博 士 論 文
（Doctoral Thesis）

Research on Efficient Vision-Based Hardware Architecture for Feature Representation and Object Detection

（特徴表現及び物体検出のための効率的なビジョンベースのハードウェアアーキテクチャに関する研究）

駱　愛文
(Aiwen Luo)

広島大学大学院先端物質科学研究科
Graduate School of Advanced Sciences of Matter
Hiroshima University

2018 年 3 月
（March 2018）

目　　次
（Table of Contents）

Extended Abstracts of the International Conference on Solid State Devices and Materials, Tsukuba, 2016, pp.457-458.

(2) Reconfigurable Block-based Normalization Circuit for On-chip Object Detection

Aiwen Luo, Fengwei An, Xiangyu Zhang, Lei Chen and Hans Jürgen Mattausch

Extended Abstracts of the International Conference on Solid State Devices and Materials, Sendai, 2017, pp.819-820.

(3) A Hardware Architecture for Cell-based Feature-Extraction and Classification Using Dual-Feature Space

Fengwei An, Xiangyu Zhang, Aiwen Luo, Lei Chen, Hans Jürgen Mattausch

IEEE Transactions on Circuits and Systems for Video Technology, vol.PP, no.99, pp.1-13, 2017.

(4) A Hardware-oriented Histogram of Oriented Gradients Algorithm and its VLSI Implementation

Xiangyu Zhang, Fengwei An, Ikki Nakashima, Aiwen Luo, Lei Chen, Idaku Ishii and Hans Jürgen Mattausch

Japanese Journal of Applied Physics, vol.56, no.4S, pp.04CF01:1-7, 2017.

(5) Pixel-based Pipeline Hardware Architecture for High-performance Haar-like Feature Extraction

Yuki Fujita, Fengwei An, Aiwen Luo, Xiangyu Zhang, Lei Chen, Hans Jürgen Mattausch

IEEE Asia Pacific Conference on Circuits and Systems, Korea, 2016, pp.611-612.

主　論　文
(Main Thesis)

Contents

Abstract

Machine learning (ML) is described by Arthur Samuel as the field of study that gives computers the ability to learn without being explicitly programmed. The methods for solving the ML problem can be classified into two groups, i.e. supervised learning and unsupervised learning. Pattern recognition is a branch of ML that focuses on the recognition of patterns and regularities in data. Pattern recognition systems can be trained from labeled training data in supervised learning, but also can be used to discover previously unknown patterns in unsupervised learning when no labeled data are available.

Feature representation is becoming an important prerequisite for building ML models to achieve high detection performance, in particular in vision-based applications. A wealth of feature-extraction techniques are being presented in this research field. To meet the requirements of embedded systems with limited resources, this thesis developed a simplified scale-invariant speeded up robust features (SURF) approach, which can decrease object-matching complexity and enhance computational performance significantly, to extract the feature vectors of sliding windows in the image. The further investigated histogram of oriented gradient (HOG) approach uses a widely accepted feature for object detection, which attains high accuracy against changes of illumination of variously textured objects.

Comparing to hardware solutions, the software technologies for feature extraction are more affected by computationally demanding algorithms so that the power dissipation of software technologies is burdensome for mobile applications and the processing speed becomes a bottleneck in case of real-time (>30 frames/s) object detection. Consequently, the achievable advantages of faster processing speed and lower power consumption lead to application-specific integrated circuit (ASIC) solutions.

Therefore, a resource-efficient coprocessor for a simplified SURF descriptor, employing Haar-like wavelets as feature vectors (FVs), is developed and a prototype is fabricated in 180nm CMOS technology in this research. A pipelined hardware architecture with low computational budget, that directly uses the serially-inputted pixel data without pre-storage, is employed for the local image cells. With a novel multiplexing process of the cell-based partial feature vectors, developed in this research work, the multidimensional feature vectors of multiple scan windows can be extracted simultaneously. The high achievable frame rate (up to 325 frames/s) is more than sufficient for real-time processing, while only 12 k bytes of on-chip memory space is consumed for feature extraction with the developed simplified SURF descriptor in this

research. The die area of the prototype is reduced by more than 98.25%, when comparing to previous works. The normalized power consumption is 8.723 mW, using constant field scaling to 65 nm CMOS, which is 96.04% less power dissipation than reported in previous works and is thus much more suitable for mobile applications with limited battery capacity.

After their completion, the cell-based local FVs are sequentially outputted for the parallelized window-level multidimensional feature representation, and for subsequent direct object matching without normalization. In this research, the concept of "regular rule of reusing times" (RRRT) of each cell is proposed so that the multiple window-based descriptor vectors can be constructed in parallel with high speed through access to look-up tables.

The first object detection framework, which combines the pipelined feature extraction, the cell-based sliding window algorithm and the nearest neighbor search (NNS) classifier, is verified in 65 nm CMOS technology. During the entire object detection procedure, the prototype coprocessor can achieve 34.6 fps VGA (640×480) frame rate when working at 200 MHz frequency. Furthermore, the requirements of 1.26 mm^2 die area and 26 k bytes on-chip memory result in 31.49 mW power dissipation at 1.0V supply voltage. The coprocessor can substantially reduce computation cost without significant degradation of classification accuracy. The achieved maximal detection accuracy for front-car images is 98.78% and 93.90% for pedestrian detection.

Since the object detection is greatly hampered by a large amount of high-dimensional FVs in the former-proposed feature-construction scheme, I further employ partial least squares (PLS) analysis to project the FVs data onto a much lower dimensional subspace, to reduce the computational amount and to save significantly in resource consumption. Up to 242.48 MHz frequency of the developed hardware architecture is verified for pedestrian detection operated on the Altera Stratix-IV field-programmable gate array (FPGA) platform.

Furthermore, a block-based sliding window algorithm with normalized feature vectors by the L1-norm scheme is developed for the second object detection framework that combines the HOG descriptor and the support vector machine (SVM) classifier. The improved detection framework aims to increase the pedestrian detection accuracy and robustness. Since fixed application-field parameters will normally make a hardware architecture unsuitable for practical scenarios, flexible regulation of image size, cell size (CS), etc., is desired in the second object detection framework to allow usage in a variety of mobile applications with the usually occurring changes under various circumstances. Variable image resolutions up to 1024 (width) × ∞ (height) pixels can be handled with

the same hardware architecture without any modification due to the reported research results. Five CS levels, i.e., 2×2, 4×4, 8×8, 16×16, 32×32 pixels with an unlimited image height, applied in the on-chip FV-extraction circuit, are implemented in another 65nm CMOS prototype ASIC. Up to 125 fps of XGA (1024×768) images can be handled for object detection by the developed coprocessor. 41 k bytes on-chip memory and 2.86 mm^2 die area are consumed in this scheme. The on-chip memory in this scheme seems to increase when compared to the first framework with the NNS classifier, but this increase is only due to the higher dimensionality of the HOG descriptor (3780 dimensions) in comparison to the SURF descriptor (1680 dimensions). Moreover, the on-chip memory requirement increases linearly with the reference number for the NNS classifier, whereas the coefficients number of the SVM classifier is fixed for a given window size. Thus on-chip storage and area requirements for the second framework can be smaller than for the first framework when many references are used in the NNS classifier. The power consumption is 21.3 mW when the coprocessor operates at 125 MHz frequency and 1.0 V supply voltage.

In conclusion, this research developed a cell-based feature extraction architecture and two object detection frameworks combined with NNS and SVM, respectively. Both the NNS and the SVM classifiers are applied for verifying the high detection performance of the proposed hardware architectures, which are sufficient to meet the requirements of mobile applications.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

1.1.1 Machine learning and pattern recognition

Machine learning (ML) is described by Arthur Lee Samuel in 1959 [1] as the field of study that gives computers the ability to learn without being explicitly programmed. As a branch of artificial intelligence (AI), ML automates analytical model building based on the idea that machines should be able to learn and adapt to experience. The study and computer modeling of learning processes in their multiple manifestations constitute the subject matter of ML [2]. ML is a science that is not a new concept [3] but one that has gained fresh momentum and is resurged with new interest due to great improvements resulting from the dynamic growth of new computing technologies and from the discovery of new facts and theories through observation and experimentation in the past decade. The acquisition of new knowledge makes researchers strive to implant intelligent capabilities into machines in the computer era. ML has become more popular than ever because it can analyze bigger, more complex data and deliver faster, more accurate results. The popular ML produces precise models more quickly and automatically for data mining and analysis.

The ML problems can be generally classified into two broad groups, i.e., supervised learning and unsupervised learning. Supervised learning infers a function from labeled training data that have a certain relationship with the output data [4]. As illustrated in Fig.1.1(a), what the correct outputs should look like in blue or red color is already known by the researcher in supervised learning problems. By contrast, unsupervised learning allows us to deal with problems even when little or nothing is known about the variables, and to derive a structure by clustering the data based on relationships among the variables in the data, as illustrated in Fig.1.1(b), with no feedback on the accuracy of prediction results [5].

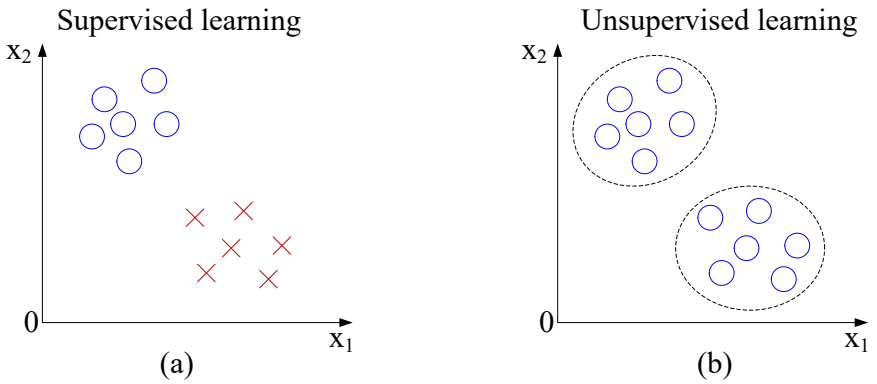


Fig.1. 1 Demonstration of supervised learning and unsupervised learning.

One classic approach to unsupervised learning is the *k*-means clustering, which partitions a set of data samples automatically into *k* clusters that are similar or related by different variables, depending on the nature of the data, the boundary information and the clustering classes [6, 7]. There is no feedback based on the prediction results with unsupervised learning, but we can apply it to build structures in a chaotic environment with mass data.

Regression and classification are two main categories for dealing with supervised learning problems [8].

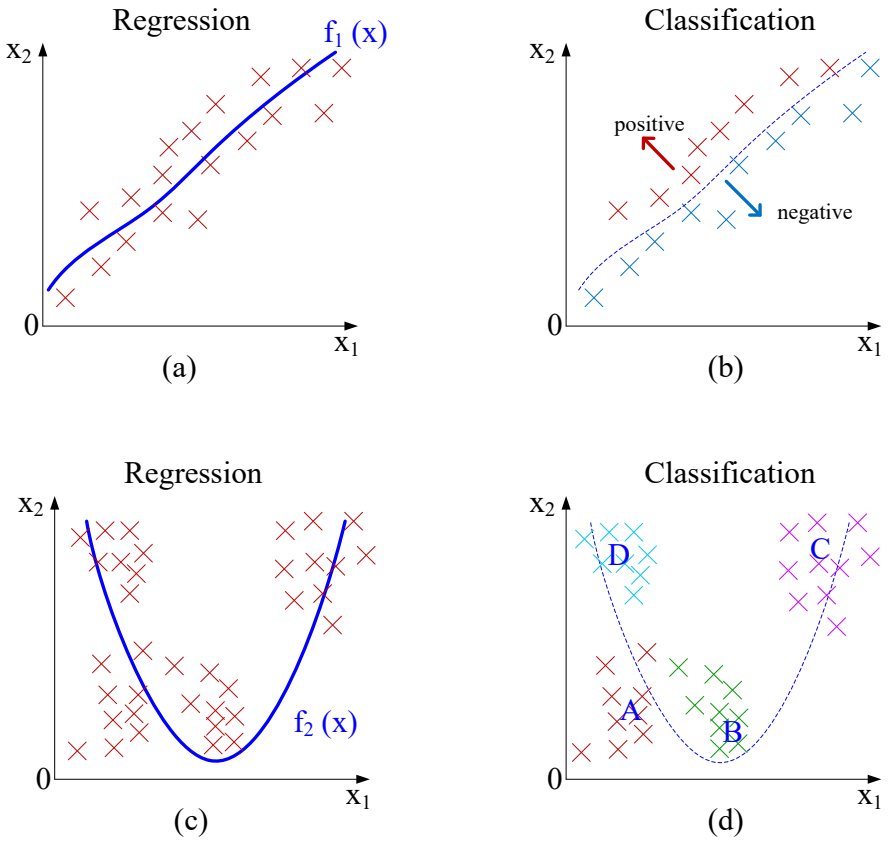


Fig.1. 2 Demonstration of regression and classification in supervised learning.

Continuous results and tendencies can be predicted by a regression model, while the discrete outputs instead are obtained on the basis of some given information by a classification model. For instance, the given input data in Fig.1.2 (a) is being mapped to the continuous function $f_1(x)$. In a classification problem, the given input variables are used to predict the sorting results into discrete categories, e.g., two discrete categories as a positive result or negative result, as illustrated in Fig.1.2 (b). More complicated demonstrations, with high dimensionality of the continuous function $f_2(x)$ for regression and more discrete categories for classification are shown in Fig.1.2 (c) and Fig.1.2 (d), respectively. Specific examples for regression and classification can be found in various previous research works [9, 10].

Pattern recognition is a branch of ML that focuses on the recognition of patterns and regularities in data [11]. The problem of searching for patterns in data is fundamental and has a long and successful history [12-15]. Pattern recognition systems can be trained with labeled training data in supervised learning such as object classification, but also can be used to discover previously unknown patterns when no labeled data are available in unsupervised learning with e.g. clustering algorithms.

The pattern recognition is concerned with the automatic discovery of regularities in data through the chosen computer algorithms and applies these regularities to classify the input image into different categories as illustrated in Fig.1.3.

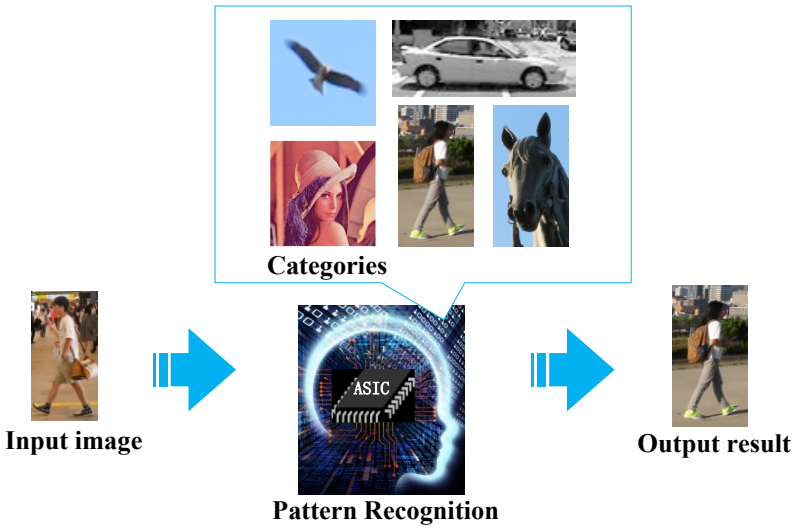


Fig.1. 3 Schematic of the main procedure of general pattern recognition.

The categories of the specific objects shown in Fig.1.3 are known in advance. Far better results can be obtained by adopting an ML approach with trained categories in a specific training set, which is used to tune the parameters of an adaptive model. The result of running the pattern recognition can be described as a function $\mathbf{f}(\mathbf{x})$. The desired

form of the function **f**(**x**) can be determined during the training phase, also known as the learning stage, on the basis of the training dataset. The identity of new input images, comprising a test dataset, can be determined once the model is trained. Taking a new image x_1 as a testing input, an output vector $f(x_1)$ can then be generated, encoded in the same way as the target vectors in supervised learning problems.

In other pattern recognition problems, i.e., unsupervised learning problems, the training data consists of a set of input vectors **x** without any corresponding target values. The goal of such problems is mainly to discover groups of similar examples within the categories called 'clusters', as indicated in Fig. 1.3, or to take density estimation, determining the distribution of data within the input's high-dimensional feature space.

1.1.2 Machine vision and feature representation

Pattern recognition based on specific feature vectors is an important methodology for digital image and video analysis in many machine-vision-based scenarios. Digital images are a convenient media for describing and storing spatial, temporal, spectral and physical components of information contained in a variety of domains (e.g. satellite images, biomedical images) [16]. Representation of image information by specific features is one of the most important factors that affect object detection performance.

In many past applications of pattern recognition, some form of fixed pre-processing concentrated on the feature extraction (FE). The objects in digital images scanned by a sensor are mapped into a multidimensional feature space through a specific feature representation by a descriptor as illustrated in Fig.1.4.

Generally, the features are applied for representing the reduced sets of values, extracted from the input data, e.g. the parts of interest in an image for vision-based applications.

Using such a reduced representation instead of the complete input data (e.g. raw pixels of image frames) facilitates the subsequent pattern recognition processing. Many kinds of feature vectors have been applied for complicated scenes that are suitable for various application fields. Selection of a FE method is probably the most important factor for achieving high recognition performance in applications.

A considerable amount of research work has been carried out on FE for different representations. Feature descriptors such as the gradient location-orientation histogram (GLOH) [17], the shape context [18], the local binary pattern (LBP) [19], and higher-order local autocorrelation (HLAC) [20] were proposed to provide specific descriptors of significant image features based on different construction principles.

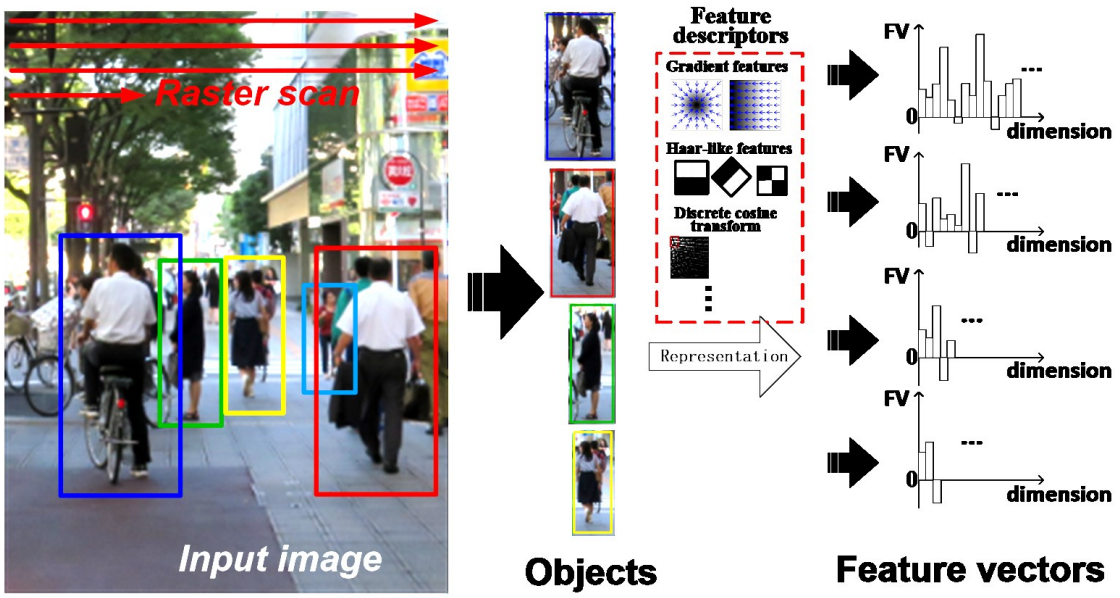


Fig.1. 4 Objects to be detected and the corresponding feature representation with specific descriptors

Scale-invariant feature transform (SIFT) [21] as one of most popular FE algorithms has shown great robustness or invariance to changes in image scale, rotation, illumination, 3D camera viewpoint, and noise. However, the SIFT algorithm has high computational cost and memory requirement, which prevents it from real-time applications, especially in the case of implementation by a pure software scheme. The scale-invariant speeded up robust feature (SURF) [22, 23] approach is regarded as an improved scheme which is partly inspired by the SIFT and can decrease object-matching complexity and enhance computational performance significantly, so that it is more suitable for embedded systems with limited resources.

The original SURF consists of two distinct stages: detection and description. The main work of the detection stage is to search for essential interest points in scaled image-frame pyramids. In contrast, the description stage of SURF aims at collecting Haar-filter responses around each interest point in chosen orientations, which constitutes the components and determines the dimension of the feature vector. The Haar sequence was proposed by Alfréd Haar [24] in 1910, and has been widely used and highly developed until now. The advantage of the Haar-wavelet responses, which are calculated for the SURF descriptor, is that they enable higher processing speed and better repeatability than other descriptors, thus decreasing object-matching complexity and enhancing computational performance. Histogram of oriented gradients (HOG) [25] is also a popular and widely accepted feature descriptor for pattern recognition and attains high accuracy against changes of illumination of variously textured objects. Feature extraction and construction is becoming an essential prerequisite for building object-classification

models to achieve high detection performance.

1.1.3 Object detection and classification

In the computer-vision-based solutions, object detection is the technology that is related to image processing. Object detection is the process of sorting the instances of semantic objects into classes such as vehicles, pedestrians, faces, trees and buildings as illustrated in Fig. 1.5. Specifically, object detection is most widely used in the research domains of face detection and pedestrian detection in the current decade.

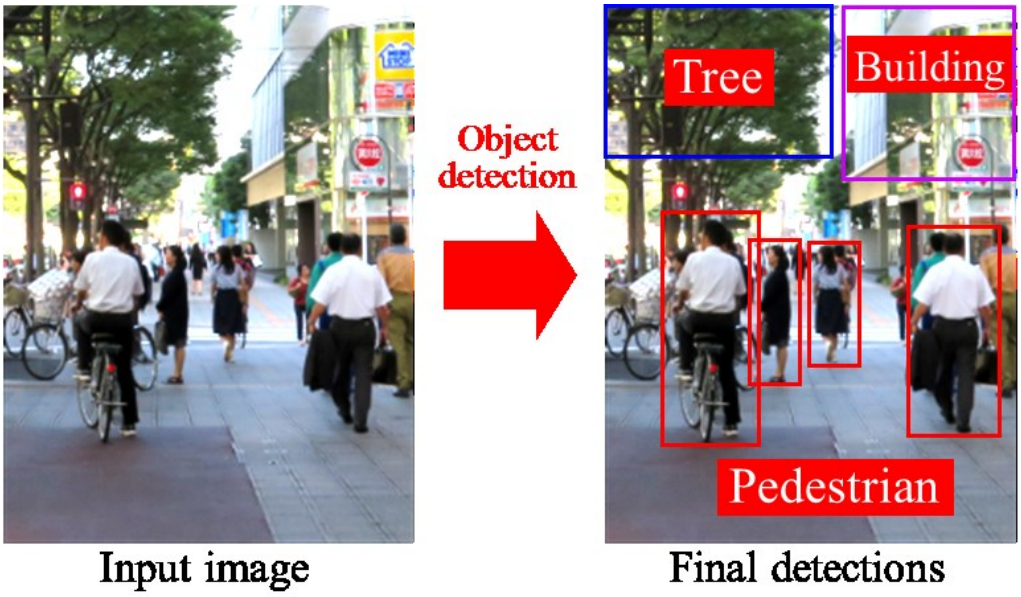


Fig.1. 5 General applied scenario for detecting different objects including tree, pedestrian and building.

Generally, there are two main steps in an object-detection system as illustrated in Fig.1.6: 1) Extracting a number of features, and 2) Training a classifier. In the feature extraction phase, different feature extraction methods, which have been introduced in section 1.1.2, are designed for different representations of the characters in images. In the second phase of training a classifier from a given dataset, including both positive image samples (responding to the target object) and negative image samples, the aim is to predict the existence of objects within input testing images, and further to detect the location of the objects.

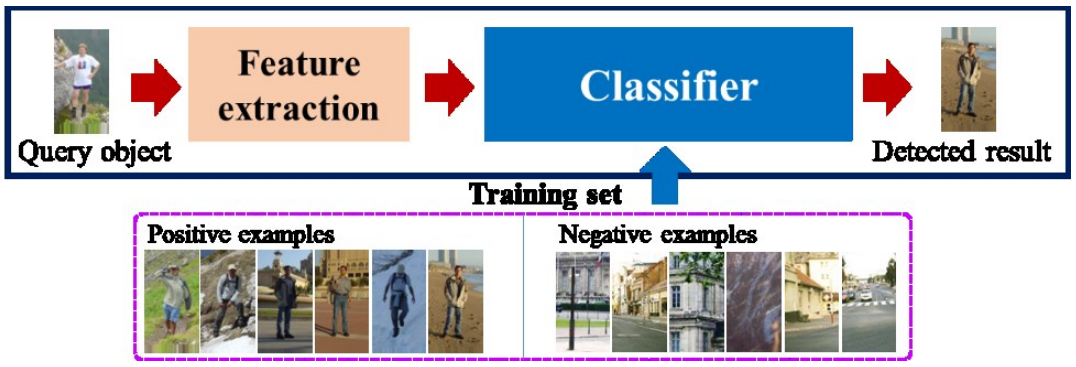


Fig.1. 6 The general flowchart for an object detection system.

A classifier is an algorithm that takes a set of features that characterize objects and uses them to determine the type (or class) of each object. For each object, the features measure a number of properties (e.g., brightness, size, shape, texture, etc.) and the classifier then uses these properties to determine the class that each object belongs to. Classifiers can give an estimate for the probability that an object belongs to each of the candidate classes. For supervised classification, a set of known objects called the training set are used to determine into which classes an object may be categorized. The positive samples and negative samples for the two known classes are used for training by the classification programs to learn how to classify objects. There are also unsupervised classification methods in which the induction engine works directly on the input images, and there are neither training sets nor pre-determined classes.

A generic image classification approach in pattern recognition collates heterogeneous training images to certain categories and finds the one category of objects that is the most likely to be present in an image given for evaluation.

Object detection can apply appropriate classifiers, such as a support vector machine (SVM), a nearest-neighbor (NN) distance estimation or a neural network as a searching engine to find out the target object with favorable properties in the evaluated visual data. Artificial vision systems can be powerful tools for automatic inspection of e.g. fruits or vegetables, as illustrated in [30]. This thesis will also employ the NN search (NNS) and the SVM to evaluate the performance of the developed circuits in chapter 5.

1.2 Motivation and purpose

1.2.1 Motivation for mobile applications

These days, many video surveillance systems are applied in public areas such as roads, airports, supermarkets, train stations, and subways, etc. Image and video capture

programs are widespread in daily life and also intended for military and other purposes. Various application fields, such as advanced driver assistance systems (ADAS) [26], wearable devices [27], emotion recognition [28] or unmanned aerial vehicles (UAV) [29] also extract feature vectors for machine vision in complicated scenarios.

The basic function of ADAS is to detect specific target objects, do classification, alert the driver of hazardous road conditions, and slow or stop the vehicle in some cases as shown in Fig.1.7.

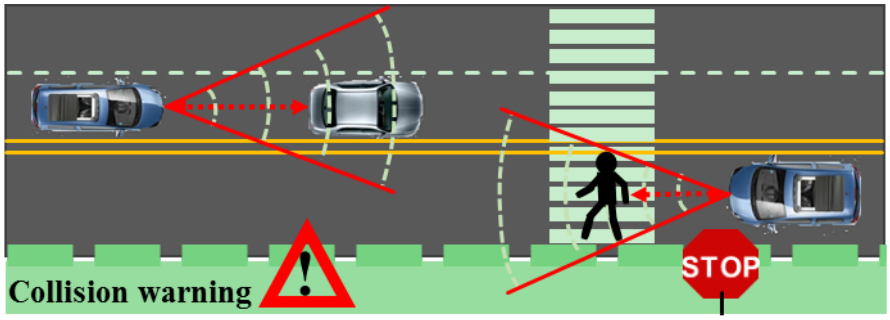


Fig.1. 7 An applied scenario for ADAS to deliver collision warnings.

The ADAS is great for fast-moving applications like blind spot monitoring, lane change assistance, and forward collision warnings. Thus ADAS is experiencing rapid adoption and growth in automotive systems enabled by advancements in the object detection and classification technology.

The movement of the current UAV system relies on global positioning system (GPS) for navigation combining low-cost inertial sensors, sonar, and computer vision techniques, as illustrated in Fig. 1.8. Navigation and control are two key technologies for an unmanned aerial vehicle (UAV). The GPS navigation becomes invalid in indoor applications or when the UAV is hidden by shelters like the trees. Therefore, the embedded guidance system on the UAV plays a more important role with the advancements in the object detection and classification technology. The UAV control shares qualities with typical robotic motion planning problems.

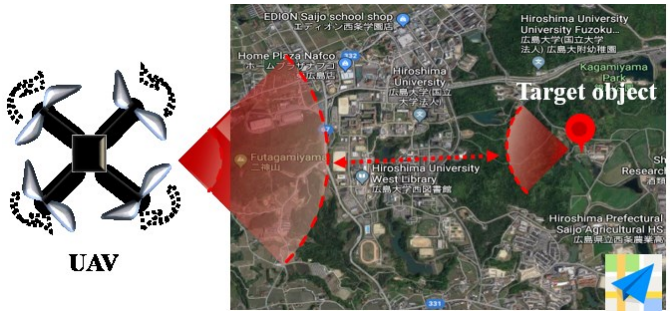


Fig.1. 8 An applied scenario for navigating and controlling a UAV.

Smartphones are widely used in the last decade and it can be regarded as the most representative device of mobile applications. More and more intelligent functions are integrated in a mobile device which is far more than its name “phone”. Specifically, in order to recognize objects from images recorded by the camera of a mobile device, feature extraction and object detection are applied for mobile applications such as human face recognition as illustrated in Fig.1.9.

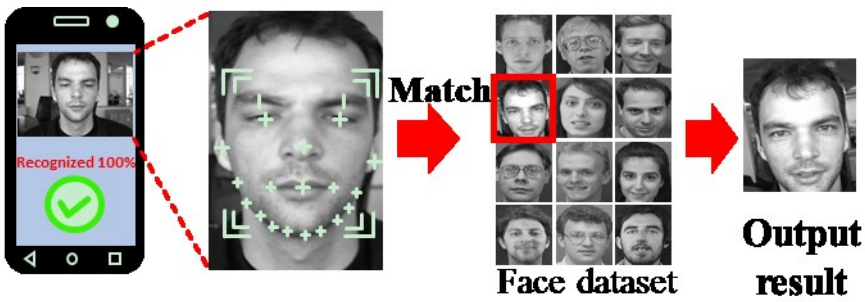


Fig.1. 9 An illustration of mobile face recognition.

For mobile applications of object detection and recognition, there are many factors that should be considered.

In this thesis, the motivation is to develop a hardware-oriented object detection system for mobile applications such ADAS, UAV or face detection as discussed above. As the onboard battery of the mobile device is the critical factor that restricts the utilization in practice, low power dissipation is the most desired performance to provide enough battery storage to run object detection on the basis of sufficient detection accuracy. Further, the resource requirements determine the power consumption and the device cost. Thus reduction of computational complexity is necessary for mobile applications in hardware designs. Additionally, sufficient processing speed (> 30 fps) is essential for practical significance. More functions are desired for mobile applications.

1.2.2 Related researches and existing problems

Many related feature-based object-detection solutions by software, hardware or combined hardware-software implementations have been reported in many experiments and applications while maintaining high enough accuracy. The machine or computer vision applications tend to deal with high image resolutions and high frame rates in recent years.

On one hand, software libraries like OpenCV [31], OpenCL [32] as well as OpenGL [33], are supported by multi-core processing systems, CPUs or graphics processing units

(GPUs), so that the conventional vision-based algorithms become feasible for various feature descriptors, enabling rapid prototyping and overall shorter development periods. However, the dependence on high-performance hardware devices or platforms makes these software-library solutions unsuitable for largely outdoor mobile-application scenarios in spite of the support of rapid prototyping and shorter development periods. On the other hand, to support the substantial computational amount for processing multiple frames per second in real time, many hardware implementations for feature-based object recognition have been proposed in the current decade. For example, a SIFT detector, using a heterogeneous methodology to partition the workload between CPU and GPU for embedded systems [34], suffers from similar power consumption (3383 mW) as the CPU-only scheme (3186 mW), and from the relatively long execution time of 148 ms per image frame even with a small image size of 224 pixels × 224 pixels. Another CPU-GPU hybrid computing method [35] for feature extraction was introduced with significant improvements in processing speed. However, this solution is still energy-hungry, while working at an extremely high frequency, and speed-limited, when compared to an application-specific integrated circuit (ASIC) [36, 37].

Many hardware implementations based on field-programmable gate array (FPGA) or ASIC solutions are also proposed in the current decade. A programmable FPGA-based solution [38] is flexible for different functional designs with relatively short developing period and high processing speed.

TABLE I. I
Performance of Recent Hardware Solutions for Feature-Based Object Detection

	WSPS [38]	TCSVT [39]	JSSC [40]	CICC [41]
Technology	FPGA	65nm CMOS	65nm CMOS	65nm CMOS
Feature type	HOG	MSEM	SIFT	SURF
Frequency(MHz)	404	20	200	200
Resolution (pixels)	1920×1080	100×100	1920×1080	1920×1080
Frame rate(fps)	209	40	30	57
Memory usage(kB)	108	14.625	40	400
Peak Power(mW)	3600	9	198.4	220

However, the power consumption for a FPGA-based solution is extremely large when compared to ASIC solutions [39-41] as shown in Table I.I. Moreover, the machine or computer vision applications tend to require high resolutions and high frame rates, which leads to more energy computation. As opposed to the rapid growth of computer vision, the development of battery-storage capacity is meeting a significant bottleneck,

which causes great limitations for energy-critical mobile applications. Moreover, the large data-storage requirements and the limited memory bandwidth for image processing also cause difficulties for high-performance implementations of object detection in mobile terminals. The hardware computational cost is mainly depended on the algorithm for the chosen feature type and the intrinsic characteristic of the hardware architecture or platform. In other words, the feature type and hardware architecture are two critical factors in hardware implementations for object detection.

There is a pretty important concept named integral image which was introduced to computer graphics by Franklin C. Crow [42] in 1984 and was used for multiple previous research works. The integral image has become one of the most popular methods applied in the realization of feature-based object-detection systems because of its convenience for calculating the pixel-intensity sum for any rectangle area of the image. The calculation time of the integral image is independent of the rectangle size as illustrated in Fig. 1.10, so that a remarkable improvements of feature-calculation speed can be achieved. However, integral images require large memory space for accumulated-sum storage [41, 43], which is hardly affordable for resource-limited on-chip systems in mobile embedded applications. Therefore, an alternative scheme is requested for feature calculation of visual data from the image sensor more efficiently and meeting this requirement is an import objective in this research.

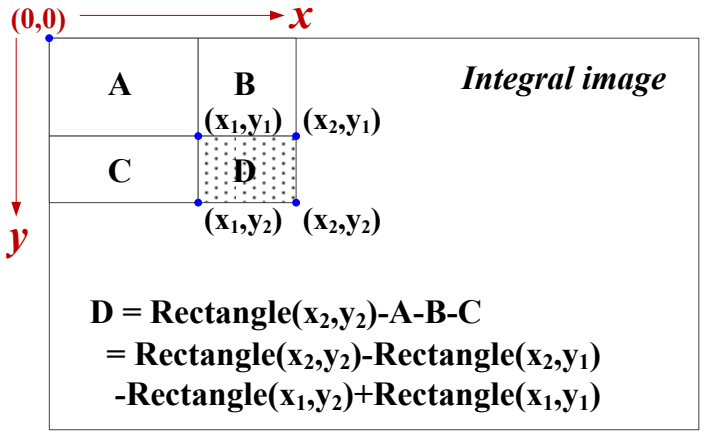


Fig.1. 10 Explanation of integral-image-based pixel-value summation in an arbitrary rectangle within the image. The integral image starts from the origin point (0, 0) of coordinates. Pixel summation of rectangle D can be computed by the four cumulative sums corresponding to its four corners, i.e., at the locations (x_1,y_1) , (x_2,y_1) , (x_1,y_2) and (x_2,y_2).

