A Real-Time Object Tracking System With On-Line Feature Learning

(オンライン形状学習機能を備えた実時間物体追跡システム)

趙普社

Supervisor: Professor Tadashi SHIBATA

Department of Electrical Engineering,

The University of Tokyo

Abstract

Object tracking plays an important role in many applications, such as video surveillance, human-computer interaction, vehicle navigation, and robot control. It is generally defined as a task of estimating the location of an object over a sequence of images. In practical applications, there are many factors that make the task complex such as illumination variation, appearance change, shape deformation, partial occlusion, and camera motion. Moreover, lots of these applications require real-time response. Therefore, the development of real-time working algorithms is of essential importance. In order to accomplish such a challenging task, a real-time tracking system has been developed and proposed in this thesis.

In this thesis, a solution to the tracking task is proposed based on consideration of efficient implementation as priority, and several critical issues are resolved as follows. At first, a hardware-friendly tracking framework is designed, which is implemented on field-programmable gate array (FPGA) technology, and compatible with very large scale integration (VLSI) technology. This framework, named multiple candidate regeneration (MCR), is developed as a simple but high-speed and high-efficiency searching algorithm. The basic idea was inherited from the particle filter (PF) but the algorithm has been greatly modified from the original particle filter so that it can be implemented on VLSI hardware very efficiently. The important difference between MCR and PF is that the MCR is developed by simplifying the visual tracking task and considering the simple hardware implementation. It can be considered as an efficient searching strategy instead of an

intensity estimation method. In the development, several problems that may limit the hardware performance have been solved, such as complex computation, data transmission and utilization of hardware resources. The proposed architecture achieved 150 frame per second (f/s) on FPGA, and can reach about 900 f/s if it is implemented on VLSI with on-chip image sensor. This solution has several advantages. First, it works at high frame rate, which can enhance the effect of localization. It also meets the requirement of higher processing speed in some complex intelligence systems, which seems difficult to achieve by conventional solutions. Second, the system can be extended to be useful in many applications because of its flexibility. Third, since the processing speed is faster than the frame rate, there is still large space for further improving the ability of the system without losing real-time performance. The system was implemented on a Terasic DE3 FPGA board. Under the operating frequency of 60 MHz, the experimental system achieved a processing ability of 0.8 ms per frame in tracking a 64×64 scale object image in 640×480-pixel size video sequences.

In tracking algorithms, how to represent the target image is of particular importance because it greatly influences the tracking performance under certain tracking framework. Color, edge, and texture are typical attributes used for representing objects. A number of other features, including active contour, scale-invariant feature transform (SIFT) feature, oriented energy, and optical flow, are also used in many works. Some works also combine these features or incorporate on-line learning of the model of an object and background. In this thesis, we have aimed to establish both robustness of object representation and the real-time performance of the processing, because feature extraction is usually a time consuming process. It was well known that the visual perception of animals relies heavily on the directional edges. In the present work, therefore, the directional-edge-based image feature representation algorithm is employed to represent the object image.

Robust performance of the directional-edge-based algorithms has already been demonstrated in various image recognition applications. In addition, dedicated VLSI chips for efficient directional edge detection and image vector generation have also been developed for object recognition systems.

Whether a tracking system can be easily extended for various purposes is also a critical issue. This thesis contains a detailed discussion on extending the function of the system, including hardware implementation on VLSI, multiple-object tracking, full-occlusion and initialization problems, and employing of state vector. The architecture of this system is compatible with VLSI design, and may reach better performance on VLSI. For the multiple-object tracking, an efficient method is proposed to allocate the limited hardware resources. For the full-occlusion and initialization problems, a searching algorithm based on proposed system is developed. By using the state vector, more attributes can be estimated for achieving more information about the object, which also helps deal with appearance change and increase the tracking accuracy.

The following parts of the thesis are focused on building learning ability for the system. For object tracking, one promising direction is to consider the object tracking as a binary classification problem, and employ discriminative methods in the tracking framework. Nearest neighbor (NN) classifier is a simple but widely used classifier. Some tracking algorithms have tried to use it because of its effectiveness in some tasks and its outstanding simplicity. Support vector machine (SVM), as a powerful classification scheme, has been also used in many tracking algorithms, benefiting the algorithms with accurate localization and flexible modeling of the target. Each of the classifier works as an appearance model of the target by changing its templates while training. For NN classifier, the training and testing process is really

simple and fit for hardware implementation. For the SVM classifier, one feature is that the boundary is represented by the combination of support vectors, and the number of support vectors is usually a small portion of the total training dataset. This feature becomes very important when implementing the tracking algorithm on hardware, because the hardware resources are always limited.

The SVM-based tracking system proposed in this thesis aims to solve the following problems. Some work builds a superior SVM classifier and gives good results in tracking vehicles. However, the off-line training mechanism employed in the work requires a large number of training samples selected manually and does not support updating the training samples. In some research, all samples learned from each frame of an image sequence are stored for training the SVM. This causes a large memory cost if it is used in a long-duration task. In some work, a simple strategy is employed to determine new training samples, which may cause "drift problem". Moreover, these algorithms do not consider their real-time performances, which is in fact of great importance in object tracking applications. This is mainly because of the complex computation of SVM. Especially for the on-line learning SVMs, frequently repeated training and predictions make this problem even worse. Therefore, in order to extend the power of SVM in most of the general tracking applications, it is necessary to develop a proper tracking framework and a VLSI hardware-implementation friendly structure for the SVM-based algorithm.

A real-time visual tracking algorithm is presented employing an on-line support vector machine scheme. In this system, a novel training framework is proposed, which enables the system to select reliable training samples from the image sequence. The tracking framework includes how to update training samples and how to select test samples and make prediction of the target location. Different from other

algorithms, this framework gives a rule guiding the selection of target training samples. When the target changes its appearance significantly, the system may fail to localize the target because the classifier misclassifies the target image to the background image category. In order to solve this problem, background samples are utilized to predict the location of the target image. Unlike the moving target image, most of the background sample images are stable. As a result, high-accuracy tracking has been established. In addition, regarding the selection of target samples for on-line training of SVM, a new selection method has been introduced.

The on-line SVM learning requires repeated training and predicting process. The predicting process always contains computation of thousands of test samples in conventional algorithms, preventing these algorithms from working in real-time. In this process, not only the SVM, but also the feature extraction of each sample will cost lots of time. Based on a SVM chip developed in our group, the most complex part in this algorithm can be computed efficiently. At the same time, multiple candidate regeneration is employed to reduce the computational cost without sacrificing the tracking accuracy. In addition, the directional-edge-feature vector representation, whose VLSI implementation has been proposed, is employed to represent the sample images. The algorithm has been evaluated on challenging video sequences and showed robust tracking ability with accurate tracking results. The hardware implementation is also discussed, while verification has been done to prove the real-time ability of this algorithm.

After development of the SVM-based tracking system, the essential facts of the tracking task were analyzed, and an NN-based tracking system has been proposed. The basic idea is that a classifier specially designed for tracking task is more efficient. The classifiers mentioned

above are usually used in object recognition in computer vision. Compared with object recognition task, object tracking contains much less categories of objects, and it is obvious that object recognition is a time-consuming work in most of the time. Therefore, a "weak classifier to recognition can be sufficient for the tracking task, and it is also very important to the hardware implementation. In this part, relationship between similarity, APED vector, and NN classifier is analyzed. Based on the analysis, a new appearance model use basic NN has been proposed. The accuracy of this system has been evaluated and the simplicity of hardware implementation has been discussed.

In summary, this thesis presents a novel real-time solution to object tracking task with learning ability. The robust feature learning ability of the system is realized by introducing the SVM and NN classifiers into the tracking system, and designing a new tracking framework for the classifier-based algorithm. The hardware implementation problem was considered carefully. Hardware-friendly architecture has been designed and the real-time tracking system has been finally implemented on an FPGA board with a dedicated VLSI chip. Extensive experiments have been performed for evaluation on the tracking accuracy of this system. The thesis also contains very detailed discussions about the system.

Acknowledgements

First of all, I am extremely grateful to my supervisor, Professor Tadashi Shibata, for his guidance and all the useful discussions and suggestions, especially during the difficult time. His deep insights helped me at various stages of my research. His enthusiasm impressed me deeply, and becomes encouragement to me all the time.

I would like to thank Professor Keikichi Hirose, Professor Kunihiro Asada, Professor Kiyoharu Aizawa, Professor Akira Hirose, and Professor Makoto Ikeda, for their preview of the thesis and their extremely valuable comments to my research. Their constructive suggestions were indispensable for making my dissertation study successful.

Thanks to Professor Yoshio Mita for his instructive discussion. In discussions, he gave me plenty of helpful comments. Although working in different research fields, his interesting suggestions always inspired me.

I express my appreciation to Ms. Kimiko Mori for her help on so many documentaries, and to Ms. Motoko Inagaki for her helpful support.

I wish to express my sincere appreciation to those who have helped me in the past three years. I want to express my gratitude to all the lab members. During these three years, I received the most encouragement from them. I am so lucky to study and work with all of them. I would like to thank my friends, Hongbo Zhu, Ruihan Bao, Renyuan

Zhang, Wenjun Xia for their help on my research and in my daily life. Whenever I need help, they are always there. Their support is of great importance and really precious to me. Being with them made my college life an amazing experience.

Thanks to the GCOE program for providing me with a platform to communicate with those excellent students. The activities gave me precious opportunities to learn from other people and to express my own idea. I received very important support from GCOE program.

Finally, I would like to thank my father, mother, and my wife Yaoyao Shan. Without their help, it would be impossible for me to complete my study. I always feel their warm care and support, which encourages me all the time.

Contents

List of Figures

LIST OF FIGURES

LIST OF FIGURES

List of Tables

1

Introduction

1.1 Background

Object tracking is an important research topic in several fields, such as computer vision and video processing. Generally speaking, the purpose of an object is to generate the trajectory of an object over time by locating its position in every frame of the video. In other words, a tracker assigns consistent labels to the tracked objects in different frames. In some cases, a tracker also provides object centric information, such as orientation, area, or shape of an object. However, location of the tracked object is still the most essential information. Object tracking is well-studied during the past two decades, and has many practical applications, such as video surveillance, robot control, medical care, and human-computer interaction (2, 3, 4, 5, 6).

Since in human brain recognition and tracking are two basic and important abilities, which are realized in ventral pathway and dorsal pathway in brain according to some researches in biology, psychology, and neuroscience. The tracking ability tells us where is the object that we are interested in, and also sends the information for motion detection and some other high-level processing. The brain structure is a high efficient system, which is very different from modern computer. It can achieve robust tracking ability as well as high-speed performance. Although the mechanism in human brain is still not clear, it is possible to learn from present knowledge about brain. This method has achieved many successes in other fields, in which neural network is a famous example.

There are two aspects that we learn from the brain. The first aspect is the system structure (or tracking framework). In order to achieve real-time performance, we need to consider both the algorithms and implementations. The other aspect is the learning ability. The learning ability can solve difficult problems in object tracking, such as deformation, occlusion, rotation, illumination change, and multi-target tracking. These are very common problems in practical applications.

A typical tracking system consists of two components: 1) an appearance model, which can evaluate the likelihood that the object of interest is at some particular location, and 2) a search strategy for finding the most likely location in the current frame. In this thesis, the tracking algorithm will be discussed in detail based on this two-part system structure.

1.2 Related Works

During the recent more than two decades, many algorithms and hardware implementations on object tracking have been proposed. In this part, a brief summary of researches on object tracking is given according to some different standards.

As described above, a typical tracking algorithm contains an appearance model (7, 8, 9, 10). In many cases, the model itself is a representation of the object image using some features. Color (11), shape (12), and texture (13) are very common features. Especially the color feature is widely used in some algorithms because of its simple computation. However, it is obvious that the color is not an effective feature in many situations, and it is usually sensitive to illumination change. Object boundaries usually generate strong changes in image intensities. An important property of edges is that they are less sensitive to illumination changes. Texture is a measure of the intensity variation of a surface which quantifies properties such as smoothness and regularity. It is not sensitive to illumination change and shape change (non-rigid object). However, it requires the object should be rich of texture information that is sufficient for discriminating different objects.

There are also some other complex features, which can describe the object efficiently, including optical flow, active contour (14), SIFT (15), PCA (16), oriented

energy (17, 18) and many others (19, 20, 21, 22). These features have their own characteristics and are effective for corresponding situations. Since in most of the situations, a single feature is not sufficient to deal with all situations, many works consider to use combination of some of these features (23, 24, 25, 26). This is related to problems of feature selection and feature combination. There are several aspects to evaluate a feature representation algorithm. For example, whether it is a global representation or local representation is important in object tracking algorithm. Another aspect can be the computational cost of this algorithm. No matter the algorithm is implemented by software or hardware circuits, high-speed processing is always a critical issue. Generally speaking, along with the algorithm becomes complex, the processing time increases. Therefore, a balance between the accuracy and speed becomes important recently. Take SIFT as example, which is one of the most effective feature extraction algorithms, there are many researches on both development on algorithm and hardware implementation (27, 28). SIFT extracts key features from image with size and orientation information. These features are selected for effectiveness, and the number of featured depends on the image. In order to make SIFT usable in more applications, there are some researches focused on implementing SIFT on VLSI chip for high-speed processing (29, 30, 31, 32). It has been claimed that some VLSI can generate one SIFT feature from an image in only 3600 clock cycles (32). The selection of features will affect the performance of tracking algorithm (33, 34). Especially in a tracking system, the feature extraction is usually of great important to the performance.

How to evaluate the similarity between two images is related to the feature extraction algorithm (35), but sometimes related to localization algorithms (36). In order to estimate the object tracking algorithms there are many kinds of localization strategy (37, 38). These algorithms can also be considered as an object detection algorithm in the tracking algorithm, although there is slight difference between tracking and object detection. Object detection is a task to find some object or tell whether some object exists in an scene. Object tracking for one frame of image is similar that it also requires finding the object in the image. However, the simplest difference between object detection and object tracking is that in object tracking the objects location always has some limitation. For

example, the object location in current frame should be in a region determined by the object location in previous frame.

For localization, many kinds of methods can be used, such as background subtraction (2, 39), segmentation, and searching (40, 41). Among them, mean-shift (15, 42, 43) and particle filters (24, 44, 45, 46, 47, 48) are very commonly used. Particle filter is one of the most powerful algorithm for estimating the object state in different frames (49). It provides an effective framework for solving object tracking problems in nonlinear, non-Gaussian systems. Particle filter is based on the Bayes principle and is a sequential Monte-Carlo simulation method indicated by probability density of particles. Simply speaking, it usually contains two processes: predicting and updating. By using this two processes, the particles are updated in a way of "fit for surviving". The distribution of particles is used for estimation of the object state (location). Despite its outstanding performance (50), its computational cost is the disadvantage. Although there are researches working on implementing particle filter on GPU, FPGA, or VLSI (51, 52, 53, 54), a satisfied processing time is still difficult to achieve.

Hardware implementation is a hot topic in object tracking research. This is a little different from other researches in computer vision. One of reason is what need to process in object is a consecutive image sequence, which can be considered as a data stream. Therefore, the processing speed is more important than in other researches (55). There are many implementations trying to realize real-time object tracking, built on GPU, FPGA, or VLSI (56, 57, 58). The basic idea of these researches is to parallelize the algorithm and then do computation in parallel. In order to build a real-time tracking system, there are several critical issues. Firstly, the camera as video data input is alway a limitation of the tracking system, because most of the camera used transferring pixel data in serial way. The data transfer and exposure time in fact take most of the computational time, although the process can be organized in pipelined way. Secondly, it is more convenient to deal with simple integers than floating numbers. This is related to the complexity of the system, and also determine the memory cost. Thirdly, the cost of hardware resources and the data transfer issue is especially important in video processing. The process of object tracking contains operations and transfer of large amount of data. Organization of the data stream usually is a critical

issue in the hardware system. Until now, most of the implementations achieves processing speed close to video frame rate, which is usually 25 to 30 f/s (59, 60).

In the development of object tracking, there are two interesting research directions, which are promising for significant improvements. One is object tracking that considers the context information (61). For example, in a vehicle tracking application, the location of vehicles should be constrained to paths on the ground. Another direction is to employ learning algorithms to make robust tracker (62, 63, 64). This is because the tracker always deal with very complex situations, which is always difficult to describe before tracking or summarizing in rules. For example, some classifiers are built to detect the object from the background, but it is very difficult to determine what classifier should be used. In different situations, the results can be very different. If the tracking algorithm has some learning ability, especially on-line learning ability (65), it will be adaptive and robust. A learning tracking algorithm has been developed using support vector machine classifier (66). For SVM-based trackers, the positive examples consist of the images of the object to be tracked, and the negative examples consist of all things that are not to be tracked. This is an example of tracking algorithm with learning ability and using context information. One advantage of this approach is that knowledge about background objects is explicitly incorporated in the tracker.

1.3 Evaluation Methodology

To evaluate an object tracking system is itself a challenge. The difficulties lay in that the tracking in computer vision is a process similar to the tracking ability in human brain. In most of the cases, it is simple to tell whether the system has tracking ability as robust as human, but it is difficult to tell how far away it is from human ability when compare different systems. Another aspect is that the evaluation on tracking ability sometimes depends on the applications. For example, there must be some error between the predicted location and the real object location. So what kind of error is acceptable? Despite the difficulties, there must be some way to evaluate a research, which is essential for development of the research.

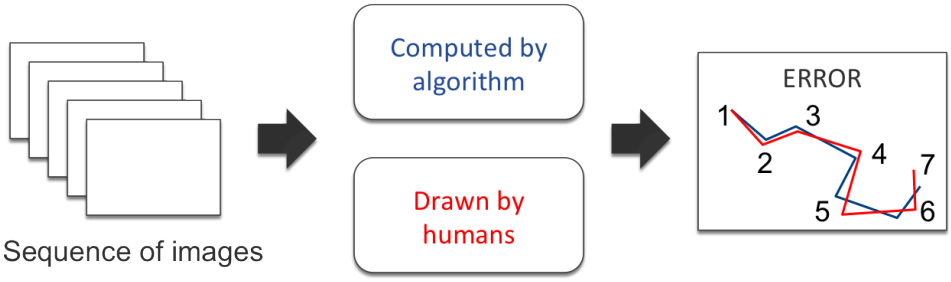


Figure 1.1: Illustration of evaluation methods on object tracking systems.

Generally speaking, there are three methods that are commonly used: qualitative comparison, location error, and precision. The qualitative comparison method is most common. Usually, the researcher will use some video database to test their algorithms, and the result is examined by researcher to tell whether the algorithm track the object all the time. If the algorithm is not robust enough, and lost target at a certain frame of image, then the situation in this image will be discussed to show the advantage or disadvantage of an algorithm.

The location error analysis shows more information about tracking performance, as shown in Fig. 1.1. At first, in order to build a database, some videos must be collected and human subjects are asked to draw the object location in every frame of the image, served as ground truth. Then the predicted locations from some algorithm will be compared with the ground truth by calculating the distance between the predicted location and ground truth location in each frame. This distance is called location error. This method can evaluate algorithms more accurately. However, there are still problems. Since the error versus frame plot can be difficult to interpret, it is useful to summarize performance by computing the mean error over all the frames of the video. However, this value sometimes fails to correctly capture tracker performance. For example, if a tracker tracks an object closely for most of the video, but loses track completely on the last several frames, the mean location error may be higher than a tracker that sticks with the object though not as precisely. The preference between these two behaviors inevitably depends on the final application.

The precision comparison is a method proposed in (67) to provide percentage of the correct tracked frame within a certain threshold for location error. This

method shows the percentage of frames for which the predicted object location was within some threshold distance of the ground truth. For example, if we choose threshold 20 to calculate the precision, the result of accuracy on a certain video is the percentage of frames for which the tracker is less than 20 pixels away from the ground truth. In this thesis, in order to evaluate the system from different aspects, we adopted all the three methods.

1.4 Real-Time Object Tracking System

In this thesis, we proposed a solution to the tracking task based on consideration of efficient implementation as priority, and solved problems as follows. Firstly, a hardware-friendly tracking framework is designed, which is implemented on FPGA, and compatible with VLSI technology. This framework, named multiple candidate regeneration (MCR), is developed as a simple but high-speed and high-efficiency searching and predicting algorithm. A simple illustration of the MCR is shown in Fig. 1.2. In this algorithm, some candidates are used for searching the desired location (location with maximum similarity in this case). By updating and regenerating the candidates as illustrated, the candidates tend to accumulate at the maximum location. Although it is very similar to particle filter (PF), it can be considered as a fast searching strategy instead of a approximation of intensity distribution. In the development, several problems, which may limit the hardware performance, have been resolved, such as complex computation, data transmission and cost of hardware resources. This solution has several advantages. First, it can work at high frame rate, which can simplify the work of localization. Second, the system can be extended to use in many applications because of its flexibility. Third, since the processing speed is faster than real-time standard, it is probable that the ability of this system can be improved in the future.

We have aimed to establish both robustness of object representation and the real-time performance of the processing, because feature extraction is usually a time consuming process. The directional-edge-based image feature representation algorithm is employed to represent the object image. Robust performance of the directional-edge-based algorithms has already been demonstrated in various image recognition applications. In addition, dedicated VLSI chips for efficient

directional edge detection and image vector generation have also been developed for object recognition systems.

Whether a tracking system can be easily extended for various purposes is also very important. This thesis contains a detailed discussion on extending the function of the system, including hardware implementation on VLSI, multiple-object tracking, full-occlusion and initialization problems, and employing of state vector. The architecture of this system is compatible with VLSI design, and may reach better performance on VLSI. For the multiple-object tracking, an efficient method is proposed to allocate the limited hardware resources. For the full-occlusion and initialization problems, a searching algorithm based on proposed system is developed. By using the state vector, more attributes can be estimated for achieving more information about the object. It also improves the tracking ability and accuracy.

The following parts of the thesis are focused on the learning ability of the system. For object tracking, one promising direction is to consider the object tracking as a binary classification problem, and employ discriminative methods in the tracking framework. Support vector machine (SVM), as a powerful classification scheme, has been used in many tracking algorithms, benefiting the algorithms with accurate localization and flexible modeling of the target. The SVM works as an appearance model of the target by changing its boundary while training. In order to extend the power of SVM in most of the general tracking applications, it is necessary to develop a proper tracking framework and a VLSI-hardware-implementation-friendly structure for the SVM-based algorithm. A real-time visual tracking algorithm is presented employing an on-line support vector machine (SVM) scheme. A novel training framework is proposed, which enables us to select reliable training samples from the image sequence for tracking. The tracking framework includes how to update training samples and how to select test samples and make prediction of the target location. Different from other algorithms, this framework gives a rule guiding the selection of target training samples. In addition, regarding the selection of target samples for on-line training of SVM, a new selection rule has been introduced. Based on a SVM chip developed in our group, the most complex part in this algorithm can be computed efficiently. At the same time, multiple candidate regeneration is employed

to reduce the computational cost without sacrificing the tracking accuracy. In addition, the directional-edge-feature vector representation, whose VLSI implementation has been proposed in, is employed to represent the sample images. By using this hardware-friendly structure, real-time tracking ability can be achieved. The hardware architecture for realizing this kind of real-time tracking system is discussed in detail.

After development of the SVM-based tracking system, the essential facts of the tracking task were analyzed, and an NN-based tracking system has been proposed. The basic idea is that a classifier specially designed for tracking task is more efficient. The classifiers mentioned above are usually used in object recognition in computer vision. Compared with object recognition task, object tracking contains much less categories of objects, and it is obvious that object recognition is a time-consuming work in most of the time. Therefore, a "weak classifier to recognition can be sufficient for the tracking task, and it is also very important to the hardware implementation. In this part, relationship between similarity, APED vector, and NN classifier is analyzed. Based on the analysis, a new appearance model use basic NN has been proposed. The accuracy of this system has been evaluated and the simplicity of hardware implementation has been discussed.

In summary, this thesis presents a real-time solution to object tracking task with learning ability. The robust feature learning ability of the system is realized by introducing SVM and NN classifiers into the tracking system, and designing a new tracking framework for the classifier-based system. The hardware implementation problem was considered carefully. Hardware-friendly architecture has been designed and a real-time tracking system has been implemented. Extensive experiments were performed for evaluation on the tracking system. The thesis also contains very detailed discussion about various aspects of the tracking system.

