-parallel hardwired deep convolutional neural network for 1-ms dual-hand tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 12, pp. 8192–8203, 2021.

[4] S. Du, P. Cai, **T. Hu**, and T. Ikenaga, "Automatic Foreground Detection at 784 FPS for Ultra-High-Speed Human-Machine Interactions," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3587–3600, 2021.

[5] **T. Hu**, R. Fuchikami, and T. Ikenaga, "High Temporal Resolution-Based Temporal Iterative Tracking for High Framerate and Ultra-Low Delay Dynamic Tracking System," *IEICE Transactions on Information and Systems*, vol. 105, no. 5, pp. 1064–1074, 2022.

[6] **T. Hu**, R. Fuchikami, and T. Ikenaga, "Temporal prediction-based temporal iterative tracking and parallel motion estimation for 1 ms rotation-robust LK-based tracking system," *IEEE Transactions on Instrumentation and Measurement*, 2023. (To be published).

International Conferences

[1] **T. Hu**, H. Wu, and T. Ikenaga, "FPGA Implementation of High Frame Rate and Ultra-Low Delay Tracking with Local-Search Based Block Matching," *International Conference on Machine Vision and Information (CMVIT)*, pp. 93–98, 2017.

[2] **T. Hu**, and T. Ikenaga, "FPGA Implementation of High Frame Rate and Ultra-Low Delay Vision System with Local and Global Parallel based Matching," *IAPR International Conference on Machine Vision and Applications (MVA)*, pp. 286–289, 2017.

[3] Y. Xu, **T. Hu**, S. Du, and T. Ikenaga, "Partial descriptor update and isolated point avoidance based template update for high frame rate and ultra-low delay deformation matching," *International Conference on Pattern Recognition (ICPR)*, pp. 1827–1832, 2018.

[4] J. Zhang, D. Huang, **T. Hu**, R. Fuchikami, and T. Ikenaga, "Critically Compressed Quantized Convolution Neural Network based High Frame Rate and Ultra-Low Delay Fruit External Defects Detection," *International Conference on Machine Vision and Applications (MVA)* , pp. 1–4, 2021.

[5] Z. Dong, **T. Hu**, R. Fuchikami, and T. Ikenaga, "Encoding-free Incrementing Hough Transform for High Frame Rate and Ultra-low Delay Straight-line Detection," *International Conference on Machine Vision and Applications (MVA)* , pp. 1–4, 2021.

[6] D. Huang, J. Zhang, **T. Hu**, R. Fuchikami, and T. Ikenaga, "Contextual Information based Network with High-Frequency Feature Fusion for High Frame Rate and Ultra-Low Delay Small-Scale Object Detection," *International Conference on Machine Vision and Applications (MVA)* , pp. 1–5, 2021.

[7] W. Yang, Y. Li, **T. Hu**, R. Fuchikami, and T. Ikenaga, "Relative Vectors Clustering and Temporal Constraint based Generalized Hough Transform for High Frame Rate and Ultra-low Delay Arbitrary Shape Detection," *International Conference on Computer Vision and Information Technology (CVIT)*, 2022.

[8] X. Zha, Y. Li, **T. Hu**, R. Fuchikami, and T. Ikenaga, "Multi-line Buffer Based Pipeline Architecture with Junction Connectivity Analysis for High Frame Rate and Ultra-Low Delay Contour-Based Corner Detection," *International Conference on Computer Vision and Information Technology (CVIT)*, 2022.