1.2.3 Purpose of this research

The purpose of this research is to analyze the main impact factors in relation to

resource efficiency, high processing speed, low computational cost and low power consumption, so as to construct reconfigurable hardware structures for flexible regulations of feature extraction and object detection that are suitable for dedicated mobile applications and handheld devices.

The novel proposals to approach the realization of this purpose are as follows:

Firstly, considering the insufficiencies in current strategies for feature-based object detection, this research proposes a resource-efficient hardware architecture for a representative feature descriptor, i.e., the simplified SURF descriptor employing Haar-like wavelets as feature vectors. It can substantially reduce computational cost without significant degradation of classification accuracy due to the applied simplified structure. A low computational-budget hardware architecture, that directly uses the serial input-pixel data without pixel pre-processing such as for integral images, is employed for feature vector extraction of local image cells.

Secondly, an innovative window-based search approach is proposed for high-dimensional feature extraction and parallel object detection at the same time. The partially computed feature vectors of multiple simultaneously-processed scan windows are sent to the detection engine to start already searching for target objects while the feature vectors of the scan windows are constructed. The detection operation synchronizes with the serial output of cell-based local feature vectors. Due to the efficiency of the cell-based multi-window parallel process, a significant reduction of power and storage consumption is achieved in this research. Furthermore, feature-dimensionality reduction, which employs the partial least squares (PLS) regression, are included to reduce the computational cost.

Finally, experimental detection-performance verifications with different classifiers are reported in this work. In particular, the detection procedure, employing nearest neighbor search (NNS) or SVM as the object classifier, is applied to quantitatively verify the obtainable achievement of a significant reduction in power and storage consumption without significant degradation of detection accuracy. Flexible image resolutions, handled by the coprocessor hardware architecture without any modifications, enhance the compatibility and versatility of the desired detection system for multi-scale or multi-object applications.

Hardware-friendly prototypes for the proposed hardware designs are mainly verified with SOTB (Silicon on thin BOX) CMOS technology so that it can be easily be applied to mobile applications as e.g. the popular handheld devices. An FPGA platform is also employed in this research for verifying the feasibility of the proposed algorithms and evaluating the detection performance.

1.3 Dissertation outline

The thesis is organized into six parts as follows:

Chapter 1 introduces the background including traditionally implemental approaches and discusses their pros and cons with respect to the vision-based object detection in the field of the popular machine learning. A highly-performing hypothesis for a feature-based detection scheme is proposed and formulated in this section.

Chapter 2 proposes a simplified SURF descriptor and its pipelined scheme for local cell-based feature extraction, which is a cornerstone of object detection in computer vision. The typical integral image is substituted by an immediate-processing engine for the serially input-pixel data from the image sensor without normalization, resulting in resource efficiency and real-time processing.

The developed window-based feature-space-construction algorithm, applying local cell FVs, is illustrated in chapter 3. Instead of extracting feature vectors only around the interest points as in the previous SURF algorithms, this research applies an overlapping scan-window approach for covering the entire image without detection of interest points.

In chapter 4, an improvement scheme with both FV normalization and dimensionality reduction is proposed for enhancing the detection robustness as well as decreasing the computational cost. The high-dimensional feature spaces are substantially reduced, to avoid the processing of dense and unnecessarily large data amounts during the recognition procedure. Flexible regulation strategies for highly-robust performance are explored with the L-norm scheme. Local feature vectors are further computed in terms of programmable cell size and window size with unlimited image height, to increase the flexibility for usage in many different applications.

Chapter 5 illustrates the hardware results with different object-detection frameworks, in combination with either simplified-SURF or HOG descriptors. The detection performance is compared and discussed for the cases of NNS and SVM classifiers.

The conclusions of this dissertation as well as the future-work tasks are summarized in chapter 6.

References

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol.3, no.3, pp.211-229, Jul. 1959.

[2] J. G. Carbonell, R. S. Michalski, T.M. Mitchell, "An Overview of Machine Learning," In *Machine Learning*, Symbolic Computation, Springer, Berlin, Heidelberg, 1983, pp.3-23.

[3] D. M. Dutton, and G. V. Conroy, "A review of machine learning," *The Knowledge Engineering Review*, vol.12, no.4, pp. 341-367, 1997.

[4] M. Mohri, A. Rostamizadeh, A. Talwalka, Foundations of Machine Learning, *The MIT Press*, 2012, ISBN: 9780262018258.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no.3, pp.264-323, 1999.

[6] G. H. Ball and D. J. Hall, "A Clustering Technique for Summarizing Multivariate Data," *Behavioral Science*, vol.12, no.2, pp.153-155, 1967.

[7] J. A. Hartigan, and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society*, Applied Statistics, vol.28, no.1, pp.100-108, 1979.

[8] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, "RA Olshen Classification and Regression Trees Chapman and Hall," *Classification and regression trees*, CRC press, 1984.

[9] A. Liaw, and M. Wiener, "Classification and regression by randomForest," *R news*, vol.2, no.3, pp.18-22, 2002.

[10] G. De'ath, K. E. Fabricius, "Classification and regression trees: a powerful yet simple technique for ecological data analysis," *Ecology*, vol.81, no.11, pp.3178-3192, Nov. 2000.

[11] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[12] S. Wold, "Pattern recognition by means of disjoint principal components models," *Pattern recognition*, vol.8, no.3, pp.127-139, 1976.

[13] J. S. Taylor, and N. Cristianini, *Kernel methods for pattern analysis*, Cambridge university press, 2004.

[14] T. Pavlidis, *Structural pattern recognition*, vol.1, Springer, 2013.

[15] K. Fukunaga, *Introduction to statistical pattern recognition*, Academic press, 2013.

[16] A. K. Jain, and A. Vailaya, "Image retrieval using color and shape," *Pattern recognition*, vol. 29, no.8, pp. 1233-1244, 1996.

[17] K. Mikolajczyk, and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.27, no.10, pp. 1615-1630, Aug. 2005, 10.1109/TPAMI.2005.188.

[18] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no.4, pp.509-522, Aug. 2002, 10.1109/34.993558.

[19] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.24, no.7, pp.971-987, Jul. 2002, 10.1109/TPAMI.2002.1017623.

[20] Q. Gu, T. Takaki, I. Ishii, "Fast FPGA-based multiobject feature extraction," *IEEE Trans. Circuits and Systems for Video Technology*, vol.23, no.1, pp.30-45, Jan. 2013, 10.1109/TCSVT.2012.2202195.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. journal of computer vision*, vol.60, no.2 pp. 91-110, 2004.

[22] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proc. Computer vision–ECCV2006*, Graz, Austria, 2006, pp. 404-417.

[23] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol.110, no.3, pp. 346-359, Jun. 2008.

[24] A. Haar, Math. Ann, "Zur theorie der orthogonalen funktionensysteme," *Mathematische Annalen*, vol.69, no.3, pp.331-371, 1910 [in German].

[25] N. Dalal, and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE CVPR*, San Diego, CA, USA, 2005, pp.886-893.

[26] F. Zaklouta, and B. Stanciulescu, "Real-time traffic sign recognition in three stages," *Robotics and autonomous systems*, vol.62, no.1, pp.16-24, Jul. 2012.

[27] O. D. Lara, and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorials*, vol.15, no.3, pp.1192-1209, 2013.

[28] Suchitra, P. Suja, and Shikha Tripathi, "Real-time emotion recognition from facial images using Raspberry Pi II," in *Proc. IEEE SPIN*, Noida, India, 2016, pp.666-670.

[29] A. Gaszczak, T. P. Breckon, and J. Han, "Real-time people and vehicle detection from UAV imagery," in *Proc. SPIE-IRCV XXVIII*, San Francisco, California, USA, 2011, pp. 78780B-1-13.

[30] S. Cubero, N. Aleixos, E. Moltó, J. Gómez-Sanchis, and J. Blasco, "Advances in

machine vision applications for automatic inspection and quality evaluation of fruits and vegetables," *Food and Bioprocess Technology*, vol.4, no.4, pp.487-504, May 2011.

[31] OpenCV library: http://opencv.org/.

[32] G. Wang, B. Rister, and J. R. Cavallaro, "Workload analysis and efficient OpenCL-based implementation of SIFT algorithm on a smartphone," in *Proc. IEEE Global Conf. SIP*, Austin, TX, USA, 2013, pp. 759-762.

[33] OpenGL library: https://www.opengl.org/.

[34] B. Rister, G. Wang, M. Wu, and J. R. Cavallaro, "A fast and efficient SIFT detector using the mobile GPU, " in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, 2013, pp. 2674-2678.

[35] S. Lee, H. Kim, D. Park, Y. Chung, and T. Jeong, "CPU-GPU hybrid computing for feature extraction from video stream," *IEICE Electronics Express*, vol.11, no.22, pp.1-8, 2014.

[36] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. Kelly, A. J. Davison, M. Luján, M. F. O'Boyle, G. Riley, N. Topham, and S. Furber, "Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM," presented at *Int. Conf. Robotics and Automation*, WA, USA, May. 26-30, 2015.

[37] A. K. Kushsairy, M. K. Kamaruddin, H. Nasir, S. I. Safie, Z. A. K. Bakti, M. R. Isa, and S. Khan, "Embedded vision: Enhancing embedded platform for face detection system," in *Proc. I2MTC*, 2016, pp. 1-5.

[38] C. Kelly, C. Kelly, F. M. Siddiqui, B. Bardak, and R. Woods, "Histogram of oriented gradients front end processing: an FPGA based processor approach," in *Proc. IEEE Workshop on Signal Processing Systems*, 2014, pp. 1-6.

[39] H. Zhu, and T. Shibata, "A Real-Time Motion-Feature-Extraction VLSI Employing Digital-Pixel-Sensor-Based Parallel Architecture," *IEEE Trans. Circuits and Systems for Video Technology*, vol.24, no.10, pp.1787-1799, Mar. 2014, DOI: 10.1109/TCSVT.2014.2313899.

[40] Y. C. Su, K. Y. Huang, T. W. Chen, Y. M. Tsai, S. Y. Chien, and L. G. Chen, "A 52mW full HD 160-degree object viewpoint recognition SoC with visual vocabulary processor for wearable vision applications," *IEEE Journal of Solid-State Circuits*, vol.47, no.4 pp.797-809, 2012.

[41] L. Liu, W. Zhang, C. Deng, S. Yin, S. Cai, and S. Wei, "SURFEX: A 57fps 1080P resolution 220mW silicon implementation for simplified speeded-up robust feature with 65nm process," in *Proc. IEEE CICC*, San Jose, CA, USA, 2013, pp. 1-4.

[42] F. C. Crow, "Summed-area tables for texture mapping," *ACM SIGGRAPH computer*

graphics, vol.18, no.3, pp.207-212, 1984.

[43] L. C. Chiu, T. S. Chang, Chen, J. Y., & Chang, N. Y. C. , "Fast SIFT design for real-time visual feature extraction," *IEEE Trans. Image Processing*, vol.22, no.8, pp.3158-3167, Apr. 2013.

Chapter 2

Local Cell-Based Feature Extraction

2.1 Introduction

The speeded up robust feature (SURF) descriptor is not only faster but also better repeatable than other descriptors by relying on integral images for image convolutions [1, 2]. The original SURF consists of two distinct stages, i.e., detection and description. The main work of the detection stage is to select and search essential interest points (IP) at distinctive locations in image pyramids. As illustrated in Fig.2.1, different numbers of IP can be selected to represent the image with repeatable results during the detection stage of the original SURF algorithm. Whereas, the description stage of the original SURF algorithm collects and describes Haar wavelet responses [3] around each interest point in chosen orientations, constituting the highly-dimensional feature vector for feature representation of various objects. But the generally computational-expensive SURF algorithm still consumes relatively high power, unacceptable for practical implementations of computer vision.

Fig.2. 1 An illustration of selection results of the original SURF algorithm with different numbers (2, 5, 10, respectively) of interest points (IP).

To ensure the overall quality of feature extraction and to reduce the power consumption, a pixel-based pipelined algorithm, necessitating feature extraction from the entire frame, was proposed in [4]. However, the preprocessing to generate an integral

image from raw pixels is still indispensable in both detection and description stages.

Instead of extracting feature vectors only around the IP as in the previous SURF algorithms, this research applies a simplified SURF approach for covering the entire image without detection of IP. The extracted Haar-like responses are used for window-feature-vector construction to recognize the target objects in the image pyramids, enhancing the computational efficiency and flexibility.

The Haar-wavelet-response calculation makes the SURF descriptor faster and more repeatable than other descriptors, thus decreasing object-recognition complexity and enhancing computational performance. A set of Haar-like feature types such as edge features, line features, center-surround features, diagonal feature and their separate rotated features in certain degrees, have been introduced with different restrictions in previous works [5-9], as illustrated in Figs.2.2 (a-d). Compared with raw pixels, the Haar-like features focus more on the information within a certain area of the image rather than every single pixel, which reduces/increases the in-class/out-of-class variability efficiently and thus makes classification easier [5].

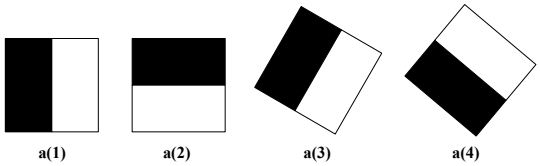


Fig. 2.2 (a) Edge Haar-like features and corresponding rotated features

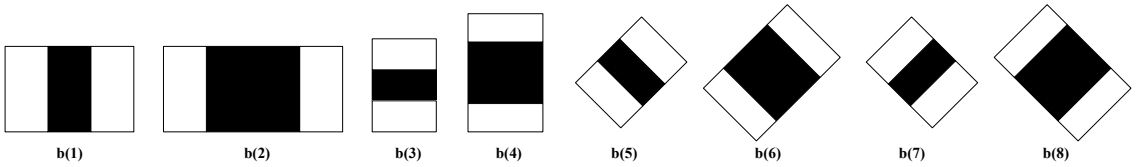


Fig. 2.2 (b) Line Haar-like features and corresponding rotated features

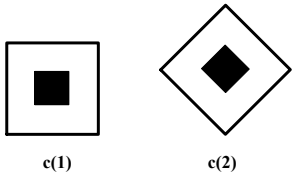


Fig. 2.2 (c) Center-surround Haar-like feature and corresponding rotated feature

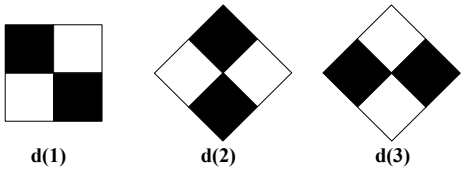


Fig. 2.2 (d) Diagonal Haar-like feature and corresponding rotated features

Fig.2. 2 Four types of Haar wavelet representations where black areas have opposite weights to white areas.

The Haar-like wavelets can be widely applied to object detection and recognition in a plenty of research works with respect to face recognition [6], hand gesture recognition [7] and pedestrian detection [8], etc. Black areas in Fig.2.2 (a-d) have opposite weights to white areas for each Haar wavelet representation to capture the structural similarities between instances of an object class [9].

Specifically, the edge Haar-like features shown in Fig.2.2 (a) are employed for the proposed simplified SURF descriptor in this research, whose black areas have positive weight "+1" while white areas have negative weight "-1".

2.2 Local cell-feature component extraction algorithm

In order to recognize the input object or to detect the target object in the input image, the input object or target object must be efficiently expressed in the feature representation. Let $\mathbf{F}(x, y)$ be a two-dimensional image pixel array of a image in x and y direction, which are directly scanned from the image sensor. $\mathbf{F}(x, y)$ can be used to denotes the color value at pixel $p(x, y)$ for color images represented in terms of the three primary RGB colors, i.e., red, green and blue, which be written as $\mathbf{F}(x, y) = \{F_R(x, y), F_G(x, y), F_B(x, y)\}$. On the other hand, $\mathbf{F}(x, y)$ can also be applied for denoting the grayscale intensity value, as illustrated in Fig.2.3, at pixel $p(x, y)$ for gray images.

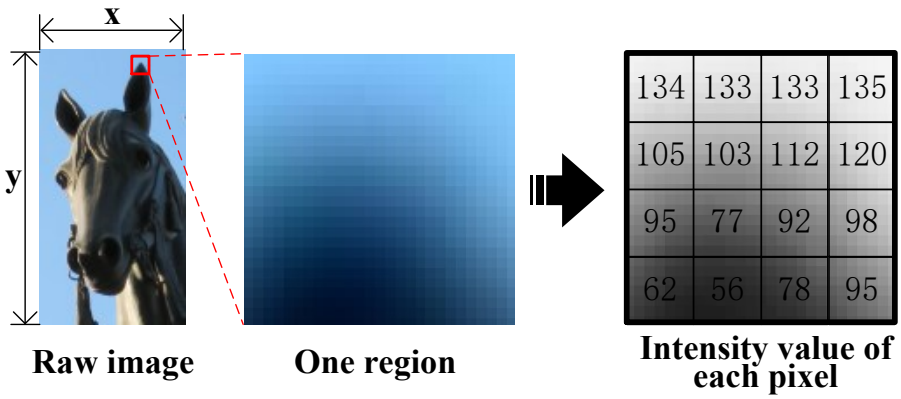


Fig.2. 3 Example of a raw image to be processed, one selected region and its corresponding grayscale-intensity values at each pixel.

Therefore, the image space can be mapped onto the n-dimensional feature space $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ by the mapping f:

$$f: \mathbf{F} \rightarrow \mathbf{X}, \qquad (2.1)$$

where n is the number of features used to represent the image.

In this research, the Haar wavelets, which are robust in decreasing object-matching complexity as well as enhancing computational performance, are applied for feature representation used in the SURF descriptor.

Specifically, the edge Haar-like features shown in Fig.2.4 are employed for the proposed simplified-SURF descriptor in this research, whose black areas have positive weight "+1" while white areas have negative weight "-1". A rectangular sub-division constructed by a 4×4-pixel array is defined as a sub-cell and applied for basic feature calculation for edge Haar-like wavelets.

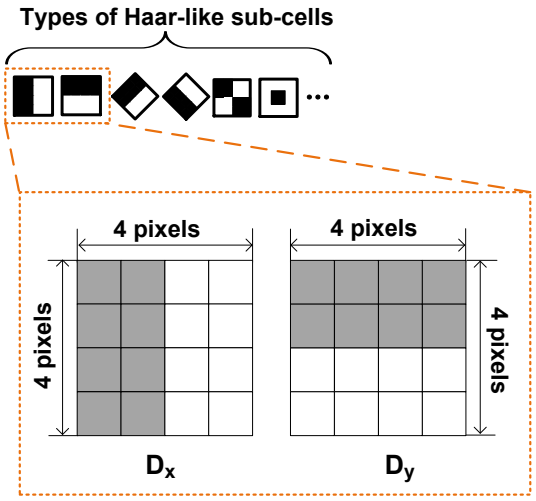


Fig.2. 4 Selected edge Haar-like feature extraction operated in a 4×4-pixel sub-cell.

For horizontal sub-cell responses, the left part and the right part of the 4×4-pixel sub-cell are summed up respectively before subtraction operation according to Fig. 2.5 and Eq. 2.2.

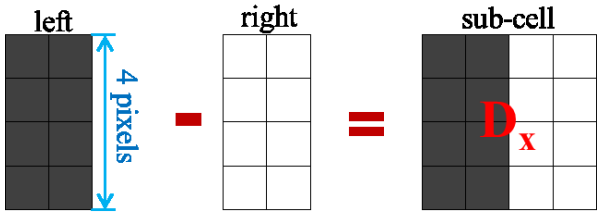


Fig.2. 5 Horizontal sub-cell response employed Haar-like wavelets.

$$D_x = \sum_{p(x) \in left_{sub-cell}} p(x) - \sum_{p(x) \in right_{sub-cell}} p(x) \qquad (2.2)$$

Here p(x) represents the pixel intensity values of a gray-scale image in horizontal (x) direction, which are directly scanned from the image sensor.

For the vertical sub-cell responses, the up part and the down part of the 4×4-pixel sub-cell are summed up respectively before subtraction operation according to Fig. 2.6 and Eq. 2.3.

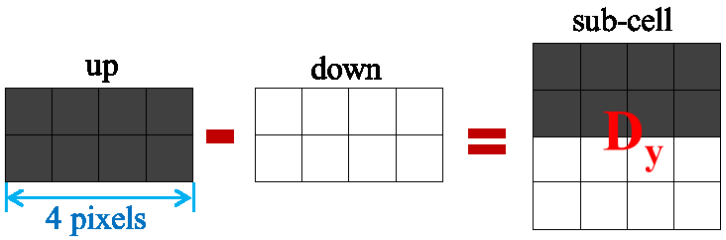


Fig.2. 6 Vertical sub-cell response employed Haar-like wavelets.

$$D_y = \sum_{p(y)\in up_{sub-cell}} p(y) - \sum_{p(y)\in down_{sub-cell}} p(y) \qquad (2.3)$$

Here p(y) represents the pixel intensity values of a gray-scale image in vertical (y) direction, which are also directly scanned from the image sensor.

According to the raster scan manner of the general image sensors, the inputted line-by-line raw pixels are immediately processed to progressively determine the Haar-like features D_x and D_y of related sub-cells used by the simplified SURF descriptor.

In order to bring in information about the polarity of the intensity changes, the absolute values $|D_x|$ and $|D_y|$ for each sub-cell are also extracted according to Eq. 2.4 and Eq. 2.5 respectively.

$$|D_x| = \left| \sum_{p(x)\in left_{sub-cell}} p(x) - \sum_{p(x)\in right_{sub-cell}} p(x) \right| \qquad (2.4)$$

$$|D_y| = \left| \sum_{p(y)\in up_{sub-cell}} p(y) - \sum_{p(y)\in down_{sub-cell}} p(y) \right| \qquad (2.5)$$

Important is, this research proposes a novel feature extraction algorithm that the intensity values p(x) and p(y) can be taken directly from the raw pixels of the image sensor serially inputted in raster manner, without any preprocessing or pre-storage as integral image. The feature calculating speed is depended on the working frequency of the image

sensor due to the operation synchronization. Thus the computational complexity and memory-resource consumption will be significantly reduced in comparison to previous solutions such as integral-image-based schemes.

A non-overlapped rectangle area with 2×2 sub-cells is further used to construct a 'cell', as illustrated in Fig. 2.7, which is the basic processing unit of the proposed simplified SURF descriptor. For the local Haar-like wavelet response of one cell, the response values D_x, D_y, $|D_x|$ and $|D_y|$ of four sub-cells that belong to the same cell are added up and form the local four-dimensional cell-response vector v_{cell} of Eq. 2.6 at the cell layer.

$$v_{cell} = \left\{ \sum_{x \in cell} D_x , \sum_{y \in cell} D_y , \sum_{x \in cell} |D_x| , \sum_{y \in cell} |D_y| \right\} \qquad (2.6)$$

The pixel-group differences of the four sub-cells in each cell are accumulated and form local four-dimensional cell-based feature-vector components. Synchronization with the image sensor's and immediate usage of each input pixel for the feature-construction process avoids the dependence on memory-intensive conventional strategies like integral-image construction or frame buffers.

An efficient control conception is essential for immediate processing of the serially inputted pixels for local cells feature components. In this research, a counter group in cooperation with the logic gates are applied for instant stepwise execution and enable signal generation of the proposed pipeline feature extraction for local cells.

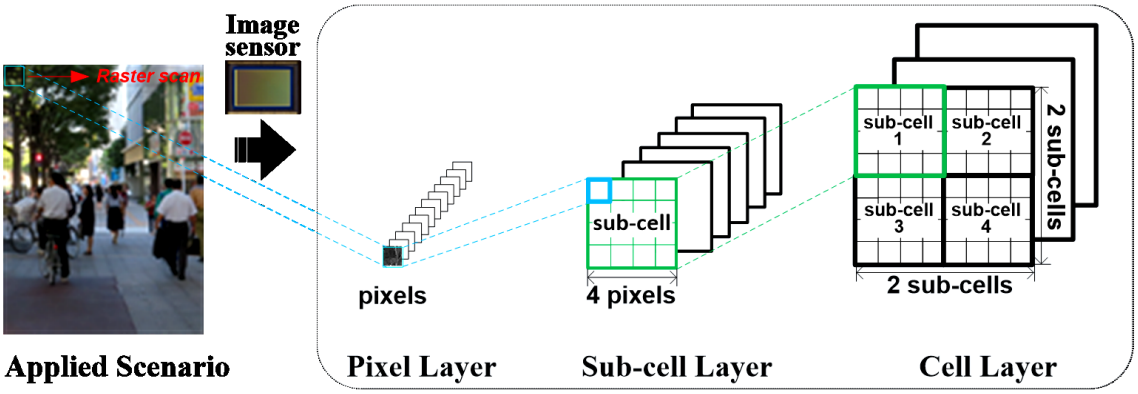


Fig.2. 7 Relationships between the processing units (i.e., pixel, sub-cell and cell) in the proposed cell-based feature extraction algorithm.

2.3 Hardware implementation of cell-based simplified-SURF feature extraction

2.3.1 Overall feature extraction architecture for local cells

Feature extraction and construction is becoming a real prerequisite for building object-classified models to achieve high detection performance. With regard to feature representation on the basis of the sliding-window strategy, the cell generally is the basic component for a search window in many feature descriptors. Specifically, there is another sub concept as sub-cell for the proposed simplified SURF descriptor applying the edge Haar-wavelet features in this research as illustrated in Section 2.2. This section is going to present a pipeline hardware architecture for feature extraction on cells which will be further applied for constructing a high-dimensional window-based feature vector in the following designs.

Given an entire input image in resolution of w (width) × h (height) pixels as shown in Fig.2.8, there are w/4 sub-cells and w/8 cells each row in response to a fixed sub-cell size (4 ×4 pixels) and cell size (8 ×8 pixels). With the serially-input pixels scanned in raster manner by an image sensor, each pixel will be sent instantly to be summed up inside the corresponding sub-cell and cell for both horizontal and vertical responses without buffering or any pre-processing. An idea that each pixel will be abandoned without retreatment after being invoked for response calculation of the non-overlapped sub-cell and cell is presented in this research. The intermediate calculated results will be temporarily stored or sent for secondary calculation.

Since the pixel scan procedure from image sensor runs consecutively without pause, the pixel processing for feature vectors should be also taken successively so as to avoid additional storage requirement or data loss. For each sub-cell row, the calculation results of the first three rows of pixels should be buffered until the last pixel of each sub-cell.

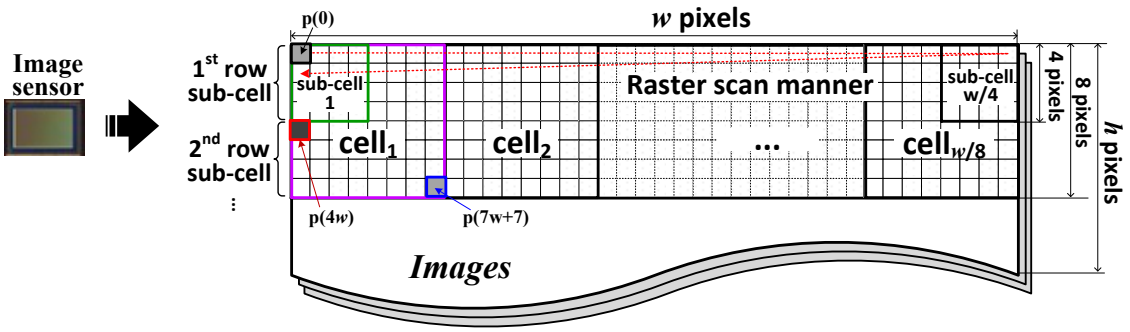


Fig.2. 8 Pixel scan manner of data supply form the image sensor and image-region division for feature extraction.

Figures 2.9 and 2.10 briefly illustrate the representative generation process of the four-dimensional feature components of the first cell 'cell$_1$' and the second cell 'cell$_2$' in the first cell row. Due to the raster scan manner of the image sensor, the pixels located at the image from left to right are inputted serially for feature calculation according to Eq.2.2 and Eq.2.3 on a basic unit of 4×4-pixel sub-cell.

The responses calculation of first sub-cell for the horizontal D_x and vertical D_y results will not complete until the fourth-pixel p[3w+4] in the fourth-pixel row in a given image width w is inputted.

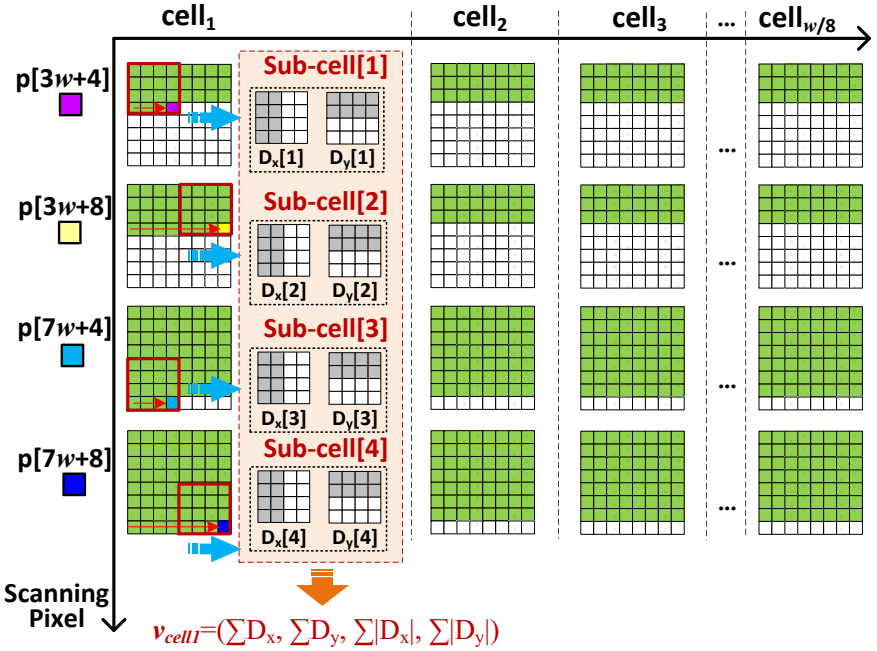


Fig.2. 9 Main generation-process flow of the four-dimensional feature components for the first cell *cell$_1$*.

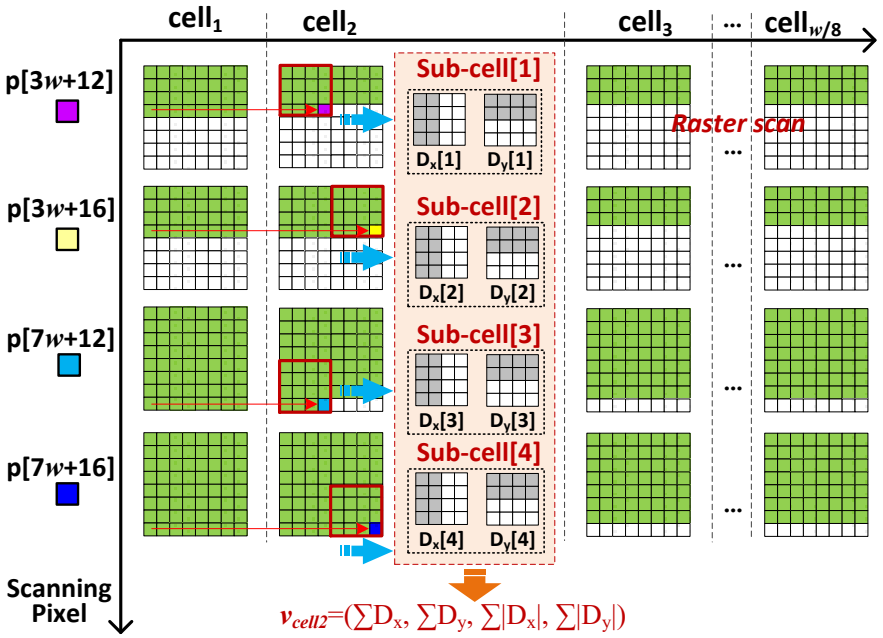


Fig.2. 10 Main generation-process flow of the four-dimensional feature components for the second cell *cell*$_2$.

The responses $D_x[1]$ and $D_y[1]$ of the first sub-cell should be buffered for summation of cell-based feature components until the calculations of the other three sub-cells in the first cell '$cell_1$' complete. As the 3rd -row pixels are kept scanning forwards from left to right, the 8th pixel p[3w+8] in the 3rd row is inputted and the responses $D_x[2]$ and $D_y[2]$ of the second sub-cell will be achieved instantly. Similarly, along with the input of the pixel p[7w+4] and p[7w+8], the calculation of the other two sub-cells is also completed sequentially for accumulation result v_{cell1} of the first cell '$cell_1$'.

Before accomplishment of calculation of sub-cells, all the intermediate values should be temporarily stored. However, once the calculation of the non-overlapped sub-cell is complete, all pixels and intermediate value related to the current sub-cell can be released and reset. The final sub-cell responses, i.e., the horizontal D_x and vertical D_y results, are the only necessary data to be transferred for secondary calculation for cells and windows.

The situation of the second cell '$cell_2$' in the first cell row is similar to the processing procedure of the first cell '$cell_1$'. The sub-cell responses should be computed and outputted sequentially for the following accumulation in the same cell. The generation of local four-dimensional feature components of other cells has similar processing procedure. The pixels shown in green color in Fig.2.9 and Fig.2.10 can reflect the accomplishment status of each cell and sub-cell. The scanned pixels in green color would be computed only once and then abandoned because they turned to be redundant and the values of the sub-cells are the only essential data for next calculation on cells, to refrain from unnecessary storage requirement.

Considering the desired transformation characteristics analyzed above during edge Haar-like feature vector extraction procedure for the local cell, this research proposes an alternative solution of feature vector construction with a pixel-based pipelined architecture, which avoids both image buffers and integral image calculations.

To recognize the target objects among complex backgrounds, a pixel-based pipelined architecture for real-time feature extraction on the local cell, as illustrated in Fig. 2.11, is proposed for efficient cell-based feature representation.

Other than processing one pixel per cycle in a pipeline for the integral image as in previous works like [4], the serial-input pixels from the image sensor are used immediately to proceed with D_x and D_y calculation of the corresponding sub-cells by the 'sub-cell calculator' according to Eq.2.2 and Eq.2.3, and then to sum the four sub-cells up as cell-feature vectors v_{cell} according to Eq.2.6.

Intermediate sub-cell responses are temporarily saved in the first storage while the cell results are then buffered in the second storage as shown in Fig. 2.11, respectively. The first storage is actually divided into two parts for computing D_x and D_y responses of each sub-cell respectively.

The calculation of D_x and D_y sub-cell responses are operated at the same time, but the generation of intermediate D_x and D_y sub-cell results are asynchronous due to the different regional divisions as shown in Fig.2.4.

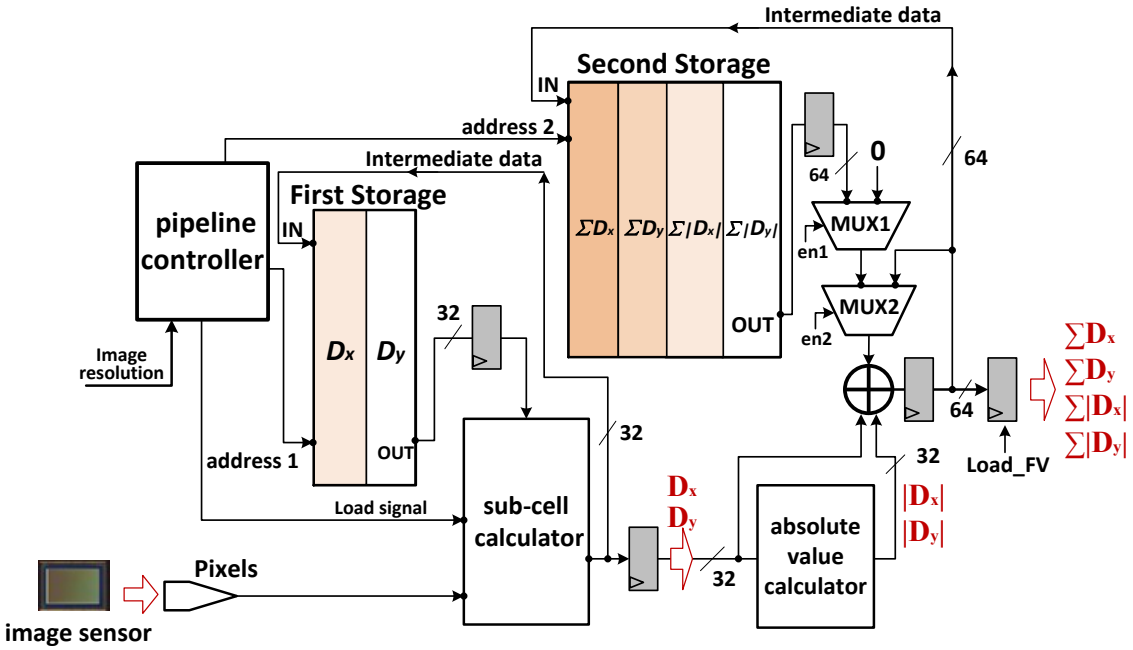


Fig.2. 11 Integrated hardware architecture for extracting the local four-dimensional cell-feature vectors $v_{cell}=\{\sum D_x, \sum D_y, \sum|D_x|, \sum|D_y|\}$ with the simplified-SURF descriptor.