1.5 Scope of This Thesis

Object tracking (visual tracking) is a research of a wide scale related to many different fields, although it belongs to computer vision field basically. In this

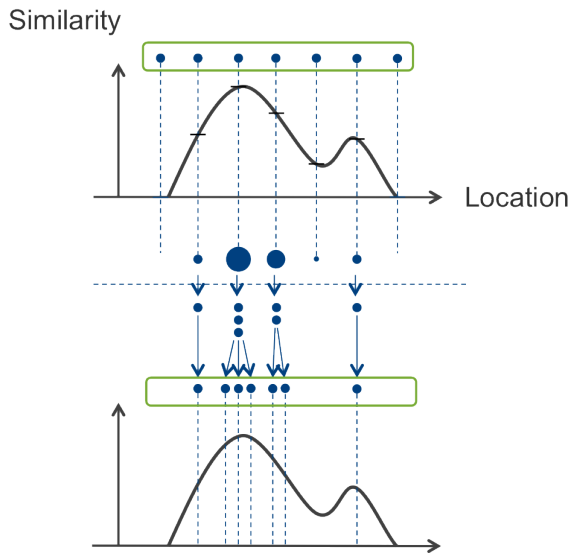


Figure 1.2: Illustration of the process of multiple candidate regeneration.

part, the scope of the thesis is described to clarify the focus of my work in object tracking. The scope is described from several aspects in the following.

From the view point of how to represent an object in the video (scene), object tracking algorithms can be classified into several groups. The most common approaches are feature-based, which means the image of the object is represented by some features extracted from the image. Color, edge, texture are usually used as basic features. Some special features are also used, such as motion feature, SIFT, and active contour. My research is focused on the feature-based approaches, and the directional-edge feature is adopted as the representation. Many object tracking algorithms are based on a two-part model as described in the previous section. My research follows this typical structure, contains object model and location prediction.

In this thesis, the hardware implementation of the tracking system is of great importance. A real-time object tracking system is the basic goal of this research. Therefore, in this thesis, the issues and solutions for hardware implementation are discussed in detail. There are several critical issues for implementation, such as parallelism, power consumption, processing time and efficiency, and system stability. But in the discussion of this thesis, I discuss mainly on the processing

speed and simple implementation of the system in order to focus on the main contribution of my research and give some convincing verification.

In this research, some related research topics other than object tracking are also included, such as feature extraction, machine learning, and neuroscience. These researches help improve the object tracking system in various aspects. Only the most important technologies that are closely related to my research are discussed.

In summary, this thesis is focused on important issues and solutions in realization of a real-time object tracking system, including both algorithm and hardware implementation. It includes topics: tracking frame work, feature extraction, learning algorithm, extensive evaluations, architecture on VLSI technology, FPGA implementation for verification, and detailed discussions.

2

A Real-Time Object Tracking System Employing Multiple Candidate Regeneration

2.1 Introduction

Object tracking plays an important role in many applications, such as video surveillance, human-computer interface, vehicle navigation, and robot control. It is generally defined as a problem of estimating the position of an object over a sequence of images. In practical applications, however, there are many factors that make the problem complex such as illumination variation, appearance change, shape deformation, partial occlusion, and camera motion. Moreover, lots of these applications require a real-time response. Therefore the development of real-time working algorithms is of essential importance. In order to accomplish such a challenging task, a number of tracking algorithms(14, 17, 33, 40, 68, 69) and real-time working systems(39, 49, 52, 57, 70, 71) have been developed in recent years.

These algorithms usually improve the performance from two major aspects of the object tracking task, i.e. the target object representation and the location prediction. In the location prediction, the particle filter(72) shows a superior tracking ability, and has been used in a number of applications. It is a powerful method to localize target, which can achieve high-precision results in complex situations. Some works have proposed improvements based on the particle filter

framework for better tracking abilities in very challenging tasks(69). Despite the better performance of these algorithms with more complex structures, they suffers from the high computational cost which prevents their implementation from working in real time.

Some implementations using dedicated processors always result in power-hungry systems(51, 52). Many implementations parallelize the time-consuming part of algorithms, increasing the processing speed to real-time(55, 60, 73). These solutions depend heavily on the algorithms and may be limited by them, if the algorithms are not designed for efficient hardware implementation. Sometimes it is not an easy work to break the tradeoff between different aspects and make improvement in this situation. Some specific implementations can be employed to speed up certain part of the algorithm, such as feature extraction(29) or localization(54). In this case, it is necessary to consider how to combine them into an efficient system, because several problems rise when building parallel system, such as transmission of large amount of data.

In this thesis, we have explored a solution to the object tracking task considering an efficient implementation as the first priority. A hardware-friendly tracking framework has been established and implemented on FPGA, thus verifying its compatibility with VLSI technology. Several problems that limit the hardware performance, such as complex computation, data transmission, cost of hardware resources, etc., have been resolved. The proposed architecture has achieved 150 f/s on FPGA, and if it is implemented on VLSI with on-chip image sensor, it is possible to achieve the frame rate as fast as 900 f/s.

Since our solution provides a high flexibility in its configuration, it can be integrated into a lot of other more-complex intelligent systems as their sub-systems. Due to its real-time performance much faster than the video rate it would provide a lot of opportunities for building real-time-operating highly-intelligent systems.

In tracking algorithms, how to represent the target image is of particular importance because it greatly influences the tracking performance under certain tracking framework. Color, edge, and texture are typical attributes used for representing objects(11, 23). A number of other features, including active contour(57), SIFT feature(24), oriented energy(14), and optical flow(66), are also used in many works. Some works also combine these features or incorporate on-line learning

of the model of an object and background(17, 23, 33, 74). In our research, we have aimed to establish both robustness of object representation and the real-time performance of the processing, because feature extraction is usually a time consuming process.

It is well known that animals have excellent ability in visual tracking, but the biological mechanism has not yet been clarified. However, it was revealed that the visual perception of animals relies heavily on the directional edges(75). In the present work, therefore, the directional-edge-based image feature representation algorithm developed in(76) is employed to represent the object image. Robust performance of the directional-edge-based algorithms has already been demonstrated in various image recognition applications. In addition, dedicated VLSI chips for efficient directional edge detection and image vector generation have also been developed for object recognition systems(77, 78).

The purpose of this work is to develop a real-time object tracking system which is robust against such disturbing situations like illumination variation, object shape deformation, and partial occlusion of target images. By employing the directional-edge-based feature vector representation, the system has been made robust against illumination variation and small variation in object shapes. In order to achieve real-time performance in tracking, a VLSI hardware-implementation friendly algorithm has been developed. It employs a statistical approach in which multiple candidate locations are generated during tracking. The basic idea was inherited from the particle filter but the algorithm has been greatly modified and simplified from the original particle filter so that it can be implemented in VLSI hardware very efficiently. The algorithm was first proposed in(79) and the performance was verified by only simulation. In this article, however, the algorithm was actually implemented on an FPGA, and the real-time performance and robust nature have been demonstrated by the measurement of the working system. In order to further enhance the robustness of the tracking ability, an on-line learning technique has been introduced to the system. When the target object changes its appearance beyond a certain range, the system autonomously learns the altered shape as one of its variations, and continues its tracking. As a result, for a large variation in the shape and for partial occlusion, the system has also shown a robust performance. The system was implemented on a Terasic DE3 FPGA board.

Under the operating frequency of 60 MHz, the experimental system achieved a processing ability of 0.8 ms per frame in tracking a 64×64 scale object image in 640×480-pixel size video sequences.

Object tracking is still a challenging task for application in real world due to different requirements in complex situations. Based on the tracking system developed in this work, we also proposed solutions to some important tracking problems, which was not included in the algorithm of(79). We have designed a flexible architecture for multiple target tracking, using only a limited number of parallel processing elements. A new image scanning scheme has been explored to realize automated initialization of the tracking system instead of manual initialization. In this scheme, the image of the tracking target is autonomously localized in the initial frame of image sequences. The same scheme has also been used to solve a group of similar problems: full occlusion, target disappearance from the scene, and accidental loss of the target image, while requiring only a few additional logic functions in the circuitry.

2.2 Algorithm

The most essential part of this algorithm is a recursive process called multiple candidate regeneration (MCR), which is similar to the prediction and update in the particle filter. The task of object tracking in a moving image sequence is defined as making a prediction for the most probable location of the target image in every consecutive frame. The iteration process is shown in Fig. 2.1.

At the very beginning of the tracking (the initialization stage), the target image is specified manually by enclosing an image by a square window and the center coordinates (x, y) of the window is defined as the image location. The target image enclosed in the window serves as a template in the following tracking process. At the same time, a fixed number of candidate locations are generated as possible locations for search in the next frame. In the initialization, these candidate locations are uniformly placed around the target image location so that their average location coincides the target location.

In the second frame, the similarity between the target image and the local image at each candidate location is calculated and a weight is assigned to each

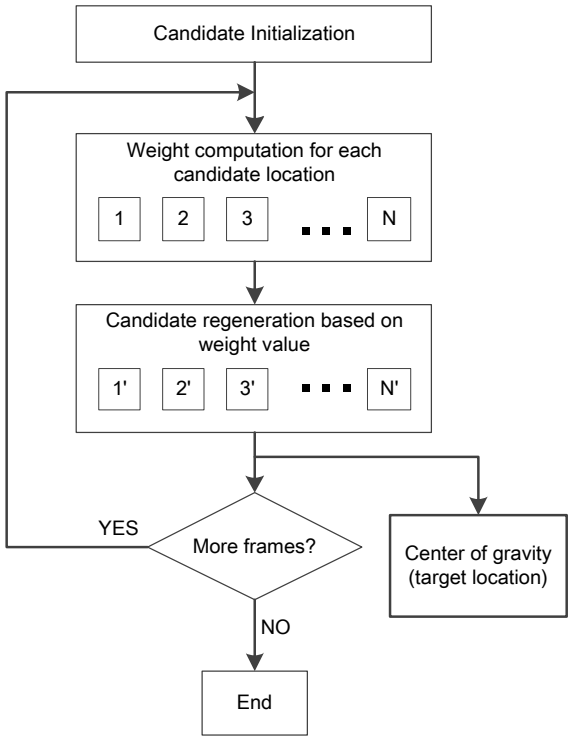


Figure 2.1: Process of multiple candidate regeneration (MCR).

location based on the similarity. The larger the similarity is, the larger weight is assigned. Then, new candidate locations are regenerated reflecting the weight (similarity) at each location. Namely, a larger number of new candidate locations are generated where the weight is large. The average of newly generated candidate locations yields the new target location in the second frame. The process continues iteratively for each coming frame.

Fig. 2.2 illustrates the procedure of weight computation and regeneration of new candidate locations using a simple example having only four candidates. In the previous frame shown at the top, there are four candidate locations represented by black dots around the target image of a smiling face. In the present frame below, the target moves to right and comes closer to location 3. The dotted line squares indicate the local images at candidate locations. The images at candidate locations are matched with the template image, and the weights are calculated according to their similarities, which are shown as solid black circles below. A larger similarity corresponds to a larger weight, being represented by a

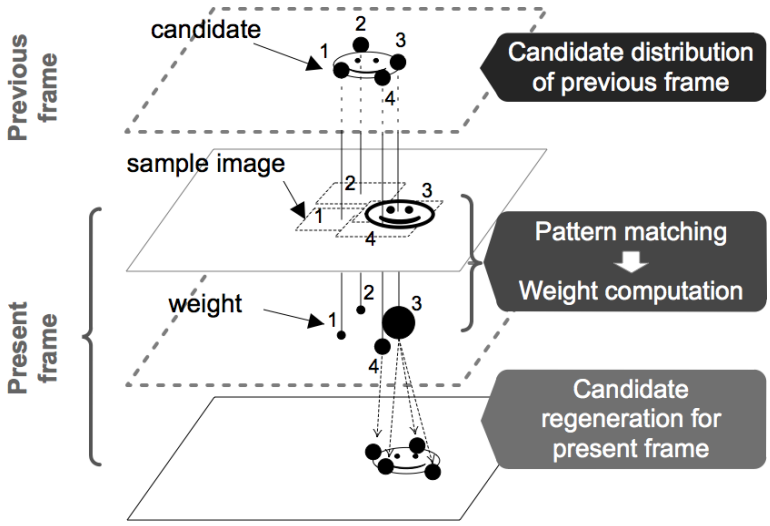


Figure 2.2: A simplified four-candidate example illustrating weight computation and candidate regeneration.

larger solid circle. Then the same number (four) of new candidate locations are generated in the regeneration process, following the rule that a candidate with a higher weight regenerates more new candidates around its location. The old candidates are discarded after regeneration so that the total number of candidates stays constant. As shown at the bottom, four new candidates are generated and the average of their locations yields the most probable location of the target in the present frame.

In the following, the entire algorithm is explained in detail, including the representation of object image, the weight generation, the candidate regeneration and the on-line learning function. They are all designed specifically aiming at easy and efficient hardware implementation.

Figure 2.3: Main structure of the present object tracking algorithm (left). Examples of candidate points distribution in the initial frame as well as in a new frame (right).

2.2.1 Algorithm Structure

Fig. 2.3 shows the structure of the algorithm. The principal component is the multiple candidate regeneration block (MCR). The algorithm starts with the initialization block at the beginning, which sets up all necessary parameters, including candidate locations and the target template. The candidate container and the template container are two memory blocks that store the candidate locations and feature vectors of the templates, respectively. Initialization is carried out with the first frame image, where the target for pursuit is identified by enclosing the image with a square window as shown at the top right. This is done manually. The points in the tracking window represent locations of candidates. These points are distributed uniformly in the tracking window in the initialization step, and stored in the candidate container. A feature vector of the target is generated from the image in the tracking window, and stored in the template container. Throughout the algorithm, we use reduced representation of local images and the procedure of feature vector representation is explained later in this section.

There are two loops in this algorithm: loop A and B as shown in the figure. In loop A the output of MCR is sent back to the candidate container as inputs to the next iteration. The MCR keeps updating the candidate distribution whenever there is a new frame coming. One example is shown at the bottom right in Fig. 2.2, in which the points are candidate locations and the square is located at the center of gravity of all the candidates at the present time. This yields the most probable location of the target in the present frame. Loop B represents the process of learning feedback. The on-line learning block generates new templates during the tracking process and stores new templates into the template container. Detail about this process is explained at the end of this section.

In summary, the algorithm starts from initialization block using the first frame of image, then processes each new coming frame and outputs the target location continuously until there is no more input image.

2.2.2 Object Representation

As explained above, in order to calculate the weight of each candidate, we need to evaluate the similarity between the candidate image and the template image. This

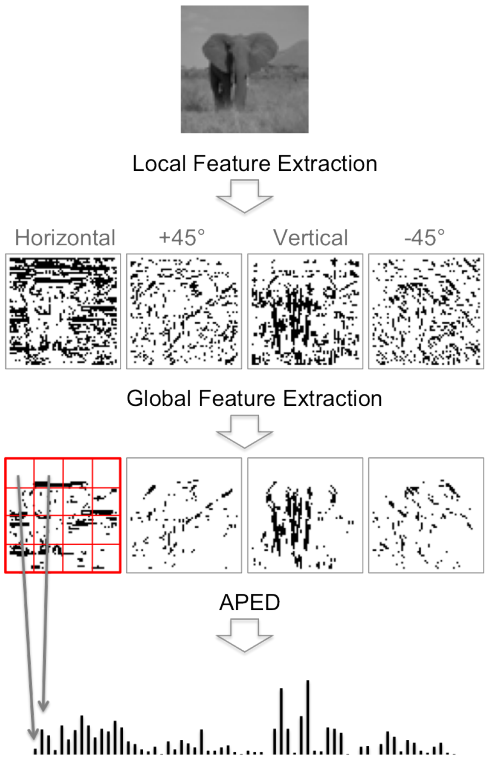


Figure 2.4: Feature extraction from a 64×64-pixel gray-scale image and conversion to a 64-dimension feature vector(1).

is done by calculating the distance between the two feature vectors representing the two images. Therefore, employing a suitable feature representation algorithm is very important. We employed the directional-edge-based image representation algorithm(1, 80, 81) which was developed being inspired by the biological principle found in the animal visual system(75). This method needs only the gray-scale information of an image as input, and the output is a 64-dimentional feature vector. It consists of three successive steps: local feature extraction (LFE), global feature extraction (GFE), and averaged principal-edge distribution (APED)(1). Fig. 2.4 shows the function of each step.

2.2.2.1 Local feature extraction (LFE)

The function of LFE is to extract the edge and its orientation at each pixel location in an image. For every pixel location, the convolutions of a 5×5 pixel region

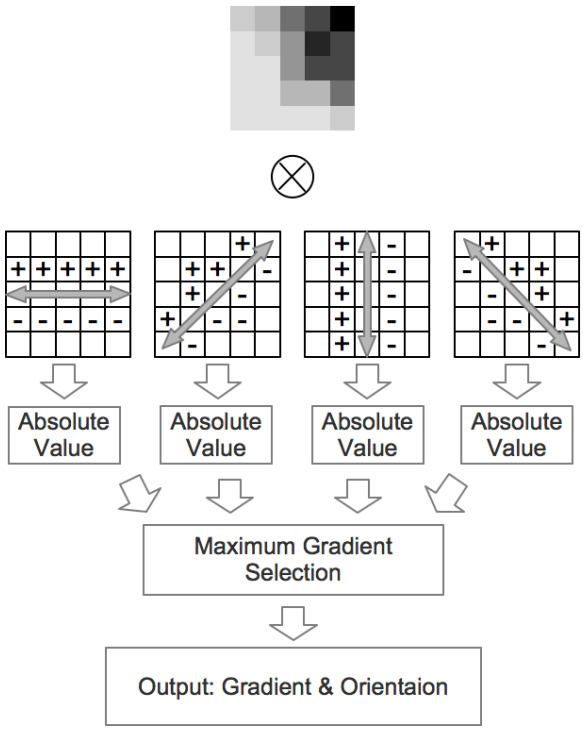


Figure 2.5: Process of directional edge detection using 5×5-pixel filtering kernels.

with four directional filtering kernels (horizontal, $+45°$, vertical, $-45°$) are calculated as shown in Fig. 2.5. Then, the absolute values of these four convolution results are compared, and the maximum value and its corresponding orientation are stored as the gradient and edge orientation at this pixel location, respectively. The processing is carried out at all pixel sites in an input image except for the two rows and two columns adjacent to the four boundaries. Therefore, from a 68×68 input image, a 64×64 gradient map with orientation information at each pixel is produced.

2.2.2.2 Global feature extraction (GFE)

The gradient map produced in the previous step contains edge orientation at all pixel sites. In this step, only the edges of significance are left by setting a threshold to the gradient map. All the gradient data are sorted, and we find out a certain number of pixels that have larger gradient values than others. The pixel locations with these larger gradients are marked as edges in four-directional edge

maps. The number of edges to be left is specified by a percentage to the total pixel number.

2.2.2.3 Averaged principal-edge distribution (APED)

Although the information has been compressed by extracting edges in LFE and GFE, the amount of information is still massive in quantity. Therefore, a method called APED(1) is employed to transform the four edge maps into a 64-dimension vector. In the APED vector representation, each edge map is divided into 16 square bins, and the number of edge flags in each bin is summed up, which constitutes an element of the vector. The 64-dimensional feature vector is the final output of the feature extraction processing, and is used throughout the entire algorithm as the representation of local images, including candidate images as well as template images.

2.2.3 Weight Computation and Candidate Regeneration

Since the basic principle has already been explained, how to implement the principle is described here. In order to make all computations easily and efficiently implementable in the VLSI hardware, each mathematical operation was replaced by a hardware-implementation friendly analogue, which are different from that in the regular particle filter algorithm.

The local image taken from each candidate location is converted to a feature vector and the Manhattan distances are calculated with template vectors. In this algorithm, there are more than one templates in the template container to represent the target. The first template is generated at the initialization step, while others are generated during the on-line learning process. Therefore, the minimum Manhattan distance is utilized to determine the weight for this candidate as described in the following.

$$MD_{i,j} = \sum_{k=1}^{n} |V_{Ci}[k] - V_{Tj}[k]| \tag{2.1}$$

$$D_i = \min(MD_{i1}, MD_{i2}, ..., MD_{in}) \tag{2.2}$$

$$W_i = \begin{cases} 0, & (D_i \geq C) \\ INT[N_0 \times (1 - D_i/C)], & (D_i < C). \end{cases} \tag{2.3}$$

Here, MD_{ij} stands for the Manhattan distance between the candidate i and the template j, and $V_{Ci}[k]$ and $V_{Tj}[k]$ denote the k-th element of the candidate vector $\mathbb{V}_{Ci}$ and the template vector $\mathbb{V}_{Tj}$, respectively. D_i is the minimum distance of candidate i with all the templates and W_i represents the weight for the candidate i. N_0 is a constant value determining the scale of the weight. In (3), C is a threshold defining the scale of weight values, which is determined by experiments. INT means taking the integer component of the value. In this manner, those candidates that have at least one Manhattan distance value smaller than the threshold C are all preserved to regenerate new candidates in the next frame. At the same time, larger weight values are assigned to candidates having smaller distances.

In the third step, new candidates are regenerated as described below. Firstly, the maximum weight value W_{max} is found and it is used as a threshold number (N_{th}) for new candidate regeneration. Note that $N_{th} = W_{max}$ ($\leq N_0$) is an integer number. At old candidate locations whose weights are equal to W_{max}, a new candidate is generated in the vicinity at each location. Then the threshold number is decreased by one, and a new threshold is obtained as $N_{th} = W_{max} - 1$. Then all weight values are compared again with the new threshold number, and at those old candidate locations whose weights are greater than or equal to N_{th}, one more new candidate is generated in each vicinity. Then N_{th} is decreased by one again ($N_{th} = W_{max} - 2$). The process is repeated until the total number of new candidates reaches a constant number N. After obtaining N new candidate locations, old candidates are all discarded.

2.2.4 On-line Learning

In many practical applications, the target we are concerned about is a non-rigid object, which may change its appearance and size. In addition, sufficient knowledge about the target is in general not available before tracking. This problem causes tracking failure if the algorithm does not flexibly learn the appearance change in the target. An on-line learning method is introduced to solve this

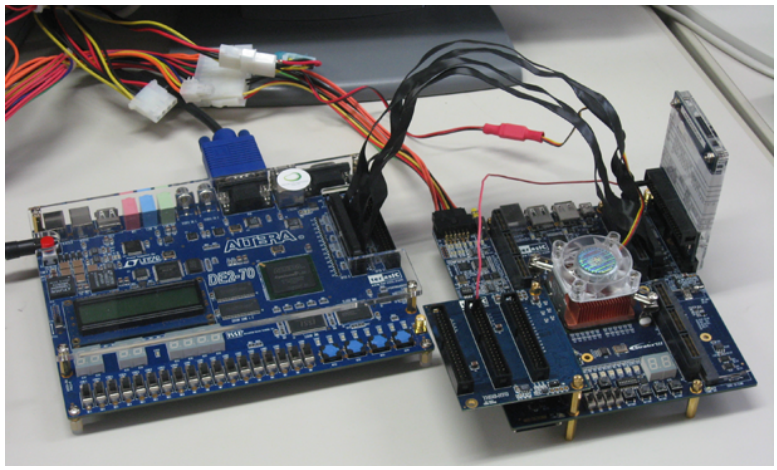


Figure 2.6: Object tracking system implementing the algorithm developed in this work.

problem in this work. The learning process begins after the estimation of the target location. One feature vector is generated from the image at the target location in the present frame. Then the Manhattan distances between this feature vector and all the templates are calculated, and the minimum distance is found. If the minimum distance is larger than a certain threshold, it is interpreted as that the target has changed its appearance substantially, and the feature vector is stored as a new template in the template container.

2.3 Implementation

This tracking system has been implemented on Terasic DE3 FPGA board which uses Altera Stratix III chip. Terasic TRDB-D5M camera is used as the image input device, and a Terasic DE2 FPGA board is used for saving and displaying the tracking result. A photo of this system is shown in Fig. 2.6. The following sections explain each part of the system and give the evaluation of the processing time.

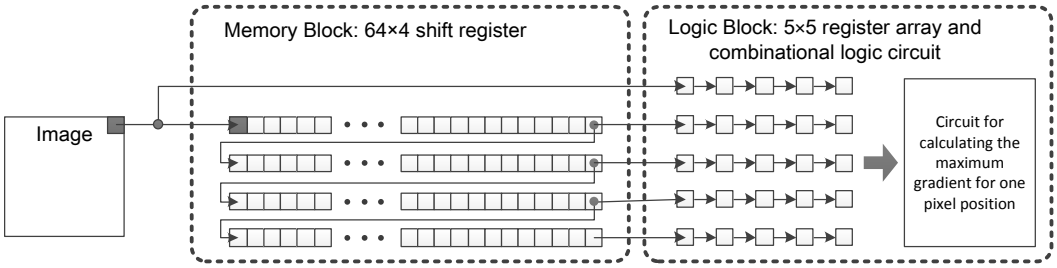


Figure 2.7: Implementation of local feature extraction (LFE) block.