To enable the pixel-based pipelined-adding operation with an immediate accumulation of pixel values after transfer from the image sensor, the dual-port memory is employed for the implementations of both the first storage and the second storage shown in Fig. 2.11. Accesses conflicts between reading and writing are also avoided by a fixed clock-cycle delay when accessing to the same address, which in particular guarantees continuity of the whole pipeline processing. In particular, the two storages are applied for buffering the intermediate data under the control of the pipeline controller.

The pipeline controller is mainly composed of a counter group, logic gates, and registers, to manage the writing and reading addresses for the two dual-port memories, as well as selection signals for multiplexers and load signals for the registers, etc. Specifically, variable counters can be employed for flexible regulation about image size to be processed, which is considered important for reconfigurable designs and will be expanded and discussed in the following contents.

The cascaded multiplexers (i.e. MUX1 and MUX2) cooperated with the second storage and the adder shown in Fig. 2.11 are designed to aid for summing up the adjacent sub-cells that are assigned to the same cell.

There are two iterative structures in the sub-cell calculator, to compute sub-cell responses according to Eq.2.2 and Eq.2.3 respectively, which will be discussed in detail in the next sections.

Consequently, the processing speed only relies on the pixel-transfer frequency from the image sensor by the proposed pixel-based pipeline cell-feature extraction architecture, to implement the calculation in hardware for the local four-dimensional cell-feature vector v_{cell}.

2.3.2 Pixel-based pipelined circuitry for sub-cell responses

2.3.2.1. Horizontal response D_x

In the case of computing D_x responses, adjacent pixels that are assigned to the same part (left or right) of the same sub-cell, are transfer to the sub-cell calculator by another two cascaded multiplexers. Two methods can be employed for calculating the horizontal sub-cell response D_x in this research with different arithmetic priorities and storage policies. The first strategy for computing horizontal response D_x is to separate the left part and right part of each sub-cell exactly the same as Fig. 2.4, accumulate pixels according to Eq.2.2 and store the intermediate results of the left part and right part independently. Another strategy for generating the D_x responses is to transform addition and subtraction instantly according to the pixel position inside the sub-cell so that the

final horizontal sub-cell response D_x can be obtained by a combined computation on all pixels inside a same sub-cell. Both of these two methods own their respective advantages and disadvantages.

Figure 2.12 illustrates the circuitry and its operating principle of the first method mentioned above. The pixels on the image are scanned in raster manner and inputted serially to the D_x calculator, which is a part (another part is the D_y calculator) of the sub-cell calculator as shown in Fig. 2.11. Intermediate partial-addition results of left- and right-part pixels of each sub-cell are temporarily stored in the first storage (D_x part) which employs a dual-port memory. Given an image with w width, $w/2$ words of the first storage must be consumed because the left part and right part of each sub-cell are stored independently and occupy one word respectively.

To accumulate the pixels inputted in raster manner immediately, each temporarily stored value in the first storage is read out successively for accumulation with the adder, shown in the more detailed representation of the *D_x calculator* in Fig.2.12. The serial pixel stream from the image sensor is sent to the one port (‘a’ port) of the adder located in the Fig.2.12, adds up with the current value (e.g. the ‘0’ transferred from two multiplexers MUX3 and MUX4 for initialization) from the other port (‘b’ port) of the adder and generate an instant sum value simultaneously.

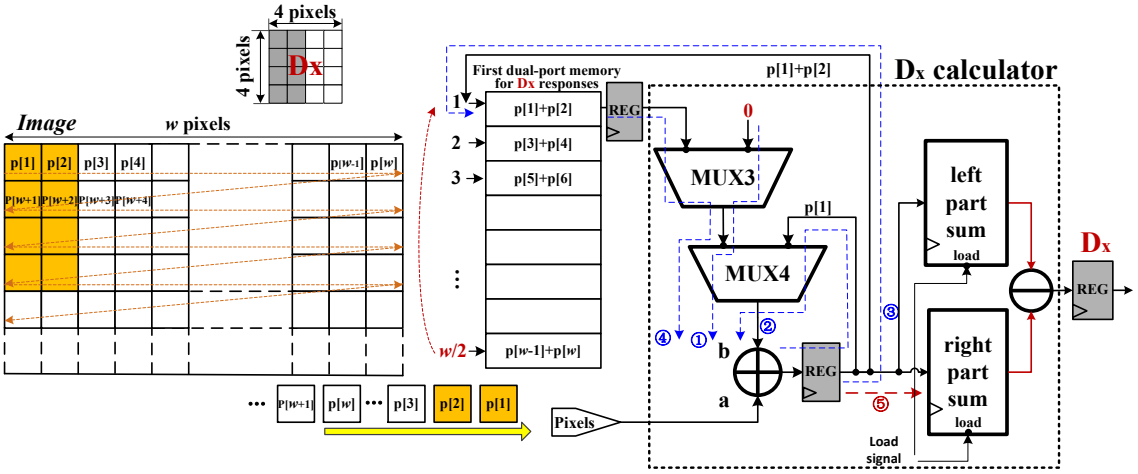


Fig.2. 12 Proposed circuitry with independent left- and right-part summation for generating horizontal sub-cell responses D_x according to Eq.2.2.

The blue lines shown in the right circuitry in Fig.2.12 illustrate the pixel processing procedure when accumulating a horizontal response D_x for one sub-cell.

Take the first sub-cell as an example for timing analysis. The first two pixels, i.e., p[1] and p[2], in the first pixel row in the image belong to the left part of the first sub-cell

response $D_x[1]$, thus the first pixel p[1] will plus the second pixel p[2] by the adder in the D_x calculator and then the summing result (i.e., p[1]+ p[2]) will be sent to the memory (i.e., the 'First dual-port memory for D_x responses' in the circuitry in Fig.2.12) after some certain delays for temporary storage as illustrated in Fig.2.13. The intermediate sum 'p[1]+ p[2]' will be read out from the memory to the 'b' port of adder through the cascaded multiplexers MUX3 and MUX4 when the first pixel p[w+1] in the second row is inputted. As both the pixels p[w+1] and p[w+2] belong to the left part of the first sub-cell, they will be added up and the intermediate sum to be stored in the memory will be updated as 'p[1]+ p[2]+p[w+1]+p[w+2]'.

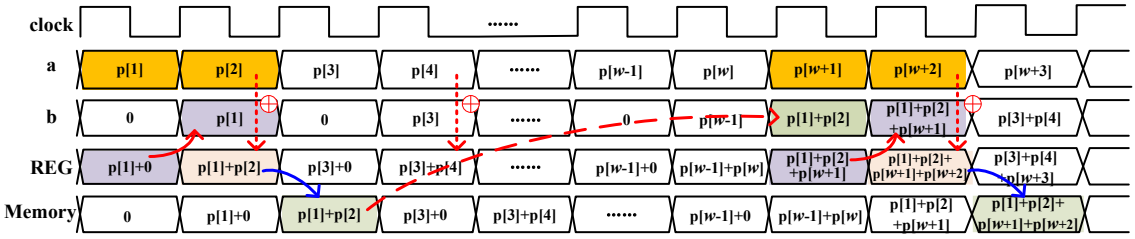


Fig.2. 13 Timing analysis with part of data flow for the first sub-cell using independent left- and right-part summation, to generate horizontal sub-cell responses D_x.

The processing procedure of the right part of the first sub-cell is similar to the left part's accumulation until the first response $D_x[1]$ is generated. We can find the first two pixels p[3] and p[4] in the right part of the first sub-cell are added up and then the sum 'p[3]+ p[4]' is revoked for accumulation when the pixel p[w+3] is inputted as illustrated in Fig.2.13. Afterwards, the updated sums are sent back to the corresponding storage unit.

This accumulative behavior of the proposed circuitry will continue until the first four pixel rows are scanned. The separate results of left- and right- part of each sub-cell are generated in sequence when processing the fourth-pixel row. Final left- and right-part summation results are transferred to separate registers for subtraction when becoming available, to calculate the sub-cell's D_x according to Eq.2.2. The D_x processing for a sub-cell row is completed by a subtractor with the sequential loading of accumulation results for left and right parts of each sub-cell into registers. In other words, the operations of addition and subtraction are supported by two independent devices in this circuitry.

It costs simpler control for frequent arithmetical transitions, i.e., the exchange between addition and subtraction for D_x responses in this circuitry shown in Fig.2.12. However, this method occupies more memory space and much more memory access for calculation, which results in more memory requirement and more power consumption.

Another circuitry for generating the D_x responses is proposed for saving memory

space in this research as illustrated in Fig.2.14. Every four pixels belong to a same sub-cell that all the related pixels in one sub-cell are calculated in each pixel row. However, this calculation involves frequent exchanges between addition and subtraction for computing horizontal D_x responses but less storage space is consumed in this strategy.

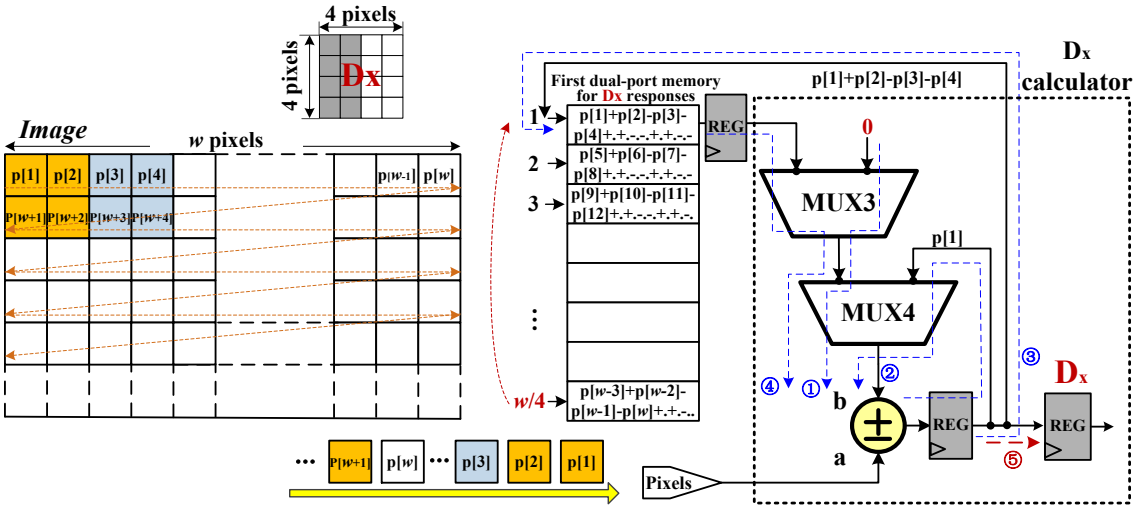


Fig.2. 14 Alternatively proposed circuitry with the arithmetical transition of addition and subtraction for generating horizontal sub-cell responses D_x according to Eq.2.2.

Also, take the first sub-cell as an example for timing analysis of circuitry illustrated in Fig.2.14 and Fig.2.15. The first four pixels in the first-pixel row belong to the first sub-cell although they also locate at different parts of the sub-cell. The addition is operated for the former two pixels p[1] and p[2] while subtraction is taken for the latter two pixels p[3] and p[4].

The intermediate computing result 'p[1]+p[2]-p[3]-p[4]' is temporarily stored in the memory and invoked for accumulation when the first four pixels are inputted in the next pixel row. Thus, each sub-cell consumes only one word in the memory so that one haft of storage space can be saved compared to the first strategy shown in the Fig.2.12.

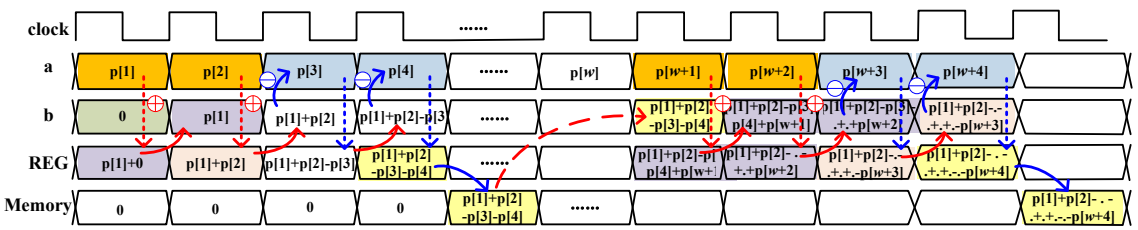


Fig.2. 15 Timing analysis with part of data flow for the first sub-cell using the arithmetical transition of addition and subtraction to generate horizontal sub-cell responses D_x.

The reduction of storage resource is worthy of more complex controls for embedded applications or integrated circuit designs because the additional operation transformations between additions and subtractions are negligible on the basis of the high processing speed of hardware circuitry. The less memory requirement generally results in the smaller occupied area and less power consumption, both of which are far more important and more critical factors for the integrated circuits.

2.3.2.2. Vertical responses D_y

The operating principle of D_y calculator in Fig.2.16 to create the sub-cell response D_y is similar to the D_x calculator in the second strategy. The D_y calculator also uses an adder/subtractor circuit for obtaining the D_y values.

In contrast to the architecture for calculating the D_x data, the hardware architecture for D_y calculation successively adds up the four adjacent pixels assigned to a same 4×4-pixel sub-cell in the two upper pixel rows according to Eq.2.3. It is not necessary to store the upper-half and lower-half of each sub-cell separately in the second dual-port memory, because the adder-subtractor sums up the pixels in the upper-half rectangle of each sub-cell and then subtracts the pixels in the lower-half rectangle of this sub-cell, as illustrated in Fig. 2.17.

Similar to the D_x implementation, another dual-port memory as the first-storage implementation for D_y responses to enable the pixel-based pipelined-adding operation with an immediate accumulation of pixel values after transfer from the image sensor.

Conflicts between reading and writing accesses are avoided by the fixed clock-cycle delay between accesses to the same address, which in particular guarantees continuity of the whole pipeline processing.

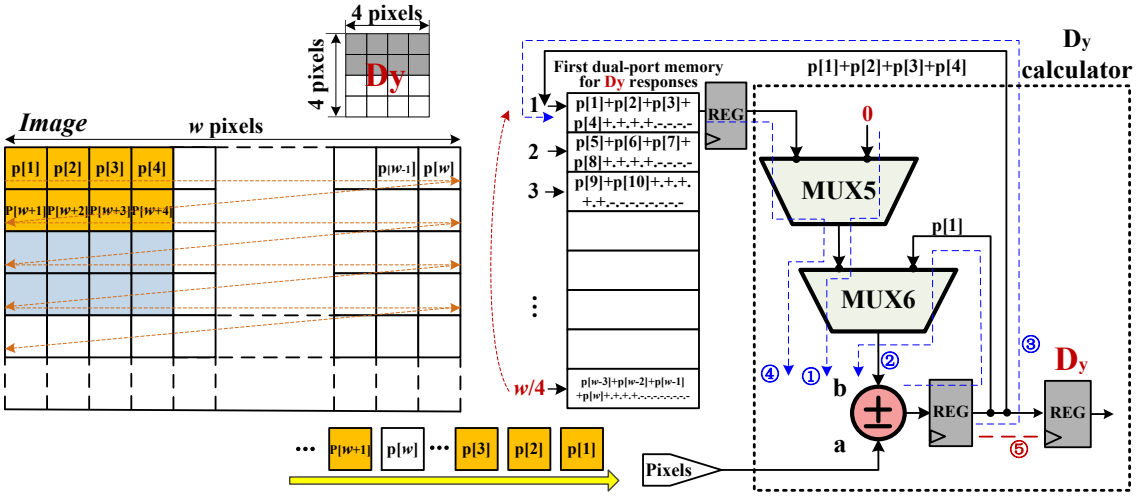


Fig.2. 16 Proposed circuitry with the arithmetical transition of addition and subtraction for generating vertical sub-cell responses D_y according to Eq.2.3.

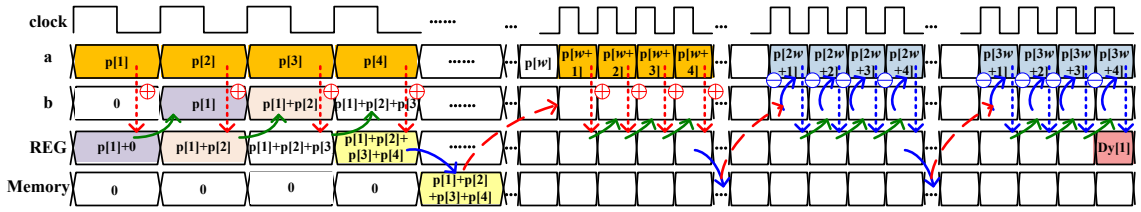


Fig.2. 17 Timing analysis with part of data flow for the first sub-cell using the arithmetical transition of addition and subtraction to generate vertical sub-cell responses $D_y[1]$.

The sequential four pixels p[1], p[2], p[3], p[4] are transferred to the adder-subtractor and accumulated with the help of the cascaded multiplexers (i.e. MUX5 and MUX6) in the D_y calculator.

The accumulated result 'p[1]+p[2]+p[3]+p[4]' will be stored in a certain-width word in the memory, read out to sum up the responding four pixels in the next row, and feedback the updated accumulated result 'p[1]+p[2]+p[3]+p[4]+p[w+1]+p[w+2] +p[w+3]+p[w+4]' to the former word in the memory. Then the adder-subtractor circuit reads out the sum of the two upper rows of each sub-cell from memory and sequentially subtracts the four adjacent pixel values of each sub-cell in each of the two lower pixel rows as they arrive from the image sensor. The same operations are performed for the remaining w/4-1 sub-cells of the sub-cell row until the pixel p[4w-1] is inputted from the image sensor.

Consequently, only w/4 words in the first storage would be sufficient for intermediate sub-cell summation results of both D_x and D_y processing of a given image width w. In other words, the intermediate result of each sub-cell only occupies one word in the memory.

The two multiplexers (i.e., MUX3 and MUX4 in the D_x calculator, MUX5 and MUX6 in the D_y calculator) form an iterative structure to initialize the calculation for each sub-cell and to execute a recursive data loop for the two dual-port memories. After the completion of calculations in one sub-cell row, the storage space of the two dual-port memories can be initialized and reused for the D_x and D_y calculation of the next sub-cell row. Then processing changes to the second sub-cell row in a similar manner and when the pixel p[7w+7] is inputted, the D_x and D_y values of the fourth sub-cell of $cell_1$ can be completed. The word precision for both the first storage is 16-bit, which is sufficient for providing accurate accumulation of eight 8-bit gray pixels in a half rectangle of each sub-cell.

It is well to be reminded that the processing of input images is in principle unlimited height h in this research, which significantly enhances the utilization ratio of on-chip

hardware resources and increases the area efficiency. The unlimited height *h* is enabled on the basis of flexible regulation with variable counters and logic gates, which will be illustrated in the following section 2.3.4.

2.3.3 Accumulation for local cell-based feature-vector construction

Upon completion, D_x and D_y results are sent to the iterative structure for the immediate accumulation of $\sum D_x$, $\sum|D_x|$, $\sum D_y$ and $\sum|D_y|$ of the corresponding cell as illustrated in Fig.2.11. In other words, once the last relevant pixel of a sub-cell has been processed, the D_x or D_y result will be transferred instantaneously for the accumulation of the local feature vector v_{cell} of the current cell, according to Eq.2.6, with a fixed size of 2×2 sub-cells.

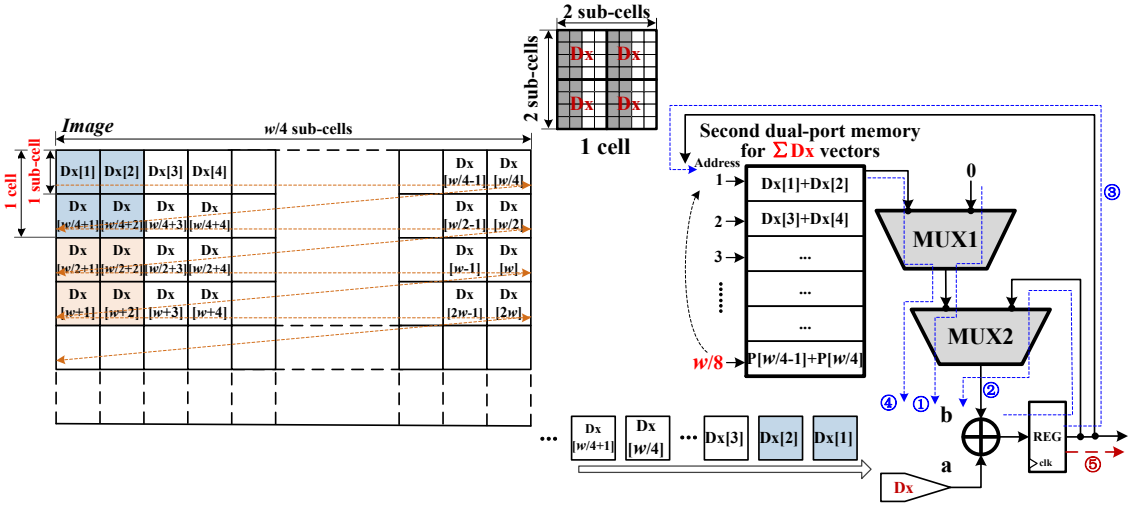


Fig.2. 18 Proposed circuitry for accumulating the horizontal D_x responses in one cell.

The architecture and operating procedure for cell-based feature vector are similar to the above schemes for D_x or D_y. Figure 2.18 illustrates the procedure for accumulating D_x responses in one cell on behalf of the cell-based feature vector accumulation. The cell-based components of other sub-cell responses D_y, $|D_x|$ and $|D_y|$ can be achieved by the almost same architecture as Fig.2.18.

Once the D_x and D_y results are generated by circuitries respectively introduced in section 2.3.2, they will be transferred to the right part of circuitry in Fig.2.18 immediately. During the input of the last pixel line of one cell row, i.e., p[7*w*+i], and i∈[0,*w*-1], a local four-dimensional cell-feature vector is completed every 8 clock cycles. Every two adjacent sub-cells in a sub-cell row are grouped into one non-overlapping cell. This subsequently enables the completion of the local four-dimensional cell-feature vector v_{cell}

($\sum D_x$, $\sum D_y$, $\sum|D_x|$, $\sum|D_y|$) of cell$_1$ as illustrated in Fig.2.19. The four-dimensional vectors, consisting of components $\sum D_x$, $\sum|D_x|$, $\sum D_y$ and $\sum|D_y|$, are eventually outputted for parallel window-level Haar-like feature vector construction and partial object recognition at the same time.

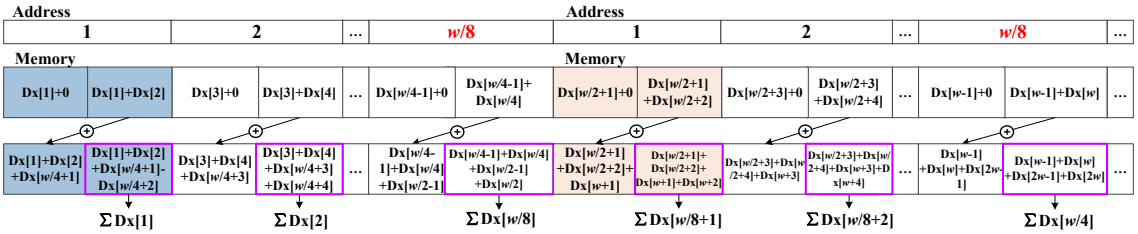


Fig.2. 19 Part of the data flow for accumulating the sub-cell responses D_x inside one cell.

Intermediate calculation results of the sequential sub-cells are saved in the second storage in Fig.2.18, which also apply dual-port memory for preventing from accessing conflicts. Given an image with width w, w/8 intermediate cell-summation results for one cell row have to be stored. The storage space of the second dual-port memory can also be overwritten afterwards during processing of the next cell row, once the processing for the previous cell row is completed. Specifically, only w/8=128 words for temporary summations of the two 16-bit horizontal cell-based dimensions (i.e., $\sum D_x$ and $\sum|D_x|$) and the two 16-bit vertical cell-based dimensions (i.e., $\sum D_y$ and $\sum|D_y|$) have to be stored in the second storage, respectively.

In conclusion, the proposed pixel-based pipeline cell-feature extraction architecture synchronizes with the image sensor, image buffering and integral image calculation become completely unnecessary.

2.3.4 Flexible regulation for unlimited image height

There is a pipeline controller shown in the overall architecture in Fig.2.11, which mainly manages to written and read addresses for the two dual-port memories, as well as selection signals for the six multiplexers and load signals for the registers.

Corresponding to the two types of circuitry for computing and accumulating the horizontal D_x responses as mentioned above, there are also two types of control circuits implemented in the pipeline controller.

The first type of implemented circuitry for pipelined processing control is shown in detail in Fig.2.20, which is composed of three 1-bit counters, two 2-bit counters, 14 logic gates and three variable counters. The left part of the controlling circuit in Fig. 2.20 can

be utilized for controlling the computational circuitry in Fig.2.12, while the right part of the circuit in Fig. 2.20 is applied for managing the accumulation inside one cell for the circuitry as illustrated in Fig.2.18.

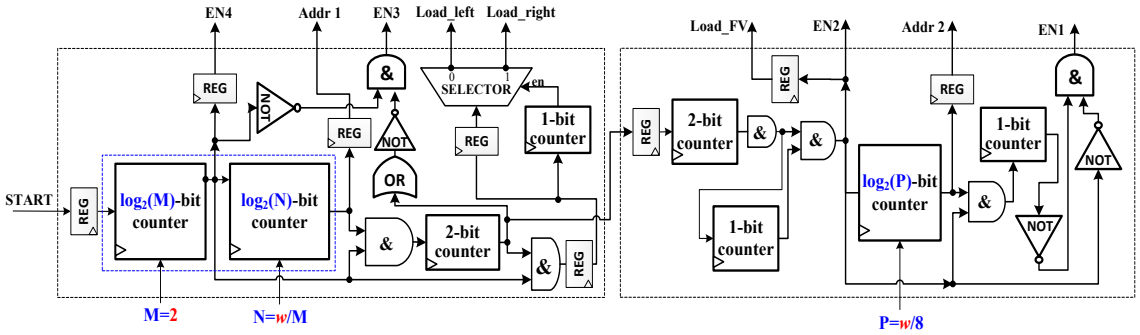


Fig.2. 20 Implemented circuitry for processing control with unlimited image height to generate cell-based feature vectors.

Specifically, variable $\lceil \log_2(M) \rceil$-bit and $\lceil \log_2(N) \rceil$-bit counters control the memory usage of the first storage during D_x calculation for one sub-cell row in images of width w. For computing D_x responses with the independent calculation for left- and right- part of a sub-cell, the parameters should be set as M=2, N= w/M. As the output signals of these variable counters are directly created under the inputted image width w, they can change the final output enable signals, such as 'EN1' for the multiplexer MUX1 and 'EN2' for the multiplexer MUX2 in Fig.2.18. The read/write addresses of the second storage for the accumulation of the 4 components of each local cell-feature vector are assigned by a $\lceil \log_2(P) \rceil$-bit counter, where P= w/8. The generated address 'Addr1' is applied for written and read address for the first storage (dual-port memory) while the address 'Addr2' is applied for written and read address for the second storage shown in the overall architecture in Fig.2.11.

Since the left- and right- parts of one sub-cell are separately computed, the computed results are temporarily stored in two registers respectively, so that two divisive enable signals 'Load_left' and 'Load_right' should be created for loading correct data to the registers. The final cell-based feature vectors of each cell will be latched into a register by the enable signal of 'Load_FV'. In conclusion, the counter group would dimension the desired memory space and processing clock circles with responding to the image width w.

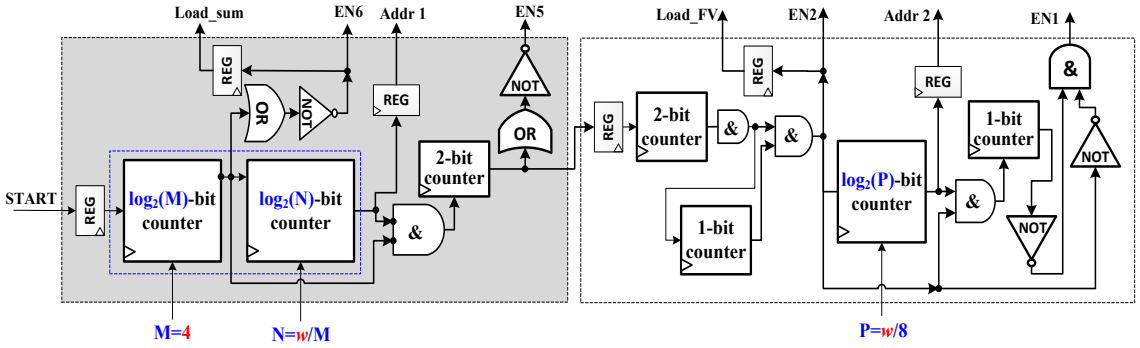


Fig.2. 21 Alternatively implemented circuitry for processing control with unlimited image height to generate cell-based feature vectors.

Comparing to the implementation shown in Fig.2.20, the left part of the circuitry in Fig.2.21 is much simpler. This scheme can be applied for controlling the computation of both horizontal D_x results and vertical D_y results of a 4×4-pixel sub-cell. Thus the variable input parameter M is equal to 4 and N is still equal to *w*/M. For instance, the output enables signals 'EN5' and 'EN6' can be applied for selecting control for multiplexers MUX5 and MUX6 respectively when computing the vertical D_y results inside one cell. Another enable signal 'Load_sum' is used for loading the final summation of one sub-cell to a register. The right part of this circuitry is the same as implementation shown in Fig.2.20. Under the control of above two types of circuitries, the feature extraction architecture shown in Fig.2.11 can be applied for flexible image resolutions with unlimited height *h*.

TABLE II. I
Necessary Bit Width of Counters and Memory Requirements as a function of Image Width

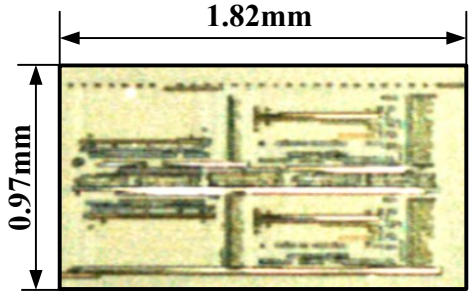
Image Width (*w pixel*)	Bit Width $[log_2(w/2)]$-bit counter	Bit Width $[log_2(w/4)]$-bit counter	Bit Width $[log_2(w/8)]$-bit counter	Memory (*words*)
256	7 bits	6 bits	5 bits	320
512	8 bits	7 bits	6 bits	640
640	9 bits	8 bits	7 bits	1280
1024	9 bits	8 bits	7 bits	1280
1920	10 bits	9 bits	8 bits	2560
2048	10 bits	9 bits	8 bits	2560

Table II.I summarizes the total number of the required 16-bit memory words and the variable-counter width for local cell-feature-vector calculation in relation to the image width *w*. Consequently, 9-, 8- and 7-bit counters are sufficient to create up to 512, 256 and 128 read/write addresses for the D_x part and D_y part in the first storage and the second storage, respectively, to support 1024 pixels of the image width.

2.4 Experimental VLSI-implementation result and discussion

A VLSI prototype of the Haar-like feature-extraction coprocessor was implemented in Rohm 180nm CMOS technology with 1.76 mm^2 (1.82 mm×0.97 mm) core area, as shown in the photomicrograph of Fig. 2.22.

Chip operation at up to 120 MHz is verified for VGA(640×480 pixels)-size video frames with a short processing time of 3.072 ms (i.e., 325 fps frame rate), the low power dissipation of 43.45 mW working at the 1.8V supply voltage. The lower working frequency leads to lower power dissipation according to different speed requirements of various applications. Figure 2.23 demonstrates the power consumptions over a range of supply voltages and operating frequencies.



Core area	**1.76 mm^2**
Supply voltage	**1.8 V**
Frequency	**120 MHz**
Power consumption	**43.45 mW**
Search time (VGA)	**3.072 ms**

Fig.2. 22 Microphotograph and parameters of the fabricated chip in 180 nm CMOS technology for cell-based Simplified-SURF feature extraction with 1.76 mm^2 (1.82 mm×0.97 mm) core area.

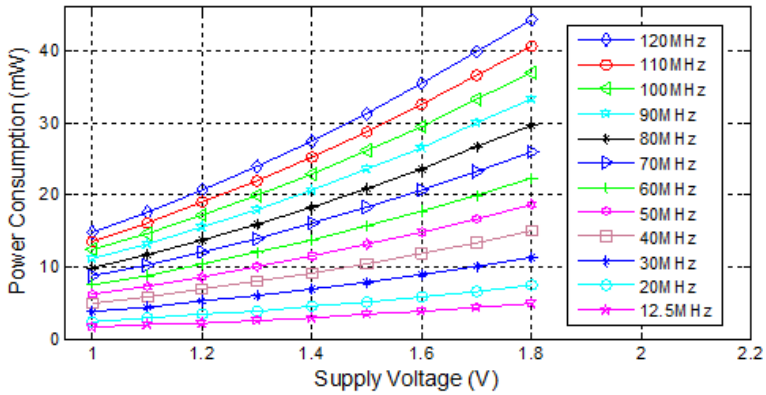


Fig.2. 23 Power consumption of the fabricated coprocessor at different working frequencies and different supply voltages.

The power consumption of this coprocessor is 4.78 mW at 1.8 V supply voltage and 12.5 MHz frequency, which is sufficient for processing 30 fps VGA images from general image sensors. The more performance of the proposed pipelined hardware scheme for

local cell-based simplified-SURF descriptor applied Haar-like feature vector is summarized in Table II.II.

TABLE II. II
Performance of the Proposed Hardware Architecture for the Simplified-SURF Descriptor when applied to Haar-like Feature Vectors

Technology	180 nm SOTB CMOS
Die area	1.76 mm^2
Design target	Feature extraction
Extraction scope	Entire frame
Image resolution	1024×∞ pixels
Feature type	Haar-like hardware
Frequency	120 MHz
Core/IO voltage	1.8 V/ 3.3 V
Power	43.45 mW
Maximum frame rate	325 fps (VGA)
Memory usage	12 kB

In our ASIC realization, a storage space of 1024 16-bit words is implemented for both D_x and D_y calculation of the sub-cells. On the other hand, 512 words of 64-bit precision are implemented in the second storage of our fabricated prototype coprocessor and are assigned for summation of the cell-feature vector components of one cell row, meaning that 32 kb of dual-port memory is consumed for completing the local four-dimensional cell-feature vectors. Consequently, only 64 kb of storage space has to be assigned to the overall circuitry for local cell-feature vector extraction to enable the processing of images with up to 1024-pixel width and unlimited height. Another 32 kb of SRAM (Static Random-Access Memory) space is served as FIFO (First In First Out) for buffering cell-based feature vectors, which are further used for high-dimensional window-based feature construction for recognizing target objects. In conclusion, only a small memory size of only 96 kb (i.e., 12 kB) is sufficient for realizing the flexible image-size processing (up to 1024 pixels × ∞ pixels) with our hardware architecture.

2.5 Conclusion

A pixel-based pipelined hardware architecture for extracting a simplified SURF descriptor, which relies on Haar-like feature vectors of image cells, is developed in this research. The architecture immediately processes every input pixel without pre-storage of raw images and reuses memory space for local cell-based feature-vector calculation of different cell rows in the processed image, making it highly flexible for multi-size image processing with low memory requirements. Due to the flexibility in image resolution, the

prototype chip can deal with a maximum width w of 1024 pixels and an unlimited height h. Specifically, a sub-cell with 4×4 pixels and a cell with fixed 2×2 sub-cells are defined in this section to capture sufficient details of the target object. Since the proposed scheme shortens the critical path and the searching time of pixel positions, the redundant computing cost is greatly reduced. A VLSI prototype for cell-based feature extraction of all sliding windows across the image frame was implemented in 180 nm CMOS technology, which is verified with the achievement of very fast real-time processing speed and high energy efficiency.

References

[1] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proc. Computer vision–ECCV2006*, Graz, Austria, 2006, pp. 404-417.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol.110, no.3, pp. 346-359, Jun. 2008.

[3] A. Haar, Math. Ann, "Zur theorie der orthogonalen funktionensysteme," *Mathematische Annalen*, vol.69, no.3, pp.331-371, 1910 [in German].

[4] D. Jeon, M. B. Henry, Y. Kim, I. Lee, Z. Zhang, D. Blaauw, and D. Sylvester, "An energy efficient full-frame feature extraction accelerator with shift-latch FIFO in 28 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol.49, no.5, pp.1271-1284, 2014.

[5] R. Lienhart, and J. Maydt, "An extended set of haar-like features for rapid object detection," In *Proc. Image Processing ICIP*, vol. 1, pp. 900-903, 2002.

[6] J. Cho, S. Mirzaei, J. Oberg, and R. Kastner, "FPGA-based face detection system using haar classifiers," in *Proc. ACM/SIGDA international symposium on Field programmable gate arrays*, pp. 103-112, Feb. 2009.

[7] Q. Chen, N. D. Georganas, and E. M. Petriu, "Real-time vision-based hand gesture recognition using Haar-like features," in *Proc. IMTC.* pp. 1-6, May, 2007.

[8] A. Mohan, C. Papageorgiou, T. Poggio, "Example-based object detection in images by components," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.23, no.4, pp. 349-361, 2001.

[9] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," In *Proc. Computer Vision*, pp. 555-562, Jan. 1998.

Chapter 3
Sliding-Window Approach

3.1 Introduction

In the former two chapters, a cell-based feature extraction architecture and a block-based normalization, applied to the sequential cell-based feature vectors (FVs), have been proposed. In order to recognize a target object, an appropriate region which matches the target object should be represented by the FVs. The image cell (e.g., 8×8 pixels) and block (e.g., 16×16 pixels) are generally too small to completely reflect the essential information for a common object, such as a car (e.g., 128×64 pixels), a pedestrian (e.g., 64×128 pixels), a face (e.g., 64×64 pixels), etc. Thus relatively large groups of square-shaped cells or blocks, which are both important for extracting sufficient pictorial information in detail for feature representation of local regions (i.e., cells and blocks), are generally combined for constructing a high-dimensional window whose size is mapped to the size of the target objects. The feature descriptor with a window-based localization is a crucial factor for accurate object recognition basing on a sliding-window strategy.

The sliding-window strategy has been proved to be competitive in terms of classification performance for a large variety of detection tasks and many state-of-the-art paradigms [1-5]. The designated sizes of the scan window (SW) are depended on location and size of different target objects in the image. The SW feature vector quantifies the similarity of contained objects to reference SWs for the recognition process.

Two schemes are proposed for window-based feature construction in this research. One of the feature representation schemes is a direct decoding from reutilization information of each cell with a new concept called “regular rule of reusing times” (RRRT). The other scheme for window-based feature representation is to arrange the normalized block-based feature components without frequent accesses to the cell-based components.

Specifically, this research applies an overlapped sliding-window approach to the simplified-SURF descriptor for covering the entire image and employs Haar-like FVs for the whole image, instead of extracting FVs only around the interest points as in the previous SURF algorithms [6-7]. The extracted Haar-like responses are used for window-

based FV construction for the proposed simplified-SURF descriptor to recognize the target objects within the SWs.

3.2 Hardware implementation in terms of cell-based feature components

On the basis of feature components on cells, the 2×2-cell block feature vector can be represented by cell-FVs. The feature components of the four cells of each block are arranged in zigzag order to form higher dimensional block-feature vectors for feature descriptor.

On one hand, with the block movement by one cell in raster manner inside a window for constructing the high-dimensional feature vector of the entire scan window (SW), as illustrated in Fig.3.1, the local cell-FV becomes the basic component and each cell can be covered by up to four overlapped blocks.

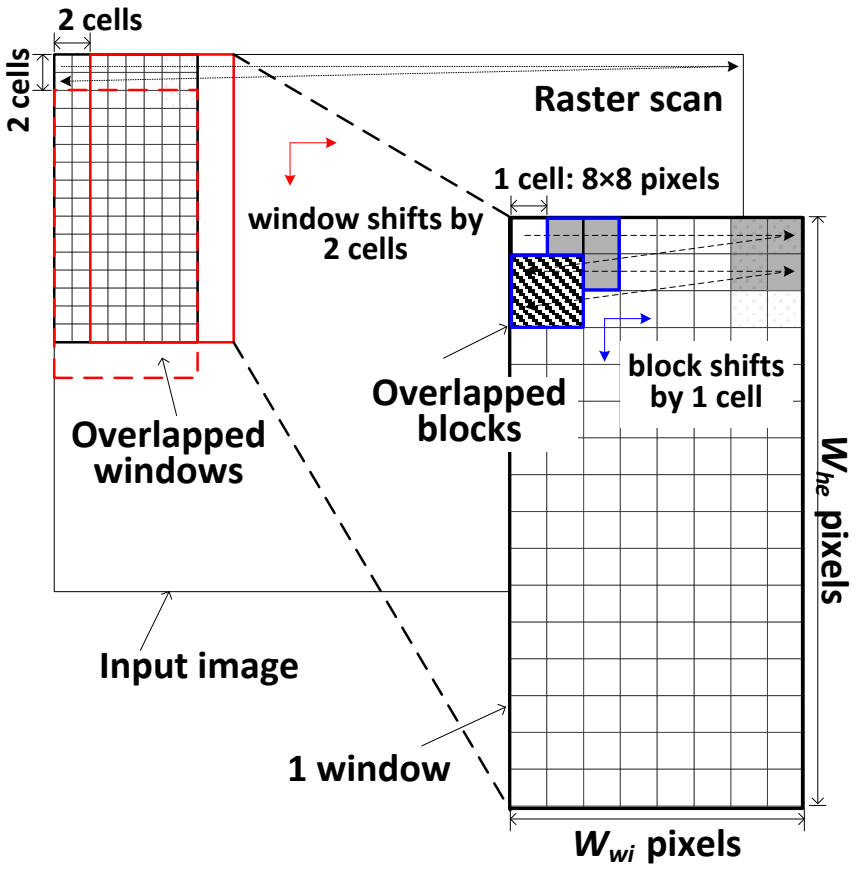


Fig.3. 1 Movement rule of detection windows and blocks.

On the other hand, with the window movement by one non-overlapped block stride (i.e., two-cell stride) in a row-raster-scan manner, the local cell can be overlapped and revoked for window construction in multiple times when sliding the window for capturing all target objects within the complete image. In other words, the local cell-FVs can be reused by both blocks and windows during the feature space construction and the synchronous detection procedure.

In order to apply the local cell-based feature components directly without recalculation for block components, the regular rule of local cells constrained by the inner block movement inside a SW and the window movement inside an image should be determined. So that each cell-based local feature vector is scanned and extracted only once without complex re-computations, and reutilized for all corresponding blocks and SWs.

3.2.1 Regular rule within one window

3.2.1.1. RRRT rule and window FV dimensionality

The image region of a block is represented by 2×2 cells in this research. As an alternative hardware solution to traditional window-sliding strategies, a cell-based perspective for window search and construction is used for confirming the reusing times of each cell located in one SW. The proposed sliding-window algorithm mainly confirms the cell location in each window to which the current cell belongs so that the partial local FV components at the corresponding window position are correctly invoked for feature representation and object detection.

The location of one cell within the SW determines the reused time of the current cell in this window due to the moving manner of the overlapped blocks. As the block slides by one cell within one SW in raster scan manner, it causes reutilization of cells in different overlapped blocks, as illustrated in Fig.3.2, depending on their respective positions within the window. The recycling time or reusing time $\boldsymbol{R}$ of a cell in a SW varies with changes of the block size.

Specifically, shifting of 2×2-cell blocks by one cell in the horizontal and vertical direction on the SW results in three reused cases for cell-FVs. As indicated in right part of Fig.3.2, the cells at corner, edge and internal positions in the SW, which are marked in different colors, are overlapped by 1, 2 and 4 blocks, respectively, illustrating that the corresponding cell-FVs are used by 1, 2 and 4 times for the window-based FV construction. All the sequential cells outputted from the feature extracting circuits such as the simplified-SURF scheme in Fig.2.8 in different windows are abided in the same way.

Consequently, the regular rule of local cells on blocks within a window can be summarized by a regular rule of reusing times (RRRT) and revoked for the feature descriptor construction in every SW by decoding according to the inner cell position within the SW.

By exploiting the cell-overlapping characteristics of blocks, the concept of the block in this research loses its sense of presence in the SW-feature vector construction.

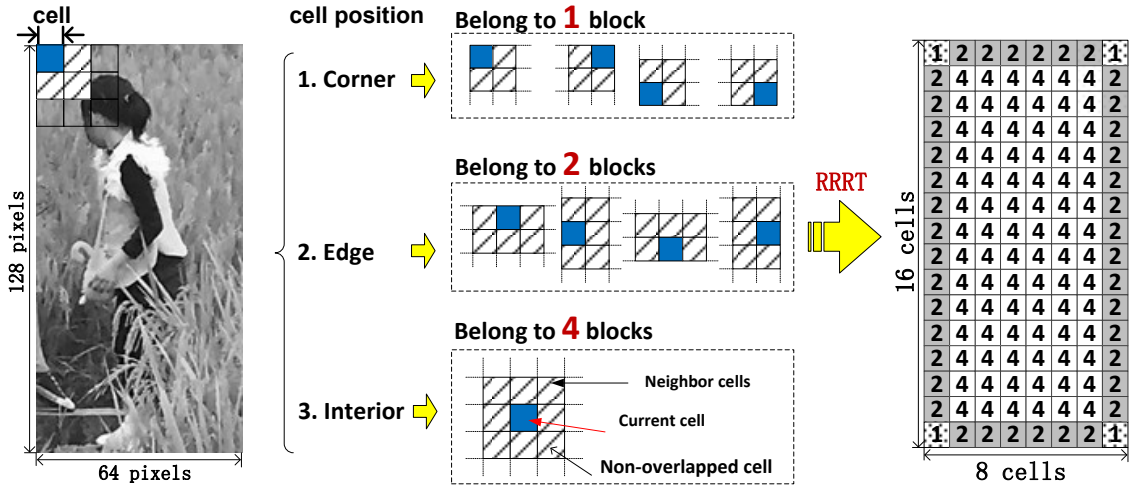


Fig.3. 2 Cell locations in one 64×128-pixel window and its covering times by overlapping blocks. Three kinds of cells, located at corner, edge, and interior of the window are reused for window-based FV construction in 1, 2 or 4 overlapping blocks, respectively.

In other words, the whole feature extraction procedure of the SW is based on the local cell-FVs and their corresponding reused times RRRT which are depended on the cell position within each SW.

Consequently, all cells of one SW are scanned only once respectively and then construct the window-level feature descriptor vector together with dimensionality d_{SW} according to Eq.3.1 as follows:

$$d_{SW} = d_c * \sum_{i=1}^{i=k} R(i) \tag{3.1}$$

where d_c is the dimensionality of the local cell-based FV, $R(i)$ represents the RRRT value corresponding to the i^{th} cell $cell_i$ within the window, and k is on behalf of the total cell number in one SW.

For instance, there are 128 cells in the 64×128-pixel SW and each local 8×8-pixel cell contains d_c=4 local dimensional feature components, as discussed in Chapter 2, for the proposed simplified SURF descriptor, resulting in d_{SW}=1680 dimensions according

to the RRRT rule.

As illustrated in Fig.3.3, The Haar-like responses of all sub-cells, located in the same cell, are added up and form a local cell-based FV. An image cell for the simplified SURF descriptor contains four sub-cells of 4×4 pixels each. The four cells within a block are arranged in raster manner to form 16 dimensional block-FVs basing on 4-dimensional cell-FVs ($\sum D_x$, $\sum|D_x|$ and $\sum D_y$, $\sum|D_y|$) each, from which the SW feature vector is constructed. Thus, 1680 dimensions Haar-like FV can be obtained by arranging the 16-dimension block-FVs from the four cell-FVs in sequence according to the block movement in a SW.

Consequently, the dimensionality d_{SW} of a SW-FV can be accumulated on the basis of the local cell-FVs by decoding the RRRT value according to the corresponding cell position within one SW. The regular rule of RRRT has already embodied the characteristics of blocks by the cell-FV reutilization. This cell-based SW scheme leads to great improvement in computational-cost reduction in comparison to previous designs.

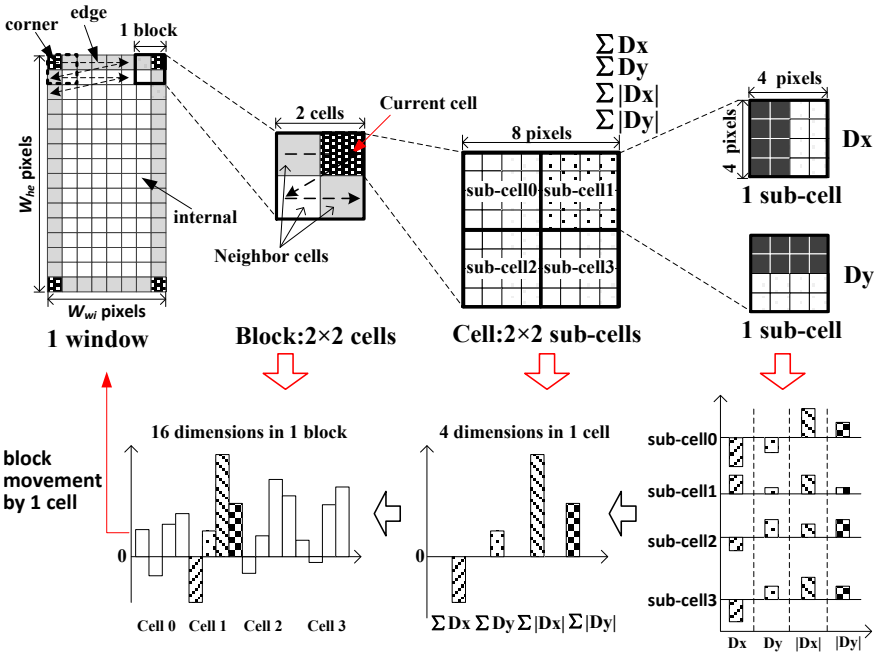


Fig.3. 3 Construction of a simplified SURF descriptor from local Haar-like feature vectors.

3.2.1.2. Regulation of window size

Similar to the effect of cell size and block size, larger window size also leads to a slower processing speed for extracting the window-level FV. The dimensionality d_{SW} of a window-level FV in flexible size can be calculated on the basis of the d_c-dimensional cell-FVs by Eq.3.1 or Eq.3.2 as follows:

$$d_{SW} = (N_{corner} \times 1 + N_{edge} \times 2 + N_{interior} \times 4) \times d_c \quad (3.2)$$

Here, the parameters N_{corner}, N_{edge} and $N_{interior}$ represent the amount of corner-, edge- and interior-located cells in the SW. The values of these three parameters are related to the window size (W_{wi} pixels × W_{he} pixels), which leads to various window-FV dimensionalities depending on the SW size.

The designated size of a SW and its sub-regions are mainly determined by the size of target objects in the image. For example, in [9], a window in size of 64 (width) × 128 (height) pixels is selected for human detection. Different components such as head, legs, and arms are further identified by various sub-regions with 42×42 pixels, 69×46 pixels, or 47×31 pixels, respectively.

Figure 3.4 shows the dimensionality-calculation method for the Haar-like SW-feature vectors applied in the simplified-SURF descriptor of our research.

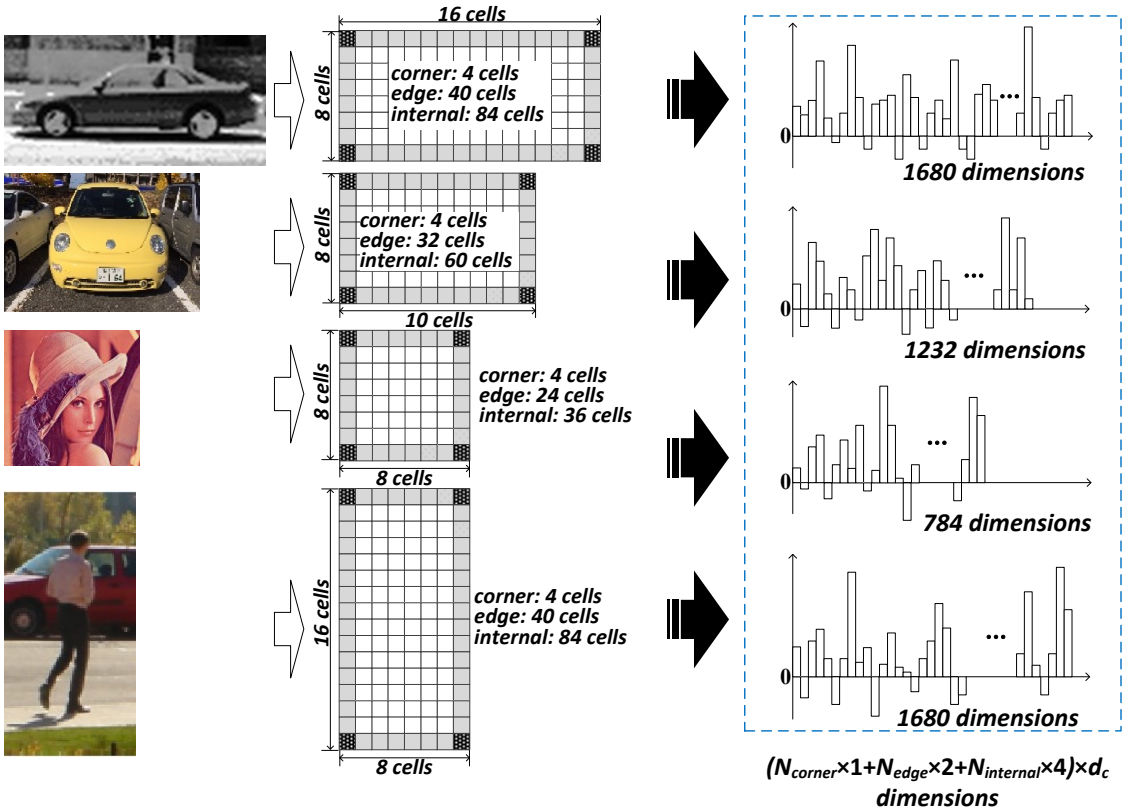


Fig.3. 4 Dimensionality determination of Haar-like SW-feature vectors for different SW-sizes, constructed from the local four-dimensional cell-feature vectors v_{cell}, which serve as the basic components for a holistic Haar-like SW-feature vector of the proposed simplified-SURF descriptor.

The SW sizes are 128×64 pixels, 96×64 pixels, 64×64 pixels and 64×128 pixels, adapted to different target objects (e.g., cars, faces, and pedestrians) of variable scales. Image resolution flexibility with different SW sizes can be applied to match varying types of multi-scale processing.

In other words, the corresponding adaptability to different object types results from the flexibility of the developed hardware architecture with respect to the SW size. Specifically, the raster block movement in a SW of 64×128 pixels or 8×16 cells results in 4 corner cells, 40 edge cells, and 84 interior cells. Therefore, the proposed simplified SURF descriptor, using Haar-like four-dimensional cell-FVs, owns (4×1+40×2+84×4) × 4 =1680 dimensions in case of 64×128-pixel SWs. On the other hand, a dimensionality of (4×1+24×2+36×4) × 4=784 is obtained for a 64×64-pixel SW. The dimensionality for other SW sizes is calculated likewise according to Eq.3.2. The variety and variability of window size can be able to apply for multi-scale image pyramid as well, which is with similar properties for image pyramid as the cell size illustrated in section 4.2.2.

3.2.2 Regular rule within an image

For recognizing the object in a designated region (i.e. SW in this research), the local cell-based feature components, such as the four-dimensional Haar-like feature components ($\sum D_x$, $\sum|D_x|$ and $\sum D_y$, $\sum|D_y|$) in the simplified SURF descriptor, have to be organized for SW-level description. Since SWs are moved in a non-overlapped block stride (i.e. 2×2 cells) for searching the complete image in the raster sliding-window paradigm, most cells are overlapped by multiple SWs.

Since the SW is shifted in block units across the image, the total amount of SWs can be confirmed. Given an entire input image with $w \times h$ pixels, the maximal window number N in horizontal direction and M in the vertical direction can be inferred from Eq.3.3, where W_{wi} and W_{he} are width and height of the designated SW, respectively, and B_s is the square-block size, which is 16 in our case. Thus, based on a 64×128-pixel SW, a VGA (640×480 pixels) image is divided into N × M = 37 × 23 = 851 mutually overlapping SWs, while there are N × M = 61 × 41 = 2501 SWs for XGA (1024 × 768 pixels) images.

$$\begin{cases} N = \frac{w - W_{wi}}{B_s} + 1 \\ M = \frac{h - W_{he}}{B_s} + 1 \end{cases} \tag{3.3}$$

In order to enhance the computational capability of SW-level feature extraction, all

SWs overlapping a given cell are processed in parallel in this work. Usually, a cell is located in multiple SWs due to the shifting by block units across the image. Therefore, the local cell-FV can be reused for multiple SWs. In other words, not only the overlapping blocks as illustrated in Fig.3.2 but also the related overlapping SWs can invoke the four-dimensional cell-feature vectors simultaneously.

To specify the reusing time by overlapping SWs of the current cell (i.e., the cell in blue shown in the input image of Fig.3.5), the number of overlapping SWs must be determined according to the SW size and the cell location in the input image.

For a designated SW size of W_{wi} pixels × W_{he} pixels, a given cell can be located in up to W_{wi}/B_s overlapping SWs in the horizontal direction and W_{he}/B_s overlapping SWs in the vertical direction. In case of SWs with a size of 64×64 pixels (8×8 cells), the number of overlapping SWs in which each cell (block) is located, is illustrated in Fig.3.5. The four cells located in the first block belong all to only one SW (i.e. the first SW), while the 7th cell in the 7th cell row, highlighted with blue color in Fig.3.5, is located in 4×4=16 SWs.

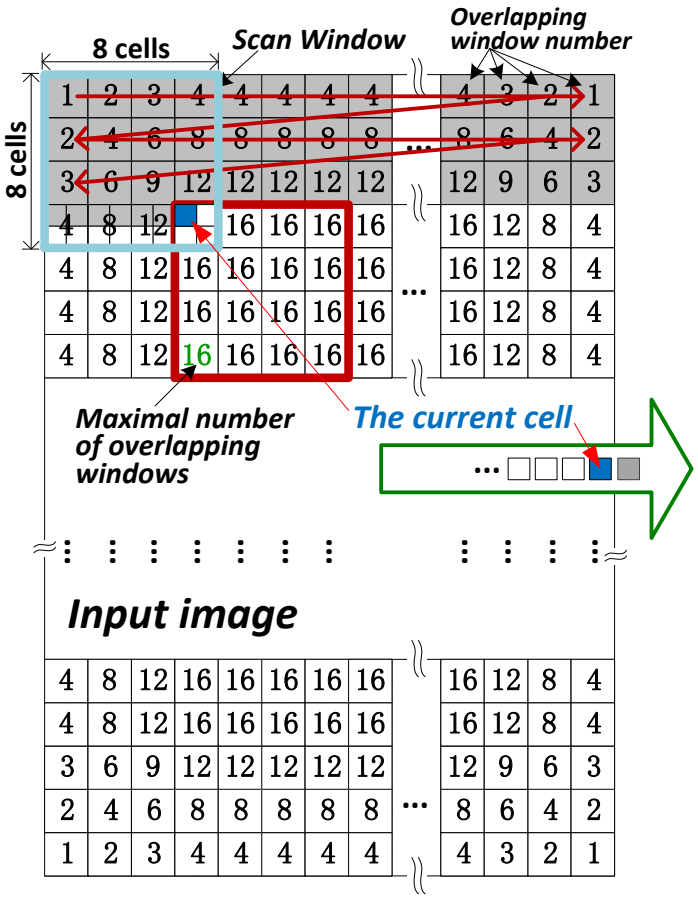


Fig.3. 5 Covering times of each cell (block) by 8×8-cell SWs shifted in block units according to a raster-sliding scheme.

For a SW with a size of 64×64 pixels (8×8 cells), up to 16 SWs can be overlapping on a cell simultaneously if the image is larger than 14×14 cells. All the 16 related windows are illustrated respectively in Fig.3.6. As the current 8×8-pixel cell marked in blue is located in all the 8×8-cell SW (even at different places), the cell-FV is a part of a components of all 16 windows. As the current cell is only scanned and computed for once, all the calculations related to the current cell-FV for each SW should be completed in sequence. The 7th cell in the 7th cell row in the image is locating at the 7th row and the 7th column in the first window W1, but it is also locating at the 7th row and the 5th column in the second window W2, etc. The sequentially inputted cell-FVs will be invoked instantly for all related SWs overlapping on the current cell, although the resulting partial feature components of different windows are available at different accomplishment ratio as shown in Fig.3.7.

Changes in the SW size cause also changes of the maximal number of SWs overlapping a given cell. For instance, in case of 64×128-pixel SWs, up to 32 SWs are overlapping an 8×8-pixel cell, because there are up to W_{wi}/B_s=4 and W_{he}/B_s=8 SWs in the horizontal and vertical direction in a large enough image, respectively.

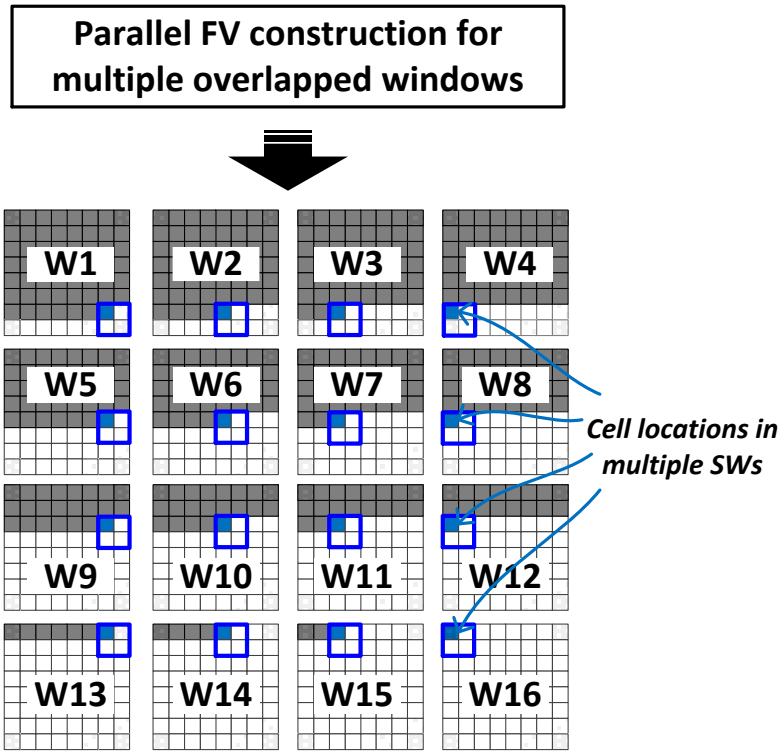


Fig.3. 6 Corresponding positions of the current cell in each related 8×8-cell SW during parallel FV construction.

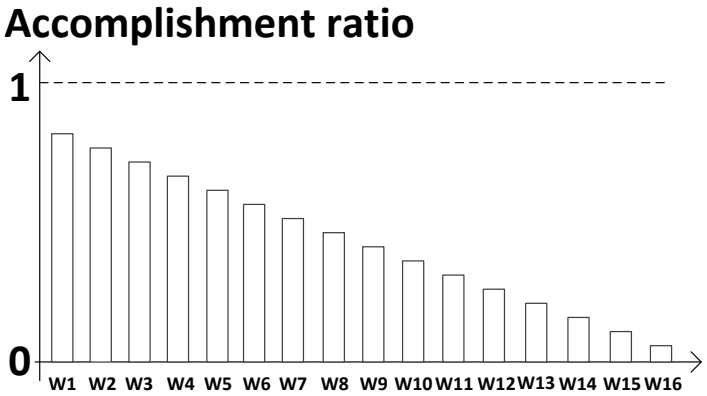


Fig.3. 7 Corresponding accomplishment ratio of the current cell in each related 8×8-cell SW during parallel FV construction.

Thus a parallelized scheme for window-based feature space construction is essential for speeding up the data processing. All cell-FVs in the entire image are extracted only once, then stored temporarily and afterwards reutilized for all related blocks and SWs, to avoid complex re-computation. Thus computational cost and complexity of the feature extraction are significantly reduced. Furthermore, the storage space for one row of SWs should be overwritten for next row of SWs, which results in much less storage requirement than in the previous research.

3.2.3 Implementation of parallel window-feature-vector construction

To compute all related SWs basing on the current cell, a parallel processing scheme has to be proposed. In this research, the window index (WI) or window address (WA) represents the relationships about all related SWs based on the current cell, which is utilized to assign the certain order for organizing the cell-FV components to each corresponding window-level FV.

A cell-based window scan circuitry is proposed for mainly confirming the cell location in each SW that the current cell belongs to, so that the partial local cell-FV components at the corresponding SW position are correctly invoked for feature construction and object recognition. The developed parallel SW (PSW) method, which exploits simultaneous processing of multiple SWs, operates with cell-FVs and takes advantage of the reusing times RRRT of cells in each related SW.

Specifically, each cell-FV from the feature extraction circuitry carries specific information corresponding to its cell position within the multidimensional SW-FV. For

example, the cell C[c,r] represents the cell that locates at c^{th} column and r^{th} row of the image.

From the sliding steps of the sliding-window algorithm and the cell's column (c) and row (r) indices, the initial-window address (IWA) $W_{i(n)}$ can be derived as illustrated in Fig. 3.8, where the index $i(n)$ represents the first SW containing the current cell C[c,r]. On the basis of the parameters c and r indices from one cell, its corresponding IWA can be computed according to the conditions constrained as the flowchart in Fig.3.8. The value of the index $i(n)$ depends on the location of the current cell C[c,r] and the SW size, where $n=c+(r-1)\cdot w/8$ represents the order number of the current 8×8-pixel cell C[c,r] in the input-cell sequences for an image width of w pixels.

Furthermore, the number of SWs, in which the 8×8-pixel cell C[c,r] is located, can be calculated from the cell's position. The multiplication parameters in horizontal and in vertical directions illustrate the number of overlapping scan windows (OSWs) containing C[c,r] in these directions. The total number of OSWs is calculated as $hor \times ver$.

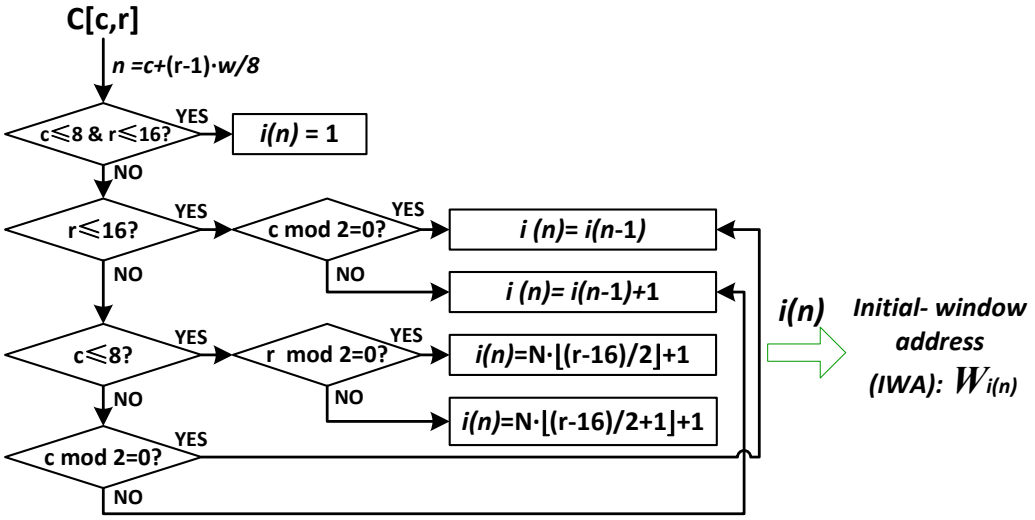


Fig.3. 8 Flowchart for deciding the index $i(n)$ and the IWA (initial-window address) $W_{i(n)}$.

Apart from the four cell-FV components, multiple SWs and cell positions in each SW are associated with each cell.

For each SW in which the cell is located, the cell position corresponds to a cell address within the window. The window number corresponding to different cell positions are initialized in a look-up-table. The numbers of SWs in a $w \times h$-pixel image in horizontal (hor) and vertical (ver) direction are summarized in Table III.I and Table III.II.

TABLE III. I
Window numbers for cell-row index ***c*** in horizontal direction of *w*-width images

c	1~2	3~4	5~6	7~8	9~10	11~12	13~14	15~*w*/8-14	*w*/8-13 ~ *w*/8-12	…	*w*/8-1~*w*/8
hor	1	2	3	4	5	6	7	8	7	…	1

TABLE III. II
Window numbers for cell-column index ***r*** in vertical direction of *h*-height images

r	1~2	3~4	5~6	7~(*h*/8-6)	(*h*/8-5)~(*h*/8-4)	(*h*/8-3)~(*h*/8-2)	(*h*/8-1)~*h*/8
ver	1	2	3	4	3	2	1

Figure 3.9 illustrates the window distribution in an input image with $w \times h$ pixels, operating with 8×8-pixel cell and 2×2-cell block, where Eq.3.3 defines N that represents the window number in one row when window sweeps within a block stride.

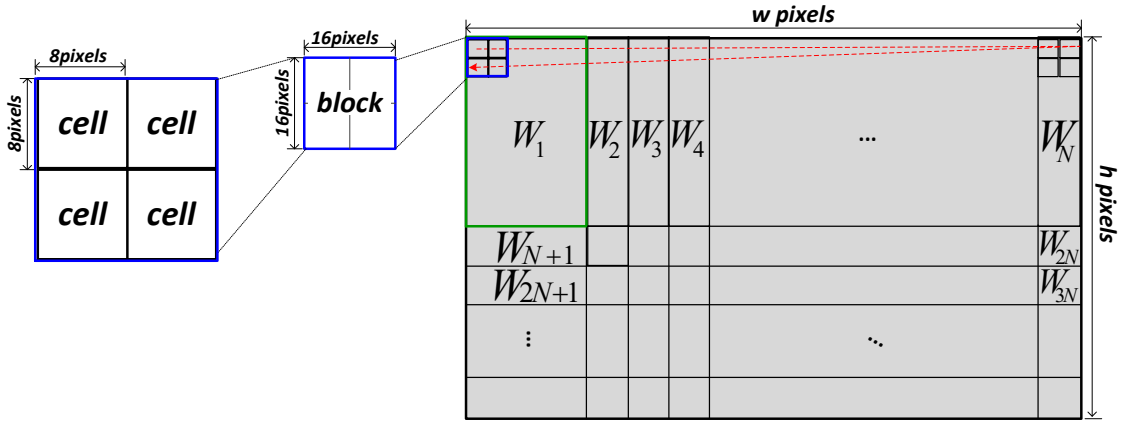


Fig.3. 9 Distribution of window order in an input image with $w \times h$ pixels, operating with 8×8-pixel cells and 2×2-cell blocks.

Based on the IWA index *i(n)*, the window number *j* and *k* in horizontal and vertical directions respectively, which are initialized in a look-up-table, the indices of all OSWs containing C[c,r] can be obtained by Eq.3.4.

$$Index = \{i(n) + \mathrm{j} + \mathrm{N} \cdot \mathrm{k}\}, \mathrm{j} \in [0, \mathrm{hor} - 1], \mathrm{k} \in [0, \mathrm{ver} - 1] \tag{3.4}$$

Figure 3.10 summarizes the design flow for parallel window construction of FV space according to the separate regular rules within a window and an image. Since the RRRT value can be obtained on the basis of the regular rule within a window and the index of each OSW can be computed in accordance with the regular rule within an image, each

OSW can be operated in parallel by applied the RRRT value.