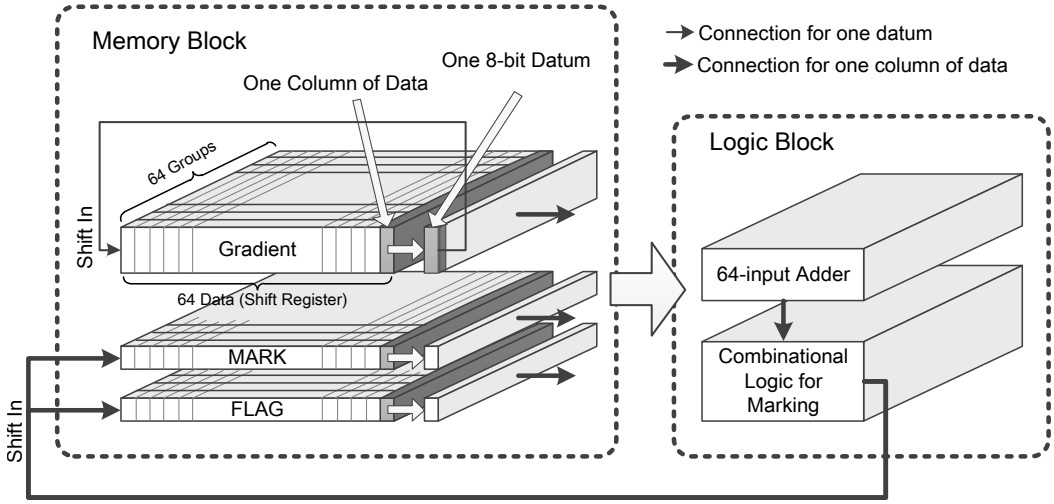


Figure 2.8: Implementation of global feature extraction (GFE) block.

2.3.1 Feature Extraction

The feature extraction stage is implemented in three serially connected functional blocks: local feature extraction (LFE), global feature extraction (GFE) and vectorization. In this system, the image transmission from camera to FPGA board is serial, one pixel per clock cycle. Therefore, at this stage, we built the feature extraction block working in pipeline for efficient computation. The whole system has eight such units working in parallel for efficient computation. The processing time is determined by the size of the candidate image, which was set as 68×68 pixels in this system. It takes about 5650 clock cycles to process one candidate image. Implementation of each part is explained in the following.

The structure of LFE block is shown in Fig. 2.7. There are four 68-stage shift registers each serially connected, and the output of each shift register is

inputted to the respective row of a 5×5 register array. It shifts pixel data of 8 bits. The shift register stores the minimum size of image data necessary for computation. The 5×5 register array works as a buffer between the shift register and the logic block. The combinational logic block deals with all the logic processing needed to calculate the gradient and orientation in two clock cycles, including doing convolution with four 5×5 kernels, taking their absolute values, and storing the largest value. The intensity values of an image are sent into the first row of the shift register and, at the same time, into the top row of the register array pixel by pixel. The four rows of data in the shift register are shifted-in to the corresponding lines of the 5×5 register array. In this manner, the 5×5-pixel filtering kernel block scans the entire image pixel by pixel and generates a directional gradient map. Because gradient values centered at the peripheral two rows and two columns are not calculated, a 64×64 gradient map is produced from a 68×68 image in 4626 clock cycles (2 more cycles for processing the last value).

The following GFE block, as explained in the algorithm section, must implement the sorting function. Since we employed a hardware-friendly sorting algorithm, the processing time is only related to the bit length of the data. This algorithm is briefly explained in the following, and the detail can be found in (77).

Supposing that we need to pick out the K largest data from a group of data, the sorting starts from the most significant bits (MSB) of the data. Before sorting, all the data are assigned a mark of "UNKNOWN". First, according to the value of MSB (1 or 0), the data are divided into two groups. The first group has all the data with "1" as MSB, while the second group owns all the data with "0" as MSB. Then the system counts how many data are in the first group. If the number is less than K, it is certain that all the data in the first group belong to the K largest, and the data are marked with "YES". If the number is greater than K, all the data in the second group can be discarded as not belonging to the K largest, and are marked with "NO". The data left will be still marked "UNKNOWN". In the second step, similar computation is repeated upon the second bit from MSB. The unknown data will be divided into two groups again, but the summation of the data in this step will also count in the data with mark

"YES". By repeating this procedure, all the largest K data will be marked with "YES" after processing all bits of the data. This is a parallel sorting method, which can be completed in several clock cycles theoretically. The difficulty in implementation is that we need an adder that sums up all single bits coming from all the data. In this tracking system, there are totally 4096 data to process in GFE. It is not easy to implement a 4096-input adder connected to 4096 15-bit registers. Therefore, we made a tradeoff between the speed and complexity, dividing the total 4096 data into 64 groups. The implementation of this part is shown in Fig. 2.8.

The 64 groups of data are processed in parallel and in a pipelined way. The "FLAG" and "MARK" are used to represent the state of each datum. The "FLAG" indicates whether the decision has already been made or still "UNKNOWN", while the "MARK" tells whether the datum is marked with "YES" or "NO". The 64 groups of data, and default values of "FLAG" and "MARK" are all stored in respective shift registers. Each shift register stores 64 data, and owns one output feeding back to its input. At the beginning, the shift register shifts data for 64 clock cycles, and a 64-input adder with accumulator sums up the MSB of all the data. In the next loop of 64 clock cycles, the "FLAG" and "MARK" are modified according to the summation result following the rules explained above. At the same time, the summation of the next bit from all the data are summed up, which will be used in the next loop. The calculation time for GFE is 1024 clock cycles.

The output of GFE is a binary map that contains the edge information. In the following step, this edge information is compressed effectively into a feature vector representation as explained in the algorithm section. We use shift registers and accumulators to realize this function in a common way, and do not describe it in detail here.

In computer vision, SIFT is a very effective feature extraction method providing local feature description. From the viewpoint of hardware implementation, we compared the APED with SIFT to illustrate the performance. Implemented on VLSI and FPGA(29, 32), the time for computing one feature of SIFT has been reduced to about 3300 clock cycles. In order to describe an sub-image effectively, at least three features are necessary, and more features are needed to describe

the scene. Matching is performed between these features. The feature extraction method in this thesis takes only about 5600 cycles to generate a global description of a candidate. Since the processing unit is not complex, it is convenient to realize parallel processing.

2.3.2 Multiple Candidate Regeneration

The next several blocks of the system, including weight generation and new location estimation, are explained in this part. Fig. 2.9 shows the hardware structure of the weight generation block and candidate regeneration block. A shift register is used to store the templates. Each time when this block receives a feature vector from feature extraction block, it sends a start signal to the template container, and the template container shifts out all the templates to the weight generation block. Then Manhattan distances between the feature vector and all the templates are calculated one by one, and the minimum value of them is retained for calculating the weight. At last, the weight will be sent to the candidate regeneration block.

In the candidate regeneration block a shift register is used to store the candidate locations, and the number of candidates is 64 in our system. The candidate regeneration block first collects all the weights for 64 candidates, and then count down from the largest weight value, which is set to 15 ($N_0 = 15$ in (3)) in this system. New candidate is generated when there is any candidate that has a weight value greater than the counter. As shown in Fig. 2.9, there are eight directions which the new candidate can choose randomly. This avoids the problem that all the candidates may tend to be generated in the same location. A 3-bit counter is used to represent the direction for regeneration. In every clock cycle while generating new candidates, the counter is added by one. Because the process of determining whether to generate new candidate is not regular, the directions read from the counter will be like random values. For determining the distance between the old and the new candidates, this system gives a small distance variance when the weight is large, because these candidates reflect the target location better. A large distance variance is assigned when the weight is small, in order to produce a wide distribution that can cover the area for detecting the target.

For example, weights larger than 12, larger than 8, and less than 8 correspond to distance of 1 pixel, 2 pixels, and 4 pixels, respectively. After the regeneration of new candidates, the new location is stored in the candidate container. The center of gravity of all the new candidate locations, is sent to both the display block and the on-line learning block as the prediction of the new target location.

The calculation time is decreased dramatically due to the hardware-friendly feature of this algorithm as compared to the software implementation of regular particle filter algorithms. For the calculation of these blocks together, a typical 620 clock cycles are needed, and the maximum number of clock cycles is 1024.

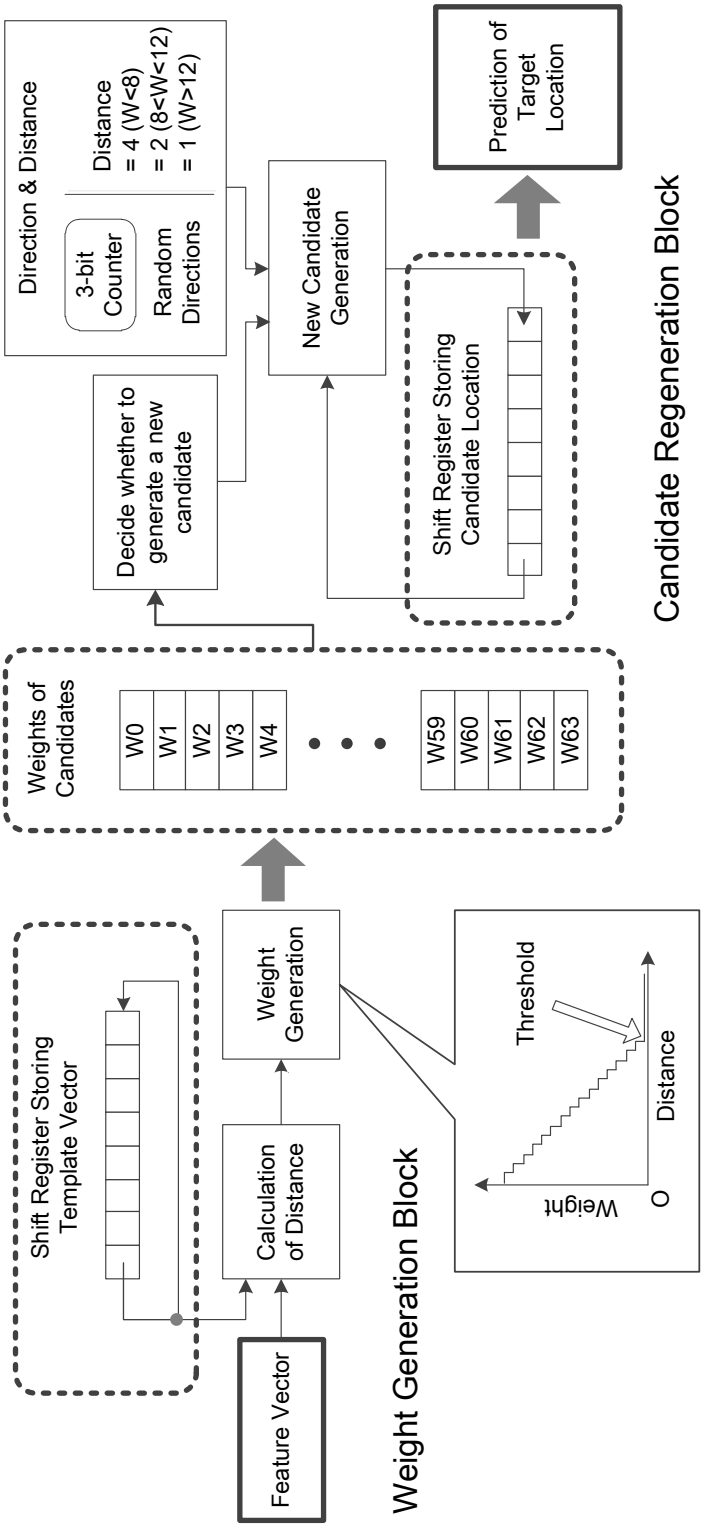


Figure 2.9: Implementation of multiple candidate regeneration block composed of weight generation block (left) and candidate regeneration block (right).

2.3.3 On-line Learning

After receiving the estimation of the new target location from the candidate regeneration block, this on-line learning block will extract the feature vector from the image at the new target location. This feature vector is compared to all the templates by using Manhattan distance to find the minimum distance. If the minimum distance is greater than a certain threshold, this feature vector will be stored into the template container as a new template. In order to start searching for new target locations, only a limited region in the input image that is four times larger than the tracking window is stored for saving memory resource.

2.3.4 Overall Structure

Fig. 2.10 illustrates the overview of the hardware organization of this system. Table 2.1 shows a summary of FPGA resource utilization of main processing blocks with processing time. After receiving the image data from camera, this system first allocates the data into corresponding memories. Then eight parallel candidate processing blocks process these data in parallel, and output the weight of each candidate. Then, these weight values are used to generate new candidate locations and the target location. At the same time, the on-line learning block updates the templates according to the tracking result in each iteration. In this system, we set up total 64 candidates for tracking. Considering the resource limitation of the FPGA board, we divided the 64 candidates into eight groups. All eight candidates in each group are processed in parallel, and all eight groups are processed serially. In the experiments on tracking with only eight candidates in total, the system still shows tracking ability, but with some degradation in the performance. Therefore, this system can be operated in different modes to achieve the balance between the tracking speed and accuracy. In the high-speed mode, the system handles a less number of candidates for higher speed search, while in the high-accuracy mode, the system takes more time and handles a larger number of candidates. At the working frequency of 60 MHz, the typical processing time for one frame (640×480 pixel-size) is 0.1 ms in the high-speed mode and 0.8 ms in the high-accuracy mode. Such a flexible configuration provides an opportunity

of realizing multiple target tracking function with a fixed number of processing elements, which is discussed in the discussion section.

Data transfer is one of the most important issues in video processing system. Figure. 2.11 illustrates the data bandwidth of the connections between the functional blocks and memories. In the figure, only the processing of one candidate is shown, but all the types of connections in the system are included. It can be observed that, after the image data are transferred to a vector, the quantity of the data becomes very small for computing. In fact, the image data received are only stored partially in registers for pipelined processing. The effective features for predicting the object location are all expressed in vectors for serial transfer and convenient storage. The massive connections can be found in the GFE part, which uses row-parallel processing to reduce computational time. There is a balance between parallelism and implementation, which is discussed in implementation of GFE in detail. In summary, the strategy for data computing in this system contains two aspects: First, the large amount of image data should be transferred to feature vector in an efficient way. Second, achieve balance between parallelism and resource consumption, and limited the massive data transfer in local region.

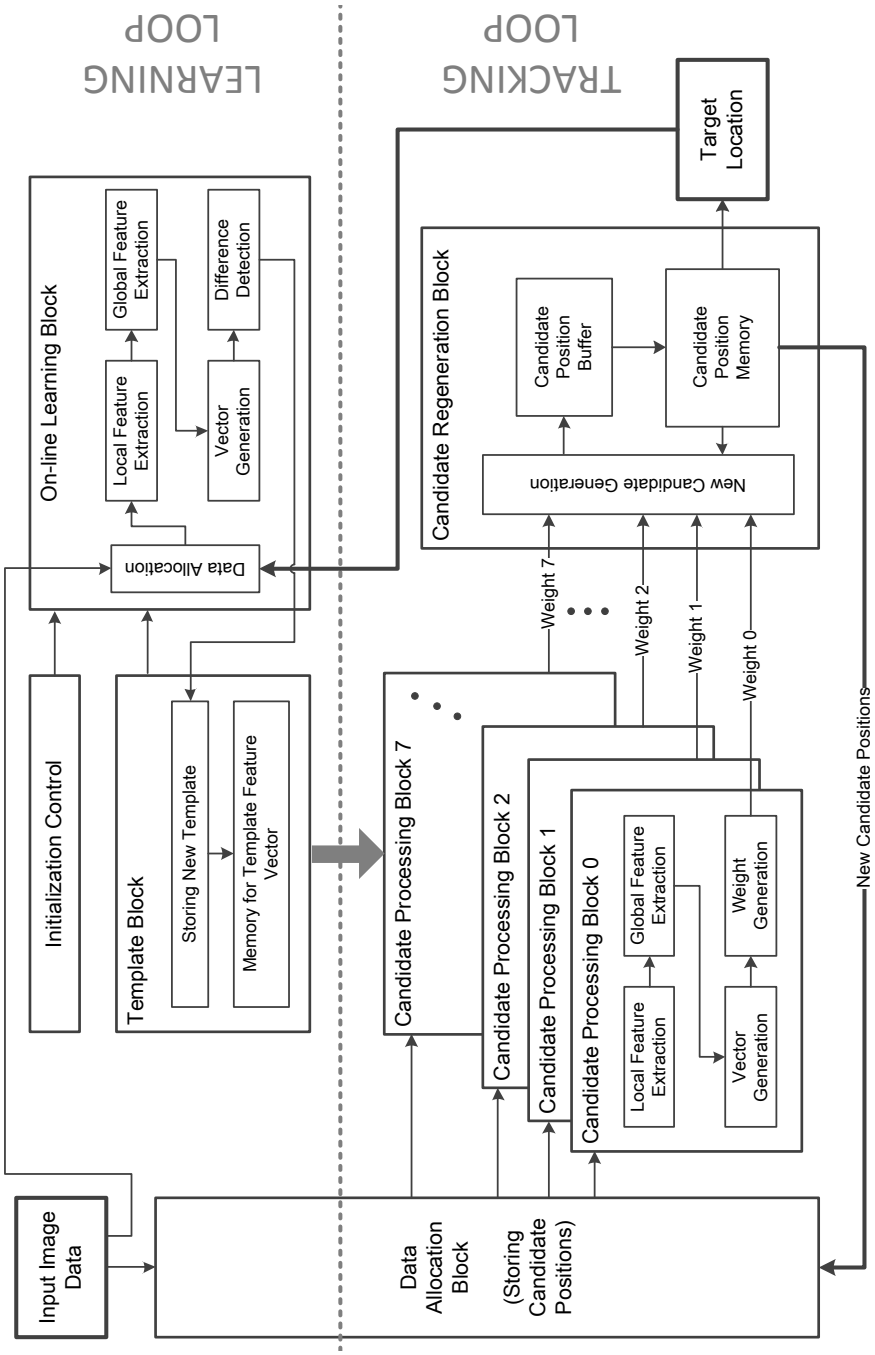


Figure 2.10: Hardware organization of this tracking system. After receiving the image data from camera, this system first allocates the data into corresponding memories. Then eight parallel candidate processing blocks work to process these data in parallel, and output the weight of every candidate. These weight values are used to generate new candidate locations and the target location. The on-line learning block updates the templates according to the tracking result in each iteration.

	Combinational ALUTs	Memory ALUTs	Dedicated logic registers	Time (Clock Cycles)
Edge Map Generator	2253	1264	2175	4626
Vector Generator	553	144	629	64
Weight Generator	541	144	419	64
Candidate Regeneration	7272	0	5073	1024
On-line Learning	5980	1660	8435	6802
Total (Entire System)	75830 (28%)	22504 (17%)	60906 (23%)	-

Table 2.1: FPGA Resource Utilization Summary

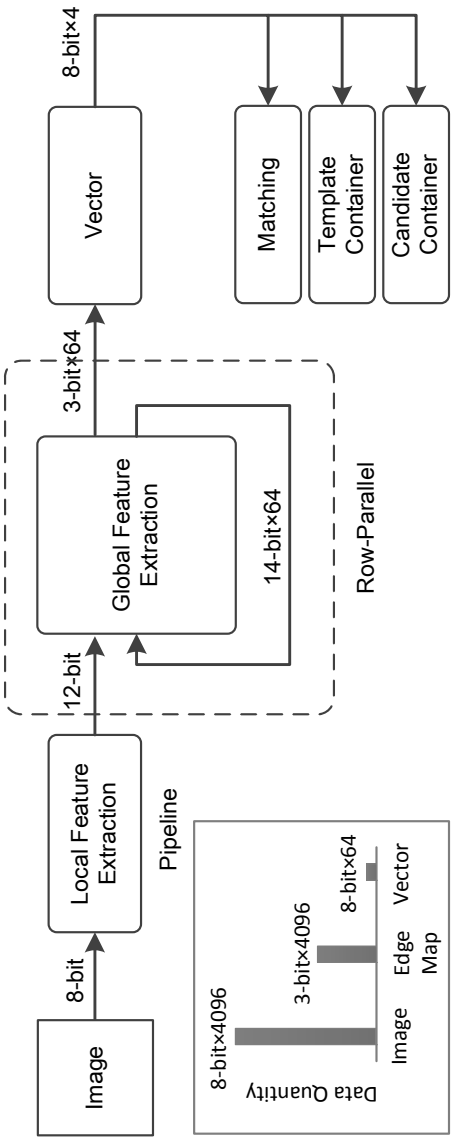


Figure 2.11: A diagram illustrating data transfer in the tracking system (processing of one candidate). It can be observed that, after the image data are transferred to a vector, the quantity of the data becomes very small for computing. The massive connections can be found in the GFE part, which uses row-parallel processing to reduce computational time.

2.4 Experiments

In this section, we evaluated the tracking system by using a group of challenging video sequences, and showed the real-time performance of the system. For evaluation on accuracy, we did the experiments through software simulation. The program was written in a way that every logic operation is same with the implementation on FPGA. From the viewpoint of logic function, the software simulation should give the same results and show the same performance on accuracy with the hardware implementation. For the experiments on the real system in the following part, output of the system was real-time displayed on a monitor screen, and it was recorded by a video camera. The results shown in figures are images extracted from the video. In all the experiments, the parameters are fixed, such as threshold and number of candidates.

2.4.1 Evaluation on Accuracy

In this part, we evaluated the proposed system by using several challenging video sequences from a public database. For comparison, we adopted an evaluation methodology proposed in (67), and compared our system with tracking system in that work. Although this evaluation was made through software simulation, we programmed in the way that every logic operation in program is same with the implementation on FPGA.

In Fig. 2.12, tracking results on these video sequences are shown. The features of these videos are: the Sylvester and David Indoor video sequences present challenging lighting, scale, and pose changes; the Cola Can sequence contains a specular object, which adds some difficulty; the Tiger sequences exhibit many challenges and contain frequent occlusions and fast motion (which causes motion blur); and the Coupon Book clip illustrates a problem that arises when the tracker relies too heavily on the first frame.