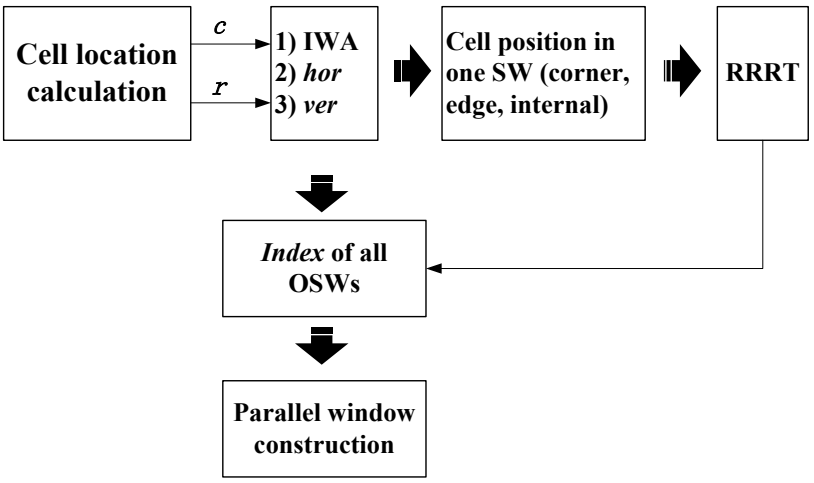


Fig.3. 10 Design flow for parallel window construction basing on the RRRT and the *Index* of all OSWs.

When a 64×128-pixel SW contains the current cell, its responding coefficients a_s, b_s, c_s, or d_s (s∈[0,7]) in Eq.3.5 are '1.' Otherwise, its responding coefficients are '0.' As listed in Table III.I and III.II for the case of a SW with 64×128 pixels, the maximal value of *hor* is '4' while the maximal value of *ver* is '8'. Therefore, one cell can be covered in up to 4×8=32 OSWs.

For example, the cell C[8,3] as in the VGA (640×480 pixels) image is taken as the current cell.

$$\begin{bmatrix} a_0 * W_{i(n)} & b_0 * W_{i(n)+1} & c_0 * W_{i(n)+2} & d_0 * W_{i(n)+3} \\ a_1 * W_{i(n)+N*1} & b_1 * W_{i(n)+N*1+1} & c_1 * W_{i(n)+N*1+2} & d_1 * W_{i(n)+N*1+3} \\ a_2 * W_{i(n)+N*2} & b_2 * W_{i(n)+N*2+1} & c_2 * W_{i(n)+N*2+2} & d_2 * W_{i(n)+N*2+3} \\ a_3 * W_{i(n)+N*3} & b_3 * W_{i(n)+N*3+1} & c_3 * W_{i(n)+N*3+2} & d_3 * W_{i(n)+N*3+3} \\ \cdots & \cdots & \cdots & \cdots \\ a_7 * W_{i(n)+N*7} & b_7 * W_{i(n)+N*7+1} & c_7 * W_{i(n)+N*7+2} & d_7 * W_{i(n)+N*7+3} \end{bmatrix} \quad (3.5)$$

Through the lookup tables (Tables III.I and III.II), C[8,3] is contained in eight OSWs, as illustrated in Fig.3.11, since *hor* is four in the horizontal direction and *ver* is two in the vertical direction. The initial window $W_{i(n)}$ of C[8,3] is the first window W_1 ($i(n)$=1) of these eight OSWs, which can be derived from the first condition in the flowchart of Fig. 3.8. Therefore, eight coefficients of the parameter corresponding to the eight OSWs can be calculated by Eq.3.5.

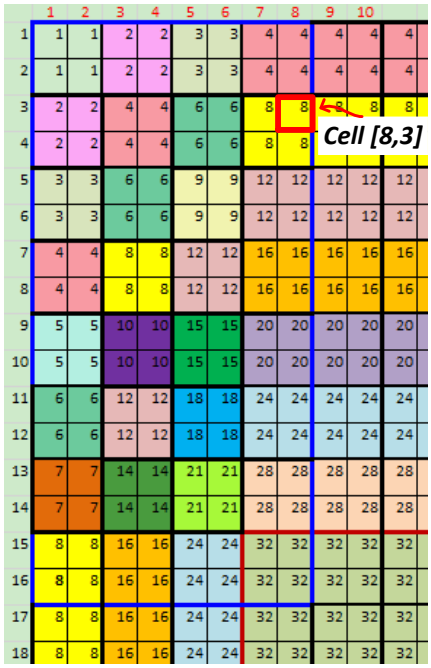


Fig.3. 11 Position of cell *C[8,3]* in a VGA image and its covering times (i.e. '8') by all OSWs when a SW-size of 8×16-cells is used.

That is, coefficients a_0, b_0, c_0, d_0, a_1, b_1, c_1, and d_1 in Eq.3.5 are '1' while all other coefficients are '0.' As a result, the OSW indices for C[8,3] are summarized as {1,2,3,4,38,39,40,41}. Thus all the overlapping OSWs related to the 8×8-pixel cell C[8,3] in the VGA image are listed in the matrix in Eq.3.6.

$$\begin{bmatrix} W_1 & W_2 & W_3 & W_4 \\ W_{38} & W_{39} & W_{40} & W_{41} \end{bmatrix} \tag{3.6}$$

For other cell size, SW sizes and image size, the maximal number of OSWs in horizontal and vertical directions can change as described in above discussion.

To implement the window algorithm as reported above, a loop control circuit for generating the window address (WA) and the RRRT value of the current cell C[c,r] for each of the SWs within the image is proposed as illustrated in Fig.3.12. The multiplication parameters *hor* and *ver* are initialized in the window-look-up-table (WLUT). A cell-position-look-up-table (CPLUT) is used to determine the reuse times according to RRRT (i.e. 1, 2, or 4) inside each OSW.

Each SW, declared by W_{index}, is allocated a separate cache space (storage unit) for storing the cell location within the SW, which determines the cell's RRRT value through look-up in CPLUT. In this way, the reuse times of the current cell-FV in all declared SWs W_{index} are calculated.

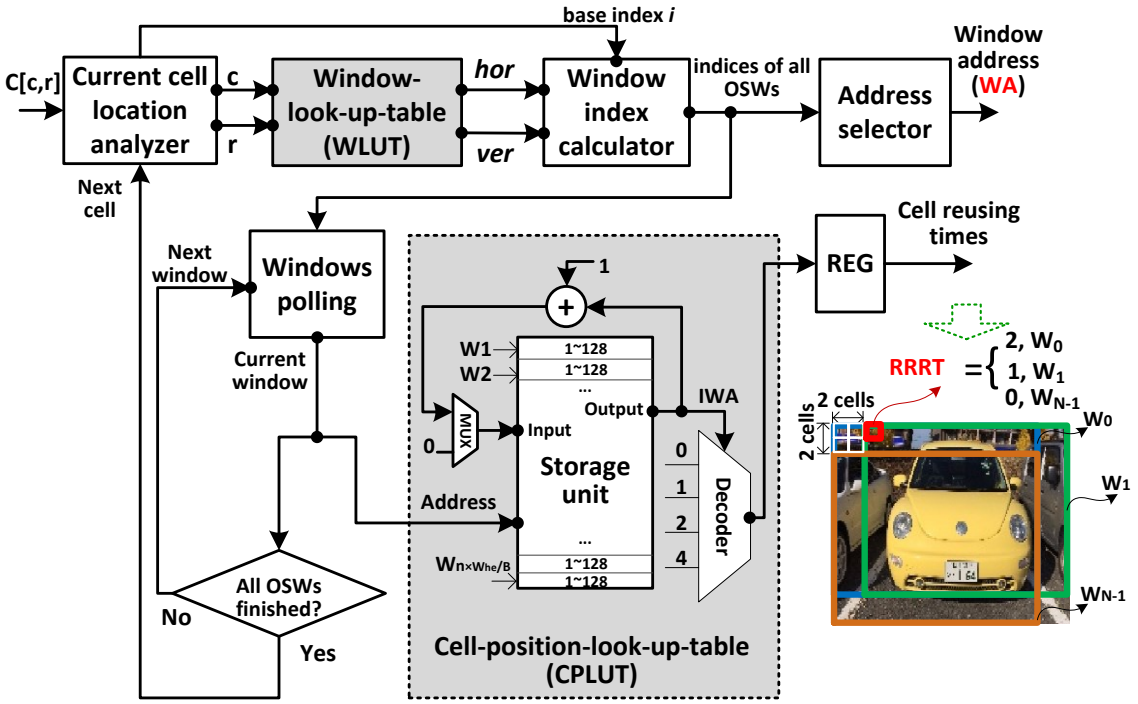


Fig.3. 12 Architecture for generating the window address (WA) and the RRRT value of the current cell C[c,r] for each of the SWs within the image, that contains C[c,r].

Each word in the storage unit for intermediate cell addresses of OSWs requires 7-bit if an OSW consists of 128 cells for 64 × 128-pixel or 128 × 64-pixel SWs. The storage capacity in CPLUT determines the practicability of the circuit in this research. Specifically, the storage requirement, i.e., the number of words in the storage unit of the CPLUT, which depends on the sizes of the input image, SW, and block, is at most $N \times W_{he}/B_s$.

The reuse times of the current cell in each OSW define how many vector components will be processed in parallel for window-level FV construction. For example, in the case of "2" as the RRRT value of one cell, the number of parallel-processed vector components is "8," as the four cell-FV components of the simplified SURF descriptor are used twice. To record to cell position in each related OSW, 16kbits of memory are assigned, which allows up to 1024 parallel processed SWs.

After OSW determination, the corresponding components of the reference FVs are invoked to execute the next stage of object detection according to the cell position in each OSW (W_{index}). For instance, the nearest-neighbor-search (NNS) is continued by partial squared Euclidean distance (PSED) calculation between the newly constructed FV of each declared OSW and the reference vectors, which will be described in detail in the following Chapter 5.

3.3 Hardware implementation with block-based normalized feature components

The parallel window construction method basing on the concept of RRRT of each cell in one SW is proposed in above section so that we can reconstruct the scan-window-based FVs based on the local cell components and a decoding scheme.

To reduce the times of memory access for the relatively large amount of cell number, another parallel window-level or window-based FV construction method is proposed in this research. Distinguished from the scheme described in section 3.2, the sliding-window-based localization for detecting the target objects in the input image is handled directly by the arrangement of the normalized block-based FVs in this method, rather than by a decoding scheme from the RRRT for each cell.

A fixed size of 2 × 2 cells is chosen for the image region of a block in this research. This results in 7×15=105 blocks within a 64×128-pixel window for pedestrian detection, while 7×7=49 blocks are contained within a 64×64-pixel window for face detection when a default CS of 8×8 pixels is used. The feature components of the four cells of each block are arranged in zigzag order to form 4×4=16 dimensional block-feature vectors for simplified-SURF descriptor and 4×9=36 dimensional block-feature vectors for HOG descriptor. Note that the regulation of window size mentioned in section 3.2.1.2 still valid in this block-based normalized FV construction scheme. Thus 1680-dimensional window-level FV for simplified SURF descriptor and 3780-dimensional window-level FV for HOG descriptor will be generated for a 64×128-pixel window.

Since the appropriate arrangement of sub-region components for feature construction in a SW can improve the performance of object detection significantly. The computational amount for local FV-component arrangement is an important factor to be considered. The developed pipeline architecture for FV construction is based on sequential cell-based FVs inputs and the developed parallel block-based L1-norm scheme reported in Chapter 4.

A feature space construction architecture including the FV normalization procedure is constituted as illustrated in Fig. 3.13. Four main functional parts, i.e. the parameter initializing circuit (PIC), the sequential pixel-based pipelined circuit (PBPC), the reconfigurable block-based normalization circuit (BBNC) and parallel window-based search circuit (WBSC), in this research.

The serially-inputted raw pixels scanned directly from the image sensor, are transferred to the sequential PBPC for extracting the local FVs for the cells, without preprocessing or pre-storage. The processing speed only relies on the pixel-transfer

frequency from the image sensor. The PIC transforms the flexible input-image resolution into the parameters CNH and CNV according to the customized CS programmable up to 32×32 pixels. As the feature normalization is performed across blocks represented by four cells, the block size is depended on the customized CS.

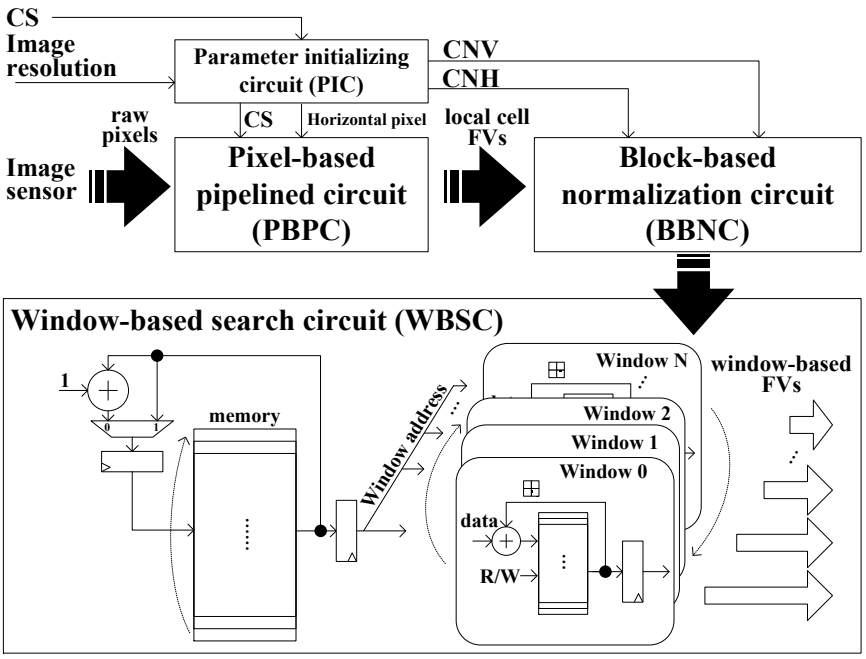


Fig.3. 13 Overall hardware architecture for window-based FV construction with block-based normalization.

The reconfigurable BBNC is implemented by addition, multiplication, bit shift, decoding, and division with variable input parameters for various cell numbers in horizontal and vertical directions from PIC. The WBSC is utilized for allocating the currently normalized block-FVs from the BBNC to all related SWs for parallel construction of the corresponding FVs for these SWs.

In WBSC, the number and the respective order of related SWs overlapping the current block are decoded according to the block position in the input image. Since all block-overlapping SWs related to current blocks are processed in parallel, the SW order for controlling the write and read addresses of the storage space for the intermediate SW-based FVs has to be buffered and invocated recursively as well. Each normalized block is invoked only once and then the local normalized results are reused by all related windows, to avoid redundant re-computation.

Rather than outputting one cell per time with the previously reported RRRT solution for cell reutilization in cell-based construction schemes, the local FV-components of four cells in one block are outputted simultaneously by the BBNC. There is no need to reuse

the normalized block-FV components inside the same SW as well since all blocks have been computed and arranged in raster manner by the BBNC. Additionally, the number of the SW-internal overlapped blocks is smaller than the cell number. In consequence, the block-based architecture prompts less computational amount and power consumption for local FV construction.

Multiple BBNCs in parallel can be applied for simultaneous multi-cell processing, e.g. applied for block-based FV inputs, so that the hardware architecture for normalization is not confined by the specific architecture for FV extraction. Moreover, the storage space for buffering the local cell-based FVs and related partial block FVs is reinitialized after completion of the processing of one block row.

3.4 Experimental results and discussion

3.4.1 Cell-based feature-vector construction

Vehicle detection is applied for an example to evaluate the efficiency of our proposed algorithm by software-based simulation as illustrated in Fig. 3.14.

A set of positive samples (i.e. cars), which are collected in the surroundings of Hiroshima University (HU), and negative samples (i.e. non-cars) from the INRIA dataset [10] are selected for verification in this work. A reasonable quantity of samples plays a significant role in establishing sufficient diversity to correctly detect the input data from the testing dataset [11].

Therefore, 1225 positive samples from the HU dataset and 12180 negative samples obtained by cropping non-car images from the INRIA dataset are resized to three kinds of SW sizes, i.e. 128 pixels × 64 pixels, 96 pixels × 64 pixels, and 64 pixels × 64 pixels, for training. Furthermore, corresponding sizes of 556 positive samples gathered from the HU dataset and 1812 negative samples cropped from the INRIA dataset are used as the image set for testing.

The results as shown in Fig.3.14 demonstrate that the classification performance enhances with the increase of SW size, which determines the feature-vector dimensionality. The obtained true positive rate (TPR) and true negative rate (TNR) results indicate a high accuracy of the proposed simplified-SURF descriptor for vehicle recognition. The high accuracy also testifies the appropriateness of the window-scan step size chosen in this research.

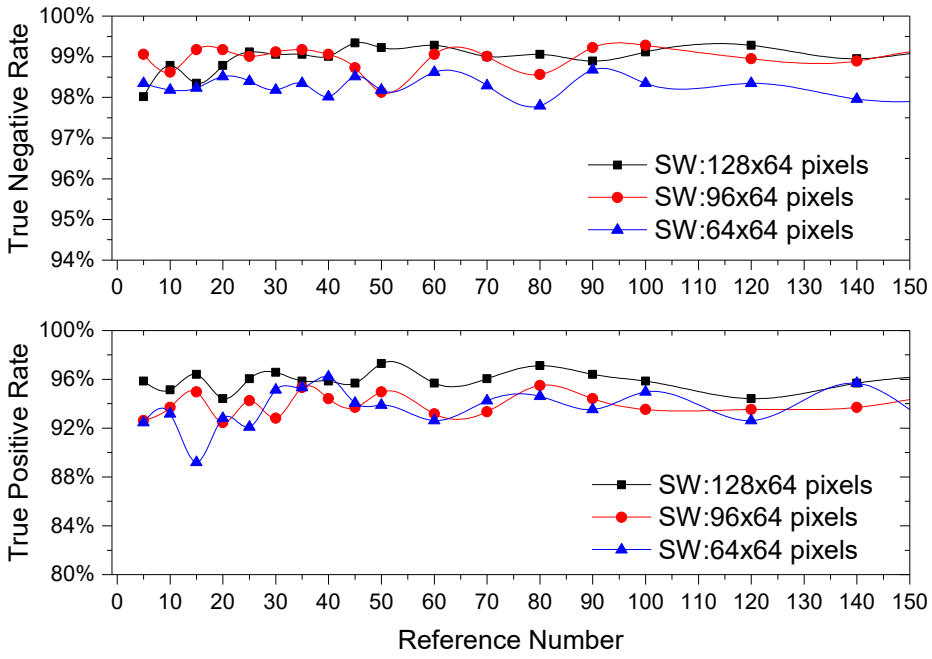


Fig.3. 14 True positive rate and true negative rate for car recognition with different SW sizes by the proposed simplified-SURF descriptor.

The one-sided increase of TPR or FPR can also impact the validity of the vehicle-detecting accuracy (ACC) with different quantities of testing samples according to the relationship in Eq.3.7. Here P is the number of all positive samples, N represents the number of non-car samples (negative samples), TP is the number of vehicle images detected correctly among all positive samples, TN is the number of correctly detected negative samples among all negative samples, and FP is the number of negative samples that are detected as positive samples. The specificity (SPC) is an associated variable of FPR (SPC=1-FPR).

$$\mathrm{TPR}=\frac{\mathrm{TP}}{\mathrm{P}}\times 100\%;\ \mathrm{FPR}=\frac{\mathrm{FP}}{\mathrm{N}}\times 100\%;\ \mathrm{SPC}=\frac{\mathrm{TN}}{\mathrm{N}}\times 100\%;\ \mathrm{ACC}=\frac{\mathrm{TP+TN}}{\mathrm{P+N}}\times 100\% \quad (3.7)$$

Rather than using the entire training dataset as reference data for the nearest neighbor search (NNS), the k-means algorithm is further used to cluster the dataset into k groups. The centroids of these k groups are finally taken as the reference data for classification.

Since only *w*/8 cell-based FVs have to be temporarily stored before constructing the FV of the first window, the memory requirement is again much lower than in the conventional integral-image-based implementations.

Synchronization with the pixel-input frequency from the image sensor and parallelized partial processing of multiple relevant SWs lead to flexibility with respect to

the size of processed cells, windows, and images. Furthermore, drastically reduces memory consumption, fast processing speed, and low power dissipation are achieved in the proposed cell-based window-level FV construction scheme.

3.4.2 Block-based feature-vector construction

Pedestrian detection is employed as an illustration to evaluate the accuracy of the block-based FV construction method on an equivalent software emulation employing an advanced 3.30GHz Intel® Core™ i5-4590 CPU and 8 GB of RAM memory.

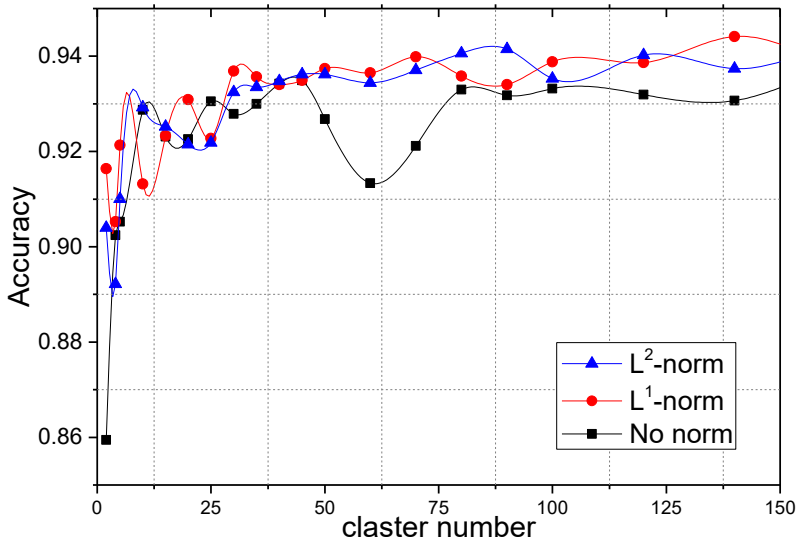


Fig.3. 15 Pedestrian-detection accuracy as a function of the cluster number, applying a framework composed by the cell-based simplified-SURF descriptor and the nearest neighbor search (NNS) classifier.

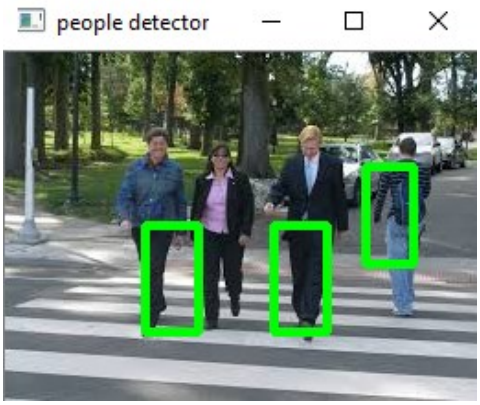


Fig.3. 16 Pedestrian detection verification in a software-implementation of the proposed architecture.

A framework composed by the cell-based simplified-SURF descriptor and the NNS

classifier shows that the L1-norm achieves comparable detection accuracy performance to the case where L2-norm is applied in Fig.3.15, but the hardware implementation for L1-norm is much simpler and requires much fewer resources than the L2-norm scheme. On the other hand, the proposed L1-norm scheme further operates with much better accuracy than without normalization (no-norm) for detection system from the INRIA dataset. Figure 3.16 shows the pedestrian detection result operating at a VGA (640×480 pixels) image by software-implementation of the proposed architecture with a 64×128-pixel SW. in practical application, the image should be adjusted to different scales or dynamically change the window size to match the object sizes in the testing image.

Multiple parallel block-based L1-norm circuits can accelerate the processing speed and the memory reutilization of each block leads to large reductions in on-chip storage requirements.

3.5 Conclusion

Since the feature descriptor with a window-based localization is a crucial factor for accurate object recognition based on a sliding-window strategy, two schemes are proposed for window-based feature construction in this chapter. One of the feature representation schemes uses direct decoding from reutilization information of each cell with the new concepts of a RRRT (regular rule of reusing times) table and a window index, which weakens the block concept and simplifies the computation by lookup-table usage during the FV construction procedure. The other scheme arranges the normalized block-based feature components without frequent accesses to cell-based components. Both of these two methods can avoid complex re-computation, thus computational cost and complexity of the feature extraction are significantly reduced. Furthermore, the storage space, required by these two methods for one row of SWs, can be overwritten for the next row of SWs, which results in much less storage requirement than in the previous research. It could be verified, that the second method with block-based normalized FVs can achieve higher detection accuracy than the former cell-based scheme without normalization of the FVs.

References

[1] Dollár, Piotr, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *Proc. IEEE Computer Vision and Pattern Recognition*, 2009, pp. 304-311.

[2] M. Enzweiler, and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.31, no.12, pp.2179-2195, 2009.

[3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.32, no.9, pp.1627-1645, 2010.

[4] C. H. Lampert, B. B. Matthew, and T. Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," in *Proc. IEEE Computer Vision and Pattern Recognition*, 2008, pp. 1-8.

[5] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.34, no.11, pp.2189-2202, 2012.

[6] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proc. Computer vision–ECCV2006*, Graz, Austria, 2006, pp. 404-417.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol.110, no.3, pp. 346-359, Jun. 2008.

[8] A. Mohan, C. Papageorgiou, T. Poggio. "Example-based object detection in images by components," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.23, no.4, pp. 349-361, Apr. 2001.

[9] N. Dalal, INRIA Human Dataset (2005) [http://pascal.inrialpes.fr/data/human/].

[10] J. A. Eichel, A. Mishra, N. Miller, N. Jankovic, M. A. Thomas, T. Abbott, D. Swanson, and J. Keller, "Diverse Large-Scale ITS Dataset Created from Continuous Learning for Real-Time Vehicle Detection," *IEEE Trans. Intelligent Transportation Systems*, vol.16, no.1, pp. 1-13, Mar. 2015.

Chapter 4

Feature-Vector Optimization

4.1 Introduction

Feature vector normalization (FVN) and dimensionality reduction (DR) are two important issues for feature-vector-based designs, to reduce the complexity of computation without a significant decrease in accuracy.

To support the substantial computational amount for processing multiple frames per second in real time, a pixel-based pipelined architecture that synchronizes to the working frequency of the image sensor was reported for both cell-based HOG [1] and simplified-SURF descriptor applied Haar-like feature vector [2]. The integral image is substituted by an immediate processing engine for the serially input-pixel data from the image sensor in our previous work. Local feature vectors (FVs) are computed in terms of a sub-region named 'cell'. The cell-based local FVs are sequentially outputted for the scan-window-based multidimensional feature representation, matching to an object directly without normalization and dimensionality reduction.

On one hand, although the non-normalized cell-based feature descriptors effectively reduced the computational complexity, the omission of normalization in the feature-extraction stage results in unstable accuracy results against circumstance changes in the detection stage. This is a serious concern, because the real-world practical application circumstances generally vary irregularly and unpredictably. Illumination intensity of light source, foreground-background contrast and the automatic gain control from a camera, etc., limit the performance of vision-based object detection and recognition systems. To avoid degradation of performance due to above issues, an effective normalization method turns out to be essential [3-5].

On the other hand, object detection and recognition are greatly hampered by a high-dimensional feature descriptor, such as the 1680-dimensional window-level FV for the simplified SURF descriptor or the 3780-dimensional window-level FV for the HOG descriptor, which will be generated for a 64×128-pixel scan window. In such high-dimensional spaces, classical pattern recognition algorithms such as the nearest neighbor

search (NNS) are nearly intractable with respect to target detection. Feature reduction without decreasing the detection accuracy turns out to be an essential step.

4.2 Hardware implementation for L1-norm

Similar to the original HOG algorithm [6], the four cell features in a block are normalized in [3], showing that the L1-sqrt-norm and L2-norm perform equally well, while more multipliers and more memory accesses make an FPGA-based design much more complex and resource consuming than without normalization. The L1-sqrt-norm is also called 'Least absolute deviation by square root', which can be illustrated in Eq.4.1. L2-norm is also called 'Least square' that is illustrated in Eq.4.2.

The L1-sqrt-norm:

$$v \rightarrow \sqrt{\frac{v}{\|v\|_1+\epsilon}} \tag{4.1}$$

The L2-norm:

$$v \rightarrow \frac{v}{\sqrt{\|v\|_2^2+\epsilon^2}} \tag{4.2}$$

Here v presents the non-normalized descriptor vector, while ϵ is a small constant. Some regularization ϵ is needed as we evaluate descriptors densely, including on empty patches, but the results are insensitive to ϵ's value over a large range [6].

Compared to the L2-norm, the L1-Sqrt-norm avoids squaring the histogram elements leading to an efficient implementation without reducing the detection rate significantly. Furthermore, multi-scale and multi-object detection is becoming a tendency for intelligent machine-vision applications [7-9]. The typical cell size (CS) is reported as 8×8 pixels for the original HOG descriptor [6], and this CS is widely used in other current research work such as in [4]. The fixed size of cells in the conventional approach to generating the feature pyramid leads to the requirement of more calculations, including linear interpolation [9]. However, the multi-scale image pyramid in state-of-the-arts results in large data expansion and higher computational complexity, necessitating fast hardware implementations to enable real-time processing. Whereas, the hardware resource is a critical factor for integrated circuit designs.

4.2.1 Block-based normalization algorithm for feature vectors

To improve detection of the target objects among complex backgrounds, this research

proposes a real-time feature-vector-normalization algorithm. The simpler L1-norm, which is defined as Eq.4.3 in [6], instead of the L2-norm or the L2-Hys-norm is applied in this research and is verified to meet a favorable tradeoff between computing complexity and detection accuracy. Here v presents the non-normalized descriptor vector and ϵ is a small constant.

$$v \rightarrow \frac{v}{\|v\|_1+\epsilon} \tag{4.3}$$

Specifically, a reconfigurable normalization circuit [10] together with a cell-based feature descriptor is performed across overlapping blocks in size of 2×2 cells in this research.

The non-overlapping rectangular pixel group named 'cell' is the elementary output unit of many cell-based feature descriptors. The dimensionality of the local cell feature vector varies in accordance with the adopted feature descriptor. For instance, a nine-dimensional vector for each cell is generated by accumulating weighted gradient magnitudes in nine bins matching with corresponding gradient-angle groups, each covering 20° between 0° and 180°, for the case of the HOG descriptor [1]. In contrast, the edge features of the simplified SURF descriptor with Haar-like feature vector in [2] compute the gray level differences and their corresponding absolute values between white and black groups of pixel rectangles, resulting in *four* dimensions of the local feature vector for each cell. In practice, the other cell-based feature descriptors, reported in various research works such as [9] and [11], are desired to be appropriate to be applied in the proposed L1-norm architecture of this research as well.

Particularly, the proposed block-based Lp-norm operation over an image in this research is defined as Eq. 4.4.

$$d_i' = d_i/\left(\left|d_{i(cell0)}\right|^p + \left|d_{i(cell1)}\right|^p + \left|d_{i(cell2)}\right|^p + \left|d_{i(cell3)}\right|^p\right)^{1/p} \tag{4.4}$$

Here, d_i ($i \in [0, n\text{-}1]$) refers to one component of the n-dimensional cell-based feature vector. For the cell-based HOG descriptor in [1], n=9, while n=4 in the cell-based simplified SURF descriptor in [2]. In the proposed L1-norm scheme, the parameter p is set to be '1', i.e., p=1.

A general-purpose normalization-circuit architecture is developed and implemented by a reconfigurable ASIC-based solution to meet for multi-scale and multi-object detection tasks. With the block movement by a fixed half-block stride (i.e., one cell stride) in a row-raster-scan manner in this research, as illustrated in Fig.4.1, the adjacent blocks

are overlapped and one cell can be shared by up to four blocks. The feature vector components of the four cells of each block are arranged in raster order to form the local block-based feature vectors, resulting in a final 9×4=36-dimensional local block-based feature vector in [1] and in a 4×4=16-dimensional local block-based feature vector in [2]. And the local block-feature vectors become the basic components for constructing the high-dimensional feature vector of the entire scan window. Sharing and reusing of one cell-based feature vector can increase the dimensionality of the block-based feature vector and the final window-based feature vector.

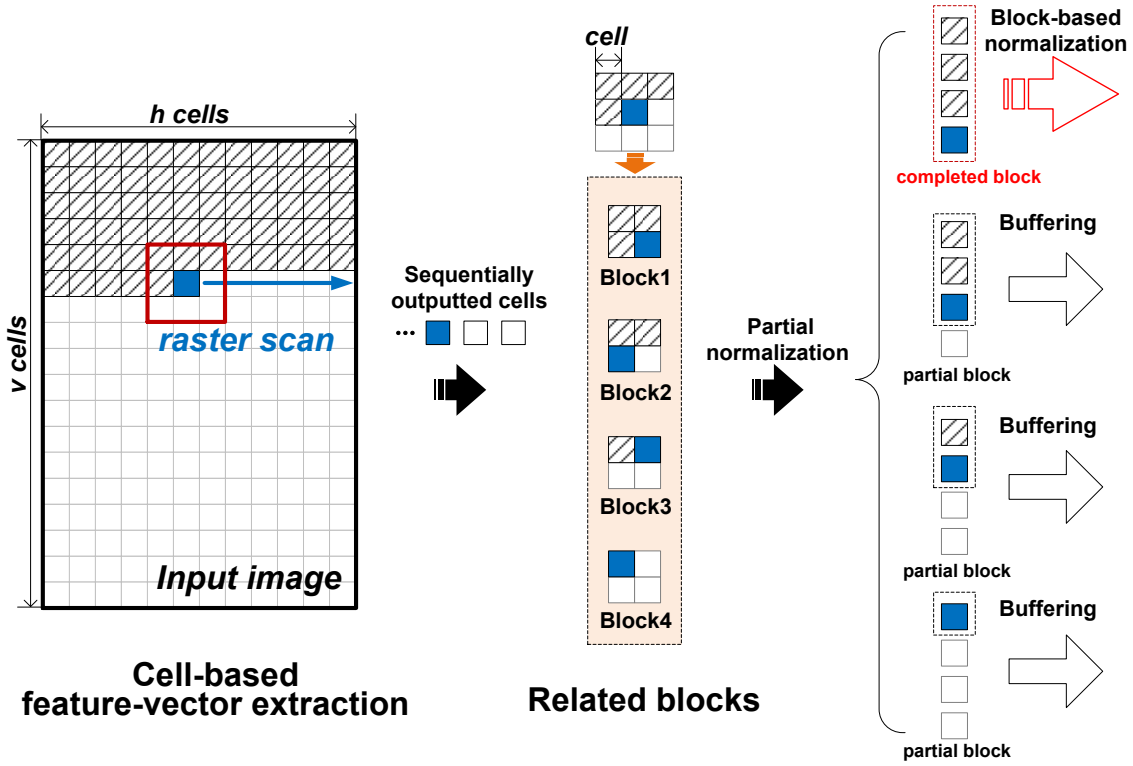


Fig.4. 1 Parallel feature vector normalization scheme of overlapping blocks with cell-based feature components. Each cell will be normalized by its neighboring cells within the same block.

In this research, each cell is normalized by its neighboring cells within the same block to increase robustness to texture and illumination variation. Specifically, the size ratio between cells of increasing size is used to define the scaling factor of the corresponding pyramid level in this research. Since the CS is not a restriction factor for the subsequent normalization and the window-based feature construction, the proposed hardware architecture can be applied to calculate components of various CSs up to e.g. 32×32 pixels, in accordance with the desired pyramid levels. The rectangular CS and the image resolution are only limited by the on-chip storage capacity for VLSI implementation. The obtained flexible CS and the unlimited vertical size of input images can be viewed as a substantial contribution to multi-scale and multi-object detection in our work.

Although the configurable normalization concept can be applied for all four block normalization schemes L2-Hys, L2-norm, L1-sqrt and L1-norm introduced in previous research, the simpler L1-norm is applied in our hardware design and is verified to meet a favorable tradeoff between computing complexity and detection accuracy. Furthermore, flexible regulation for memory allocation is executed according to customization parameters from the input so that the presented block-based L1-norm-circuit architecture has high processing-flexibility for different image-cell sizes, cell-based feature descriptors and image resolutions, which is only limited by the on-chip storage capacity. Synchronization with the pixel-transfer frequency from the image sensor enables real-time processing. The cells and blocks are scanned in row raster manner for the entire image and are arranged in sequence so that the intermediate-calculation results can be stored and invoked in the proper order.

Different CSs and window sizes (WSs) lead to different block numbers and therefore different window-based feature vector dimensionality. The designated sizes of the window are depended on location and size of different target objects in the image since the window is applied for matching the target object. The sliding-window-based localization for detecting the target objects in the input image has been discussed in Chapter 3.

In conclusion, a reconfigurable normalization algorithm with variable cell size is proposed in this research for extensive applications using image-cell-based feature vectors.

4.2.2 Flexible cell size for multi-scale image pyramid

Better performance can be achieved by customized normalization of each element (i.e., cell-FV component) for different local blocks containing the respective cell, and by treating the results as independent dimensions.

Concerning the presented block-based L1-norm circuit, the developed architecture is only constrained by the cell number in the horizontal direction. Figure 4.2 (a) shows the conventional approach to generate the feature pyramid, where the image pyramid is generated first from the input image by bilinear interpolation, and then feature pyramid with fixed size (e.g. 8×8 pixels) cells are generated [4]. Thus, each scale of the image pyramid should be computed independently, which results in a mass of redundant computation and significant requirements of the on-chip resource.

In this research, the image pyramid is skipped, and the feature vector of each image scale with different cell sizes are generated directly from the input image as shown in

Fig.4.2 (b). In other words, the cell sizes can be applied for matching the corresponding level of the feature pyramid so that the multi-scale objects can be detected from the input image. The ratio between each cell size defines the scaling factor of the corresponding pyramid level. The cut-off frequency of each filter is proportional to the scaling factor of the corresponding pyramid level. The low pass filters in response to smaller cell sizes prevent aliasing by removing high frequencies in the input image.

Since the input pixels are coming in sequence, partial feature vectors are calculated on-the-fly for all related cells that contain the input pixel.

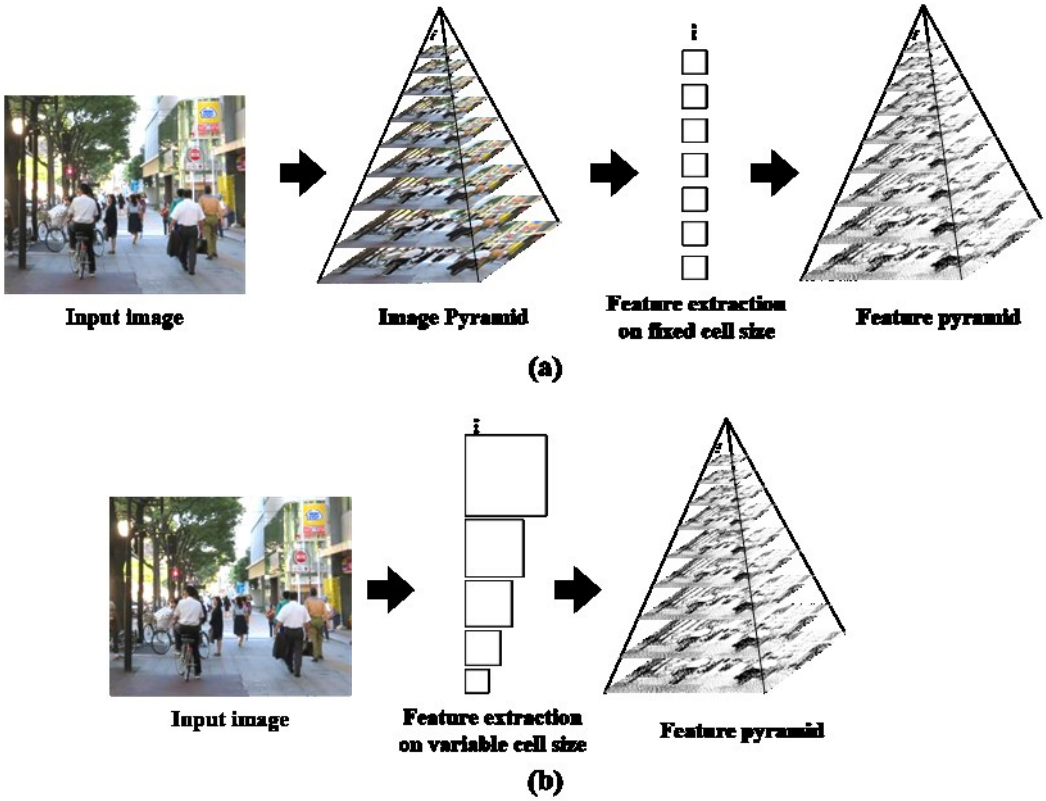


Fig.4. 2 Generation of a feature pyramid by (a) generating image pyramid first then calculating fixed-size cell FVs, or (b) calculating FVs of different cell sizes corresponding to each pyramid level.

The resulting partial features of different pyramid levels are available at different times corresponding to the cell size. In the practical on-chip FV-extraction circuit in this research, apart from the image resolution flexibility of up to 1024×∞ pixels, there are five CSs, i.e., 2×2, 4×4, 8×8, 16×16, 32×32 pixels, handled by the for multi-scale and multi-object detecting applications.

4.2.3 Hardware-oriented architecture for block-based L1-norm

4.2.3.1. Overall architecture

The developed reconfigurable circuit architecture for block-based L1-normalization consists of three main parts, as illustrated in Fig.4.3. The upper 'Block part' is used for caching and updating of intermediate block-summation results in the currently processed row of image blocks, which depends on assigned image resolution and CS in different applications. The four cells related to the same block are outputted successively in the 'Pipelined L1-norm processing circuit'. The final descriptor feature vector (FV) of a window is constructed by combining the normalized block-FVs (i.e., d_0', d_1', …, d_n') of all related blocks. Cache and extension for one row of cells are handled in the lower 'Cell part' of the circuit architecture.

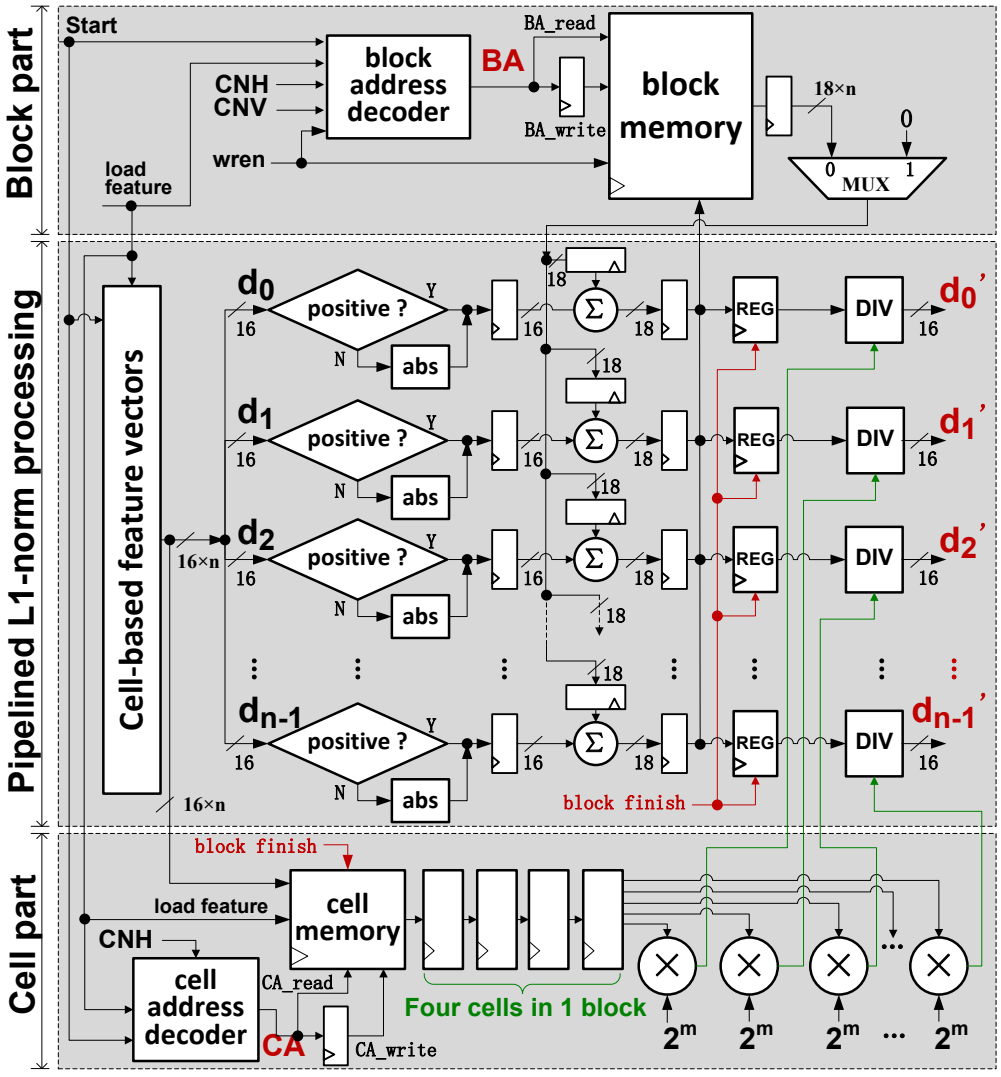


Fig.4. 3 Hardware architecture of the reconfigurable block-based normalization circuit (BBNC) for pipelined L1-norm processing with cell-based feature vectors.

The developed pipeline architecture for FV normalization scheme is based on sequential cell-based FVs inputs. A parameter initializing circuit (PIC) will assist the sequential pixel-based pipelined circuit (PBPC) to generate sequential cell-based FVs to the reconfigurable block-based normalization circuit (BBNC).

The serially-inputted raw pixels scanned directly from the image sensor, are transferred to the sequential BBNC for extracting the local FVs for the cells, without preprocessing or pre-storage. The processing speed only relies on the pixel-transfer frequency from the image sensor. The PIC transforms the flexible input-image resolution into the cell number in horizontal (CNH) and vertical (CNV) directions according to the customized CS programmable up to 32×32 pixels.

As the L1-norm according to Eq. 4.4 is performed across blocks with 2×2 cells, the block size is depended on the customized CS. The reconfigurable BBNC is implemented by addition, multiplication, bit shift, decoding, and division with variable input parameters for various cell numbers in horizontal and vertical directions. As the CNH and CNV are two parameters adjustable from circuit input, the extensibility to different cell-number-dependent image-processing solutions is only limited by the on-chip storage capacity.

4.2.3.2. Cell part

Since the local non-normalized FVs are inputted in terms of cell-based FV, these cell-based FVs have to be cached in the ‘Cell part’ until the accumulation operations with respect its related blocks are all accomplished for block-based normalization. The cell number to be buffered relies on the CNH direction of the input image. As the L1-norm (i.e., p=1) according to Eq. 4.4 is performed across blocks with 2×2 cells, “CNH+1” cells have to be temporarily stored in the ‘cell memory’.

As the cell-FVs are sequentially inputted and processed in parallel for all related blocks at the same time, the buffer space for the front-stored cells will be overwritten for the subsequently inputted cells in sequence during the block normalization, resulting in high efficiency for storage space and energy consumption. Thus the loop cell address (CA) of the cell memory for writing or reading cell-based FVs is generated by a *decoder*. Specifically, only 1024/8+1=129 cells in size of 8×8 pixels have to be stored for XGA images with 1024×768 pixels. In other words, the image width and the CSs to be handled by this circuit are only constrained by the on-chip memory capacity.

Besides, there is no constraint in image height in principle because of the reutilization of memory space. In particular, each word of the cell memory has nine-bin×16 bit=144 bits for the HOG descriptor in the practical ASIC design.

To ensure a fixed-point computation in the middle 'Pipelined L1-norm processing' part of Fig. 4.3, the numerator of Eq. 4.4, i.e. each cell-FV component, is multiplied by a factor of 2^m, which determines the computational precision of the block normalization. We set m=12 in the practical chip design to meet a tradeoff between the required data bandwidth and the precision of the calculation results. Only the integral part of the product is retained for the subsequent division by the block-internal accumulation of FVs. Thus the hardware architecture for the complicated floating-point computation of division can be implemented by a fixed-point circuit with insignificant part of the precision loss.

4.2.3.3. Block part

In this research, the sequentially outputted cells are invoked for partial block normalization immediately. The upper 'Block part' of Fig. 4.3 is used for caching and updating of intermediate block-summation results in the currently processed image blocks.

Since the block moves by a fixed half-block stride (i.e., one cell stride) in a row-raster-scan manner, the intermediate block-summation required massive redundant memory space if all image blocks are stored. For instance, (640/8-1)×(480/8-1)=4661 overlapping blocks have to be buffered for processing VGA (640×480 pixels) images on 8×8-pixel cells.

As the former blocks, which are already outputted in sequence for window-based FV construction, become invalid, these utilized blocks can be discarded and the corresponding storage space can be reinitialized in the block memory. Therefore, only 80 words and 128 words in the 'block memory' are required for processing VGA and XGA images, respectively. The four 16-bit FV-components of the corresponding four independent cells of each block are accumulated, resulting in 18-bit precision for each bin of the HOG descriptor. This means that each word of the block memory can be allocated to 9×18-bit=162bits for each local block-FV component.

The necessary storage space for one row of blocks as well as the corresponding storing locations, i.e. read and write block addresses (BAs) for the block memory, are regulated in accordance with the CNH and CNV parameters of the input image. The BA describes the number of the cell-overlapping block and its respective positions covering the current cell in the image.

A given cell can be covered by up to four blocks, in case of block movement by one cell in a row-raster manner inside a window. Cells which can be covered by 1, 2 or 4 blocks are marked identically in Fig. 4.4 and the corresponding BA_i (i from 0 to 3) are given in Eq. 4.5.

$$BA = \begin{cases} BA_0 = k - k/h \\ BA_1 = k - k/h - 1 \\ BA_2 = k - k/h - h \\ BA_3 = k - k/h - h + 1 \end{cases} \quad (4.5)$$

Here k and h represent the current-cell order number in the whole image and the image's CNH value, respectively.

Although cells marked in the same color in Fig. 4.4 are overlapped by the same number of blocks (1, 2 and 4 blocks for cells marked in black, gray and white, respectively), the BA or BAs, i.e. the sequential block order in the image that the current cell belongs to, are different for each cell.

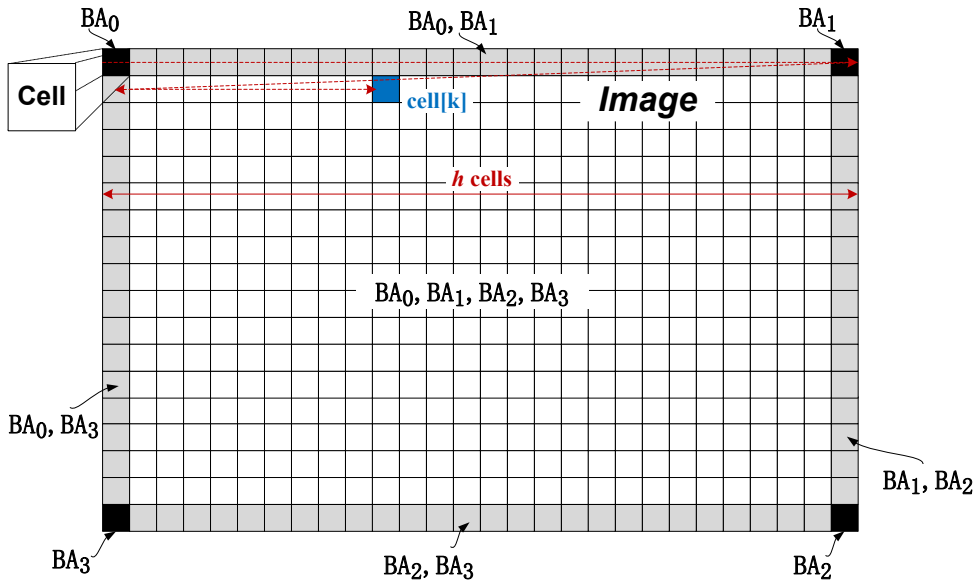


Fig.4. 4 Corresponding block addresses (BAs) for all cells according to their positions.

For instance, the four corner cells are all overlapped by one single block, but their separate BA is assigned as BA_0, BA_1, BA_3, and BA_2 by Eq. 4.5 as illustrated in Fig. 4.4. In other words, the BA related to the current cell is depended on the image resolution and the cell position in the image.

Figure 4.5 illustrates more reconfigurable-architecture details of the 'block address decoder', which generates the BAs related to a given image cell. All the cells in the input image are categorized into nine cases according to the situation analyzed above so that all BAs can be sequentially decoded by Eq. 4.5 with lower computational cost.

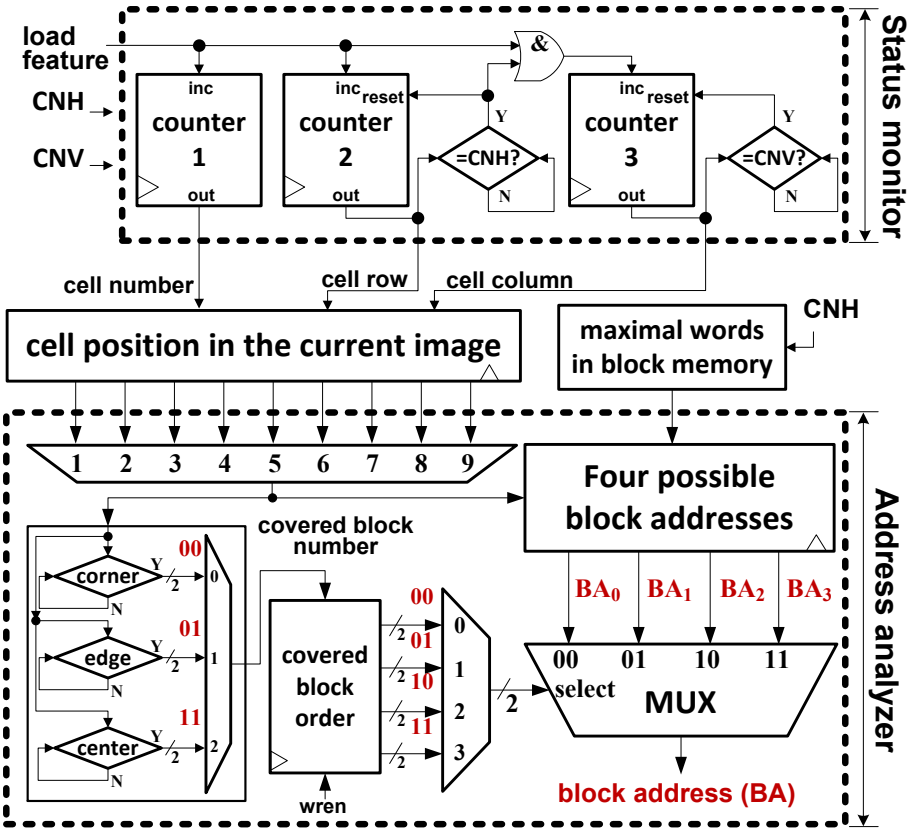


Fig.4. 5 Structure of the 'block address decoder' (see Fig.4.3) for coordinating the reconfigurable normalization according to cell position and image size, customized by the configuration parameters CNH (up to 128) and CNV (unlimited).

As illustrated in the circuit block 'address analyzer' of Fig. 4.5, the four possible block addresses are first calculated and then selected as the final outputted BAs on the basis of the assigned cell case. The input customization parameters of the circuit block 'status monitor' determine the configuration information of the currently processed image. The status monitor is composed of a counter array for simpler architecture parameterization. The counters have the function of a decoding logic which can translate the external regulation parameters into the local control signals, contributing to the architecture's processing flexibility in terms of image size (CNH×CNV cells).

4.2.3.4. Pipelined L1-norm processing circuit

Local variations in illumination and foreground-background contrast make effective local contrast normalization essential for good performance in object detection. The internal components of each inputted cell-based FV are used for normalization within a block according to Eq. 4.4 with p=1. The absolute values of each FV-component of the four cell-based FVs in a block are accumulated separately. The multiplied product of FVs

of the current cell by a factor of 2^m from the 'cell part' will be invoked for the division as soon as the accumulation operations of the four cells within a block are completed. The pipeline registers synchronously latch the transmitted data with the same rising clock edge so that the results can be transferred to the following register in the pipeline. In other words, the proposed circuit is constructed as a pipeline processing cluster.

Once the FV extraction for one block, is accomplished as indicated in Fig.4.3, the corresponding block-FV is transferred for window-based FV construction and the storage space for this block-FV is reinitialized for the subsequent block soon afterwards. Otherwise, the cell-FV and the partial block-FV results are temporarily stored in the memory for block-FV construction. The necessary storage space for one row of blocks as well as the corresponding storing locations, i.e. read and written BAs for the block memory, are regulated in accordance with the CNH and CNV of the input image. Consequently, on-chip memory requirements are strongly reduced. Moreover, multiple BBNCs in parallel can be applied for simultaneous multi-cell processing, e.g. applied for block-based FV inputs, so that the hardware architecture for normalization is not confined by the specific architecture for FV extraction.

4.3 Dimensionality reduction of feature vectors based on PLS regression scheme

The curse of dimensionality is inherently expensive for hardware circuit designs, especially for the nearest neighbor search (NNS) classifier. One of the desired functions of this research is to reduce the cardinality of the high-dimensional FV representation space. To circumvent the densely processed data during recognition procedure, we employ partial least squares (PLS) [12] analysis to project the FVs onto a much lower dimensional space. The PLS analysis can preserve significant discriminative information that makes it as an efficient dimensionality reduction technique in pattern recognition.

The dimensionality reduction scheme performed in this research is to analyze the FVs by the PLS model for each SW in the image. The basic idea of PLS is to construct new descriptor vectors as the descriptor variables (i.e., the FVs). A brief mathematical description of the procedure is provided in Eq.4.6.

$$\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{P}^{\boldsymbol{T}} + \boldsymbol{E} \tag{4.6}$$

where **X** is a $n \times d$ data matrix denoting an d-dimensional space of FVs for n original window samples, **Y** is a $n \times k$ data matrix containing k extracted FVs, the $(d \times k)$ matrix **P**

represents the loading of weight vectors, and the $n \times d$ matrix **E** is the residuals. For Haar-like descriptor, the dimensionality d of window-level FV is equal to 1680, and k ($k<d$) newly constructed components in matrix **Y** summarize the original FVs as well as possible.

In this research, a set of weight vectors (or projection vectors) $\mathbf{W}_{PLS}= \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k\}$ are pre-stored in a memory and invoked for PLS analysis with corresponding original cell-based Haar-like FV $v_{cell}=\{\sum D_x, \sum D_y, \sum|D_x|, \sum|D_y|\}$. The search of SWs overlapping current cell and dimensionality reduction of current window are two concurrent procedures, according to the RRRT value and current window value obtained by the circuitry shown as Fig.4.6.

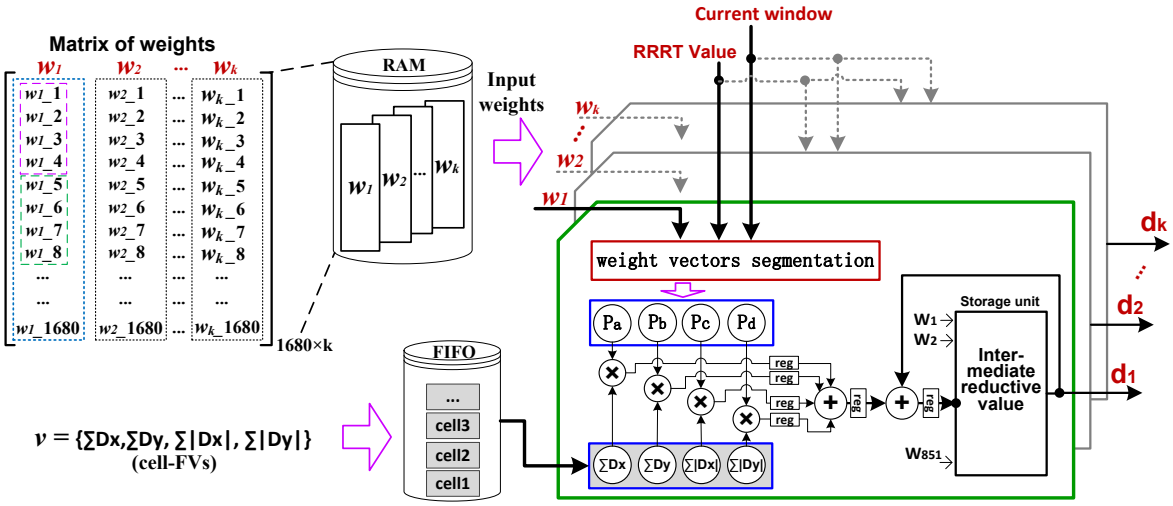


Fig.4. 6 Data flow and developed architecture for transforming the dimensionality of Haar-like FVs from 1680 dimensions to k dimensions.

The cell-based FV v_{cell} as one individual processing unit which is computed by Eq.2.6 is projected onto corresponding components in the weight vectors $\mathbf{W}_{PLS}$ that is obtaining the latent vectors d_i as a result.

Each weight component $\boldsymbol{w}_i$ is 1680 dimensions in application scene of Haar-like descriptor when operating in 64×128-pixel SWs. The 1680-dimensional space of Haar-like FV of the current window is projected onto k 1680-dimensional weight components $\boldsymbol{w}_i$ in parallel, constructing k-dimensional extracted FV $\{d_1, d_2, \ldots, d_k\}$ of the current window, which is then applied for object recognition.

Specifically, each weight component $\boldsymbol{w}_i$ is segmented to 4-dimensional sub-components for the raw cell-based Haar-like FVs on the basis of the index of current window and the RRRT value $\boldsymbol{R}$ (i.e. 1, 2, or 4) according to the current cell.

Figure 4.7 shows a part of 4-dimensional sub-components of weight vectors

segmented in each window corresponding to the cell sequence on VGA images. All the 4-dimensional sub-components which are projected onto one same cell for the simplified SURF descriptor applying Haar-like FVs will be processed in parallel. The intermediate cell-based accumulative results of dimensionality reduction of each SW in PLS analysis are temporarily stored and updated in a storage unit respectively.

Then, a simple and efficient nearest neighbor classifier is used to classify this lower-dimensional vectors $\{d_1, d_2, \ldots, d_k\}$ as either a human or non-human on the popular INRIA pedestrian dataset consisting of complex scenes in pedestrian detection applications.

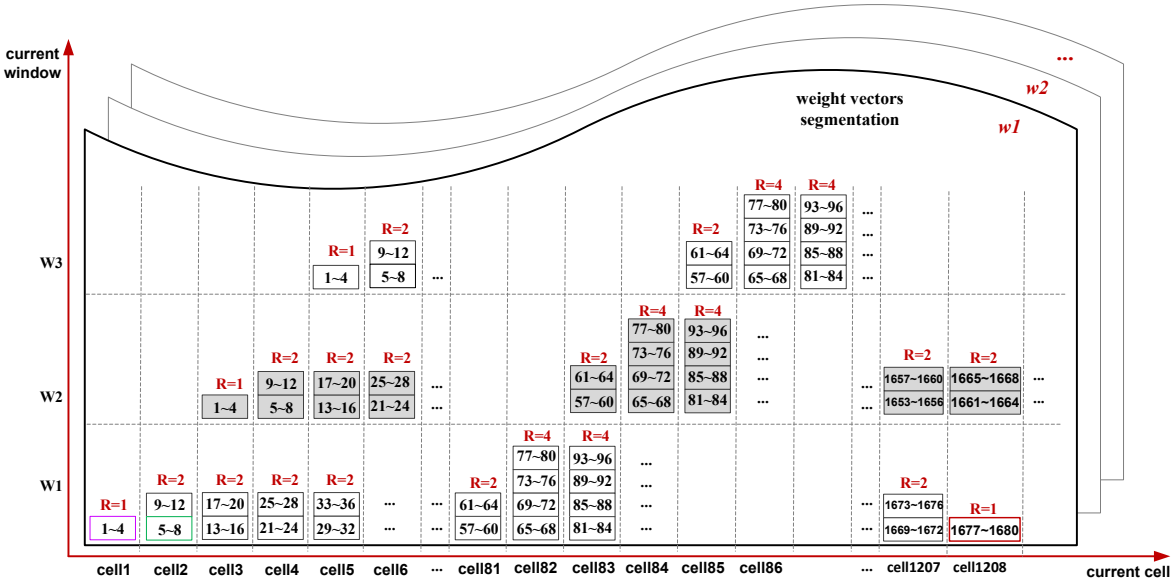


Fig.4. 7 Concurrent PLS regression analysis for invoking correct weight vectors in parallel for multiple SWs related to the current cell.

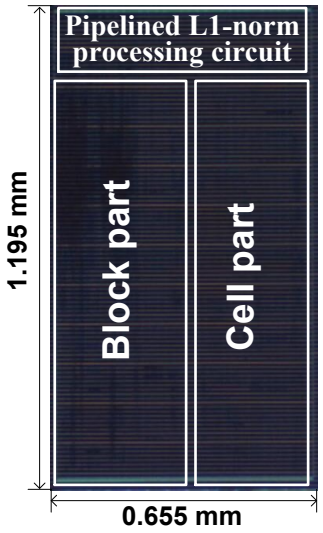
4.4 Experimental results and discussion

4.4.1 VLSI implementation results for L1-norm

A proof-of-concept prototype chip for flexible feature vector normalization is fabricated in 65 nm CMOS, as shown in the photomicrograph of Fig.4.8. The 3780-dimensional cell-based HOG descriptor is applied for reconfigurable FV extraction in this design. The total core area for the L1-norm circuit is about 0.655×1.195 mm^2. The layout of the L1-norm circuit in Fig. 4.9 verifies its logical density.

Due to the flexibility in image resolution, the prototype chip can deal with a maximum width of 128 cells and an unlimited image height with *five* different cell sizes, i.e., 2×2, 4×4, 8×8, 16×16, 32×32 pixels.

Approximately 31 fps real-time-processing capability for XGA-size image frames can be obtained when operating at 25 MHz frequency. The power consumption of this coprocessor is 21.3 mW at 1.0 V core voltage when operating at the maximal frequency of 125 MHz. Lower working frequency can be used, which leads to lower power dissipation by adjusting to the different speed requirements of various applications.



Core area	0.78 mm^2
Supply voltage	1.0 V
Maximal frequency	125 MHz
Power consumption	21.3 mW
Frame rate (XGA)	158fps (@125MHz)
On-chip SRAM	36 kbits
Image resolution	Flexible $\leqslant$ 1024×∞ pixels
Multi-scale	5-level pyramid

Fig.4. 8 Micrograph of the prototype chip in 65 nm CMOS technology and the device performance for feature vector normalization.

The required storage space for on-chip for the reconfigurable general-purpose L1-norm circuit consumes only 36 kbits dual-port memory space due to the applied reutilization scheme.

From the layout of the proposed VLSI implement of the L1-norm circuit, the memory occupies most of the area of the die chip, which illustrates that the memory requirement is a critical factor for the on-chip circuit design.

The word precision for each FV-component of the cell-based HOG descriptor is 16 bit, resulting in $9\times[16+\log_2(2\times2)]=162$bit for each local 36-dimensional normalized block-FV (2×2 cells), to provide good classification accuracy and minimized hardware cost. The on-chip storage capacity allows up to 128 cells in the horizontal direction can be handled without restrictions of the CS.

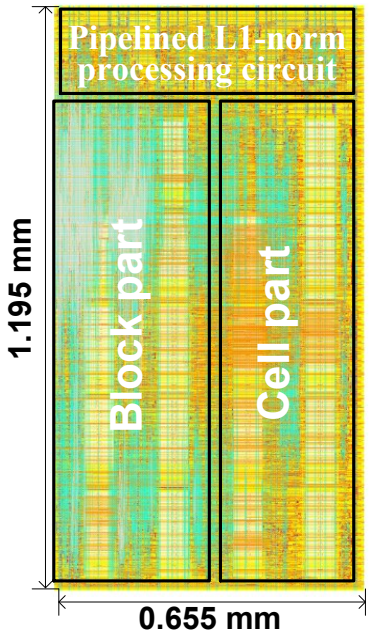


Fig.4. 9 Layout of the developed circuit for L1-normalization.

The extended bit width of each component of the local cell-FV by the factor 2^m (m=12 in real chip design), which should be recovered for the final detection results, ensures an optimized tradeoff between the data precision and the computational complexity.

Note that the developed architecture can handle also other cell-based feature descriptors for normalized SW-based FV construction to enable the realization of various object detecting applications.

4.4.2 FPGA implementation results of PLS regression

The hardware circuit of the proposed PLS analysis for dimensionality reduction for the simplified SURF descriptor with Haar-like FVs was described by Verilog HDL and then implemented on Altera® Stratic IV field-programmable gate array (FPGA).

As shown in Fig.4.10, the demo-system consists of a DE4 FPGA board, an LCD display and with or without a CameraLink camera with XGA (1024×768 pixels) frames. A 1680-dimensional Haar-like FV is transformed to 8-dimensional simplified FV, which significantly reduce the computation amount and cost for object detection.

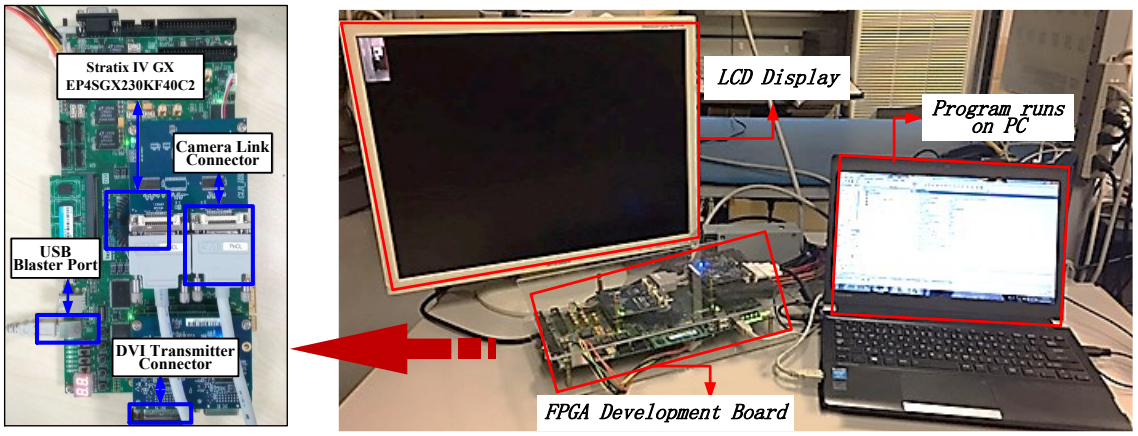


Fig.4. 10 Proposed object detection system based on FPGA and the applied Altera® Stratic IV FPGA development board.

An independent DIV transmitter-receiver board is inserted to the FPGA development board and an assistant driver is installed in the FPGA as well. The PC serves as a host to program the configuration files to FPGA device and manage the datasets.

TABLE IV. I
Physical Resource Utilization of the Proposed Circuit for Object Detection with PLS Analysis

Resources	Used	Available	Utilization
Combinational ALUTs	14730	182400	8%
Memory ALUTs	272	91200	1%
Dedicated logic registers	9676	182400	5%
Total registers	10108	N/A	N/A
Total block memory bits	397164	14625792	3%
DSP block 18-bit elements	144	1288	11%
Total PLLs	3	8	38%

The synthesis result of the proposed object detection system with PLS analysis circuit is listed in Table IV.I, the multipliers are configured by look-up tables (LUTs) and DSP block elements. Part of registers are employed as assistant units to implement pipelined data flow with memory access, and others are used for operations like fixed time circle delay, etc. The total memory usage in Table IV.I is mainly determined by the dimensionality of weight vector, which is in accordance with the dimensionality of window-level FV of the simplified SURF descriptor. The results indicate that the proposed object detection with PLS regress is able to work at up to 242.48 MHz frequency, which is sufficiently high processing speed for real-time applications.

4.5 Conclusion

On one hand, a hardware-friendly coprocessor applied for a general-purpose L1-norm engine is developed and implemented in 65 nm CMOS technology within 0.78 mm^2 core area. Multiple cell sizes (CSs) up to 32×32 pixels are used to define the scaling factor of the corresponding pyramid level. The flexible CS and the unlimited height of input images (≤1024×∞ pixels) make an important contribution to multi-scale and multi-object detection, enabling much smaller computational effort than required in previous research works. Flexible application adjustment is provided by input customization parameters, so that the presented block-based L1-norm circuit can achieve high processing flexibility for different image-cell sizes, cell-based feature descriptors, and image resolutions.