In Table 2.2, we show the tracking accuracy at a fixed threshold of 20. The threshold is a distance in pixels. If the distance between the predicted location and the ground truth in one image is larger than the threshold, the prediction in this image is considered as failed. The data in Table 2.2 show the percentage of how many successful predictions are made over the total number of images in

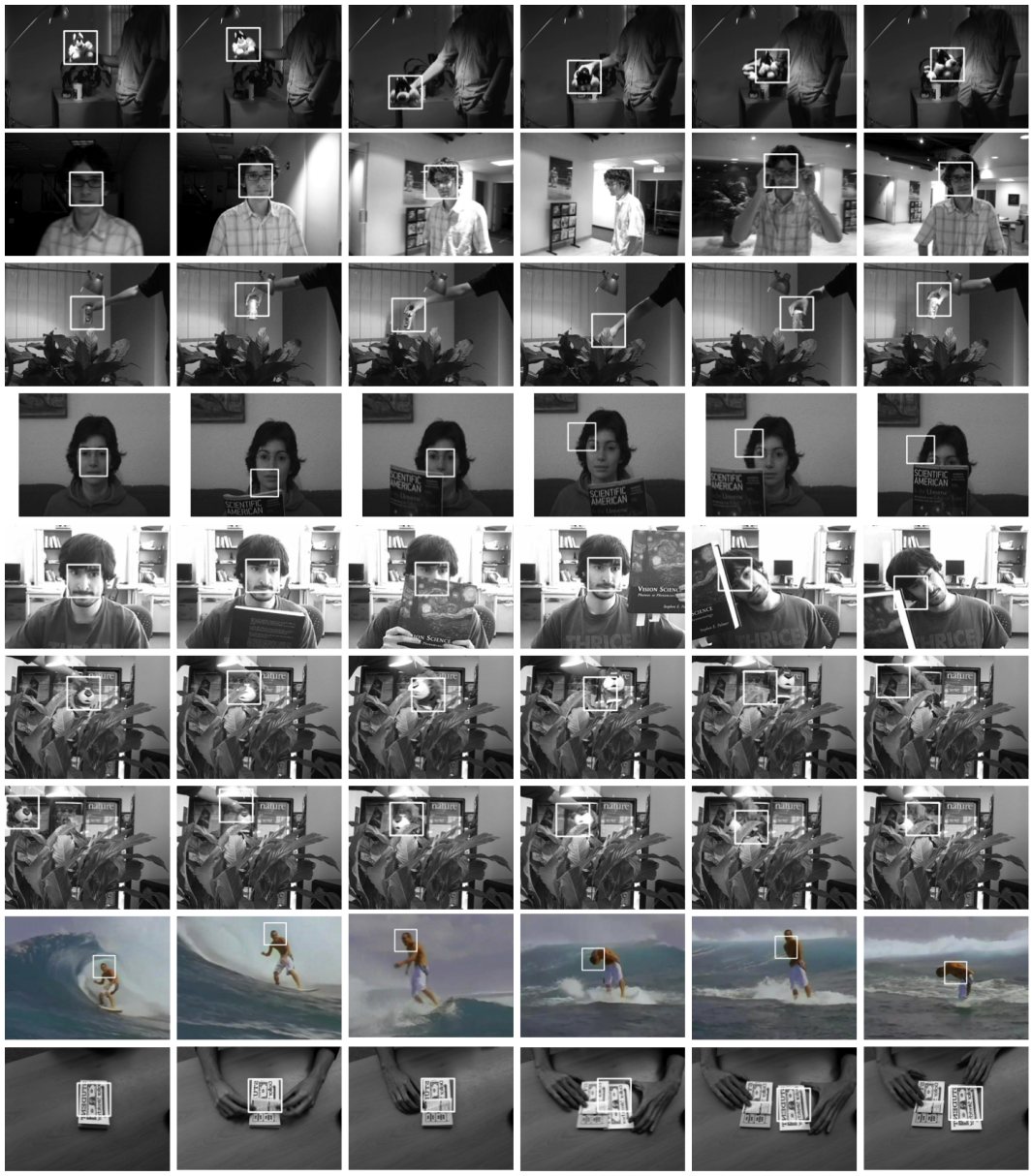


Figure 2.12: Tracking results from software simulation. Video sequences from top to bottom: Sylvester, David Indoor, Cola Can, Occluded Face, Occluded Face 2, Tiger 1, Tiger 2, Surfer, Coupon Book.

a video sequence. Detailed information about this measurement method can be found in(67). Table 2.3 shows evaluation on average location error on various tracking algorithms, including SemiBoost, Frag, MILTrack, DMLTrack(82), and

Table 2.2: Comparisons: Precision at a Fixed Threshold of 20

	OAB	SemiBoost	Frag	MILTrack	This Work
Sylvester	0.64	0.69	0.86	0.90	0.83
David Indoor	0.16	0.46	0.45	0.52	0.88
Cola Can	0.45	0.78	0.14	0.55	0.93
Occluded Face	0.22	0.97	0.95	0.43	0.12
Occluded Face 2	0.61	0.60	0.44	0.60	0.44
Surfer	0.51	0.96	0.28	0.93	0.60
Tiger 1	0.48	0.44	0.28	0.81	0.37
Tiger 2	0.51	0.30	0.22	0.83	0.50
Coupon Book	0.67	0.37	0.41	0.69	0.40

The results show the percentage of how many successful predictions are made over the total number of images in a video sequence.

this work.

In the experiments, this tracking system shows ability of dealing with illumination change, size change, deformation, and partial occlusion. In situations of severe partial occlusion or full occlusion, this system has limitation. This is mainly because the edge feature vector we employed is a global representation, which is sensitive to severe occlusion.

2.4.2 Tracking on FPGA System

We set up the parameters of this system by optimizing them through preliminary experiments, and did not change any parameter during the experiments.

Fig. 2.13 shows the results of an experiment, in which a cup was moved around continuously in a complex circumstance. There is a sudden illumination change

Table 2.3: Comparisons: Average Center Location Errors (pixels)

	SemiBoost	Frag	MILTrack	DMLTrack	This Work
Cola Can	13.09	63.44	20.13	12.84	12.42
Coupon Book	66.59	55.88	14.74	5.68	63.53
Sylvester	15.84	11.12	10.82	9.79	13.16
Tiger 2	61.20	36.64	17.85	31.39	23.80
David Indoor	38.87	46.27	23.12	8.82	12.95
Occluded Face	6.98	6.34	27.23	19.29	46.98
Occluded Face 2	22.86	45.19	20.19	4.97	29.03

on the object, produced by a spotlight coming from the right. The background also gives a disturbing brightness condition: there is light from the left side of the window, while the right side is covered by a window blind. The target changes its appearance and size while moving. According to the results, the system gave stable trace of the target in the complex situation. In this experiment, we turned off the on-line learning function, and stored several templates of the cup in different angles and sizes before tracking. Since in this system the size of the tracking window is fixed, we stored some parts of the target as template when the target size is larger than the window size. The total number of templates was eight.

Fig. 2.14 shows the on-line leaning process of the system. When the hand changed to several different gestures, the system detected the changes and stored the new gestures as templates. After the system learned a sufficient number of templates, it can track the target that moves around and changes its appearance continuously, as shown in Fig. 2.15.

Fig. 2.16 shows the situation where partial occlusion occurs. The target goes behind some obstacle (a chair), and a part of the target is lost from the scene.

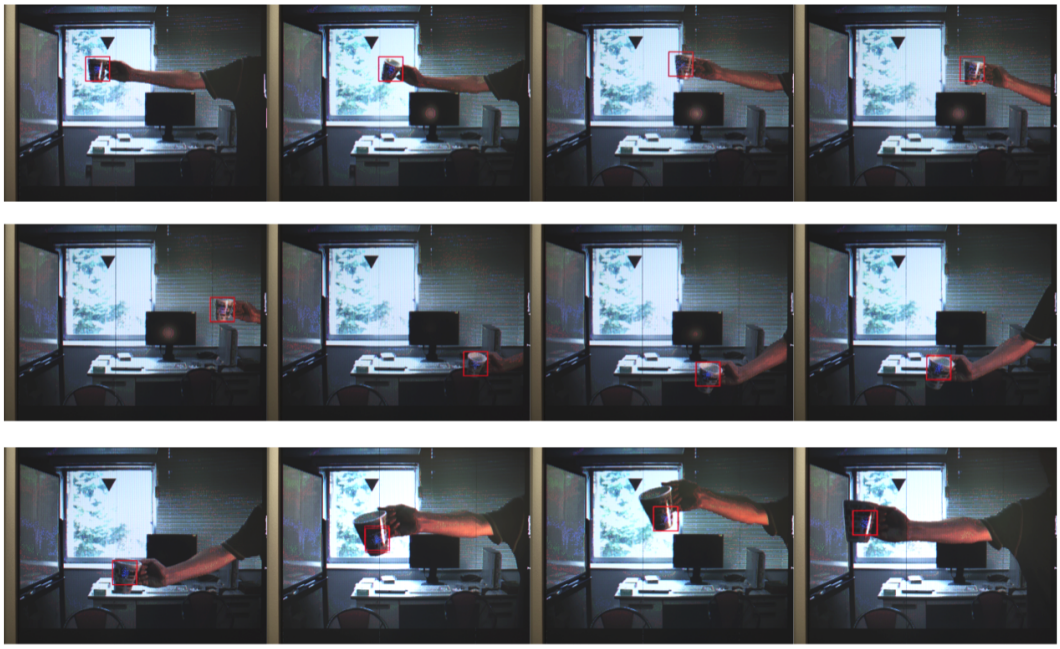


Figure 2.13: Experiment showing tracking of a cup with illumination change and deformation. In this case, the templates are set up before tracking, including appearances of cup in different angles and sizes. The on-line learning function is turned off in this case.

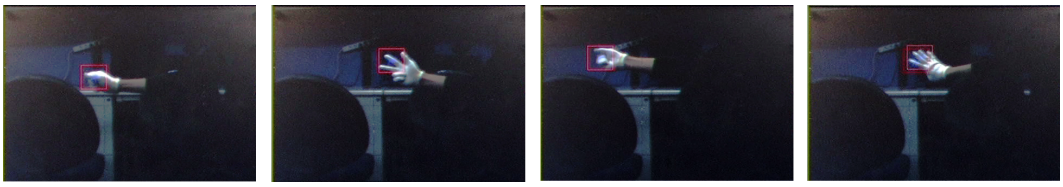


Figure 2.14: On-line learning process. The tracking system stores new templates when the target changes its appearance.

The system learns the object image partially lost by occlusion as a new template and keeps on tracking the target successfully.

In Table 2.4, we compared the performance of the proposed system with three other implementations(51, 52, 57). All three other systems use particle filter as localization method but use different feature representation algorithms. These studies claimed real-time performance, but considering the calculation time for one frame, our method is much faster than the first two systems, and is 40 times

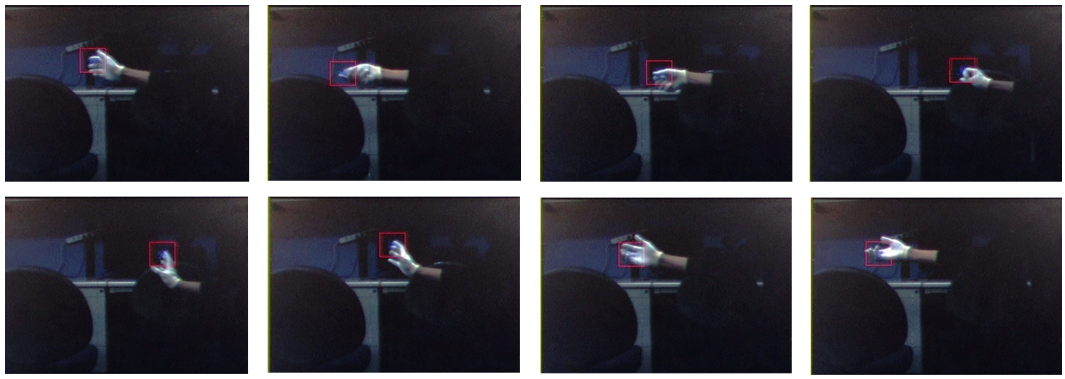


Figure 2.15: Experiment showing the tracking ability with a sufficient number of templates obtained by on-line learning. The system can track the object moving and deforming continuously.

Figure 2.16: Experiment showing the tracking including the partial occlusion of the target.

faster than the third system with the tracking window size 16 times larger. In addition, we also show the frame processing ability, which is in fact limited by the camera and the transmission between camera and processing elements. This common problem can be solved by implementing the image sensor and the processing elements on the same VLSI chip, which is discussed in the VLSI implementation section. We set up the camera to work at 25 f/s. Because the system works faster than the camera, the same frame of image is processed repeatedly (six times every frame in this system) as if it is new frame until the real new frame of image is captured by the camera. Since new candidates are regenerated for the same frame of image in each iteration, this makes the tracking result stable when object moves faster than the movement step of candidates. For this reason, we have claimed the processing ability of the present system is 150 f/s. The implementation of this tracking system on VLSI chips for improved performance is discussed in the following section.

Table 2.4: Comparisons with Three Object Tracking Implementations

	(57)	(52)	(51, 70)	This work
Feature	Local Oriented Energy	Harr-like Feature	Color	Directional Edge
Localization	Particle Filter	Particle Filter	Particle Filter	Multiple Candidate Regeneration
Processing Time	-	32.77 ms	4 ms	0.1 ms
Frame Per Second	30	30	30	150 (25*)
Tracking Window	-	Variable	15×15	64×64
Image Resolution	640×480	320×240	256×240	640×480
Implementation	FPGA	Cell/B.E.	SIMD processor	FPGA

* Limited to 25 f/s by image capturing and transmission to the FPGA. All other processing operates at 150 f/s.

See text.

2.5 Summary

In this section, we have proposed a real-time object tracking system, which is based on the multiple candidate-location generation mechanism. The system employs the directional-edge-based image features and also an online learning algorithm for robust tracking performance. Since the design of this algorithm is hardware-friendly, we designed and implemented the real-time system on FPGA, which has the ability of processing a 640×480 resolution image in about 0.1 ms. It achieved 150 f/s frame rate on FPGA, and can reach about 900 f/s if implemented on VLSI with on-chip image sensor. Evaluation of the tracking system on both accuracy and speed are shown and discussed, which clarify the features of this system. This section also presents a detailed discussion on several issues of tracking, including VLSI chip implementation for faster operation, multiple target tracking, initialization problem, and full occlusion problem. The solutions presented in the discussion part are based on our hardware system, and will give solutions in real-time applications.

3

Extending Tracking Functions

3.1 Introduction

Basically speaking, the purpose of object tracking is to predict object location in every frame of image. However, because of the complexity of practical applications, there are various requirements besides the simple location information. For example, multiple object tracking is a very common situation. In some applications, more information, such as size and orientation of the object, is also important. In some tracking task, the object may disappear from the scene or be occluded by other objects. It is important to design a method to search the missing object. Therefore, in order to fulfill different requirements, the flexibility and extensibility of the system become very important. In this part, extension of the tracking system is discussed by discussing five practical solutions: VLSI implementation, multiple object tracking, initialization, full occlusion, and state vector.

3.2 VLSI Implementation

The object tracking system of my work is implemented on FPGA, with an external CCD camera. However, the design of the algorithm and hardware architecture is compatible with implementation on VLSI technology. That means it is not required to do additional work on the architecture in order to build the tracking

system on a VLSI chip, although it could be more complex in VLSI design process. In fact, while using VLSI technology, it is obvious that the performance of the tracking system can be improved because of the flexibility of VLSI design, for example, even the image sensor can be implemented on the same chip for high-processing speed. Several differences are discussed in the following.

From the analysis of the computational time of this system, it was found that most of the time is consumed in waiting for image data input and doing feature extraction computation. This is because we cannot process the image information from camera efficiently due to the data transmission limitation from the camera to the FPGA. This problem can be resolved if the algorithm is directly implemented on VLSI chips. If this algorithm is implemented with a high performance image sensor, the performance will improve greatly. In fact, a VLSI processor has been developed for the object recognition purpose that is composed of image sensor and the feature extraction block based on the same algorithm employed in this system (77). For a 68×68 image, that processor is capable of reading image data directly from the on-chip image sensor array and calculate the intensity gradients in a row-parallel way.

The global feature extraction (GFE) is a time consuming process, which is implemented in row-parallel way on FPGA. The limitation is that this part contains a parallel adder which will sum 4096 one-bit data selected from 4096 15-bit data. This architecture requires massive connections among registers, adders and multipliers. It is not a easy work, nor an efficient solution for FPGA implementation. Therefore, the summation process is divided into 64 steps, a row of data are added up in each step. If this architecture is implemented in fully parallel way on VLSI, the GFE part can be finished in only 11 clock cycles while it needs 960 cycles in this system. Therefore, a nearly 6 times decrease in the computational time and a higher frame rate can be expected by integrating the tracking system developed in this work directly on the chip of (77).

3.3 Multiple Target Tracking

Multiple target tracking is in great demand in certain applications. The human brain acquired such an essential ability through evolution. Although the multiple

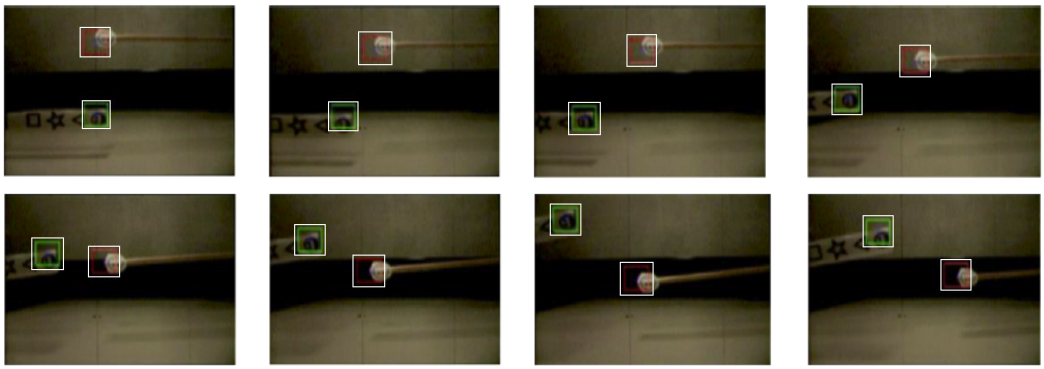


Figure 3.1: Experiment on two-target tracking. In this experiment, each target had a template container, a candidate container, and 32 processing elements. Locations of the targets were initialized separately in the first frame. The result shows that the system can track two different objects well without using additional memory or processing elements.

target tracking mechanism in the human brain is not yet known, in Pylyshyn and Storm's research (83) a widely accepted theory has been proposed, which is well supported by experiments. Their data showed that participants can successfully track a subset of up to five targets from a set of ten, and both accuracy and reaction times decline with increasing numbers of targets. One possible interpretation of their findings is that targets are being tracked by a strictly parallel preattentive process with limited resources.

In hardware systems, the problem of limited processing resources always exists, especially for the real-time applications. In our system, this problem is resolved by flexibly allocating the processing elements to multiple targets. Based on the implemented tracking system, we verified the tracking mechanism really works for a two-target experiment. In the experiment, there were two targets with their own templates. The 64 candidates were allocated to the two targets, and the tracking process was the same as the single-target tracking. The experimental result in Fig. 3.1 shows that the system still tracks the targets successfully in multi-target tracking with limited resources. In different tracking applications with different number of targets and hardware resources, the mechanism of the proposed system can provide flexible and highly efficient solutions.

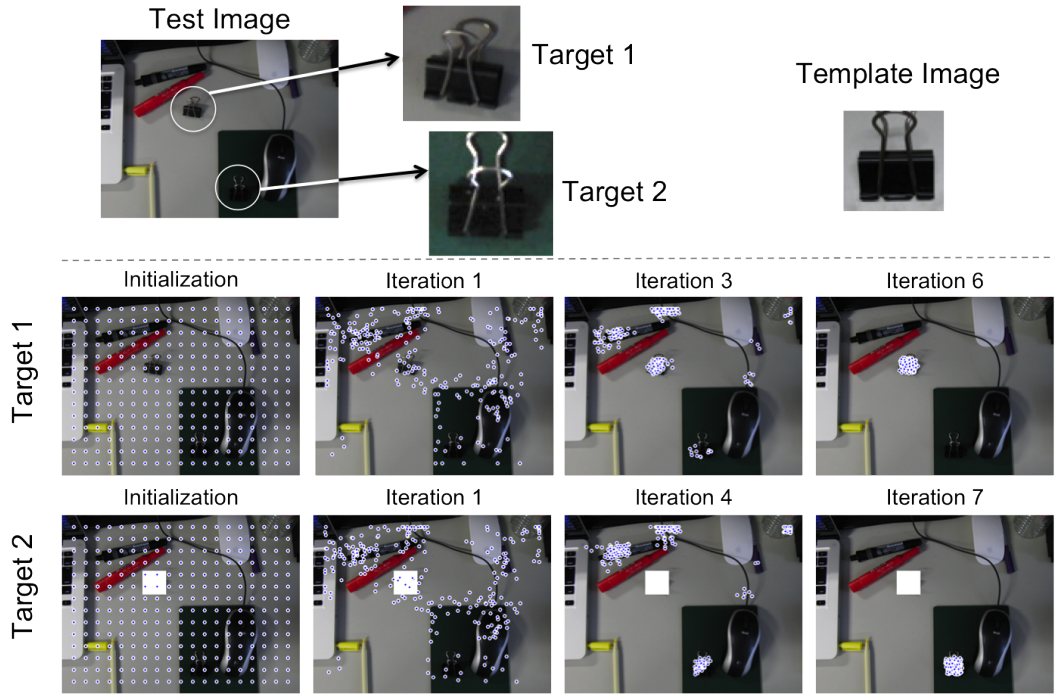


Figure 3.2: Process of searching two targets in an image based on software simulation. Images in the first row show the candidate distribution in each iteration, and the location of one object is detected as shown in the right most image. Images in the second row show the candidates distribution after giving a feedback suppression to the original image. All candidates are initialized to the default location again, and go to the location of the second object after eight iterations.

3.4 System Initialization

Initialization is a critical problem of the searching-based tracking algorithms. However, because this tracking system is not designed for a specific application, it is not easy to adopt a fixed method as initialization. In this paper and some other tracking works, the proposed solutions focus on the situation that only the object location in the first frame of image is known. The manual initialization is also used for fair evaluation on the tracking system under same condition with other works.

For initialization, we briefly discussed one possible solution, a searching method, for the situation that the system has already stored templates of the target to be

tracked, and will first try to find the initial location before tracking. In different applications, the requirements can be very different. For example, it is possible that some application requires detecting the movement first in the scene (the system has no idea about what the target is), and then determines the initial location and target by itself. For the proposed tracking system, if the performance of the system meets the requirements, it can be used in practical applications with configuration and specific initialization method.

The algorithm adopted in this tracking system is based on a regeneration mechanism. Therefore, the initial target location must be specified manually. In this subsection, it is explained that the problem of initialization can also be resolved by employing the multiple candidate regeneration mechanism developed in this work. The merit of this solution is that it does not need any additional resources except for some simple logic elements.

In our previous experiments, initialization is done by setting up the target location manually before the tracking starts. However, there are some other requirements in different applications. For example, in some case, the system has already possessed some templates of the target and starts tracking when the target appears in the scene. Then the system must search for the target first using the templates, and when it is found, the system use this location as the initial location. We focus on this situation here, and propose a solution to the initialization problem in the following.

Firstly, the image is divided into half-overlapped sub-regions having the same size with the tracking window. Secondly, all such sub-regions are treated as candidates in the multiple candidate regeneration. Similar to the tracking mechanism, all the candidates accumulate at the target location after several iterations. At last, the location determined by the candidates is stored as initial location. In the multi-target situation, the target that is already found is suppressed by giving a feedback to the image. Namely, the target found in the first round is blanked by masking. Then, the searching operation explained above is repeated to find the second target. In this iteration, the candidates accumulate at the second target location naturally. An example of finding the initial locations of two black clips in an image is shown in Fig. 3.2.

For a searching task, the simplest way is to do template matching at every location in the image. This exhaustive searching strategy needs a lot of computation, especially for a large size image. In this regard, the computational complexity has been reduced significantly by this solution because only the candidate images are processed for searching.

3.5 Full Occlusion

Full occlusion is a very challenging problem, which may cause the tracker lose the target. This is because the tracking algorithm uses the previous target location as important information. The same searching mechanism explained in the initialization using multiple candidate regeneration can also be used to solve this problem. When a certain condition is met while tracking (for instance, difference between the current target image and the templates is larger than a certain threshold), the tracking system enters the searching mode, and keep searching the target as it does in the initialization. When the system finds the target, it returns to the tracking mode. In this way, a target is found and tracked in real-time even after disappearing for some time from the scene.

3.6 State Vector

State vector is a concept commonly used to describe the state (attributes) of an object in object tracking algorithm. The simplest state vector can have only two elements: the coordinates (x, y). In many situations, the tracker not only outputs the location of the object, but also describes other information, such as object size, orientation, and appearance. For example, SIFT descriptor uses features with size and orientation parameters. Therefore, tracking algorithm employing SIFT descriptor can tell the size and orientation of the object from the features. In our system, it is simple to extend the tracking function for more useful information. In the following, a solution to this problem, which only needs adding some preprocessing into the system, is discussed.

In our tracking system, the object image is represented by a feature vector, which contains only the distribution of edge information. In order to describe

the state of the object, a state vector containing location, size and orientation has been adopted. The vector has four elements: location x, location y, size, and orientation, represented as x, y, s, o. The size is a real number with range from 0.0 to 5.0, which represent the ratio of current object size to the original object size. In this case, 1.0 means the original size. The orientation is an integer number ranging from 0 to 359, which represents the degree of orientation angle.

In order to update the additional information about the object based the developed system a specific preprocessing block is needed. Every time before extracting the image for each candidate from a frame of image, besides considering the location the size and orientation are also taken into account. In this case, the image extracted from the scene will be an image with candidate window size. Then the sub-image is rotated by the degree of the orientation angle. At last, the image is resized to the original size as normalization, because it is necessary for calculating APED. After the preprocessing, the final result is still a sub-image with normal size, but it contains more information about the object. The predicting process is same, but for updating the state vector, all the elements in state vector are updated (generating new candidates). Two experiments are performed to verify the usage of state vector. The tracking result is shown in Fig. 3.3 and Fig. 3.4.

3.7 Summary

Several extensions of the proposed tracking system are discussed. The most important feature is that all these extensions are based on the same architecture that has already been developed. The core part is the processing element called candidate. Using the same architecture VLSI implementation is possible, and will show better performance than implementation on FPGA. Multiple object tracking is realized by controlling the candidates, allocating them to different objects. An candidate-based object detection method is proposed to realize initialization and find missing object when full occlusion happens. Finally, state vector is employed in our tracking system. The candidates are distributed in a multiple dimensional feature space, and more information about the object, such as object size, can be achieved. Experiments have been performed to verify these

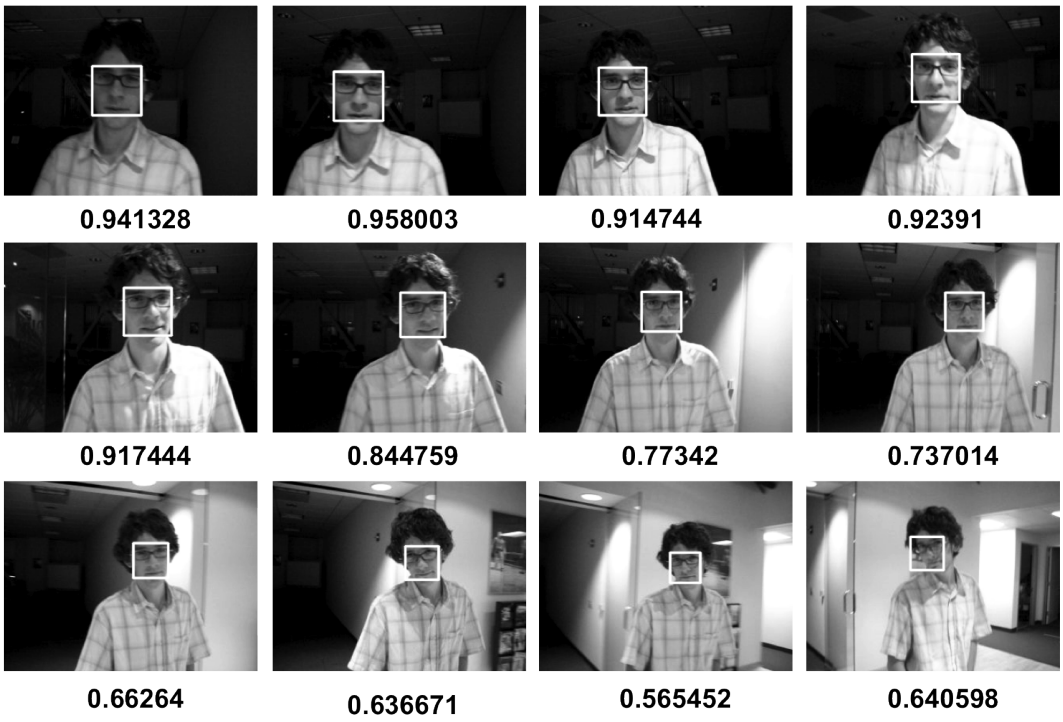


Figure 3.3: Experiment on tracking with size feature. The size value of each object state is shown under the frame of image.

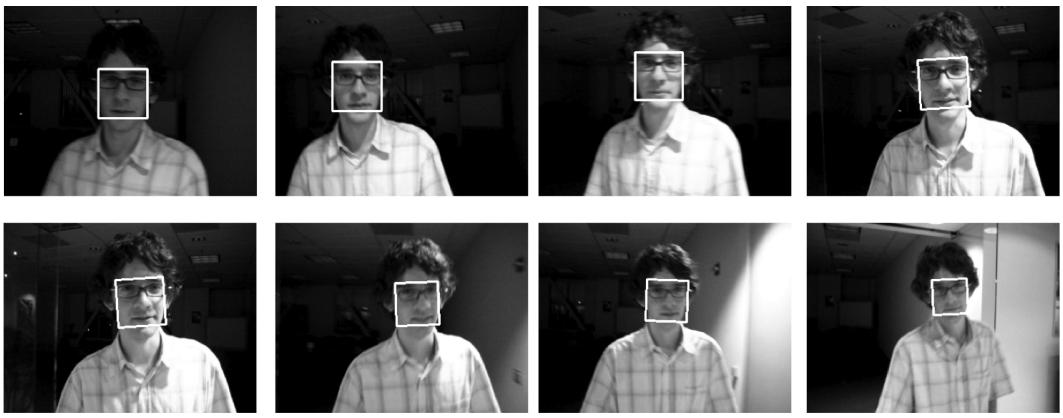


Figure 3.4: Experiment on tracking with both size and orientation feature.

methods. More functions can be expected because of the flexibility of the object tracking system.

4

A Real-Time Object Tracking System with Online Learning Support Vector Machines

4.1 Introduction

Object tracking is a critical and well-studied problem with many practical applications. A number of algorithms have been developed based on various mechanisms. One promising direction is to consider the object tracking as a binary classification problem, and employ discriminative methods in the tracking framework. Support vector machine (SVM), as a powerful classification scheme, has been used in many tracking algorithms, benefiting the algorithms with accurate localization and flexible modeling of the target (66, 84, 85). In Fig. 4.1, some sub-images extracted from a video are shown. Supposing that in every frame of image, images from some candidate locations are extracted. Some of these images belong to the object category (human face in this case), and the others belong to a combinational class - background. It is straightforward that if we can find a way to tell which of the images belong to the object, then we can use the location information of the object images to estimate the real object location. The function of discriminating images can be realized by using SVM classifier, as shown in Fig. 4.2. Fig. 4.2 shows the basic concept of SVM and the training process. Given a set of training examples (images), each marked as belonging to one of two

Figure 4.1: Illustration of the sub-images extracted from a scene. These images can be divided into two classes: target class and background class.

categories (object and background), the SVM training algorithm builds a model that assigns new examples (images) into one category or the other, as shown in Fig. 4.4. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a hyperplane with a gap that is as wide as possible. Fig. 4.3 shows the hyperplane and the support vectors. Boundary is a linear combination of all the examples along the gap, which are called support vectors. In order to set up a classifier, only the support vectors are necessary to store in the memory after training process. The SVM works as an appearance model of the target by changing its boundary while training. One feature of SVM is that the boundary is represented by the combination of support vectors, and the number of support vectors is usually a small portion of the total training dataset. This feature becomes very important when implementing the tracking algorithm on hardware, because the hardware resources are always limited.

Despite the good performance of these algorithms, they suffer from several practical problems. The work in (66) builds a superior SVM classifier and gives good results in tracking vehicles. However, the off-line training mechanism employed in the work requires a large number of training samples selected manually and does not support updating the training samples. In (84), all samples learned

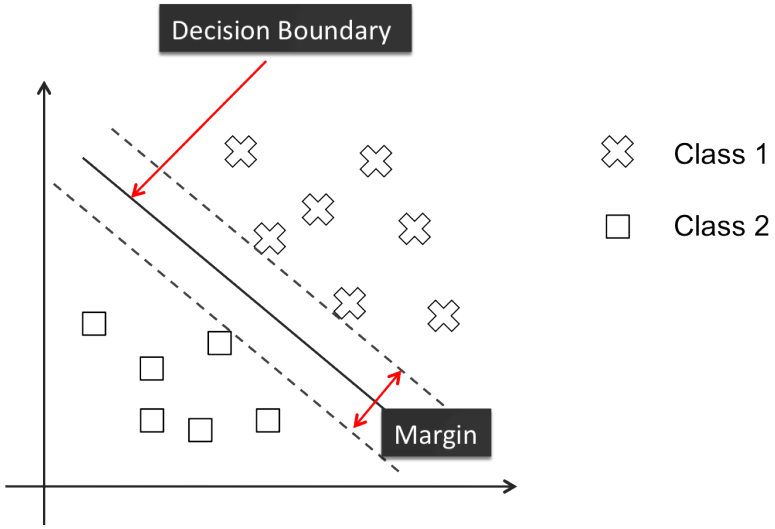


Figure 4.2: Illustration of basic concept of support vector machines.

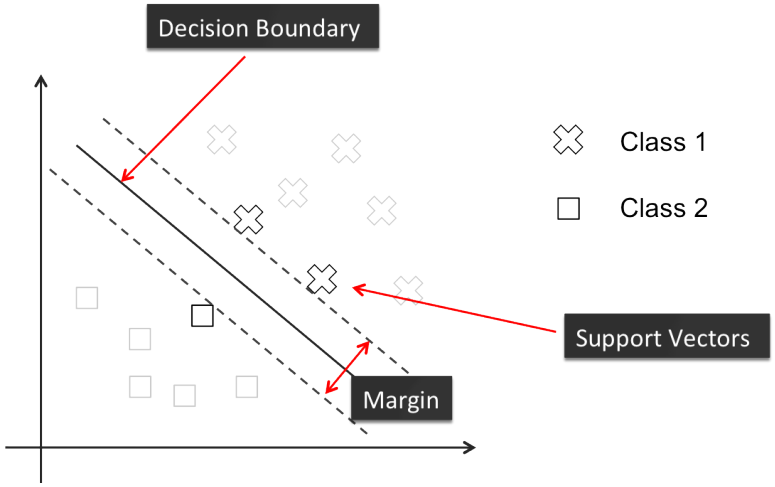


Figure 4.3: Illustration of basic concept of support vector machines.

from each frame of an image sequence are stored for training the SVM. This causes a large memory cost if it is used in a long-duration task. In (85), a simple strategy is employed to determine new training samples, which may cause "drift problem" as described in (86). Moreover, these algorithms do not consider their real-time performances, which is in fact of great importance in object tracking applications. This is mainly because of the complex computation of SVM. Especially for the on-line learning SVMs, frequently repeated training and predictions make this problem even worse. Therefore, in order to extend the power of SVM in most of the general tracking applications, it is necessary to develop a proper tracking framework and a VLSI-hardware-implementation-friendly structure for the SVM-based algorithm.

The tracking framework includes how to update training samples and how to select test samples and make prediction of the target location. In this work, a new improved tracking framework is proposed. Different from other algorithms, this framework gives a rule guiding the selection of target training samples. When the target changes its appearance significantly, the system may fail to localize the target because the classifier misclassifies the target image to the background image category. In order to solve this problem, background samples are utilized to predict the location of the target image. Unlike the moving target image, most of the background sample images are stable. As a result, high-accuracy tracking has been established. In addition, regarding the selection of target samples for on-line training of SVM, a new selection rule has been introduced.

As mentioned, the on-line SVM learning requires repeated training and predicting. The predicting process always contains computation of thousands of test samples in conventional algorithms, preventing these algorithms from working in real-time. In this process, not only the SVM, but also the feature extraction of each sample will cost lots of time. Based on a SVM chip developed in (87), the most complex part in this algorithm can be computed efficiently. At the same time, multiple candidate regeneration (79) is employed to reduce the computational cost without sacrificing the tracking accuracy. In addition, the directional-edge-feature vector representation (1), whose VLSI implementation has been proposed in (77), is employed to represent the sample images. By using this hardware-friendly structure, real-time tracking ability can be achieved.

4.2 Object Tracking Algorithm

4.2.1 On-line Learning SVM

The basic mechanism of on-line learning SVM is to train the SVM classifier repeatedly with new training samples. How to update the training samples affects the training result greatly. In (88) a training strategy is proposed, in which in each iteration of training the support vector machine is trained by a new data set and the support vectors determined by the previous learning iteration. The reasoning behind this is that the resulting decision function of an SVM depends only on its support vectors so that training an SVM on the support vectors alone results in the same decision function as training on the whole data set. This strategy is suitable to implement on hardware because of its simplicity. The following researches claimed that this strategy only give a proximate learning result (89, 90), and proposed accurate approaches dealing with huge amount of data. Although the computational complexity has been decreased in these strategies, massive memory is necessary to store all the data. In this work, we took advantage of the strategy in (88) and designed an on-line learning strategy for object tracking, which is explained in the following.

An illustration of the on-line learning is shown in Fig. 4.4. After training the SVM, we build a classifier with a boundary. It is difficult to collect all the necessary examples before training, so after some time of tracking, there may be some new examples from both the object and background. In this case, we need to determine how to modify the decision boundary according to the new examples, which is usually called incremental learning. One approximated method is to retain the support vectors in current SVM, add new examples to the support vectors, and then train the SVM again. As illustrated in the figure, the decision boundary grows with the increase of new examples. One important feature of this process is that the method focuses on the current change of decision boundary as priority. This is very convenient in object tracking application, because the purpose of tracking is discriminate object from background at present time. Fig. 4.5 shows the difference between our learning process and the conventional learning process. In the conventional learning process, the SVM is trained only once by

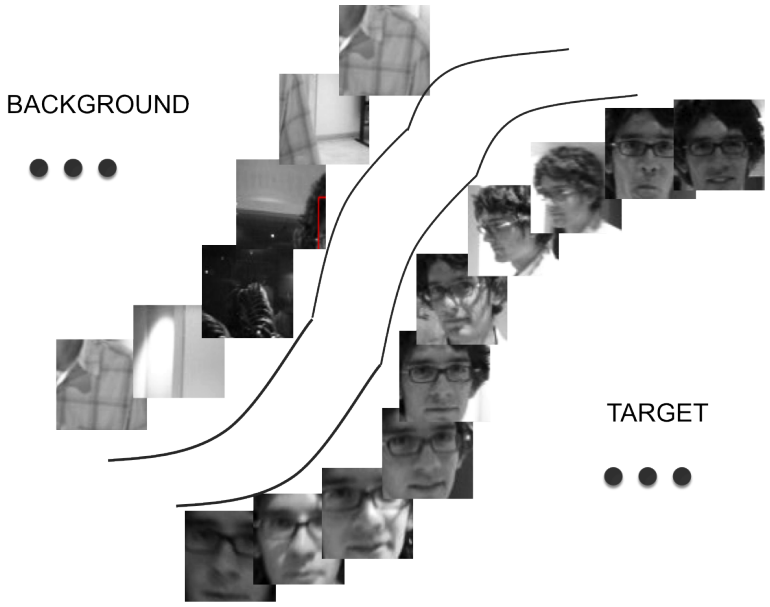


Figure 4.4: Illustration of the learning strategy based on support vector machines.

a large database. It is necessary to build this sufficient database first, and after training, it is difficult to update the SVM classifier. However, tracking application usually contains unpredictable change, such as object appearance change and background change. In our on-line learning strategy, the SVM classifier is trained for every frame of image. In each iteration, the algorithm determines new examples autonomously. The important advantages are: First, it avoids building larger database. Second, it updates along with the change of object and scene. Third, in the on-line learning process, the number of support vectors usually stays below a small number, which is very important for hardware implementation.

4.2.2 Training Framework

Fig. 4.6(a) shows the basic configuration of the on-line learning SVM in tracking algorithms. In this work, sample images are represented by directional feature vector proposed in (1). At the beginning, the SVM is trained by labeled samples from previous iteration or the initialization stage. Then, in the present image, test samples extracted from a certain region (shown as square in 4.6(b)) are classified by the trained SVM into two classes: the target and the background. Then a

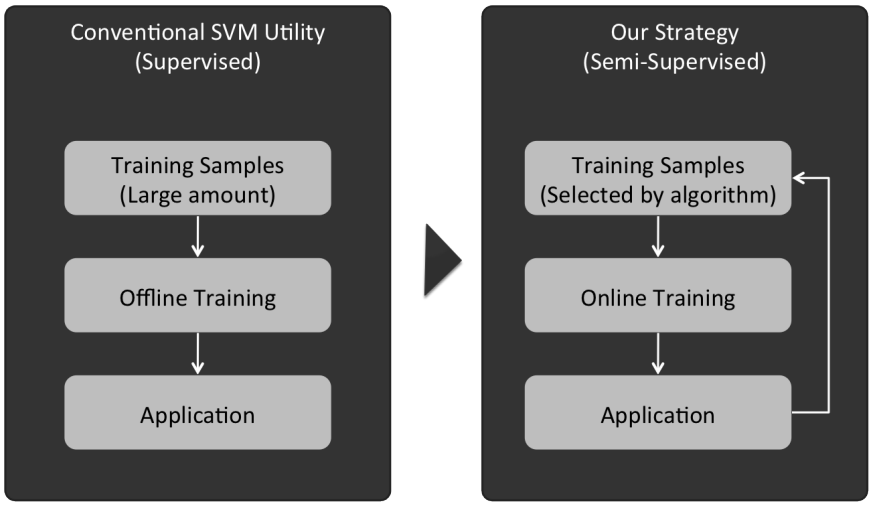


Figure 4.5: Difference between our learning strategy and conventional learning strategy.

confidence map is generated (shown on the top left corner of the image) using the locations of the test samples and their decision function values. The target location is predicted based on the confidence map. After this step, new samples for training in the next frame are selected and the next iteration starts.

The following parts focus on how to select reliable training samples in each iteration. The training samples which are not support vectors are discarded in the strategy in (88) to remove redundant training samples. In this work, the same rule is applied to the background training samples. However, the target training samples are all stored and never discarded, because the target samples are more important than the background samples in the tracking application, and the quantity of the target samples does not explode as the background samples do in complex backgrounds.

Besides the support vectors retained as mentioned above, new training samples from each frame of image are also added to the background training samples. Suppose that the algorithm has finished predicting the target location in the present frame. Then, several image patches around the predicted target location are stored as new background samples, as shown in Fig. 4.7. This is based on the assumption that in the next frame, the background images at these locations tend to be critical distracters. Together with the background support vectors,

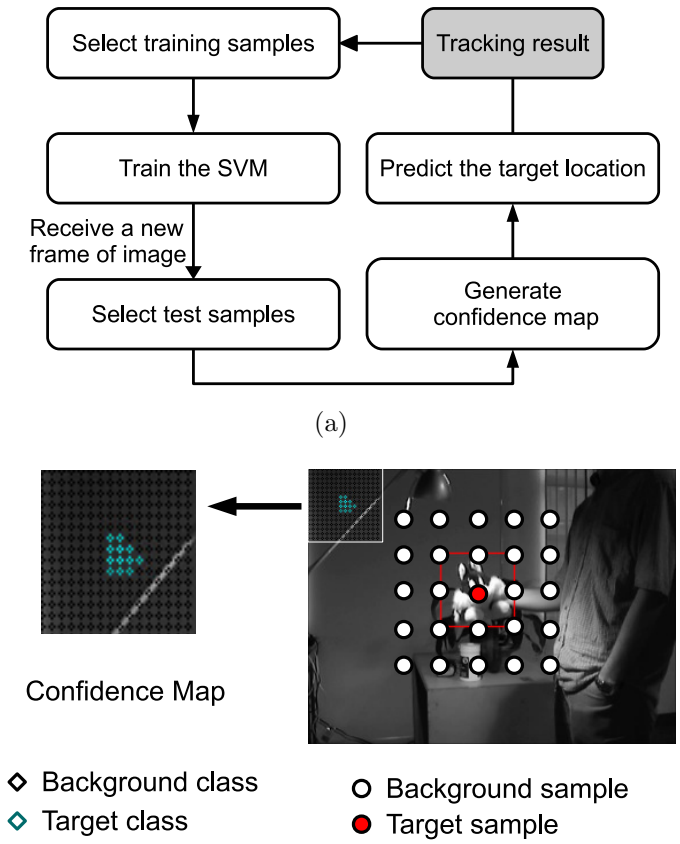


Figure 4.6: Basic mechanism of the online learning SVM-based tracking algorithm: (a) Training samples and the confidence map and (b) Basic process of the algorithm.

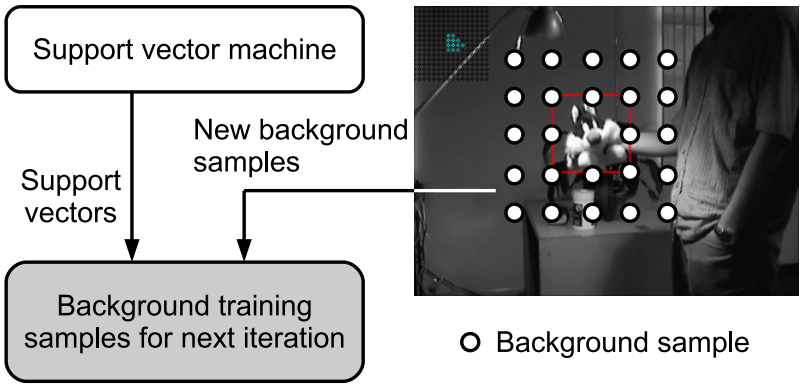


Figure 4.7: Selection of the background training samples.

the newly generated samples are stored as new background training samples for the next frame.

For the target samples, we developed an approach to select reliable new training samples. Different from the conventional algorithms, the image patch at the predicted target location in each frame is not added as new training sample. Since the prediction of the target location always has small location error, adding the image patches brings inaccurate training samples into the SVM. The error may accumulate after a long time of tracking, and cause the tracker drift away from the real target. In this work, whether to add a new target training sample is determined by the decision function values of the test samples, as shown in Fig. 4.8. If there are candidates with large function values in target class, it indicates the knowledge in the SVM is sufficient at present and no new target sample is added. However, when the target changes its appearance, it may happen that the decision function values of the samples in the target class become very small, or even there is no test samples falling in the target class. In this situation, it is difficult to predict the target location in the present frame, which also means new target sample should be learnt. From the confidence map in Fig. 4.8, it can be observed that although the target images are not found (no samples are classified into the target class), most of the background images are classified with large confidence values (shown in brighter color). Among the background samples, some samples own small confidence values (shown in darker color) because a certain region of the background is occluded by the target. Based on this assumption, the center

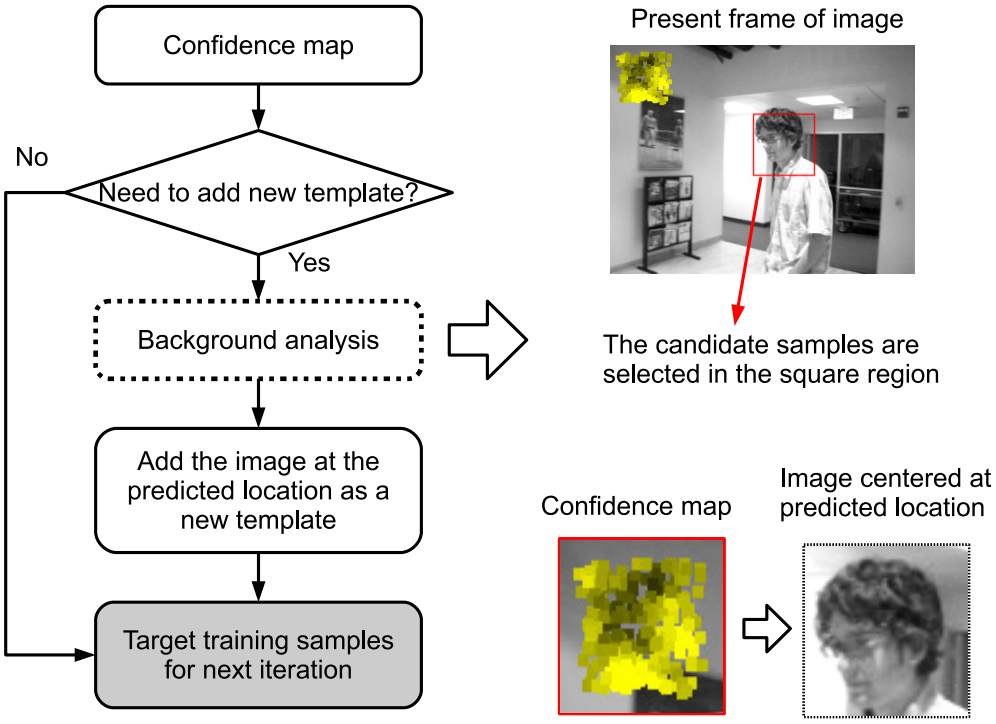


Figure 4.8: Selection of the target training samples.

of gravity of the background samples with small confidence values is calculated, and the image at this location is stored as a new target training sample.

In summary, the training samples are composed of the initial target sample, the target samples generated by background analysis, background support vectors and new back ground samples generated in the present frame.

4.2.3 Multiple Candidate Regeneration

The multiple candidate regeneration (MCR) is first proposed in (79) as a solution of real-time object tracking. It is a statistical approach which is similar with the particle filter, but simplified for hardware implementation. It has shown good performance in solving the object tracking problem. This work employs the MCR to determine the test samples in order to reduce the computational cost in the predicting process.

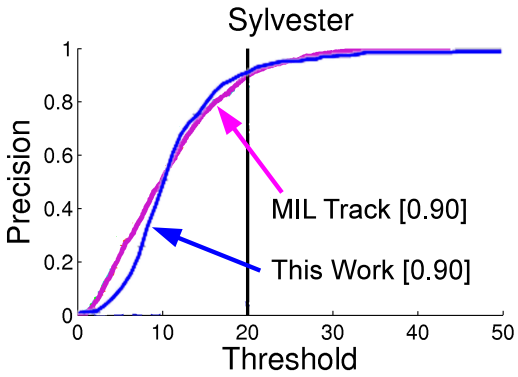
The candidates in the MCR are used to represent the test samples. The candidates are all distributed around the previous target location. In the present

frame of image, feature vectors are generated from the candidate images centered at these candidate locations and sent to SVM as test samples. The function values returned by the SVM are used to calculate weight values for the candidates. The candidate with larger decision function value is assigned a weight with larger absolute value. The positive weight and negative weight stand for the target candidate and the background candidate, respectively. Then, new candidates are generated based on the criteria as follows: in each iteration, candidates with large weight value have more new candidates in their vicinity and the total number of the candidates is constant. The target location in present frame is predicted based on the distribution of the new candidate locations. In this work, by using this approach with 256 candidates the quantity of test samples becomes much smaller compared with the conventional algorithms that generate test sample at every pixel location in the image. Detailed description of this approach can be found in (79).