On the other hand, the PLS regression scheme significantly reduces the FV dimensionality as well as the computation cost for object detection stage. Therefore, more corresponding low dimensional reference vectors can be adopted for an NNS classifier with limited storage resources for on-chip or FPGA-based designs. The developed reutilization scheme of memories for intermediate-result storage allows a significant reduction for both storage requirements and core area consumption.

Lower computational cost and pipelined data transmission result in increased efficiency with respect to power consumption and high processing speed. Consequently, the applied prototype architecture demonstrates less memory usage, lower energy consumption, and higher detection robustness for real-time object detection in various mobile applications, than previously possible.

References

[1] X. Zhang, F. An, I. Nakashima, A. Luo, L. Chen, I. Ishii, and H. J. Mattausch, "A hardware-oriented histogram of oriented gradients algorithm and its VLSI implementation," *Japanese Journal of Applied Physics*, vol.56, no.4S, pp. 04CF01, 2017.

[2] A. Luo, F. An, Y. Fujita, X. Zhang, L. Chen, and H. J. Mattausch, "Low-power coprocessor for Haar-like feature extraction with pixel-based pipelined architecture," *Japanese Journal of Applied Physics*, vol.56, no.4S, pp. 04CF06, 2017.

[3] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann, and K. Doll, "FPGA-based real-time pedestrian detection on high-resolution images," In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops*, 2013, pp. 629-635.

[4] A. Suleiman, and V. Sze, "An energy-efficient hardware implementation of HOG-based object detection at 1080HD 60 fps with multi-scale support," *Journal of Signal Processing Systems*, vol.84, no.3, pp.325-337, 2016.

[5] K. Takagi, K. Mizuno, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A sub-100-milliwatt dual-core HOG accelerator VLSI for real-time multiple object detection," In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 2533-2537.

[6] N. Dalal, and B. Triggs, "Histograms of oriented gradients for human detection," In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 886-893.

[7] K. Takagi, K. Tanaka, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A real-time scalable object detection system using low-power HOG accelerator VLSI," *Journal of Signal Processing Systems*, vol.76, no.3, pp.261-274, 2014.

[8] J. Y. Kim, M. Kim, S. Lee, J. Oh, K. Kim, and H. J. Yoo, "A 201.4 GOPS 496 mW real-time multi-object recognition processor with bio-inspired neural perception engine," *IEEE Journal of Solid-State Circuits*, vol.45, no.1, pp.32-45, 2010.

[9] A. Suleiman, Z. Zhang, and V. Sze, "A 58.6 mw 30 frames/s real-time programmable multiobject detection accelerator with deformable parts models on full HD

1920×1080 videos," *IEEE Journal of Solid-State Circuits*, vol.52, no.3, pp.844-855, 2017.

[10] A. Luo, F. An, X. Zhang, L. Chen, and H. J. Mattausch, "Reconfigurable block-based normalization circuit for on-chip object detection," In *Proc. Int. Conf. Solid State Devices and Materials*, 2017, pp. 819-820.

[11] K. Mizuno, K. Takagi, Y. Terachi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A sub-100mW dual-core HOG accelerator VLSI for parallel feature extraction processing for HDTV resolution video," *IEICE Transactions on Electronics*, vol.96, no.4, pp.433-443, 2013.

[12] W. W. Chin, "The partial least squares approach to structural equation modeling," *Modern methods for business research*, vol.295, no.2, 1998, p.295-336.

Chapter 5

Implementation of Different Detection Frameworks

5.1 Introduction

The recent rapid development in data mining has made available a wide variety of algorithms, drawn from the fields of statistics, pattern recognition, machine learning, and databases [1]. Machine learning (ML) techniques have been used to capture normal behavioral patterns and to classify the new behavior as either normal or abnormal [2]. As mentioned in the Introduction in Chapter 1, ML problems can be generally assigned to supervised learning and unsupervised learning problems. Pattern recognition systems can be trained from labeled training data in supervised learning such as object classification, but also can be used to discover previously unknown patterns, as e.g. by clustering algorithms, when no labeled data are available in unsupervised learning.

For vision-based object detection, many appropriate classifiers such as support vector machine (SVM), nearest-neighbor (NN) distance estimation, the AdaBoost classifier [3] and artificial neural networks (ANNs) can be employed as searching engines to find out target objects with favorable properties in the captured visual data. An ideal application in vision-based detection would be to gather sufficient positive and negative images for a user or a program, and then to apply a classification algorithm for training a classifier that can label or predict new unseen input images as belonging to the positive or the negative class as illustrated in Fig.5.1.

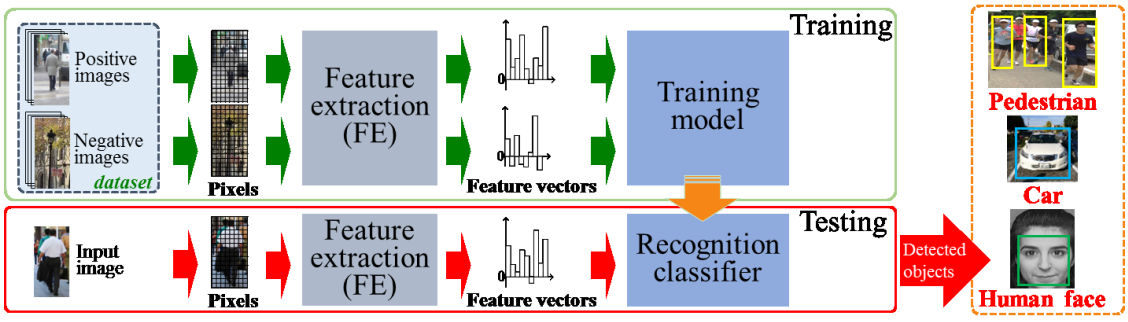


Fig.5. 1 A general flowchart of object detection in supervised learning techniques.

In order to verify the performance of the developed hardware architecture for object recognition and detection, this research employs two demonstration frameworks, which are in part motivated by the tasks of vehicle and pedestrian detection.

On one hand, a resource-efficient object recognition coprocessor, which applies the simplified SURF descriptor and the nearest-neighbor search (NNS) classifier, is proposed for frontal vehicle detection. On the other hand, another popular framework, composed of the histogram of oriented gradient (HOG) descriptor and the SVM classifier [4-5], is reported for pedestrian detection as well. Both demonstration frameworks perform binary categorization of each new input image into either the positive or the negative class.

With the proposal about feature vector (FV) extraction, construction, normalization and dimensionality reduction, described in previous chapters, the recognition and detection process is executed in parallel with the construction of the used multidimensional FV and is therefore completed shortly after FV completion for each scan/search window (SW) of an image frame.

5.2 Nearest neighbor search (NNS) combined with simplified-SURF descriptor

5.2.1 Hardware architecture for NNS classifier

The similarity is defined in terms of a more or less complex similarity function. The smaller the similarity value, the more similar are two objects. As a measurement to express the differences between the input image and reference images, the term ‘distance’ is used as illustrated in Fig.5.2. The reference pattern with minimum distance is referred to as the ‘winner’ and the reference pattern with the next smallest distance is referred to as the nearest-loser.

Fig.5. 2 Object match in an image according to minimum distance value.

Typical query types for NNS classifier are the similarity range query which is specified by a query object and a similarity distance range [0, ε], and the k-nearest neighbor query which is specified by a query object and a number k for the k most similar objects to be retrieved [6]. The k-nearest neighbor query is equivalent to a corresponding similarity range query. Figure 5.3 shows the flowchart of the object detection employed in this research. The first framework for object detection is combined with the simplified SURF feature descriptor and the NNS classifier.

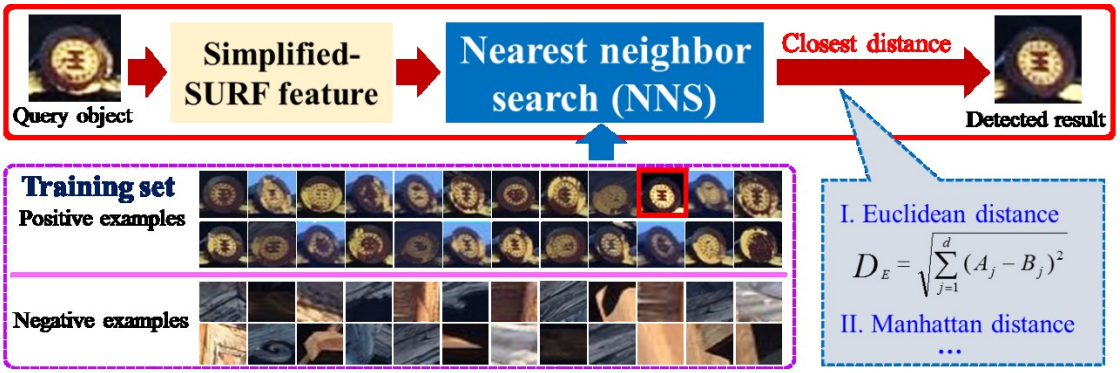


Fig.5. 3 The flowchart of the object detection system employing the NNS classifier.

For the window-based object classification, the FV (i.e., the vector ***IN***) of one scan window (SW) in the test image can be represented in Eq.5.1, while the FV of the i^{th} reference (i.e., the vector $\boldsymbol{REF_i}$) is represented by Eq.5.2.

$$\boldsymbol{IN} = \{IN_1, IN_2, \cdots, IN_W\} \tag{5.1}$$

$$\boldsymbol{REF_i} = \{REF_{i1}, REF_{i2}, \cdots, REF_{iW}\} \tag{5.2}$$

There are several popular types of operators, such as Hamming distance (HD), Manhattan distance (MD) and Euclidean distance (ED) in information theory to measure the distance from the testing image to the reference image. Architectures for fully-parallel winner-search according to the Hamming-distance [7] and the Manhattan distance [8] have been proposed based on mixed digital-analog circuits.

The three above distance concepts can be summarized by a general formula called MinKowsky equation given by Eq. 5.3.

$$D_j = \sum_{i=1}^{W} \omega_i \cdot \left[(IN_i - REF_{ij})^P \right]^{1/P} \tag{5.3}$$

Where if, P=1, D_j represents the Manhattan Distance, P=2, D_j represents the Euclidean distance, P=1 and $\boldsymbol{IN}_i$, $\boldsymbol{REF}_i \in \{0,1\}$, D_j represents the Hamming distance. And ω_i is a weighting factor which is set to '1' when computing Euclidean/Hamming distances. More different metrics proposed by different communities, including Chebychev, Camberra, Mahalanobis [9], and Kullback-Leibler distance [9]. In many practical applications, particularly in the fields of image recognition and authentication, the ED, which is the most natural and appropriate distance measure [10], is known to give better results than HD or MD.

In this research, after the overlapping scan window (OSW) determination and construction, the NNS classifier is continued by partial squared Euclidean distance (PSED) calculation between the newly constructed window-level FV for the input image and the reference images. In the perspective of overall system processing, the NNS classification is operated simultaneously with the window-FV construction with multiple paralleled SWs. For each SW in which the cell is located, the cell position corresponds to an address within the reference FV. Meanwhile, the cell-based reference vectors for NNS are organized in the same way as the extraction sequence of the cell-FV.

Thus the implementation of NNS can be additionally abided by the "regular rule of reusing times" (RRRT) for the SW-FV construction which is discussed in Chapter 3.

Figure 5.4 further illustrates the main processing procedure with the NNS classifier to search the final recognition winner among the inputted reference data for the target testing image. Pixel processing for extracting FV for images is the fundamental requirement for feature-based image representation. Since the testing image is represented by cell-based FVs construction strategy, the corresponding FVs of the reference data should be organized in the same way as the testing image FVs, so as to reduce the computational cost.

As the window sweeps within a block stride (i.e. 2 cells), the number of paralleled processing windows are dynamically changes. For this purpose, a partial-storage concept and the parallel-pipelined computation architecture applied in [11-14] are also employed in this research. Specifically, the 1680-dimensional simplified-SURF reference FVs are segmented into the four local cell-FV parts for e.g. car recognition.

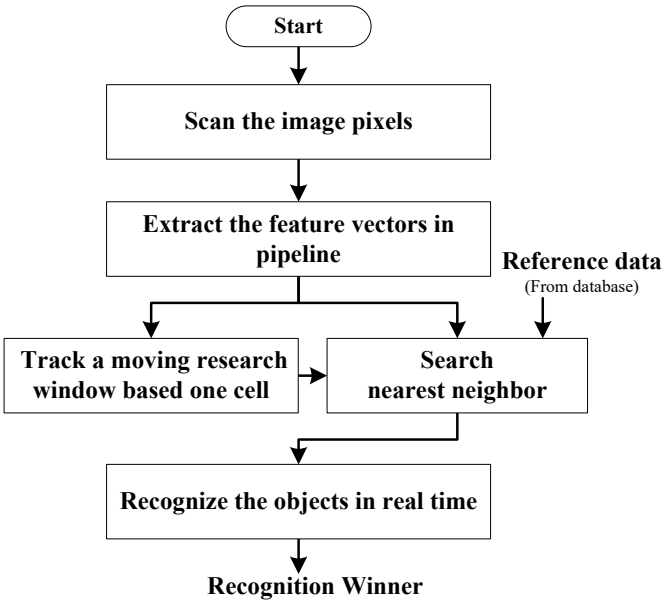


Fig.5. 4 Process flow for the proposed object recognition based on the NNS classifier.

The parallelized PSED calculation is employed with respect to the FVs of all OSWs, under consideration of the reuse time of the current cell C[c,r]. Generally, each of the simultaneously processed OSWs is in a different stage of its FV construction and PSED calculation with respect to the reference FVs.

According to the cell position in each OSW (i.e., W_{index}, where the index is defined in Eq.3.6), the corresponding components of the cell-based reference FVs are invoked to execute the next step of the PSED computation.

Until the SED results between all OSWs and reference FVs are determined, the cumulative PSED-storage unit in Fig. 5.5 has to store $N \times W_{he}/B_s$ intermediate PSED results for each reference FV. Here N is defined in Eq.3.3, W_{he} is the height of the designated SW, and B_s is the square-block size, which is 16 pixels in this research. The WA of the memory for intermediate storage of PSED values, which are related to the order number of the current window and the RRRT value, can be calculated by the hardware circuit illustrated in Fig. 3.12.

When cell row $W_{he}/8$ of the image is processed, NNS for the first row of SWs finish sequentially. Depending on the block size B_s, NNS for one SW will complete the supply of the local cell-FV of every $(B_s/8)^{th}$ cell.

Afterwards, NNS for the next row of SWs will complete, when an 8×8-pixel-cell row $(W_{he}+B_s)/8$ is processed. During the completion of NNS for a SW, the sequentially determined final SED to each reference FV is latched in Buffer 2 of Fig. 5.5, and is then compared to the intermediate minimum stored in Buffer 3.

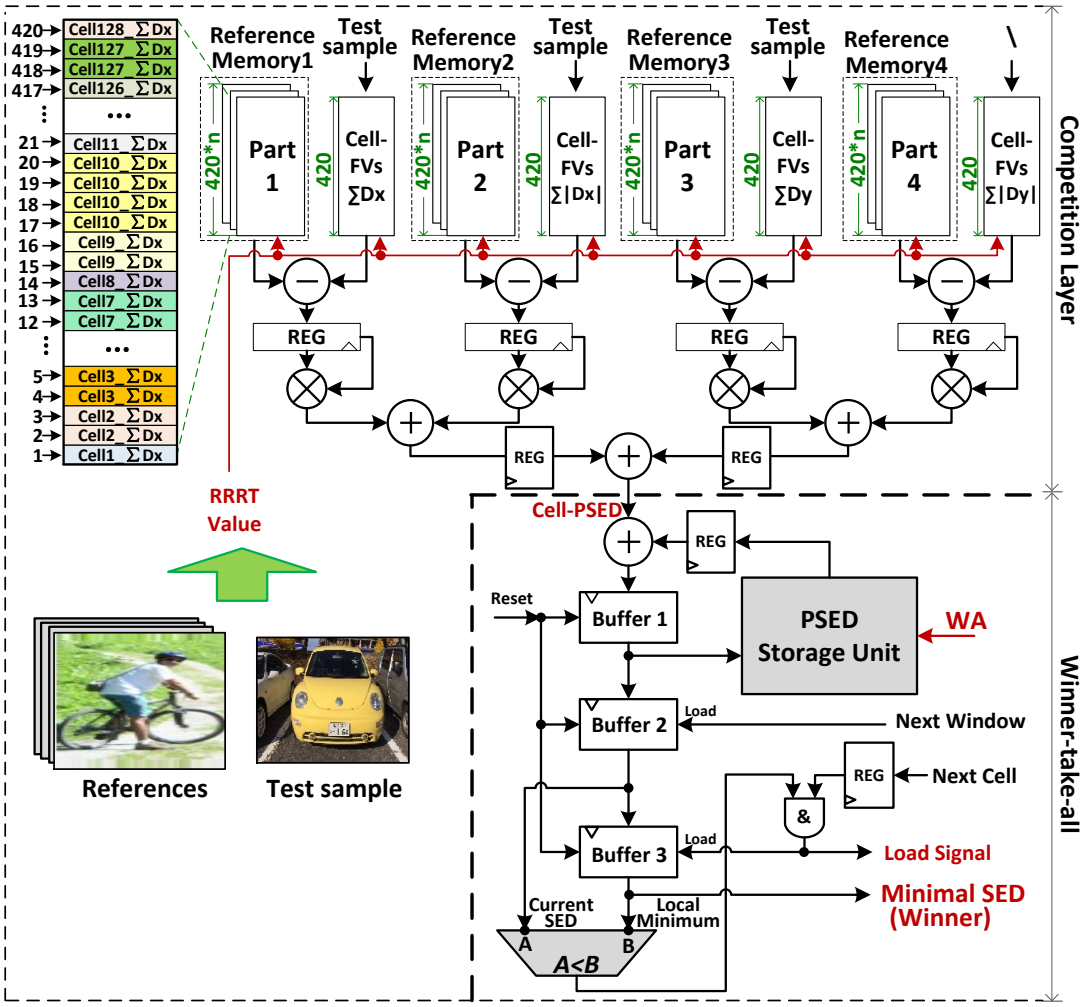


Fig.5. 5 Cell-based NNS architecture for parallel minimal SED search among multiple references.

If the SED in Buffer 2 is smaller than that in Buffer 3, it will be stored in Buffer 3 as the new intermediate minimum. After SED comparison for the last reference FV, NNS is completed, the global minimum is saved in Buffer 3, and the corresponding reference FV determines the classification result for this SW.

Important is, that each cell is scanned only once and then scanning results are reused in all related SWs. Additionally, there is no need for tracking SW-internal overlapped blocks since the NNS recognition proceeds based on cells. Consequently, the proposed parallel SW (PSW) algorithm effectively reduces the computational complexity and leads to high performance in hardware implementation.

Multi-scale and multi-object detection has become a tendency for intelligent machine-vision applications [15-17]. However, the multi-scale image pyramid results in large data expansion and higher computational complexity, necessitating fast hardware implementations to enable real-time processing.

Note that the proposed PSW concept also allows enlarging or reducing the scale of target objects in images, to match SW and target-object sizes. The PSW concept is not image-size limited and widely adjustable to pixel-transfer speed and image size of different image sensors.

Since the resolution of processed images with the customized feature-extracting circuits, such as the designated circuit for simplified SURF descriptor shown in Fig.2.11, can be dynamically changed, serial processing of a complete image pyramid on the same coprocessor hardware with a fixed SW size becomes possible. In this way, high accuracy of object recognition in real-world images with varying object sizes can be maintained.

Figure 5.6 shows an instance for a pyramid with *m* image layers in different image scales, i.e., S, S/4, S/16, S/64, etc., where S is the largest original-image scale in the image pyramid. In this case, the required operation time with *m* serial simplified-SURF descriptors for *m* image scales takes $t=t_b+t_b/4+t_b/16+t_b/64+\ldots+t_b/4^{m-1}+m\cdot t_{de}$, resulting in $t<2t_b+m\cdot t_{de}$, when the image scale in the pyramid decreases exponentially by an index "4".

Here t_b represents the operation time for the largest original-image layer and t_{de} (much smaller than t_b) represents the latency during feature extraction and object recognition.

On the other hand, the required operation time t_b+t_{de} with multiple parallel recognition engines is only limited by the size of the largest original-image layer.

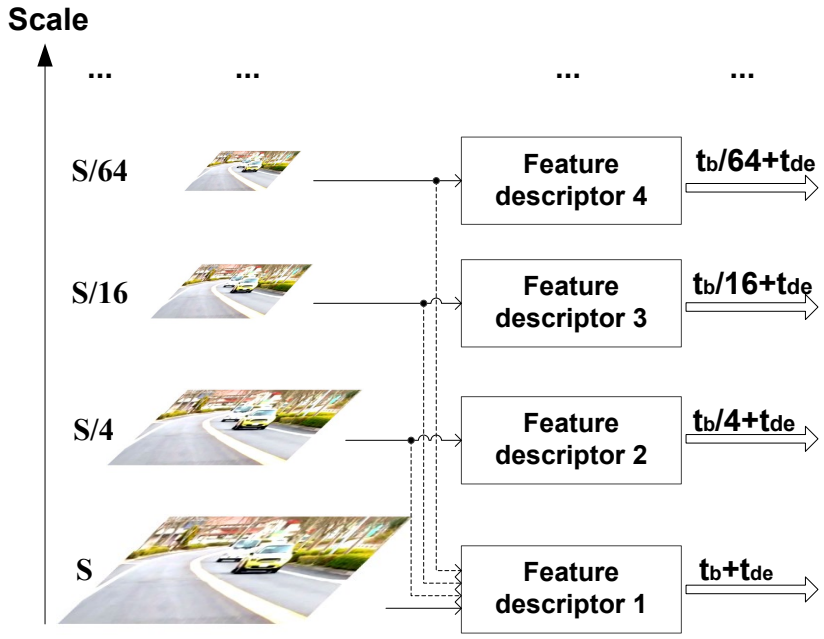


Fig.5. 6 Diagram of the proposed simplified-SURF descriptor application for multiple image layers with corresponding Haar-like wavelets.

Consequently, the serial pyramid-processing scheme with a single coprocessor is feasible for real-time applications, if the processing time of the original image is about 2 times faster than what is required for real-time processing. In conclusion, the proposed architecture and the designed prototype coprocessor can also be applied for real-time image-pyramid processing, when an appropriate original-image size is selected.

For applications recognizing objects of different sizes with the prototype-coprocessor implementation, the value of t_b determines whether a serial scheme with a single coprocessor is sufficient or a parallel scheme with multiple coprocessors is required.

5.2.2 ASIC implementation in 65nm CMOS

The FV-based recognition coprocessor, embedding the cell-based simplified-SURF FV-extraction unit and the parallel SW recognition engine with a fixed 64×128-pixel window, is prototyped in 65nm Silicon on thin BOX (SOTB) CMOS technology with 1.26 mm^2 (1.4 mm × 0.9 mm) core area, as shown in Fig.5.7.

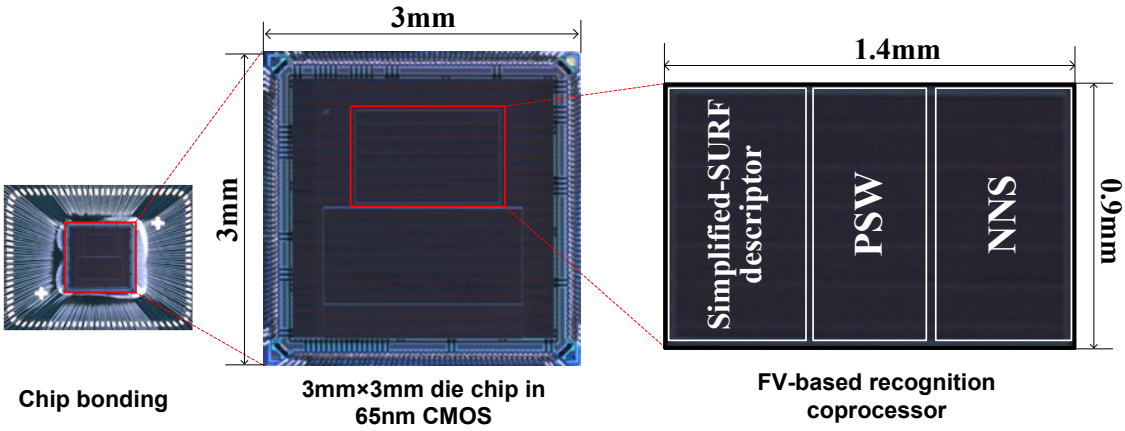


Fig.5. 7 Chip microphotograph of the FV-based recognition coprocessor, fabricated in 65 nm CMOS technology within 1.26 mm^2 (1.4 mm × 0.9 mm) core area.

In the prototype coprocessor implementation, the total on-chip SRAM is 208 kbits (i.e. 26 kB). Specifically, 64 kbits of this space are shared for calculating the simplified-SURF descriptor FVs, 32 kbits of SRAM serve as FIFO for cell-FV buffering and 64kbits on-chip memory are designated for storing 1680-dimensional references for verification of the proposed architecture, owing to the limitation of chip area. Another 16kbits on-chip memory are used for marking the newly processed addresses during NNS classification according to the current-cell location.

Additionally, the memory space for references can be overwritten from outside to

meet the needs of practical applications. The PSED storage unit consumes 16kbits memory space for intermediate PSED-computation-result storage until the current image cell. The structure overview of the on-chip memory is summarized in Table V.I.

TABLE V. I
Structure Overview of On-Chip SRAM Components of the Developed Prototype Coprocessor

Component of on-chip SRAM		Capacity	Bit width	Type
FV-extraction storage	D_x	16 kbits	16-bit	Dual-port SRAM
		16 kbits	32-bit	
	D_y	16 kbits	16-bit	
		16 kbits	32-bit	
	FV Buffer	32 kbits	64-bit	FIFO
PSW storage	CPLUT	16 kbits	16-bit	Dual-port SRAM
	WLUT	16 kbits	16-bit	
NNS storage	Temporary SED	16 kbits	32-bit	Single-port SRAM
	Reference	64 kbits	32-bit	

In comparison to previous works, less memory is consumed, even when multiple coprocessors are operated in parallel for handling scaled images because the pixel- and cell-based FV-extraction scheme makes pre-storage in buffers for raw or integral images unnecessary. Additionally, obsolete data in the storage memories are overwritten as soon as the new cell or window information becomes available. Only a small memory amount, determined by the maximum image width, is consumed for the enabling the flexibility of processing different image sizes.

The memory requirement in [18] is clearly less than in other previous works [19-21] and also this work because only the FV-extraction procedure is executed in [18]. In contrast to the previous works, the 208kbits memory in the proposed simplified-SURF coprocessor of this work is suitable for additionally enabling flexible image sizes up to $1024\times \infty$ pixels.

Practical 8-bit pixels from XGA frames and four 1680-dimensional reference vectors are inputted to the coprocessor for verifying the recognition result, utilizing a MU300-EM platform with peripherals as shown in Fig. 5.8.

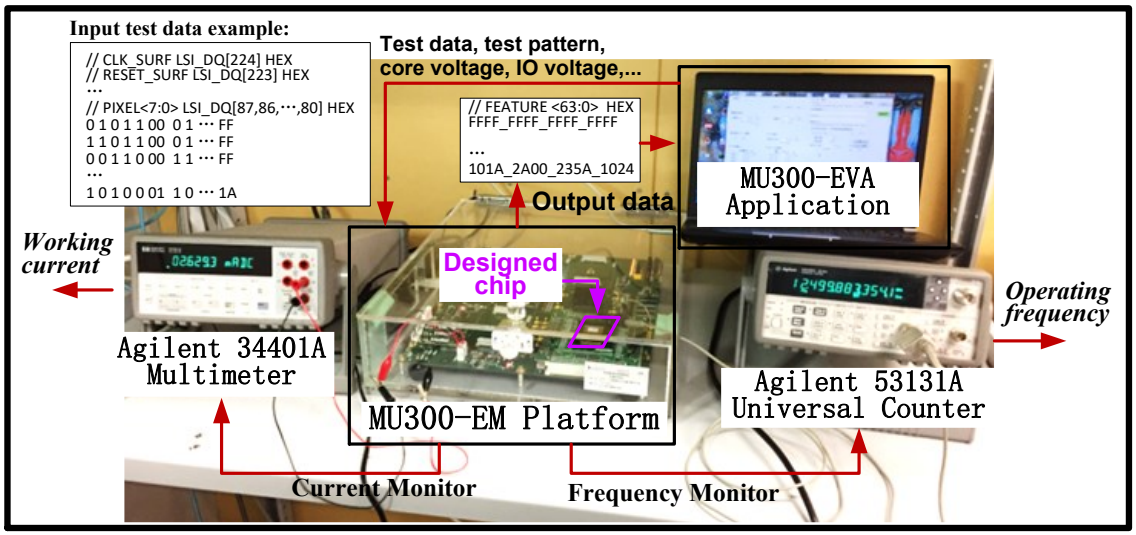


Fig.5. 8 Measurement system with MU300-EM platform for the designated chip.

The power consumption measured by the MU300-EM platform is proportional to the core chip voltage and is further related to chip area, storage capacity and operating frequency. Approximately 13.38mW is dissipated when the fabricated chip is operating at a lower core voltage of 0.5 V and 200 MHz as illustrated in Fig.5.9. The energy efficiency of the fabricated prototype for the FV-based recognition coprocessor is measured as 910 μJ per VGA frame and 1013 μJ per XGA frame when operating at 200 MHz peak frequency with 1 V supply voltage.

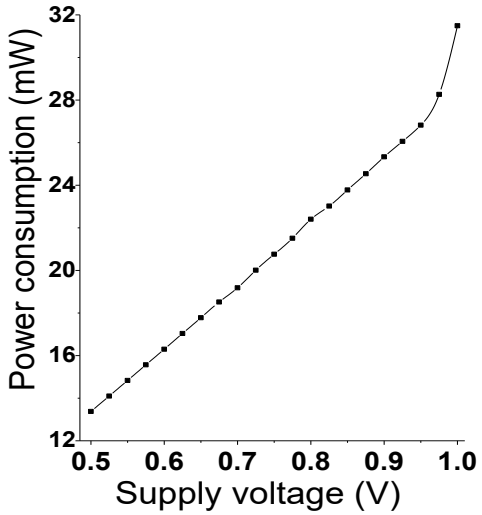


Fig.5. 9 Power consumption at different supply voltages of the designated 65 nm chip with NNS classifier and simplified SURF feature descriptor, operating at 200MHz and 1.0V core voltage.

As expected and illustrated quantitatively in Fig.5.10, the frame rate reduces and the energy consumption per frame increases with higher image resolution, resulting in 75.3 QVGA (320×240 pixels), 34.6 VGA and 31.1 XGA frames processed per second, respectively. The energy efficiency in this work is significantly improved in comparison to some of the previous works [19, 21], but is still larger when compared to other previous works [18, 20]. More advanced CMOS technology, lower operating frequency and lower supply voltage are among the reasons explaining the smaller power consumption in [18] and [19].

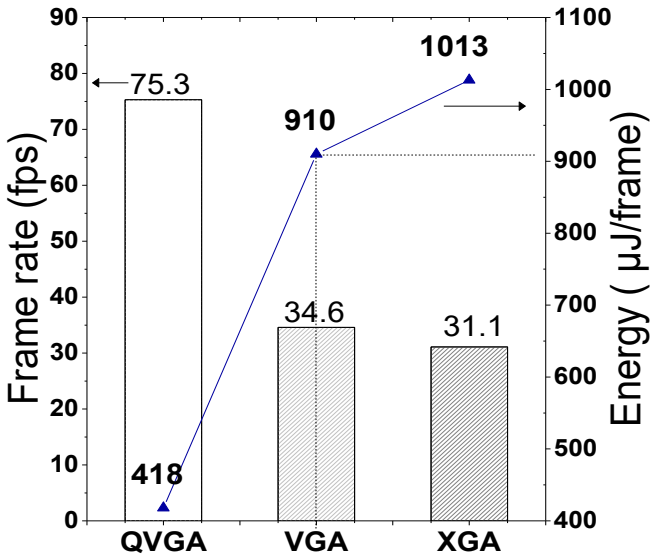


Fig.5. 10 Frame rate and energy efficiency with different image resolutions operating at 200MHz and 1.0 V core voltage.

In particular, the proposed pipelined circuitry in this research for local cell-feature extraction inside the fabricated prototype consumes only 48.6 µJ per VGA frame and 123.8 µJ per XGA frame. The frame rate in [20] is higher than for the other works due to the restricted-ROI (region of interest)-based solution, which increases the processing speed by abnegating a lot of raw-image information. Further comparison to previous related works is summarized in Table VI.II.

In particular, the word precisions for each memory word ensures a lossless fixed-point processing (no rounding required) for data computing with minimization of hardware cost. Since no floating-point calculation is involved in the data processing through the entire hardware architecture, adequate word precisions ensure a data-lossless processing and gain reasonable classification accuracy.

TABLE V. II
Comparison Results with Previous Works

	JSSC[18]	VLSI-C[19]	VLSI-C[20]	ISSCC[21]	This Work
Technology	28nm	40nm	40nm	90nm	65nm
Image resolution (pixels× pixels)	fixed 640×480	fixed 1280×720	fixed 1280×960	fixed 160×120	variable up to 1024×∞
Design function	Feature extraction	Object recognition	Object recognition	Object recognition	Object recognition
Feature type	SURF hardware	PCA hardware	Haar-like hardware	Haar-like hardware	Simplified SURF
Extraction scope	Entire frame	Entire frame	ROI only	Entire frame	Entire frame
Core area (mm^2)	2.22	5.86	9.3	28	1.26
Core / IO voltage(V)	0.47 /-	0.6 /-	0.9 / 2.5	1.2 /2.5	1.0 /3.3
Frequency (MHz)	27	100	220	200	200
Maximal frame rate (frame per second)	30	5.5	300	294	75.3 (QVGA) 34.6 (VGA) 31.1 (XGA)
Maximal accuracy	-	93%	97.4%	-	98.78% (for car) 93.90% (for pedestrian)
Power consumption (mW)	2.8	23	69	1214	31.49
Energy efficiency (mJ/frame)*	0.093	4.18	0.23	4.129	0.418 (QVGA) 0.910 (VGA) 1.013 (XGA)
Memory usage (kb)	>56 (FE* only)	3936 (FE:192; Recognition:3744)	202.4 (LDP*: 190; KTP*: 12.4)	1192 (ISPS*: 800; FSPS*: 392)	208 (FE: 96; PSW: 32; NNS: 80)

**Energy efficiency = Power consumption / Maximal frame rate*; **FE: Feature Extraction;*
**LDP: Learning-based Detection Processor; *KTP: Knowledge-based Tracking Processor;*
**ISPS: Image Stream Processing System; *FSPS: Feature Stream Processing System.*

Furthermore, data (e.g. FVs) normalization is not involved throughout the entire recognition procedure in this chip design, avoiding division operations. The fixed-point-number operation is adopted by left or right shifting the bits of one word when there is a multiple operating relationship between two parameters. Variable counters, such as the ones in the pixel pipeline controller, are also applied for multiple transformations such as node reset when detecting the image edges. All factors mentioned above assure that the recognition accuracy of the fabricated coprocessor chip becomes exactly the same as the simulated accuracy results by software-implementation of the proposed architecture.

Furthermore, the same coprocessor hardware can be applied to the serial processing of complete image pyramids (see also Fig. 5.6) to address the issue of changing object sizes with a single SW size, since the resolution of processed images with our coprocessor can be dynamically changed. The FV-extraction stage of the fabricated coprocessor needs about 0.384 ms, 1.536 ms and 3.9 ms for QVGA, VGA and XGA frame sizes, respectively, since the feature extraction speed relies only on the pixel-transfer frequency from the image sensor. These FV-extraction times are more than two times faster than what is required for real-time processing so that complete pyramid FV-extraction for videos with these frame sizes is realistic in real time.