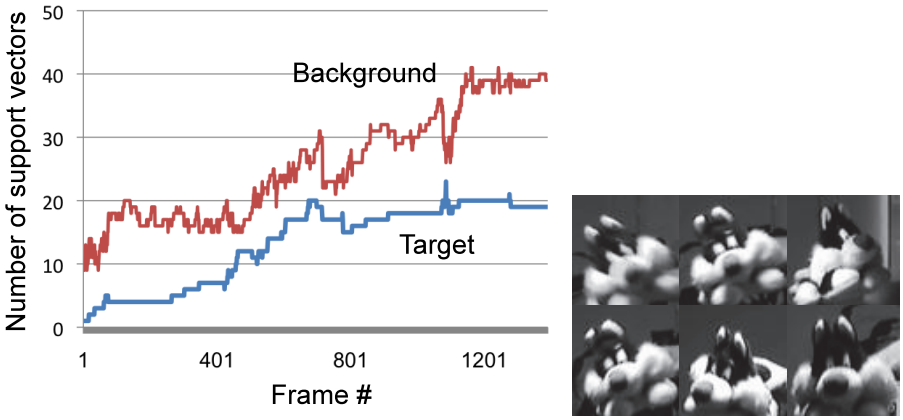
4.3 Experiments and Verification

In this section experimental results are shown, including the simulation results, evaluations on accuracy and number of support vectors. The video sequences and the evaluation method are proposed in (67). The proposed algorithm showed robust tracking ability with accurate tracking results in the experiments. Fig. 4.8(b) and Fig. 4.9(b) compare this work with the algorithm proposed in (67) on two video sequences. Under the threshold of 20, this algorithm achieved 0.90 and 0.95 accuracy in Sylvester and David video sequence, respectively. The number of support vectors stayed under a small value as expected, and the total number of target samples is 68 and 82, respectively. Some examples of the target samples are also shown. More experimental results on publicly available videos are shown in Table 4.1. The algorithm shows high-accuracy tracking ability in the experiments. Since this algorithm does not contain specific solution to the occlusion between objects, it does not give accurate results in the two face occlusion videos, in which the target (human face) is partially or almost fully occluded by a book. More results are shown in Fig. 4.12.

(a)



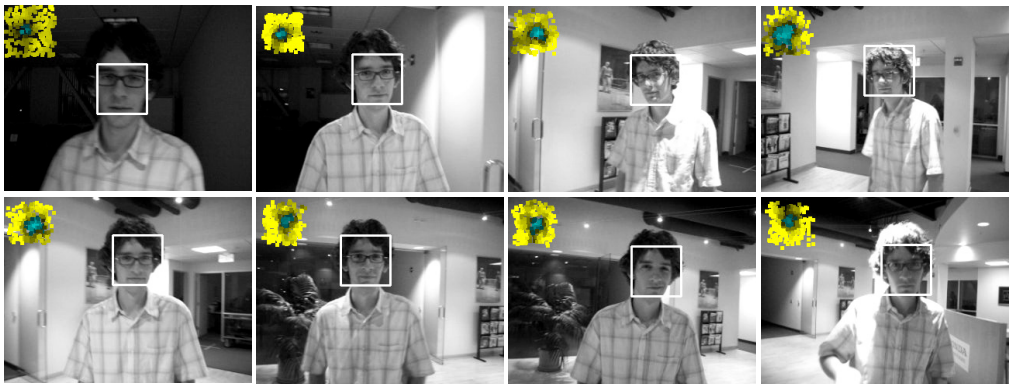
(b)



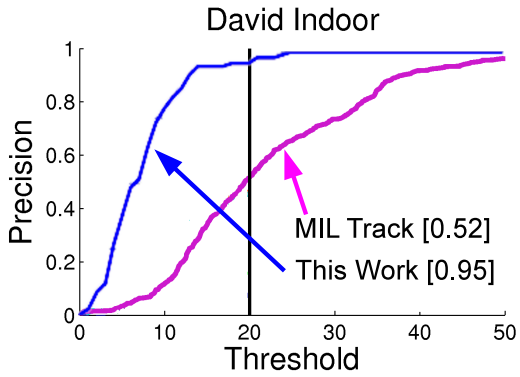
(c)

(d)

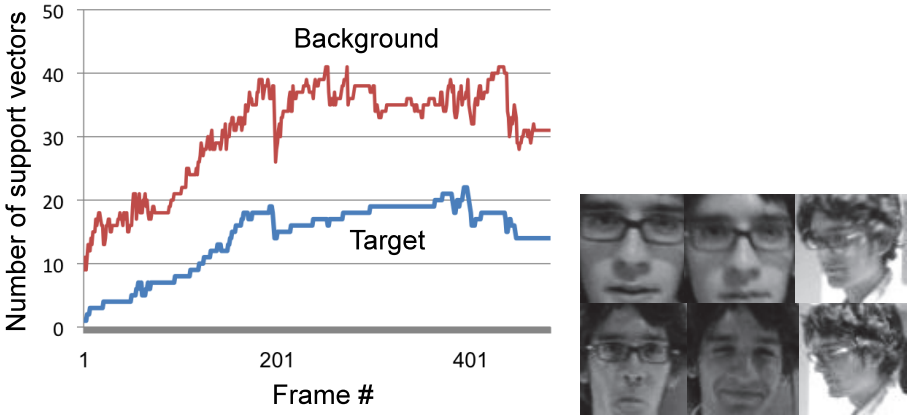
Figure 4.9: Evaluation of this algorithm with Sylvester video sequences: (a) tracking result; (b) precision evaluation; (c) number of support vectors and (d) examples of target training samples.



(a)



(b)



(c) (d)

Figure 4.10: Evaluation of this algorithm with David video sequences: (a) tracking result; (b) precision evaluation; (c) number of support vectors and (d) examples of target training samples.

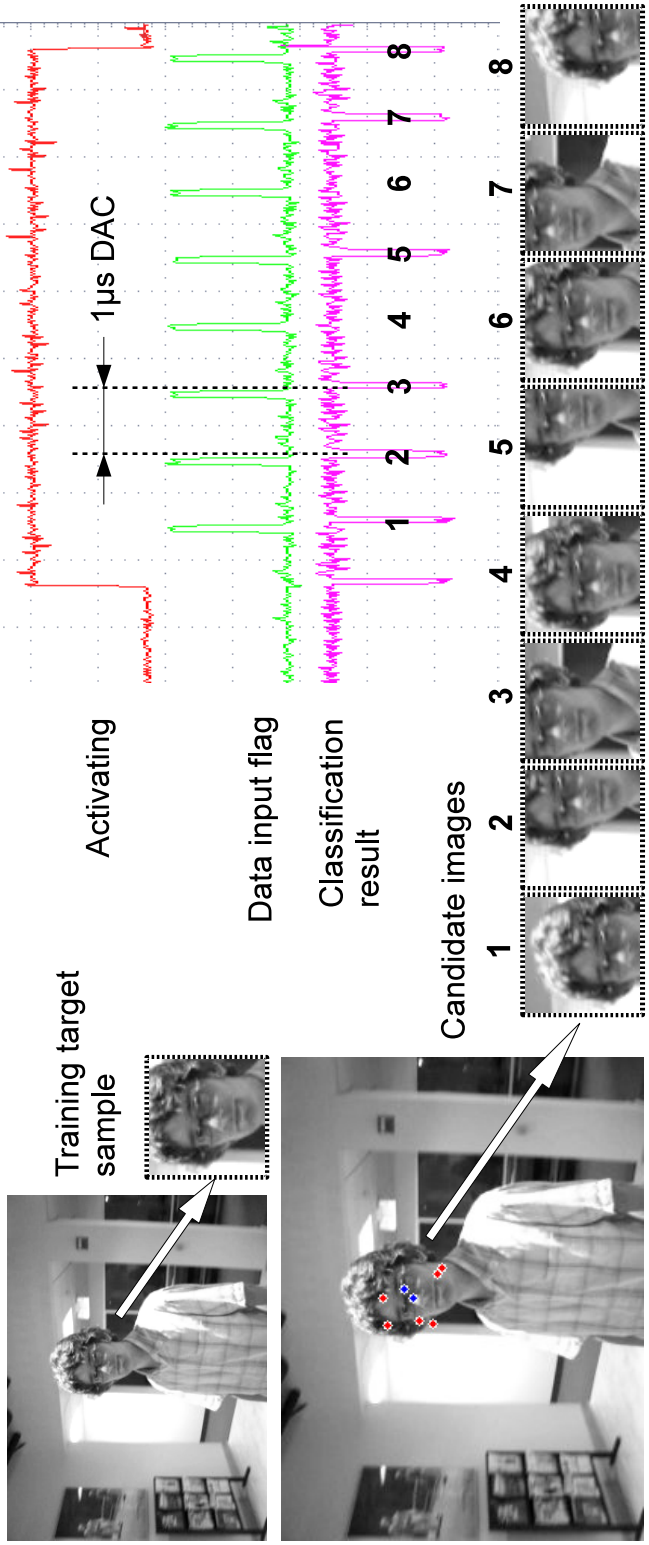


Figure 4.11: Verification of the training and predicting process on SVM chip.

Table 4.1: Evaluation of tracking accuracy at a fixed threshold of 20.

	Sylvester	David Indoor	Cola Can	Occluded Face	Occluded Face 2
OAB	0.64	0.16	0.45	0.22	0.61
SemiBoost	0.69	0.46	0.78	0.97	0.60
Frag	0.86	0.45	0.14	0.95	0.44
MILTrack	0.90	0.52	0.55	0.43	0.60
This work	0.90	0.95	0.93	0.27	0.44

In order to give a hardware-friendly solution to the object tracking task, we considered the implementation of each part of the algorithm. For the training of SVM, a dedicated VLSI chip proposed in (87) can be employed. It is a fully-parallel self-training SVM system with high training speed. Based on this chip, computational time caused by SVM computation can be decreased dramatically. A functional verification of this algorithm on the chip is shown in Fig. 5. Vectors were extracted from the image in advance and sent to the chip in real-time. After $100ns$ of training process (not shown), the chip received and gave out the classification result for each vector every $1\mu s$, in which the 4th and 6th candidate images were classified into the target class. In addition, we also employed a directional-edge-based representation algorithm (1) to represent sample images, for which a dedicated VLSI chip has been proposed in (77).

Figure 4.12: Experiments on five video clips: Sylvster, David Indoor, Cola Can, Face Occluded, Face Occluded 2.

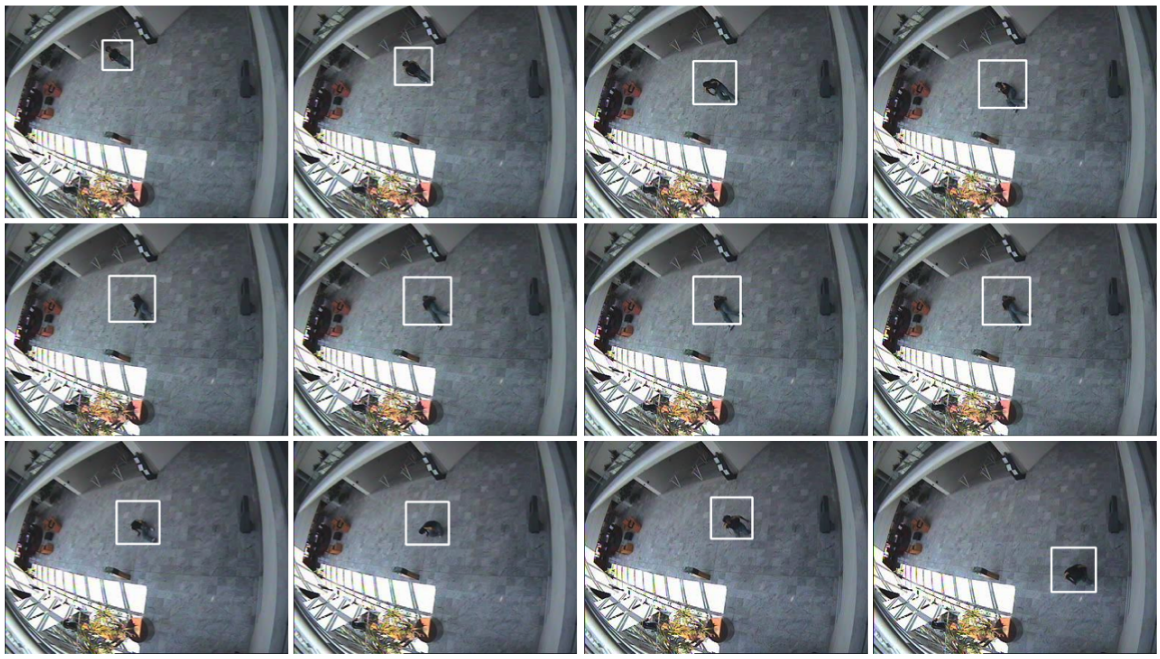


Figure 4.13: Experiment on "fallonfloor" video clip. See text for detail.

4.3.1 Experiments on Real-World Data

The SVM-based tracking algorithm with state vector (size only in this part) has been tested using several real video clips. There are various challenges in these videos. In the first video fallonfloor, a person walks and falls down on the floor, which contains severe deformation. The second video is a comparatively long video clip, in which a person walks back and forth, and gets close to some other objects. The difficult in the third video clip is that two people get together fighting, which is very confusing for the single target tracker. In this case specially, our tracking algorithm tracked different person at different time. The tracker should be improved to enhance the discriminative ability between similar objects. The fourth video clip shows the front view of a car. We selected to track the back of a car in front. There is some disturbance, such as rain and other automobiles. The size of the object changes greatly, though the appearance does not change much.

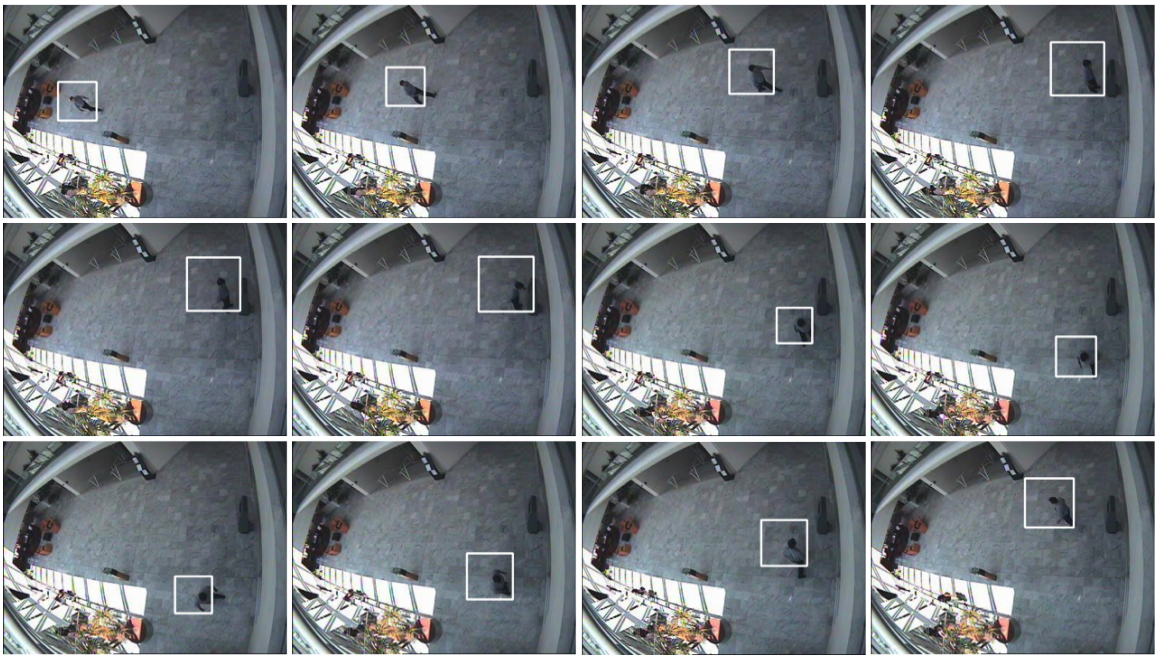


Figure 4.14: Experiment on "browse" video clip. See text for detail.

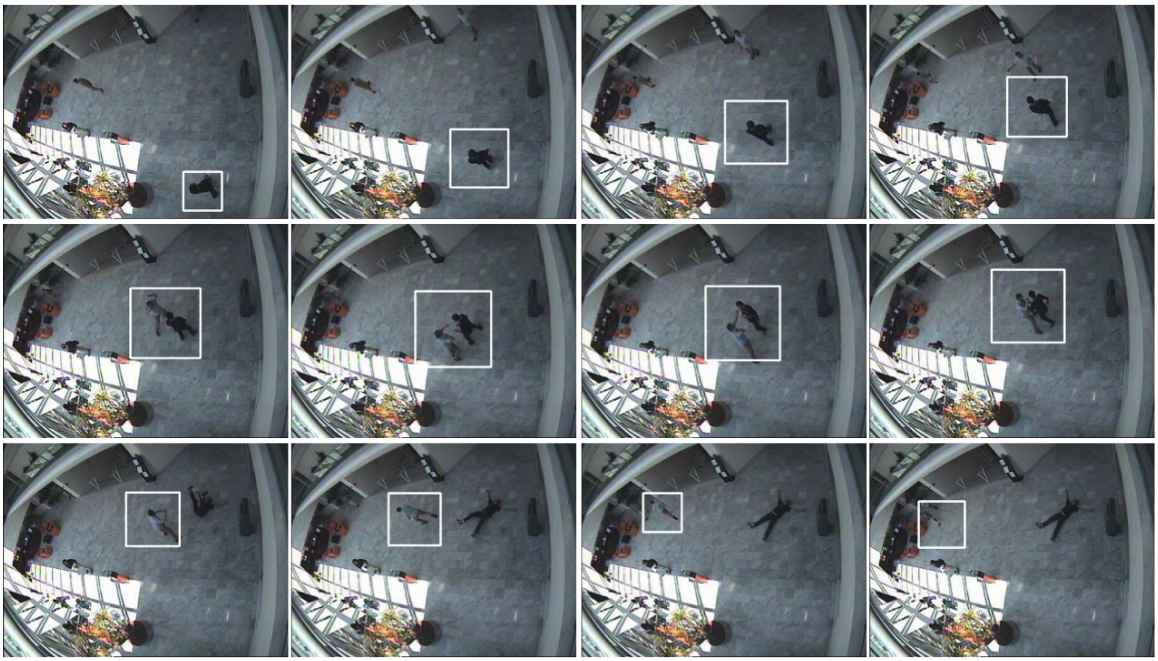


Figure 4.15: Experiment on "twomanfight" video clip. See text for detail.

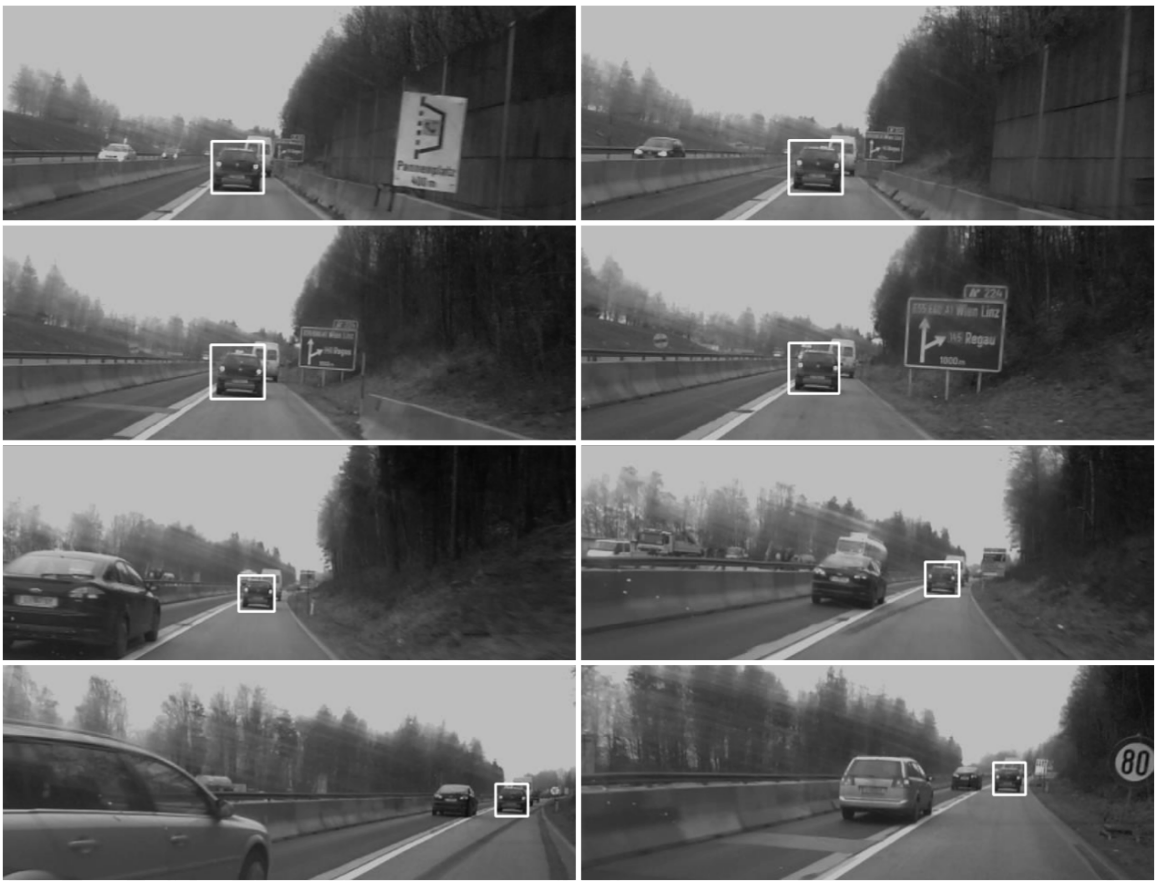


Figure 4.16: Experiment on "cartracking" video clip. See text for detail.

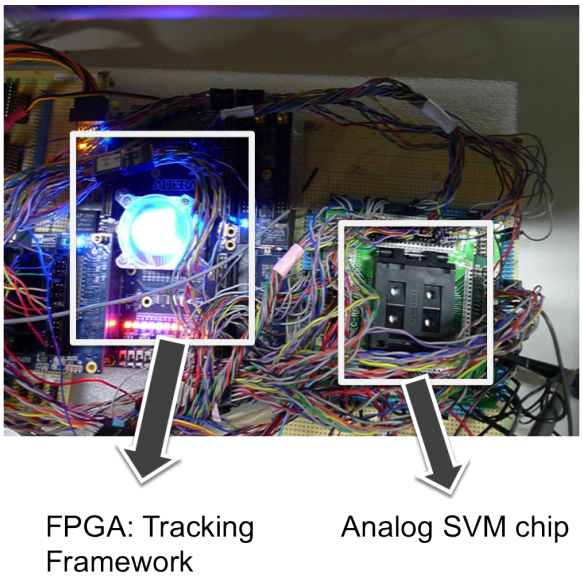


Figure 4.17: Hardware implementation of the tracking system.

4.4 Hardware Implementation

In this part, the hardware architecture implementing the SVM-based tracking system is described. There are mainly three parts on the FPGA: feature extraction, multiple candidate regeneration, and SVM-chip controller. In the following, each of the part is explained in detail.

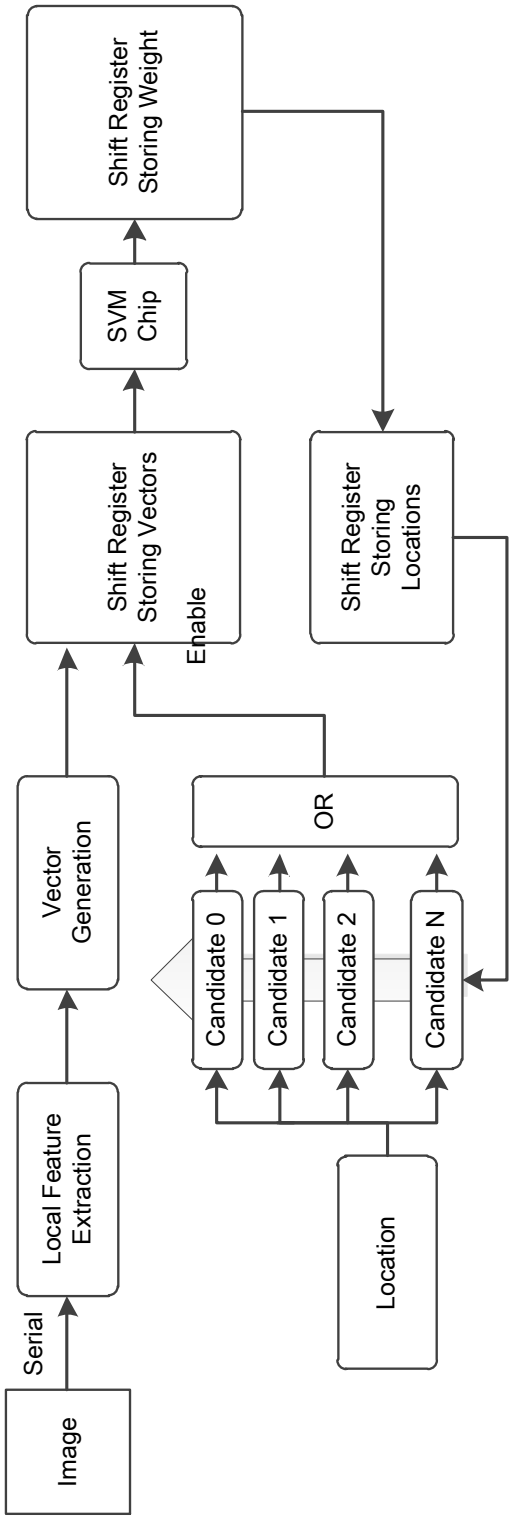


Figure 4.18: Hardware structure of the SVM-based object tracking system.

Fig. 4.18 shows the main structure of the system. At first, the image data is sent from camera to the local feature extraction block pixel by pixel for generating edge map. Then the edge information is sent to vector generation block to calculate APED feature vector for every pixel location. These three parts work in pipelined way, and one feature vector is outputted in every clock cycles. Because of the way of data transfer, this process can be considered as a scanning process. There is a location block, which counts current scanning location in the entire image. The location information is sent to the candidate blocks. Each candidate block has its own location. If the current location equals to one candidates location, the candidate will send an enable signal to the shift register that stores the feature vector. Then the shift register store current vector for a specific location. After all the feature vectors for all the candidates are calculated, these feature vectors are sent to the SVM-chip for testing. The results will be stored in a shift register that stores weights. Based on all these information, new candidate locations are generated and the private locations of the candidates are updated by the new locations.

4.4.1 Feature Extraction

The function of feature extraction in this system is same with the feature extraction in the system presented in §2. However, in this part, a new architecture is introduced for simple and high-speed computation. The entire process is implemented in pipelined structure. The image data is received pixel by pixel, and the feature vector for every pixel location (64 × 64-pixel window) is calculated. The structure is shown in Fig. 4.19. The image data is received and stored in a shift register, which contains all the necessary data for computation. The shift register is a 320 × 4 register array with 8-bit register for data of each pixel. The four rows of data are retained for calculating the convolution of the image data and four 5 × 5 kernels. This structure is similar to the directional-edge-feature extraction in §2, but the difference is that in this system the edge map of the entire frame is calculated. More information can be found in description of previous system. In the previous structure, only sub-image of the candidate is processed into edge

map. The input of image data is one pixel per clock cycle, and the output of edge data is also one pixel per clock cycle.

The serial pipelined output of edge data is sent to the second shift register, which is used for calculating feature vector (APED in this case). The goal of this part is to output the feature vector at every pixel location and make the output one vector per clock cycle, which is compatible with the transfer speed of input image and edge data. In order to reach this processing speed, some calculating structure must be designed. The calculation of APED is in fact a counting process. If the window size is 64 $\times$ 64-pixel, all the 4096 1-bit numbers (0 or 1) should be added to corresponding counter. There are totally 64 elements (counters) in one APED vector. In present structure, the edge data in a certain tracking window is sent to logic part in a column-parallel way. This means it will take 64 clock cycles to complete summation of one tracking window. But by observing the APED vector, it is obvious that the APED result can be reusable. Suppose that the APED at present location has already been calculated. The next APED vector can be calculated based on this known vector. According to the algorithm of directional-feature vector, the difference between two adjacent locations is only the new column of data and the oldest column of data, which comes at first. Therefore, the new APED can be calculated by do some simple summations and subtractions on the counters with the new column of data, as shown in Fig. 4.19. The edge data are transferred to the vector generation block in row-parallel way. A 320 $\times$ 64 shift register is used for storing the edge data which are necessary for one computation. Corresponding edge data are added up and sent through the shift registers. The element of the feature vector is calculated by summed up corresponding registers at every clock cycle. By using this pipelined structure, one feature vector for a certain pixel location is calculated and outputted in every clock cycle. In this way, the APED vector can be calculated at a very high speed. The loss in this structure is the accuracy, because the ratio threshold is replaced by a fixed-value threshold. But the advantage is that this structure provides other processing with APED vector at every pixel location in the scene, while the previous system can only calculate vectors for the selected candidates.

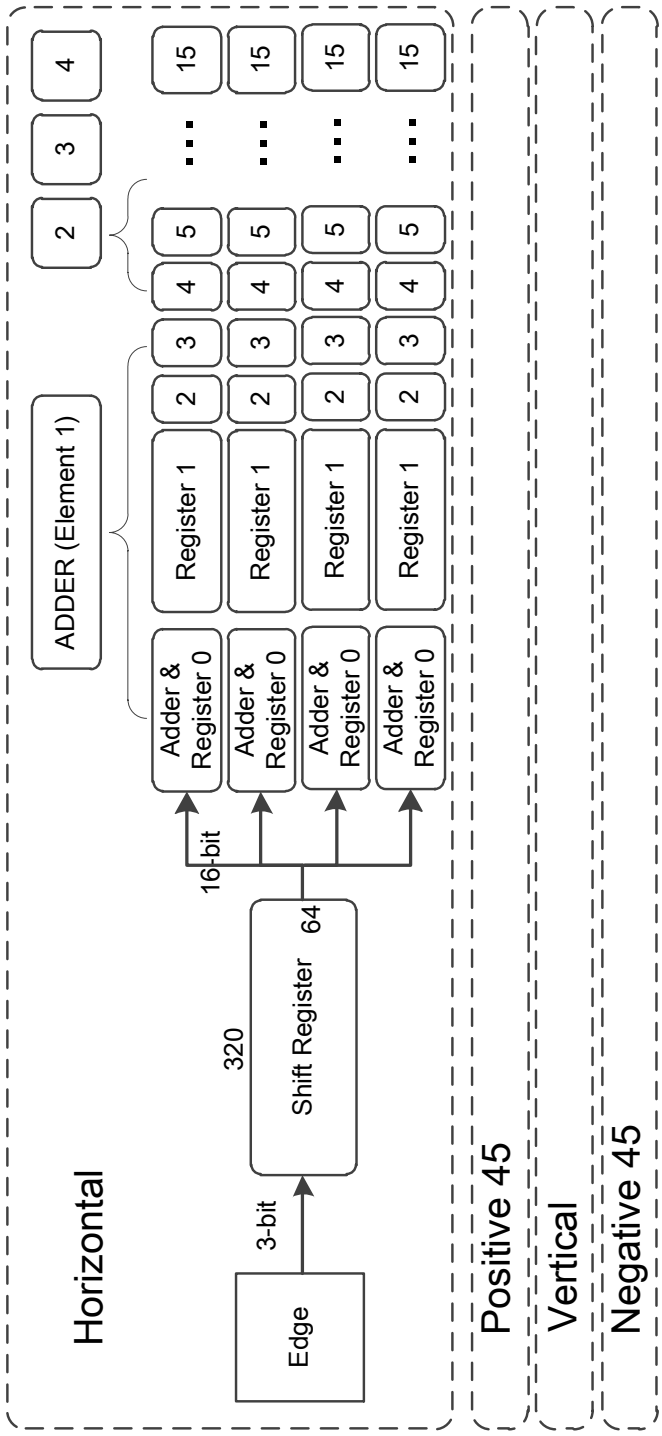


Figure 4.19: Hardware implementation of the vector generation block.

In this system, we still use candidates to predict the object location. It can be observed in the software simulation that the number of candidates is larger than in previous system. This is one of the reasons why we need to design a new structure for feature extraction. Along with the increase of the candidate number, there are several problems. Firstly, the comparison of candidates, which corresponds to the weighting process, takes more time then before. Secondly, it also takes much more memory to store all the vectors. Thirdly, it is very difficult to process all the candidates in parallel, which require too much of hardware resources. In fact, we have to divide the candidates into several groups, and then process each group in serial as we have done in the previous system. The serial processing without pipelined structure waste a lot of time in the image data transfer. The new structure solved these problems. For the first problem, we do not use Manhattan distance in this system. Instead, we use SVM to give weight value to each candidate. The testing of SVM can only be serial, which means we can calculate the weight (similarity) of one candidate at one time. But the advantage is the processing takes very short time. For the second problem, in our structure, the vectors for all the candidates can be discarded just several clock cycles after it is calculated. Therefore, it is not necessary to store the vectors for the candidates. In this case, it will be very efficient to just store the weight values of the candidates in the memory. The following processes are operations on the weights and locations.

4.4.2 Weighting and Regeneration

Because it is the weighting function is similar to the implementation in §2, the implementation of weight computation is not described in detail in this part. Brief information is as the follows, emphasizing on the communication between the FPGA and VLSI chip. After generating feature vectors for all the candidates, the candidate vector is sent to the SVM chip by eight clock cycles. In each clock cycle, one bit of the 64 elements (a 64-bit data) in one vector is sent. In the following two clock cycles, the SVM chip will test the vector, and send back the classification result, “0 or “1. Since the feedback signal is a analog signal lasting for about one clock cycles, a circuit block working at a frequency ten times faster than

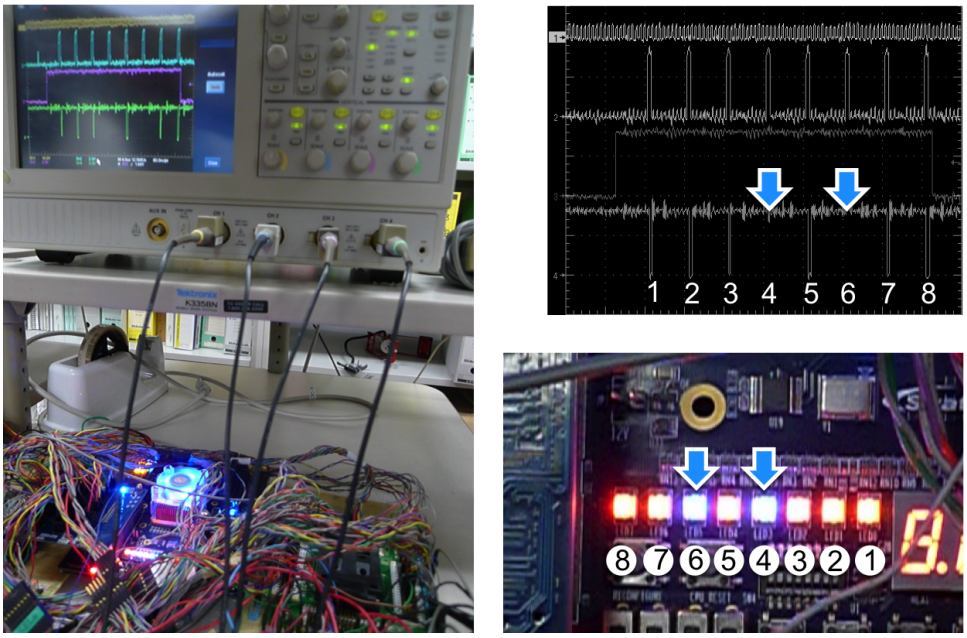


Figure 4.20: The connection between FPGA and VLSI chip.

the system clock is designed for sampling from the analog signal. All the control signals for the SVM chip are stored on the FPGA. In fact, in the entire process, there is training procedure at the end of every prediction of object location. Fig. 4.20 shows an experimental result verifying the control and computation of the system. At first, the SVM chip is trained using nine examples, as shown in Fig. 4.11. After training, eight test vectors are sent to the SVM chip and the FPGA tracking block read the classification results. The eight results are shown as the "ON and "OFF of the LED lights on the FPGA. The regeneration in the architecture is same with the previous system. Detailed information can be found in §2.

4.5 Summary

In this part, we proposed a real-time object tracking algorithm based on the on-line learning support vector machine with hardware-friendly structure and a newly proposed training framework. The tracking framework give a solution to the problem of updating reliable training samples in object tracking. The hard-

ware structure employs several real-time algorithms which have been implemented into VLSI chips. Software simulation results were evaluated, which showed robust and accurate tracking ability. Hardware verification was carried out on a VLSI SVM chip, which proved the real-time performance of this algorithm.

5

A Real-Time Object Tracking System with Online Learning Nearest Neighbor Classifier

5.1 Introduction

In the multiple-candidate-regeneration-based (MCR-based) system that has already been explained in previous chapter, there are still two important problems that need to solve. First, new templates are frequently added into the system. The quick increase of the templates may cause memory waste and increase the time for template matching. Second, like in many tracking algorithms with learning ability, it is of great importance to select new reliable templates. A wrong template may cause the shift problem or even leads to failure. The goal of this paper is to resolve the above problems to improve the tracking capability without sacrificing the other advantages, such as simplicity of hardware implementation.

In fact, these two problems are both related to the appearance model. The representation of object image and the measurement of the likelihood between two images are usually defined as appearance model, which affects the tracking accuracy and robustness greatly (68). Therefore, in this chapter we propose a novel appearance model with an online learning strategy. Instead of measuring the likelihood by Manhattan distance between the APED vectors of the two images, we adopt the nearest neighbor (NN) classifier to discriminate the object

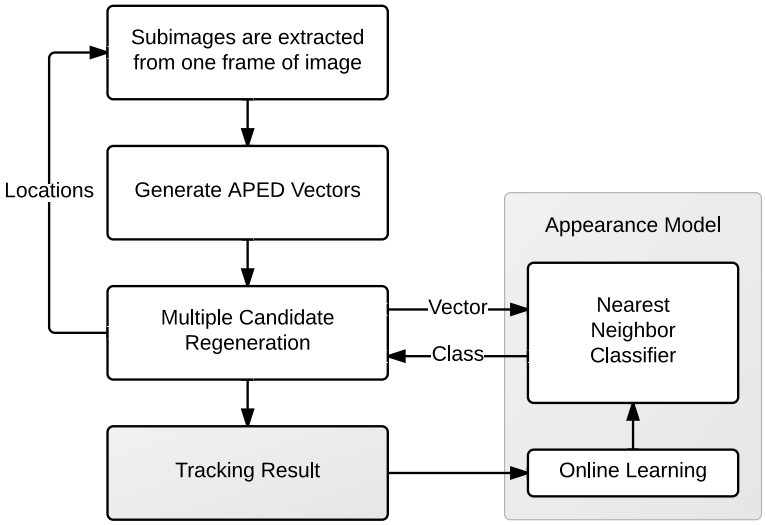


Figure 5.1: Structure of the tracking system.

images from the background images. The obvious merit of this method is that the likelihood is well expressed by the distance because of considering background information. By analyzing the likelihood map generated by the NN classifier, we found that only a small number of templates can fulfill the requirement of tracking in each frame provided that the templates are updated in each frame. Therefore, in the new learning strategy, old templates are replaced by new templates very quickly. The object templates are only updated when the system detects the classifier does not have sufficient effective templates to discriminate the object and the background. In this situation, the background region, which is more reliable, is analyzed to deduce the most possible location of the object. Then the object image at this location will be stored as a new object template. Although the analysis is focused on a specific tracking system, the solution can be utilized in many other situations with similar tracking mechanism.

Figure 5.2: Likelihood maps. (a) Generated by using Manhattan distance. (b) Generated by using NN classifier. (c) An example showing an all-background situation.

5.2 Algorithm

5.2.1 Basic Tracking Framework

A tracking system usually consists of two parts: object representation and search strategy. In this work, the bio-inspired algorithm APED that has already been implemented on VLSI chip is employed. By using APED, a grayscale image is processed by four directional filters to generate directional edge maps. Then a 64-D feature vector is generated by calculating the number of edges in certain subregions in each edge maps. It is simpler and faster to manipulate these feature vectors than many other representation features, such as scale-invariant feature transform (SIFT). The performance of APED and detailed information can be found in (1).

For the search strategy, we employed the MCR algorithm, which is first proposed in (79) and implemented on hardware in (91). It is a statistical approach that has been designed for efficient hardware implementation. The MCR uses a group of locations called candidates to search for the target object in a certain region. At the location of each candidate, a sub-image is extracted and transformed into a feature vector. Then each candidate is assigned with a weight according to the likelihood between its feature vector and the templates. In the next step, new candidates are generated following the criteria as follows. First, candidates with large weight have more new candidates in their vicinity. Second, in every iteration, the number of candidates is constant. In the last step, the object location is predicted by analyzing the distribution of the candidates in location space.

Combining the APED and MCR, Fig. 5.1 shows the structure of the tracking system. At the beginning, an object image is selected as object template and the locations of candidates are initialized. When a new frame of image comes, the subimages at candidates location are extracted and transformed into APED vectors. Then in the MCR process, new candidates are generated in the vicinity of the old candidates that are similar to the object template. The center of gravity of all the candidate locations is calculated as the prediction of the object location. At this point, the object image in this frame is checked to determine

whether it is necessary to store a new template. After this learning process, the algorithm receives the next frame of image and repeat the entire process.

5.2.2 Nearest Neighbor Classifier

Many object tracking systems use classifiers, such as the support vector machines (SVM) (66) and the NN classifier (92), to form the appearance model. These algorithms consider the object tracking task as a binary classification problem, and realize accurate and flexible description of object. Although the classifiers have different mechanisms and advantages, they all benefit from considering the background information. In this work, we use the NN classifier because of its simplicity and we also verified its capability to describe the likelihood between images. We use an example to illustrate the effectiveness of the NN classifier combined with APED vector, as shown in Fig. 5.2. The likelihood maps show the likelihood values between the subimage at each pixel location and the templates. Fig. 5.2(a) shows the confidence map generated by using Manhattan distance. Smaller distance gives higher likelihood score. The subimage at the center location works as the template. In Fig. 5.2(b), the NN classier is used, and the suppression of likelihood score by background templates is observed. The object is the subimage at center location too, while 24 background templates are selected as shown in Fig. 5.3(a). In this figure, the larger the value is, the closer the subimage is to the object class. On the opposite, the smallest value means the closest to the background templates. We can observed that in a small tracking region, a small number of templates are sufficient to cover the background and the object. In fact, for a typical tracking algorithm, it is efficient to build a classifier that can just discriminate the object from background in current frame of image. It is not efficient to make the classifier as powerful and accurate as in object detection or categorization tasks. Therefore, this system only retains the most critical templates. For instance, only the background examples extracted from last frame is retained. In the first two figures, there is no difference when use MCR to search the object (the peak location). In Fig. 5.2(c), a real example generated by Manhattan distance is shown which makes the MCG fail to locate accurately and stably. The solution is discussed in the learning strategy.

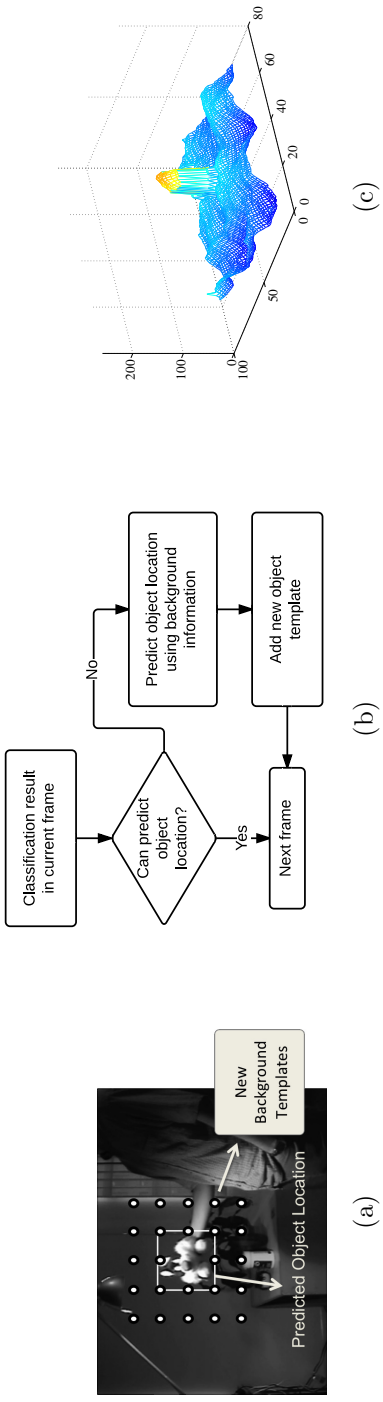


Figure 5.3: Online learning strategy for the NN classifier. (a) Selection of background templates. (b) Selection of object templates. (c) An example of adding new object template corresponding to Fig. 5.2(c)

5.2.3 Online Learning Strategy

Instead of offline training with a large database, many algorithms choose to online update the classifier, which makes it adaptive to the appearance change of the object. Like the SVM-based tracking system, we propose an online learning strategy specifically designed for tracking task.

For the background templates, the capacity is limited to hold only a certain number of templates and only the most recent examples are retained. After predicting the object location in each frame, 24 background subimages are extracted from current frame at locations shown in Fig. 5.3(a). In the tracking in next frame, these new examples are critical distractors according to the illustration in Section I.

Fig. 5.3(b) shows the updating mechanism of the object templates. The object templates are not updated in every frame in order to avoid the "shift problem" in tracking. New object examples are added only when the classifier can not find sufficient reliable candidates belonging to the object class for predicting the object location. This always happens when the object changes its appearance and it is possible that all candidates fall into the background class. But it is still possible to predict the object location by analyzing the candidates in the background class. In this case, some candidates have large distances between itself and both the object templates and the background templates. Fig. 5.3(c) shows the likelihood map from the same frame with the example shown in Fig. 5.2(c). It is obvious that we can still deduce the object location from this map, because the background does not change much in the consecutive frames and they are detected with confidence (high likelihood). In this case, we store the image at the location where is unknown (neither object nor background) as a new object template. In summary, when the "knowledge" about the object in the classifier is sufficient, no new template is updated. When it is not sufficient, the system tracks the background to deduce the possible location of object, and generates new template.

Table 5.1: Comparisons: Precision at a Fixed Threshold of 20

	OAB	SemiBoost	Frag	MILTrack	MCR (91)	This Work
Sylvester	0.64	0.69	0.86	0.90	0.83	0.88
David Indoor	0.16	0.46	0.45	0.52	0.88	0.86
Cola Can	0.45	0.78	0.14	0.55	0.93	0.96
Occluded Face	0.22	0.97	0.95	0.43	0.12	0.38
Occluded Face 2	0.61	0.60	0.44	0.60	0.44	0.62
Surfer	0.51	0.96	0.28	0.93	0.60	0.92
Tiger 1	0.48	0.44	0.28	0.81	0.37	0.41
Tiger 2	0.51	0.30	0.22	0.83	0.50	0.50
Coupon Book	0.67	0.37	0.41	0.69	0.40	0.38

5.3 Experimental Results

We evaluated the tracking system by using a group of challenging video sequences from a public database. The same evaluation methodology employed by the SVM-based tracking system is adopted, including evaluation on both accuracy and location error. In Table I, we show the tracking accuracy at a fixed threshold of 20. This threshold is a distance in pixels. If the distance between the predicted location and the ground truth is smaller than the threshold, the prediction in the corresponding frame is considered as successful. The accuracy of tracking in each video is expressed as the percentage of frames tracked correctly in the video. Some selected frames from the experimental results are shown in Fig. 5.4. The result shows this tracking system can deal with appearance change well as expected by using only 32 templates for each class. In most of the videos, this system gave better results than the precedent system. It shows less accurate when the videos contain heavy occlusion. This is mainly because the feature vector we employed is a global representation, which is sensitive to severe occlusion.

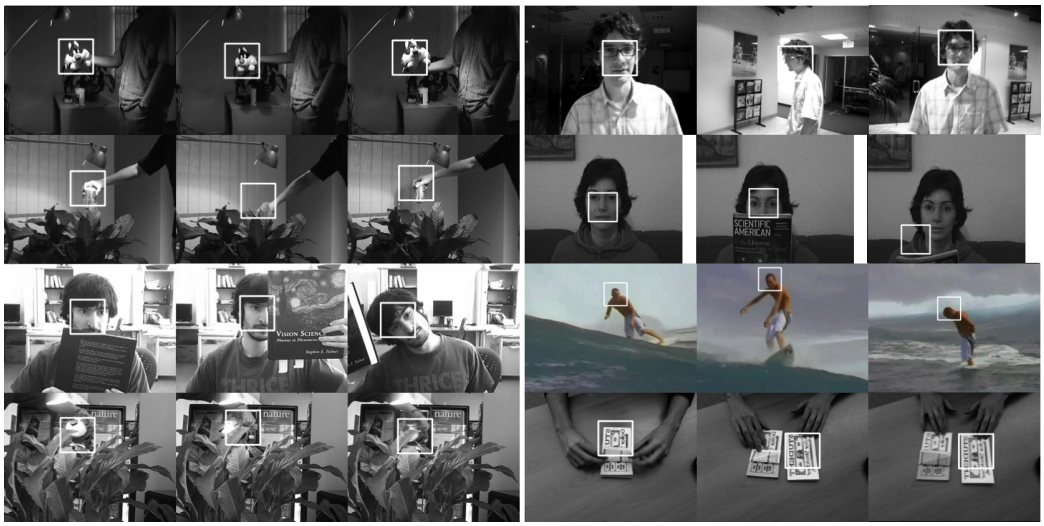


Figure 5.4: Some selected frames from the tracking results.

5.4 Discussion

Object tracking applications usually require real-time performance. Moreover, in some situations, it is necessary to reach a processing speed much higher than the frame rate. In order to solve this problem, a VLSI-based hardware architecture of the object tracking system is proposed in (91). This architecture achieves processing speed of 150 f/s on FPGA and can reach 900 f/s if implemented on VLSI. The tracking system proposed can be implemented by using similar architecture. Only some logic elements for controlling and matching operations are needed, which will not bring any drawbacks to the system.

5.5 Conclusion

In this chapter, we proposed a real-time object tracking system with online learning capability realized by a novel appearance model based on the NN classifier. We verified the effectiveness of the combination of APED feature vector and the NN classifier, which showed the merit of using background information. An online learning strategy, including selection of reliable templates, was designed specifically for the object tracking task. Evaluation of the system on accuracy was shown and discussed to clarify the features of the system. Since this tracking

system is designed for simple hardware implementation, the realization of the high-speed system is also discussed.

6

Conclusion

6.1 Summary

This thesis is focused on feature-based object tracking system, which discusses the tracking problem and its implementation and presents a complete solution to real-time object tracking problem.

At first, we proposed a real-time object tracking system, which is based on the multiple candidate-location generation mechanism. The system employs the directional-edge based image features and also an online learning algorithm for robust tracking performance. Since the design of this algorithm is hardware-friendly, we designed and implemented the realtime system on FPGA, which has the ability of processing a 640×480 resolution image in about 0.1 ms. It achieved 150 f/s frame rate on FPGA, and can reach about 900 f/s if implemented on VLSI with on-chip image sensor. Evaluation of the tracking system on both accuracy and speed are shown and discussed, which clarify the features of this system.

In the following, this thesis also presents a detailed discussion on several issues of tracking, including VLSI chip implementation for faster operation, multiple target tracking, initialization problem and full occlusion problem, and state vectors. The solutions presented in the discussion part are based on our hardware system, and can be easily implemented. Software simulation is preformed for verification of the design.

At last, this work proposed a real-time object tracking algorithm based on the on-line learning support vector machine and nearest neighbor classifiers with

hardware-friendly architecture and a novel training framework. The tracking framework gives a solution to the problem of updating reliable training samples in object tracking. The hardware structure employs several real-time algorithms which have been implemented on VLSI chips. Software simulation results were evaluated, which showed robust and accurate tracking ability compared with other superior works. In addition, the hardware architecture was also presented. Hardware verification was carried out on a VLSI SVM chip with tracking framework on FPGA, which proved the real-time performance of this algorithm.

6.2 Perspectives

Significant progress has been made in object tracking during the last few years. Several robust trackers have been developed for real-time tracking. However, it is also obvious that there are still some unsolved problems, and object tracking still can not be used in many practical applications. Therefore, the first direction which we can put effort on is to improve the robustness of the trackers in various complex situations. Improving feature representations, analyzing context information, and realizing on-line learning ability are all promising directions at present.

For representation of the object, feature selection and combination become more and more important. Many papers are focused on this direction in recent years. They use multiple cues for discriminating different objects. And in different situations, the most effective features are used. One problem in this method is that computation of multiple features brings much more computational cost.

In many researches, the information from the video is exploited deeply. For example, not only the object information, but also the background information is of great importance. From the background, we can analyze the context of the scene, and find where the object is possible to be. The background information can also be used for selecting features, because the purpose of using some feature is to discriminate the object from the background. The entire video, or consecutive frames can also be analyzed as a complete motion. This is very useful for predicting the behavior of object.

Since the object tracking is more like a cognitive process, more and more learning algorithms will be introduced into tracking system. The learning ability is not limited to the building the appearance model of the object. It can be used in analyze the state of the object, such as distribution of object features. The learning ability solve the problem that most of the tracking algorithms are designed for some kinds of scene, and there is few general solution working as human visual system. By learning, we can extract important rules from experience, and on-line learning provide the system with adaptive ability to changes, which usually can not predict before tracking.

Real-time performance of the tracker will always be a hot topic in object tracking. In fact, there are two ways to achieve this purpose. The first is to simply and parallelize the algorithm, and implement the algorithm on high-speed processor. The second way is to design high-efficiency hardware architecture for object tracking. For example, there are some implementations using on-chip image sensor, which can achieve really high-speed processing with low power consumption.

References

[1] Y. Suzuki and T. Shibata. **Multiple-clue face detection algorithm using edge-based feature vectors**. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, pages 737–740, 2004. vii, 21, 23, 56, 58, 67, 84

[2] W.W.Y. Ng, X. Ma, P.P.K Chan, and D.S Yeung. **Multi-object tracking in video using Localized Generalization Error model based RBFNN**. In *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, pages 1825–1831, 2011. 1, 4

[3] L. Yang, B. Georgescu, Y. Zheng, Y. Wang, P. Meer, and D. Comaniciu. **Prediction Based Collaborative Trackers (PCT): A Robust and Accurate Approach Toward 3D Medical Object Tracking**. *Medical Imaging, IEEE Transactions on*, **30**(11):1921–1932, 2011. 1

[4] L. Meng and J.P. Kerekes. **Object Tracking Using High Resolution Satellite Imagery**. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, **5**(1):146–152, 2012. 1

[5] D.P. Dogra, A.K. Majumdar, S. Sural, J. Mukherjee, S. Mukherjee, and A. Singh. **Toward Automating Hammersmith Pulled-To-Sit Examination of Infants Using Feature Point Based Video Object Tracking**. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, **20**(1):38–47, 2012. 1

[6] L. Yuan, Y.F. Zheng, J. Zhu, L. Wang, and A. Brown. **Object Tracking With Particle Filtering in Fluorescence Microscopy Im-**

ages: Application to the Motion of Neurofilaments in Axons. *Medical Imaging, IEEE Transactions on*, **31**(1):117–130, 2012. 1

[7] J. Ding, Y. Huang, K. Huang, and T. Tan. **Robust object tracking via online learning of adaptive appearance manifold**. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1863–1869, 2011. 2

[8] W. Li, X. Zhang, W. Luo, W. Hu, H. Ling, and O. Wu. **Robust object tracking with boosted discriminative model via graph embedding**. *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1666–1672, 2011. 2

[9] B. Liu, J. Huang, L. Yang, and C. Kulikowsk. **Robust tracking using local sparse appearance model and k-selection**. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1313–1320, 2011. 2

[10] Y. Wu, J. Cheng, J. Wang, H. Lu, H. Ling, E. Blasch, and L. Bai. **Real-time Probabilistic Covariance Tracking with Efficient Model Update**. *Image Processing, IEEE Transactions on*, **21**:2824–2837, 2012. 2

[11] P. Li. **An Adaptive Binning Color Model for Mean Shift Tracking**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **18**(9):1293–1299, 2008. 2, 14

[12] J. Chiverton, X. Xie, and Majid Mirmehdi. **Automatic Bootstrapping and Tracking of Object Contours**. *IEEE Transactions on Image Processing*, **21**(3):1231–1245, 2012. 2

[13] M. Sheu, W. Tsai, C. Hu, and C. Tsao. **Fast Texture-Based Object Tracking Algorithm on Embedded Platform**. In *Frontier of Computer Science and Technology (FCST), 2010 Fifth International Conference on*, pages 511–514. IEEE Computer Society, 2010. 2

[14] Q. Chen, Q. Sun, P.A. Heng, and D.S. Xia. **Two-Stage Object Tracking Method Based on Kernel and Active Contour**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **20**(4):605–609, 2010. 2, 13, 14

[15] H. Zhou, Y. Yuan, and C. Shi. **Object tracking using SIFT features and mean shift**. *Computer Vision and Image Understanding*, **113**(3), 2009. 2, 4

[16] G. Yuan, M. Xue, L. Yao, C. Xing, and K. Xie. **PCA-Based Estimation of Velocity Feature for Visual Tracking**. In *Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on*, pages 522–526, 2010. 2

[17] H. Wang, D. Suter, K. Schindler, and C. Shen. **Adaptive Object Tracking Based on an Effective Appearance Filter**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **29**(9):1661–1667, 2007. 3, 13, 15

[18] E. Norouznezhad, A. Bigdeli, A. Postula, and B.C. Lovell. **Object tracking on FPGA-based smart cameras using local oriented energy and phase features**. In *ICDSC '10: Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras*. ACM Request Permissions, 2010. 3

[19] R.S. Prakash and R. Aravind. **Object tracking using AM-FM image features**. *Computer Vision, IET*, **4**(4):295–305, 2010. 3

[20] X. Mei and H. Ling. **Robust Visual Tracking and Vehicle Classification via Sparse Representation**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **33**(11):2259–2272, 2011. 3

[21] S. Samarah, M. Al-Hajri, and A. Boukerche. **A Predictive Energy-Efficient Technique to Support Object-Tracking Sensor Networks**. *Vehicular Technology, IEEE Transactions on*, **60**(2):656–663, 2011. 3

[22] T.N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M. Meinecke. **Stereo-Camera-Based Urban Environment Perception Using Occupancy Grid and Object Tracking**. *Intelligent Transportation Systems, IEEE Transactions on*, **13**(1):154–165, 2012. 3

[23] J. Wang and Y. Yagi. **Integrating Color and Shape-Texture Features for Adaptive Real-Time Object Tracking**. *Image Processing, IEEE Transactions on*, **17**(2):235–240, 2008. 3, 14, 15

[24] S. Fazli, H.M. Pour, and H. Bouzari. **Particle Filter Based Object Tracking with Sift and Color Feature**. In *Machine Vision, 2009. ICMV '09. Second International Conference on*, pages 89–93, 2009. 3, 4, 14

[25] H. Ying, X. Qiu, J. Song, and X. Ren. **Particle Filtering Object Tracking Based on Texture and Color**. *Intelligence Information Processing and Trusted Computing (IPTC), 2010 International Symposium on*, pages 626–630, 2010. 3

[26] P. Hidayatullah and H. Konik. **CAMSHIFT improvement on multi-hue object and multi-object tracking**. *Visual Information Processing (EUVIP), 2011 3rd European Workshop on*, pages 143–148, 2011. 3

[27] L. Ledwich and S. Williams. **Reduced SIFT features for image retrieval and indoor localisation**. *Australian Conference on Robotics and Automation*, **322**, 2004. 3

[28] Y. Tao, M. Skubic, T. Han, Y. Xia, and X. Chi. **Performance Evaluation of SIFT-Based Descriptors for Object Recognition**. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, **2**, 2010. 3

[29] Y. Lin, C. Yeh, S. Yen, C. Ma, P. Chen, and C. Kuo. **Efficient VLSI design for SIFT feature description**. *Next-Generation Electronics (ISNE), 2010 International Symposium on*, pages 48–51, 2010. 3, 14, 28

[30] K. Mizuno, H. Noguchi, G. He, Y. Terachi, T. Kamino, H. Kawaguchi, and M. Yoshimoto. **Fast and Low-Memory-Bandwidth Architecture of SIFT Descriptor Generation with Scalability on Speed and Accuracy for VGA Video**. In *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, pages 608–611. IEEE Computer Society, 2010. 3

[31] D. Yang and A. Sluzek. **Performance evaluation of low-dimensional sifts**. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2729–2732, 2010. 3

[32] F. Huang, S. Huang, J. Ker, and Y. Chen. **High-Performance SIFT Hardware Accelerator for Real-Time Image Feature Extraction**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **22**(3):340–351, 2012. 3, 28

[33] Y. Yeh and C. Hsu. **Online Selection of Tracking Features Using AdaBoost**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **19**(3):442–446, 2009. 3, 13, 15

[34] Z.H. Khan and I.Y.H. Gu. **Joint Feature Correspondences and Appearance Similarity for Robust Visual Object Tracking**. *Information Forensics and Security, IEEE Transactions on*, **5**(3):591–606, 2010. 3

[35] L. Chen, W. Li, and W. Yin. **Joint feature points correspondences and color similarity for robust object tracking**. In *Multimedia Technology (ICMT), 2011 International Conference on*, pages 403–407, 2011. 3

[36] N. Jiang, W. Liu, and Y. Wu. **Learning Adaptive Metric for Robust Visual Tracking**. *Image Processing, IEEE Transactions on*, **20**(8):2288–2300, 2011. 3

[37] X. Xiang. **A brief review on visual tracking methods**. In *Intelligent Visual Surveillance (IVS), 2011 Third Chinese Conference on*, pages 41–44, 2011. 3

[38] G. Li, G. Qu, and Q. Huang. **A Multiple Targets Appearance Tracker Based on Object Interaction Models**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **22**(3):450–464, 2012. 3

[39] Z. Kim. **Real time object tracking based on dynamic feature grouping with background subtraction**. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008. 4, 13

[40] B. Han, Y. Zhu, D Comaniciu, and L.S. Davis. **Visual Tracking by Continuous Density Propagation in Sequential Bayesian Filtering Framework**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **31**(5):919–930, 2009. 4, 13

[41] J. Kwon and K.M. Lee. **Tracking by Sampling Trackers**. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1195–1202, 2011. 4

[42] C. Beyan and A. Temizel. **Adaptive mean-shift for automated multi object tracking**. *Computer Vision, IET*, **6**(1):1–12, 2012. 4

[43] J. Ning, L. Zhang, D. Zhang, and C. Wu. **Robust mean-shift tracking with corrected background-weighted histogram**. *Computer Vision, IET*, **6**(1):62–69, 2012. 4

[44] X. Tao, S. Zhang, and S. Yan. **A Robust Visual Tracking Approach with Adaptive Particle Filtering**. In *Communication Software and Networks, 2010. ICCSN '10. Second International Conference on*, pages 549–553, 2010. 4

[45] Y. Song and Q. Li. **Visual tracking based on multiple instance learning particle filter**. *Mechatronics and Automation (ICMA), 2011 International Conference on*, pages 1063–1067, 2011. 4

[46] M. Li, T. Tan, W. Chen, and K. Huang. **Efficient Object Tracking by Incremental Self-Tuning Particle Filtering on the Affine Group**. *Image Processing, IEEE Transactions on*, **21**(3):1298–1313, 2012. 4

[47] S. Das, A. Kale, and N. Vaswani. **Particle Filter With a Mode Tracker for Visual Tracking Across Illumination Changes**. *Image Processing, IEEE Transactions on*, **21**(4):2340–2346, 2012. 4

[48] Da T. and Y.J. Zhang. **Combining Mean-Shift and Particle Filter for Object Tracking**. In *ICIG '11: Proceedings of the 2011 Sixth International Conference on Image and Graphics*, pages 771–776. IEEE Computer Society, 2011. 4

[49] J. Cho, S. Jin, Jae W. Pham, Xuan D.and J., J. Byun, and H. Kang. **A Real-Time Object Tracking System Using a Particle Filter**. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2822–2827, 2006. 4, 13

[50] X. Zhang, W. Hu, W. Qu, and S. Maybank. **Multiple Object Tracking Via Species-Based Particle Swarm Optimization**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **20**(11):1590–1602, 2010. 4

[51] H. Medeiros, X. Gao, J. Park, R. Kleihorst, and A. Kak. **A parallel implementation of the color-based particle filter for object tracking**. In *Proceedings of the ACM Sensys Workshop on Applications, Systems and Algorithms for Image Sensing (ImageSense'08)*. Citeseer, 2008. 4, 14, 41, 43

[52] T. Ishiguro and R. Miyamoto. **An efficient prediction scheme for pedestrian tracking with cascade particle filter and its implementation on Cell/B.E.** In *Intelligent Signal Processing and Communication Systems, 2009. ISPACS 2009. International Symposium on*, pages 29–32, 2009. 4, 13, 14, 41, 43

[53] H.A. El-Halym, I. Mahmoud, and S. Habib. **Efficient hardware architecture for Particle Filter based object tracking.** In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 4497–4500, 2010. 4

[54] H. El, I. Halym, and S. ed Habib. **Proposed hardware architectures of particle filter for object tracking**. *EURASIP Journal on Advances in Signal Processing*, **2012**(1):17, 2012. 4, 14

[55] X. Lu, D. Ren, and S. Yu. **FPGA-based real-time object tracking for mobile robot**. In *Audio Language and Image Processing (ICALIP), 2010 International Conference on*, pages 1657–1662. IEEE, 2010. 4, 14

[56] H Zhuang, K.S Low, and W.Y Yau. **On-chip pulse based parallel neural circuits for object-tracking system**. *Electronics Letters*, **46**(9):614–616, 2010. 4

[57] E. Norouznezhad, A. Bigdeli, A. Postula, and B.C. Lovell. **Robust object tracking using local oriented energy features and its hardware/software implementation**. In *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pages 2060–2066, 2010. 4, 13, 14, 41, 43

[58] M. Schaeferling and G. Kiefer. **Object Recognition on a Chip: A Complete SURF-Based System on a Single FPGA**. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pages 49–54, 2011. 4

[59] T. Lee and S. Soatto. **Learning and matching multiscale template descriptors for real-time detection, localization and tracking**. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1457–1464, 2011. 5

[60] S. Liu, A. Papakonstantinou, Hongjun Wang, and Deming Chen. **Real-Time Object Tracking System on FPGAs**. In *Application Accelerators in High-Performance Computing (SAAHPC), 2011 Symposium on*, pages 1–7. IEEE, 2011. 5, 14

[61] Z. Sun, H. Yao, S. Zhang, and X. Sun. **Robust visual tracking via context objects computing**. *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 509–512, 2011. 5

[62] B. Zhong, H. Yao, S. Chen, R. Ji, X. Yuan, S. Liu, and W. Gao. **Visual tracking via weakly supervised learning from multiple imperfect oracles**. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1323–1330, 2010. 5

[63] Z.H. Khan and I.Y.H. Gu. **Adaptive appearance learning for visual object tracking**. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 1413–1416, 2011. 5

[64] X. Liu, L. Lin, S. Yan, H. Jin, and W. Jiang. **Adaptive Object Tracking by Learning Hybrid Template Online**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **21**(11):1588–1599, 2011. 5

[65] P. Wang and H. Qiao. **Online Appearance Model Learning and Generation for Adaptive Visual Tracking**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **21**(2):156–169, 2011. 5

[66] S. Avidan. **Support vector tracking**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **26**(8):1064–1072, 2004. 5, 14, 53, 54, 85

[67] B. Babenko, M. Yang, and S. Belongie. **Robust Object Tracking with Online Multiple Instance Learning**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **33**(8):1619–1632, 2011. 6, 37, 38, 63

[68] A. Yilmaz, O. Javed, and M. Shah. **Object tracking: A survey**. *ACM Computing Surveys*, **38**(4), 2006. 13, 81

[69] Z. Khan, I.Y.H. Gu, and A.G. Backhouse. **Robust Visual Object Tracking Using Multi-Mode Anisotropic Mean Shift and Particle Filters**. *IEEE Transactions on Circuits and Systems for Video Technology*, **21**(1):74–87, 2011. 13, 14

[70] H. Medeiros, J. Park, and A. Kak. **A parallel color-based particle filter for object tracking**. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–8, 2008. 13, 43

[71] S.A. Li, C. Hsu, W. Lin, and J. Wang. **Hardware/software co-design of particle filter and its application in object tracking**. In *System Science and Engineering (ICSSE), 2011 International Conference on*, pages 87–91, 2011. 13

[72] A. Doucet. *On sequential simulation-based methods for Bayesian filtering*. Tech. Rep., Cambridge Univ., 1998. 13

[73] D.C. Cherng, S.Y. Yang, C.F. Shen, and Y.C. Lu. **Real time color based particle filtering for object tracking with dual cache architecture**. In *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pages 148–153. IEEE, 2011. 14

[74] S. Avidan. **Ensemble Tracking**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **29**(2):261–271, 2007. 15

[75] D.H. Hubel and T.N. Wiesel. **Receptive fields of single neurones in the cat's striate cortex**. *The Journal of Physiology*, **148**(3):574–591, 1959. 15, 21

[76] T Shibata, M. Yagi, and M. Adachi. **Soft-computing integrated circuits for intelligent information processing**. In *Proc. Int. Conf. Information Fusion*, pages 648–656, 1999. 15

[77] H. Zhu and T. Shibata. **A real-time image recognition system using a global directional-edge-feature extraction VLSI processor**. In *ESSCIRC, 2009. ESSCIRC '09. Proceedings of*, pages 248–251, 2009. 15, 27, 46, 56, 67

[78] N. Takahashi, K. Fujita, and T. Shibata. **A Pixel-Parallel Self-Similitude Processing for Multiple-Resolution Edge-Filtering Analog Image Sensors**. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, **56**(11):2384–2392, 2009. 15

[79] H. Zhu, P. Zhao, and T. Shibata. **Directional-edge-based object tracking employing on-line learning and regeneration of multiple candidate locations**. In *Circuits and Systems (ISCAS), Proceedings of*

2010 IEEE International Symposium on, pages 2630–2633, 2010. 15, 16, 56, 62, 63, 84

[80] A. Nakada, T. Shibata, M. Konda, T. Morimoto, and T. Ohmi. **A fully parallel vector-quantization processor for real-time motion-picture compression**. *Solid-State Circuits, IEEE Journal of*, **34**(6):822–830, 1999. 21

[81] M. Yagi and T. Shibata. **An image representation algorithm compatible with neural-associative-processor-based hardware recognition systems**. *Neural Networks, IEEE Transactions on*, **14**(5):1144–1161, 2003. 21

[82] G. Tsagkatakis and A. Savakis. **Online Distance Metric Learning for Object Tracking**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **21**(12):1810–1821, 2011. 38

[83] Z.W. Pylyshyn and R.W. Storm. **Tracking multiple independent targets: Evidence for a parallel tracking mechanism**. *Spatial vision*, **3**:179–197, 1988. 47

[84] F. Tang, S. Brennan, Q. Zhao, and H. Tao. **Co-Tracking Using Semi-Supervised Support Vector Machines**. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007. 53, 54

[85] M. Tian, W. Zhang, and F. Liu. *On-line ensemble SVM for robust object tracking*, **4843** of *Lecture Notes in Computer Science*. Springer-Verlag, 2007. 53, 56

[86] L. Matthews, T. Ishikawa, and S. Baker. **The template update problem**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **26**(6):810–815, 2004. 56

[87] R. Zhang. **Real-Time On-Line-Learning Support Vector Machine Based on a Fully-Parallel Analog VLSI Processor**. *Artificial Intelligence and Soft Computing*, 2012. 56, 67

[88] N. Liu. *Incremental learning with support vector machines.* Proc Int. Joint Conf. on Artificial Intelligence (IJCAI-99), 1999. 57, 59

[89] G Cauwenberghs. **Incremental and decremental support vector machine learning**. *Advances in Neural Information Processing Systems*, 2001. 57

[90] S. Ruping. **Incremental learning with support vector machines**. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 641–642, 2001. 57

[91] P. Zhao, H. Zhu, H. Li, and T. Shibata. **A Directional-Edge-Based Real-Time Object Tracking System Employing Multiple Candidate-Location Generation**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **PP**(99):1, 2012. 84, 88, 89

[92] S. Gu, Y. Zheng, and C. Tomasi. **Efficient visual object tracking with online nearest neighbor classifier**. *Computer Vision–ACCV 2010*, pages 271–282, 2011. 85