For the entire recognition procedure including NNS, the processing time at 200 MHz frequency with a 64×128-pixel SW becomes 13.28 ms, 28.9 ms and 32.13 ms in case of

QVGA, VGA and XGA frame sizes, respectively. Please note that due to the image-cell FV reuse concept, the processing-time increase for larger image sizes in much smaller than the pixel-number increase. As a consequence of above processing-time results, multiple coprocessors in parallel are needed to achieve real-time processing for the image pyramid of XGA frame size. However, in the case of QVGA frame size, a serial scheme with the same coprocessor hardware is feasible for real-time variable-size object recognition. On the other hand, smaller size SWs consume less processing time for each frame because of the greatly reduced FV-dimensionality to match the target objects. Further, the smaller number of SWs overlapping a given cell results in less repetitive usage of the cell-based FVs as well. Therefore, the adjustment of the SW size can also be applied to reduce image-pyramid-processing time by a considerable degree for achieving real-time recognition.

In conclusion, considering the processing time for feature extraction or the entire object recognition procedure, a serial single-core solution or a parallel multi-core solution (e.g. multiple feature extraction engines or entire recognition engines) can be applied for image-pyramid processing.

Consequently, a highly energy-efficient hardware architecture for object recognition has been developed, which can be applied to extensive visual applications based on cell-based FVs. The proposed feature extraction architecture is particularly suitable for mobile applications, due to the low power dissipation and the low hardware resource requirements. In addition, the image-size flexibility of the proposed architecture can be controlled by the implemented bit-alterable counters for real-time processing of higher-resolution images than VGA, e.g., XGA (1024×768 pixels).

5.2.3 A accuracy estimation for detection of different objects

The reported hardware architecture is capable of handling different applications for object detection and recognition. To verify the performance of the proposed hardware algorithm in extensive visual applications, different objects, including the front car, lateral car, pedestrian, are served as positive samples shown in Fig.5.11 (a), (b) and (c) respectively.

Fig.5. 11 Positive examples for object detection. The front car examples (a) are collected from the surroundings of Hiroshima University (HU), the lateral car examples (b) are selected from the car detection dataset UIUC, while the pedestrian examples (c) are selected from INRIA dataset.

For the case of vehicle detection, an NNS classifier employing 1680-dimensional Haar-like FVs is implemented as an example to evaluate the efficiency of the proposed algorithm by software-based simulation. A set of positive samples (i.e., lateral cars) from the car detection dataset UIUC [22-23] and negative samples (i.e., non-cars) from INRIA dataset [24] are selected for verification as shown in Fig.5.12 in this work.

Fig.5. 12 Negative examples selected from INRIA dataset for different object detections.

A reasonable quantity of samples plays a significant role in establishing sufficient diversity to correctly detect the input data from the testing dataset [25]. Therefore, 550 positive samples from the UIUC dataset and 12180 negative samples obtained by cropping non-car images from the INRIA dataset are resized to 128×64 pixels (16×8 cells) for training. Furthermore, 199 different positive samples from the UIUC dataset and 1812 different negative samples cropped from the INRIA dataset are used for the image set for testing.

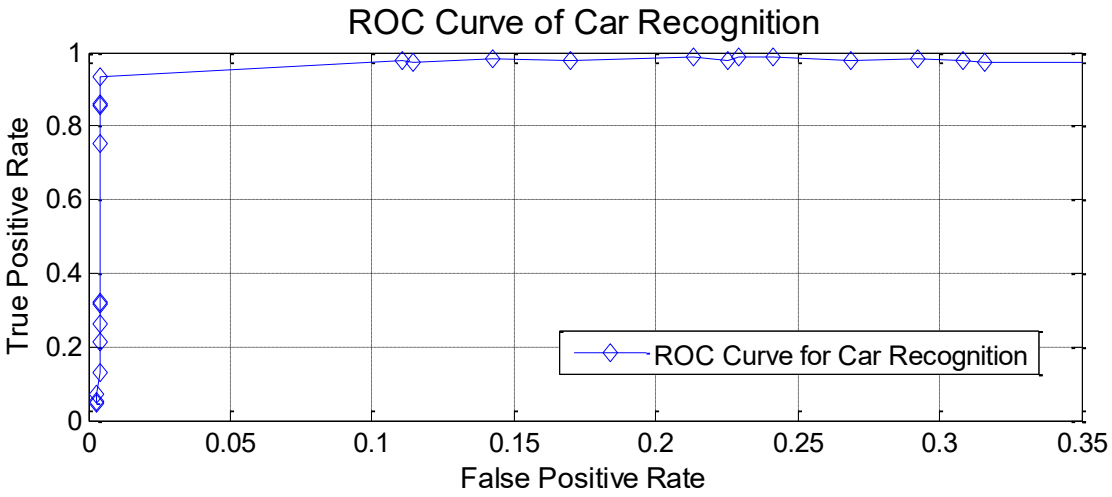


Fig.5. 13 ROC curve that indicates the detection accuracy of the fabricated coprocessor for the case of the lateral car detection application.

For evaluating the recognition performance of our proposed algorithm, SEDs between the extracted FVs of the SWs and the reference vectors are calculated for NNS classification. The receiver operating characteristic (ROC) curve in Fig. 5.13 is evaluated with the image set for testing. The ROC curve indicates that the proposed classifier's detection accuracy increases with an increased quantity of testing samples.

In the case of vehicle detection, the integral image strategy of previous software solutions [26, 27] attained an accuracy range from 85.7 to 100%. In our verification, we have achieved a maximal true positive rate (TPR) of 97.49%, a specificity of 99.67%, and a comparably good accuracy (ACC) of 99.45%.

Furthermore, the front-car recognition and pedestrian recognition are also evaluated in this research. Specifically, on one hand, the front cars images collected from the surroundings of Hiroshima University (HU) are further applied for front car-recognition evaluation.

On one hand, the scenic images from the INRIA are cropped to 12180 negative training images and 4530 negative testing images for evaluating both pedestrian and car recognition, as shown in Fig.5.12. The negative images stand for non-cars or non-pedestrian in this solution.

On the other hand, 2416 positive training images and 1126 positive testing images (i.e. containing humans) from the posture-complex INRIA dataset are employed for pedestrian-recognition evaluation. By contrast, 1225 positive training images and 556 positive testing images (i.e. containing front cars) collected from the surroundings of HU are applied for front-car recognition evaluation.

The classification-performance comparisons with the original SURF [28] are illustrated in Fig.5.14. The proposed simplified-SURF coprocessor with flexible image sizes is found to achieve a maximum of 91.03% TPR and 96.12% TNR for pedestrian recognition with a 64×128-pixel SW in the INRIA dataset. For car recognition with a 128×64-pixel SW, the proposed coprocessor achieves a maximum of 97.30% TPR and 99.34% TNR without normalization, which outperforms the previous normalized SURF descriptor with feature-based NNS classification, as illustrated in Fig.5.14 (a) and (b). The precision v.s. recall graph is shown in Fig.5.14 (c), confirming the accuracy improvement as the recall becomes higher. The average precision (AP) of the simplified-SURF coprocessor for pedestrian recognition is 82.84% based on 1680-dimension FVs, which is lower than with the original SURF (88.89%). But significant efficiency is verified for car recognition, where the reported simplified-SURF coprocessor achieves 98.67% AP.

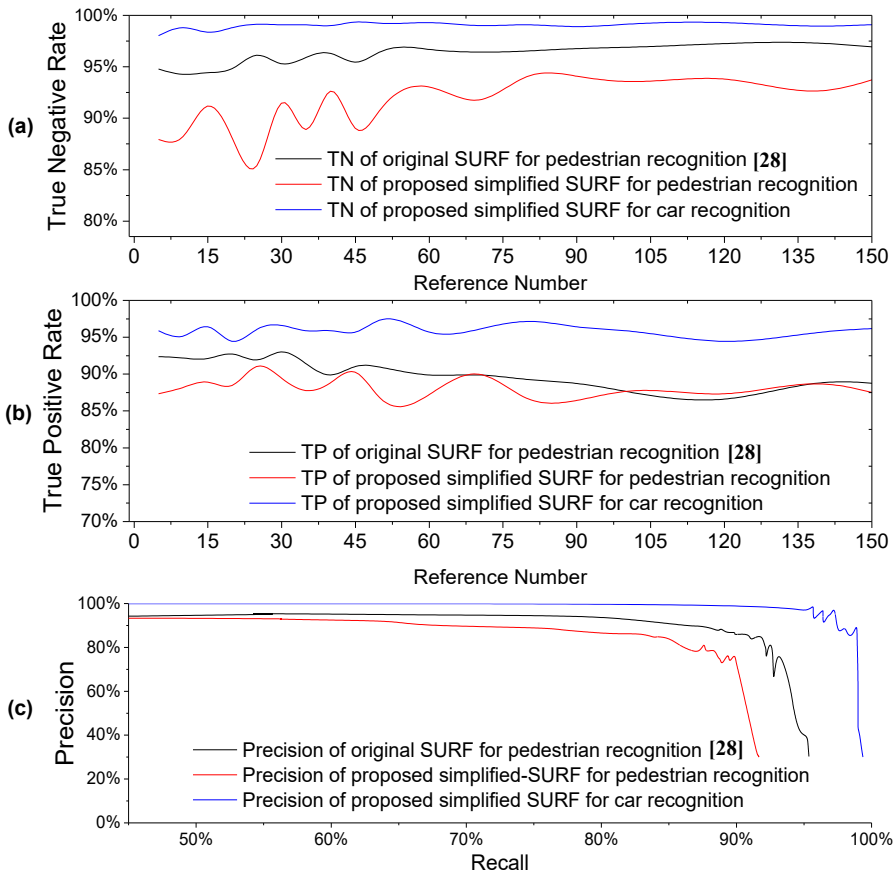


Fig.5. 14 Fixed-point software verification of recognition TPR (a), TNR (b) and Precision-Recall (c) performance of the proposed simplified-SURF descriptor compared to the original SURF descriptor [28] with feature-based NNS classification.

The accuracy comparison, as further illustrated in Fig.5.15, shows that the simplification of feature calculation introduces about 6% accuracy deterioration for pedestrian recognition in most cases, while the hardware resources and performance benefits are far more significant. The reported algorithm for different object detection and recognition achieves sufficient performances in resource requirement, power consumption, calculation cost, and flexibility, with a trade-off of a reasonable accuracy.

The recognition accuracies with different SW-sizes in Fig.5.15 also indicates that the proposed coprocessor is more efficient for car recognition than for pedestrian recognition.

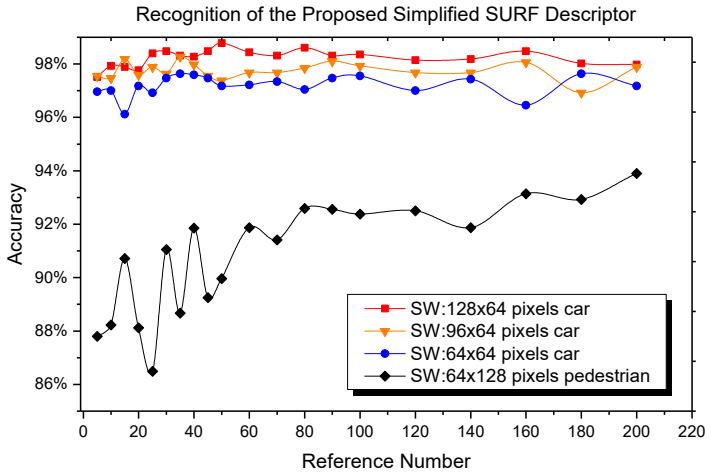


Fig.5. 15 Recognition accuracies of the proposed simplified-SURF FV-based NNS classification with different SW-sizes for car and pedestrian recognition applications.

In conclusion, the recognition accuracy enhances with increased window size. The efficient computational and flexible hardware architecture can enable the proposed object-recognition coprocessor to perfectly meet outdoor mobile-applications requirements without significant degradation of classification accuracy.

5.3 Support vector machine classifier carrying the HOG descriptor

5.3.1 Hardware architecture for SVM classifier

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, the SVM algorithm outputs an optimal hyperplane which categorizes new examples according to the given labeled training data

(supervised learning). In order to intuitively define a criterion to estimate the worth of the hyperplanes, the operation of the SVM algorithm is finding the hyperplane that gives the largest minimum distance to the training examples.

Many recent hardware implementations, such as [4] and [5], have demonstrated that the support vector machine (SVM) classifier [29] carrying the HOG descriptor leads to one of the most accurate and therefore presently most popular detection frameworks. Basing on the L1-norm scheme as mentioned in section 4.2, this research further developed an object detection and recognition circuit employing cell-based HOG descriptor and linear SVM classifier as illustrated in Fig.5.16.

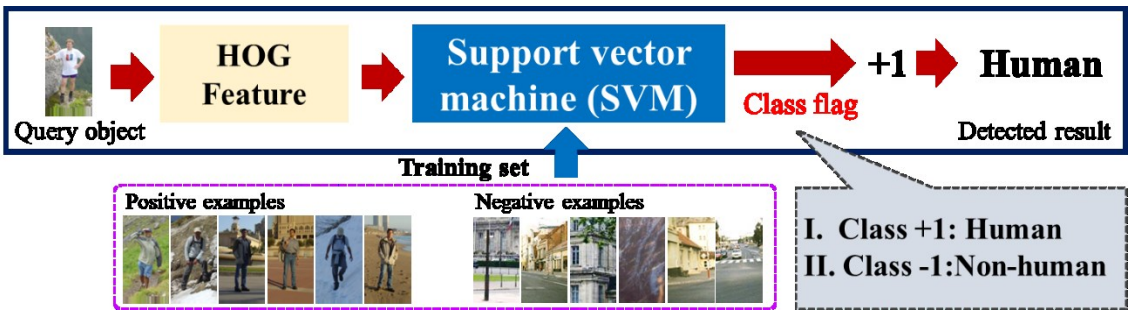


Fig.5. 16 The flowchart of object detection which combines with the HOG feature descriptor and the SVM classifier.

There are various cases about SVM models, so that different SVM mathematical formulations can be found. This research applies a linear case for supervised binary classification problem as illustrated in Fig.5.17, supposing that the training dataset consists of M vectors from the d-dimensional feature space $\mathbf{x}_i \in R^d$. Assume that the two classes are linearly separable in the linear separate SVM.

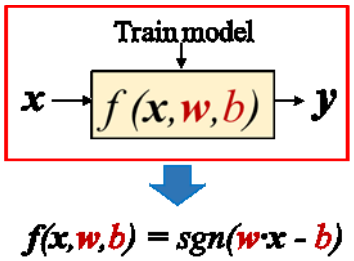


Fig.5. 17 The linear SVM model employed in this research.

It is possible to find at least one hyperplane (i.e., linear surface) defined by a d-dimensional vector $\mathbf{w} \in R^d$ (normal to the hyperplane) and a constant bias $b \in R$ that can separate the two classes without errors in linearly separable SVM approach.

The decision rule of the linear separate SVM can be based on the function $sgn[f(x)]$,

where the sign function *sgn*(*x*) is defined as Eq.5.4.

$$sgn(x) = \begin{cases} -1, & if\ x < 0 \\ 0\,, & if\ x = 0 \\ +1, & if\ x > 0 \end{cases} \tag{5.4}$$

And *f(x)* is the discriminant function associated with the hyperplane and defined as Eq.5.5.

$$f(\boldsymbol{x}, \boldsymbol{w}, b) = \boldsymbol{w} \cdot \boldsymbol{x} - b \tag{5.5}$$

In order to find such a hyperplane, a target $y_i \in \{-1,+1\}$ is associated to each vector $\mathbf{x_i}$ and one should estimate and so that the following product s is computed to be a positive value as illustrated in Eq.5.6.

$$y_i \cdot (\boldsymbol{w} \cdot \boldsymbol{x_i} - b) > 0, \;\; i = 1,2, \dots, M \tag{5.6}$$

The linear SVM approach consists in finding the optimal hyperplane that maximizes the distance between the closest training image sample and the separating hyperplane as shown in Fig.5.18. The geometrical margin between the two classes (i.e., +1 and -1) is given by the quantity '2/||**w**||'. The concept of margin, which is a measure of its generalization capability, is central in the linear SVM approach. The larger the margin, the higher the expected generalization [30].

The classical linearly constrained optimization SVM problem can be translated into a discriminant function using a Lagrangian formulation [31], so that the linear SVM approach becomes an equation depending both on the Lagrange multipliers (related to the parameters y_i, **w**, and $\mathbf{x_i}$) and on the training samples associated with the optimal hyperplane.

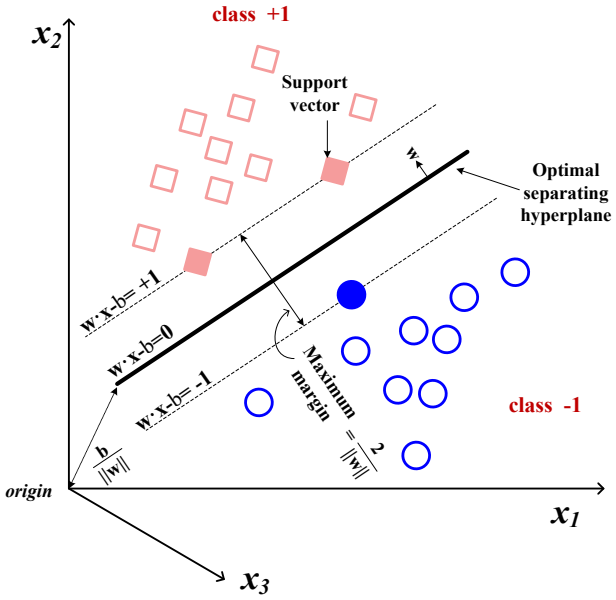


Fig.5. 18 Optimal separating hyperplane in the linear SVM approach.

The Lagrange multipliers effectively weight each training sample according to its importance in determining the discriminant function. The training samples associated with nonzero weights are called ‘support vectors’.

In order to implement the linear SVM approach in hardware circuit, this research developed an offline training model with both negative images and positive images for generating d-dimensional vector **w** (normal to the hyperplane) and a constant bias *b* for pedestrian detection. Since the 3780-dimensional normalized HOG descriptor is employed for feature representation in this research, the dimensionality d of the vector **w** is equal to 3780 for one SW. The trained weight vector **w** is stored by the cell in nine divided SRAMs. Since the cell-based FV for HOG descriptor has *nine* local dimensional components, there are *nine* block storages from SRAM1 to SRAM9, which are distributed to store separate one dimensional local component of each cell respectively as illustrated in Fig.5.19.

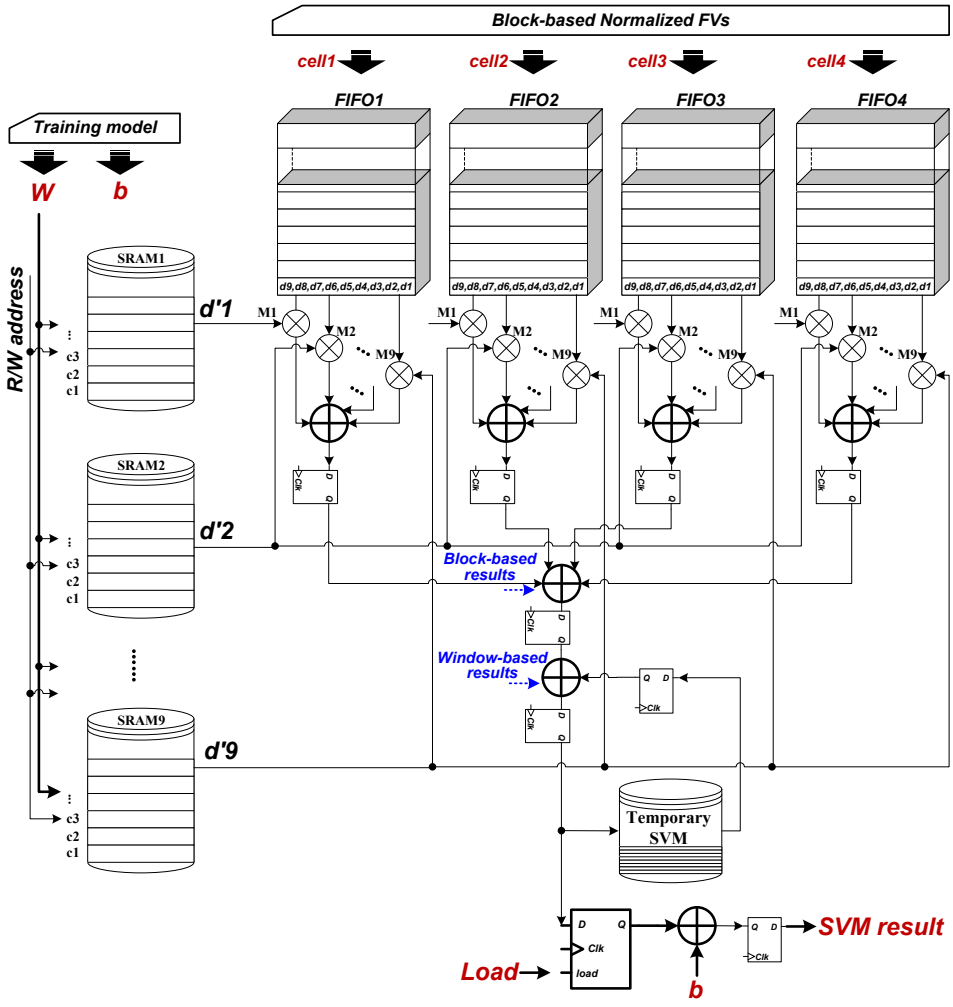


Fig.5. 19 The proposed hardware circuit for calculating the linear SVM value, based on trained weight vector **w** and constant value *b*.

Since the block-based L1-norm is operated in a 2×2-cell region, the normalized FVs are separated to 4 independent cell-FVs and buffered in 4 corresponding FIFOs for the consecutive calculation of SVM.

An additional state machine is applied to control the read/write (R/W) address for each SRAM and enable signals for FIFO so that the correct weight component of each dimension in each cell can be multiplied with the corresponding FV of each dimension in each cell from the testing image. There should be *nine* multipliers for cell-FVs outputted from each FIFO so that four cell-FVs (i.e., block-based processing) can be handled in parallel for speeding up the processing procedure. The products of each cell-based components are accumulated by essential levels of addition. The intermediate results for accumulating 3780-dimensional FV of each SW are stored in a memory until the completion of window-level computation. The accumulated results ($\boldsymbol{w} \cdot \boldsymbol{x}$) is applied to

subtract the trained bias *b* according to Eq.5.5 to generate the final SVM result for each SW, which can be applied for determining the recognition result ('+1' as positive or '-1' as negative).

5.3.2 ASIC implementation in 65nm CMOS

The developed parallel architecture illustrated in the Fig.5.19 exploits simultaneous processing of multiple SWs operates with normalized block-FVs and takes advantage of the parallel processing of blocks in related SWs, while the conventional hardware designs apply an integral-image scheme which results in significant computational cost and frame latency. The reconfigurable L1-norm-circuit architecture of block-based FVs is applied for the object-detection hardware and makes a favorable tradeoff between computing complexity and detection accuracy.

A proof-of-concept prototype chip for flexible feature space construction and pedestrian detection is fabricated in 65 nm CMOS, as shown in the photomicrograph of Fig. 5.20. The 3780-dimensional cell-based HOG descriptor is applied for reconfigurable FV extraction.

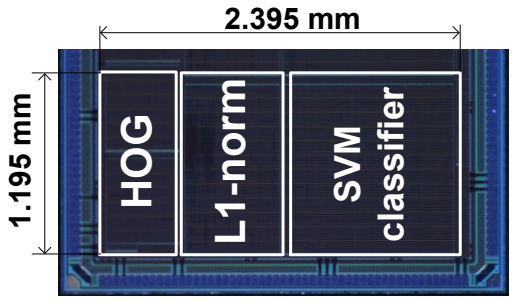


Fig.5. 20 Micrograph of the prototype chip for object recognition in 65 nm CMOS technology, which applies HOG descriptor, L1-norm scheme, and SVM classifier.

The total core area for object detection is about 2.86 mm^2, while the general-purpose L1-norm circuit occupies about 27.3% of this core area.

When operating at 25 MHz frequency, approximately 31 fps real-time-processing capability for XGA-size image frames can be obtained. The power consumption of this coprocessor is 21.3 mW at 1.0 V core voltage when operating at the maximal frequency of 125 MHz. Lower working frequency can be used, which leads to lower power dissipation by adjusting to the different speed requirements of various applications. Performance comparison to the state-of-the-art previous research works is presented in Table V.III.

TABLE V. III
Performance Comparison Results with Previous Works

	JSSC [16]	JSSC [17]	IEICE [32]	This Work
Technology	130 nm CMOS	65 nm CMOS	65 nm CMOS	65 nm CMOS
Feature descriptor	SIFT	Cell-based HOG	Cell-based HOG	Cell-based HOG
Core area(mm^2)	7.0×7.0	3.58×3.58	3.3×1.2	1.195×2.395
Power consumption	496mW (average@200MHz, 1.2 V)	58.6mW (@62.5MHz, 0.77 V)	40.3mW (@42.9MHz,0.7 V)	21.3mW (@125MHz,1.0 V)
On-chip SRAM(Kbit)	396×8= 3168	280.1×8=2240.8	610 (one core)	328 (norm:36)
Normalization scheme	Different ways by software	Same as Ref.11	L2-Hys-norm	L1-norm
Maximal frequency	200 MHz IPs / 400 MHz NoC	125 MHz	110 MHz	125 MHz
Frame rate	60fps(VGA @ 200MHz)	30-60fps(HDTV @125MHz)	30fps (HDTV @42.9MHz)	31fps(XGA,@ 25MHz) 158fps(XGA,@ 125MHz)
Image resolution	Fixed 640×480 pixels	Fixed 1920×1080 pixels	Fixed 1920×1080 pixels	Flexible ≤ 1024×∞ pixels
Flexibility for cell size	-	12 sizes up to 104×104 pixels	Fixed 8×8 pixels	5 sizes up to 32×32 pixels
Multi-scale	Multi-scale	12-level pyramid	Single scale	5-level pyramid

The required storage space for on-chip object recognition is just 328kbits SRAM, where the reconfigurable general-purpose L1-norm circuit consumes only 36 kbits dual-port memory space due to the applied reutilization scheme.

Multiple SWs to match different target objects are constructed simultaneously, although each SW is in a different stage of its FV construction. For different target object detection and recognition, corresponding weight vector **w** and constant bias *b* can be re-trained offline basing on different datasets and written to nine SRAMs.

5.3.3 Applications for pedestrian detection

The pedestrian-detection application as an illustration to evaluate the accuracy of the proposed algorithm and VLSI architecture with an equivalent software emulation employing an advanced 3.30GHz Intel® Core™ i5-4590 CPU and 8 GB of RAM memory. The object detection is carried out along with the procedure of SW-based FV construction. SVM classifier is applied to verify the significance of FVs for object-detection performance with the 3780-dimensional block-based normalized HOG descriptor.

Figure 5.21 shows that a detection system with L1-norm achieves comparable accuracy performance to the case where L2-norm is applied and further operates with much better accuracy than without normalization (no-norm) for different feature descriptors and classifiers from the INRIA dataset.

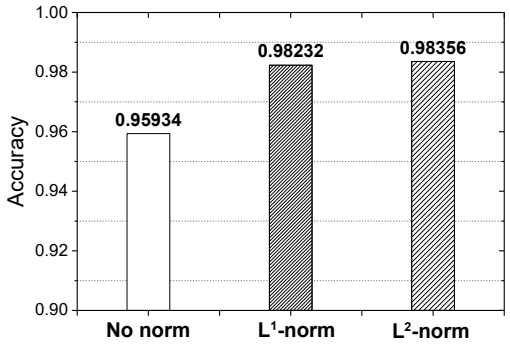


Fig.5. 21 Pedestrian-detection accuracy with cell-based HOG feature descriptor and a pre-trained SVM classifier, based on 5656 SWs of 64×128-pixel size.

It is confirmed that 98.23% pedestrian-detection accuracy with L1-norm can be achieved by a pre-trained linear SVM classifier, which allocates a trained threshold and a fixed weight for each dimension of the 3780-dimensional FV of a HOG descriptor using 5656 SWs with a size of 64×128 pixels.

5.4 Conclusion

Two frameworks (i.e., simplified SURF feature descriptor + NNS classifier, and HOG feature descriptor + SVM classifier) for object detection and recognition are applied for evaluation of different objects such as front cars, lateral cars, and pedestrians. Both hardware and software implementations are developed in this section to verify the detecting performance of the proposed architecture. All results of the evaluation show that the proposed algorithm in this research is efficient for mobile application. Flexible application adjustment is provided by input customization parameters for different image-cell sizes, cell-based feature descriptors, and image resolutions. The developed reutilization scheme of memories for intermediate-result storage allows a significant reduction of on-chip storage requirements. Lower computational cost and pipelined data transmission lead to increased efficiency with respect to power consumption. Consequently, the applied prototype architecture demonstrates less memory usage, lower energy consumption, and higher detection robustness for real-time object detection in various mobile applications.

References

[1] W. Lee, and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM transactions on Information and system security (TiSSEC)* vol.3, no.4, pp.227-261, 2000.

[2] V. N. P. Dao and V. R. Vemuri, "Computer Network Intrusion Detection: A Comparison of Neural Networks Methods," *Differential Equations and Dynamical Systems*, vol.10, no.1&2, 2002.

[3] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of machine learning research*, vol.4, pp.933-969, Nov. 2003.

[4] A. Suleiman, and V. Sze, "An energy-efficient hardware implementation of HOG-based object detection at 1080HD 60 fps with multi-scale support," *Journal of Signal Processing Systems*, vol.84, no.3, pp.325-337, 2016.

[5] K. Takagi, K. Mizuno, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A sub-100-milliwatt dual-core HOG accelerator VLSI for real-time multiple object detection," In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 2533-2537.

[6] T. Seidl, and H. P. Kriegel, "Optimal multi-step k-nearest neighbor search," *ACM SIGMOD Record*, vol. 27, no. 2, pp.154-165, 1998.

[7] H. J. Mattausch, T. Gyohten, Y. Soda, and T. Koide, "Compact associative-memory architecture with fully parallel search capability for the minimum Hamming distance," *IEEE J. Solid-State Circuits*, vol.37, no.2, pp. 218-227, 2002.

[8] Y. Yano, T. Koide, and H. J. Mattausch, "Fully parrallel nearest Manhattan-distance search memory with large reference-pattern number," *Ext. Abstr. Solid State Devices and Materials*, pp. 254-255, 2002.

[9] J. W. Williams, Y. Li, "Comparative Study of Distance Functions for Nearest Neighbors," In: *Elleithy K. (eds) Advanced Techniques in Computing Sciences and Software Engineering*, Springer, Dordrecht (2010), pp 79-84.

[10] M. Miyazawa, P. Zeng, N. Iso, and T. Hirata, "A systolic algorithm for Euclidean distance transform," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.28, no.7, pp.1127-1134, 2006.

[11] X. Zhang, F. An, L. Chen, and H. J. Mattausch, "Reconfigurable VLSI implementation for learning vector quantization with on-chip learning circuit," *Japanese Journal of Applied Physics*, vol.55, no.4S, pp.04EF02:1-6, Mar. 2016.

[12] F. An, T. Akazawa, S. Yamasaki, L. Chen, and H. J. Mattausch, "VLSI realization

of learning vector quantization with hardware/software co-design for different applications," *Japanese Journal of Applied Physics*, vol.54, no.4S, pp. 04DE05:1-5, Mar. 2015.

[13] F. An, and H. J. Mattausch, "K-means clustering algorithm for multimedia applications with flexible HW/SW co-design," *Journal of Systems Architecture*, vol.59, no.3, pp.155-164, Mar. 2013.

[14] F. An, T. Koide, and H J. Mattausch, "A K-means-based multi-prototype high-speed learning system with FPGA-implemented coprocessor for 1-NN searching," *IEICE Trans. Information and Systems*, vol.E95-D, no.9, pp.2327-2338, Sep. 2012.

[15] K. Takagi, K. Tanaka, S. Izumi, H. Kawaguchi, and M.Yoshimoto, "A real-time scalable object detection system using low-power HOG accelerator VLSI," *Journal of Signal Processing Systems*, vol.76, no.3, pp.261-274, 2014.

[16] J. Y. Kim, M. Kim, S. Lee, J. Oh, K. Kim, and H. J. Yoo, "A 201.4 GOPS 496 mW real-time multi-object recognition processor with bio-inspired neural perception engine," *IEEE Journal of Solid-State Circuits*, vol.45, no.1, pp.32-45, 2010.

[17] A. Suleiman, Z. Zhang, and V. Sze, "A 58.6 mw 30 frames/s real-time programmable multiobject detection accelerator with deformable parts models on full HD 1920×1080 videos," *IEEE Journal of Solid-State Circuits*, vol.52, no.3, pp.844-855, 2017.

[18] D. Jeon, M. B. Henry, Y. Kim, I. Lee, Z. Zhang, D. Blaauw, and D. Sylvester, "An energy efficient full-frame feature extraction accelerator with shift-latch FIFO in 28 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol.49, no.5, pp.1271-1284, 2014.

[19] D. Jeon, Q. Dong, Y. Kim, X. Wang, S. Chen, H. Yu, D. Blaauw, and D. Sylvester, "A 23mW Face Recognition Accelerator in 40nm CMOS with Mostly-Read 5T Memory," in *Proc. Symp. VLSI Circuits*, Kyoto, Japan, 2015, pp. C48 - C49.

[20] Y. M. Tsai, T.J. Yang, C.C. Tsai, K. Y. Huang, and L. G. Chen, "A 69mW 140-meter/60fps and 60-meter/300fps Intelligent Vision SoC for Versatile Automotive Applications," in *Proc. Symp. VLSI Circuits*, Honolulu, HI, USA, 2012, p.152-153.

[21] T. W. Chen, Y. L. Chen, T. Y. Cheng, C. S. Tang, P. K. Tsung, T. D. Chuang, L. G. Chen, and S. Y. Chien, "A multimedia semantic analysis SoC (SASoC) with machine-learning engine," in *Proc. ISSCC*, San Francisco, CA, USA, 2010, p.338-339.

[22] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse,

part-based representation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.26, no.11, pp.1475-1490, 2004.

[23] S. Agarwal, and D. Roth, "Learning a sparse representation for object detection," in *Proc. European Conf. on Computer Vision*, Springer, Berlin, Heidelberg, 2002.

[24] N. Dalal, INRIA Human Dataset (2005) [http://pascal.inrialpes.fr/data/human/].

[25] J. A. Eichel, A. Mishra, N. Miller, N. Jankovic, M. A. Thomas, T. Abbott, D. Swanson, and J. Keller, "Diverse Large-Scale ITS Dataset Created from Continuous Learning for Real-Time Vehicle Detection," *arXiv preprint arXiv:1510.02055* (2015).

[26] K. Y. Park, and S.Y. Hwang, "An improved Haar-like feature for efficient object detection," *Pattern Recognition Letters*, vol.42, pp.148-153, 2014.

[27] Y. Tang, C. Zhang, R. Gu, P. Li, and B. Yang, "Vehicle detection and recognition for intelligent traffic surveillance system," *Multimed Tools Appl*, vol.76, pp.5817, 2017.

[28] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol.110, no.3, pp. 346-359, Jun. 2008.

[29] C. Cortes, V. Vapnik, "Support-vector networks," *Machine Learning*, vol.20, no.3, pp. 273–297, 1995.

[30] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[31] F. Melgani, and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geoscience and Remote Sensing*, vol.42, no.8, pp.1778-1790, 2004.

[32] K. Mizuno, K. Takagi, Y. Terachi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A sub-100mW dual-core HOG accelerator VLSI for parallel feature extraction processing for HDTV resolution video," *IEICE Transactions on Electronics*, vol.96, no.4, pp.433-443, 2013.

Chapter 6

Summary and Future Research

6.1 Summary

As an essential prerequisite for building object-classification models to achieve high detection performance, a resource-efficient hardware architecture for feature extraction and representation, applying a simplified SURF descriptor which employs Haar-like wavelets as feature vectors (FVs), is implemented in this research for dedicated mobile applications and handheld devices. A low computational-budget hardware architecture, that directly uses the serially-inputted pixel data without pixel pre-processing, is employed for FV extraction of local image cells. The typically applied integral image in previous work is substituted by an immediate processing engine for the serial input of pixel data from the image sensor, resulting in resource efficiency and real-time processing. The unlimited height of input images ($\leq 1024 \times \infty$ pixels) enables an important contribution to extensive flexibility for processing different applications.

Further, an innovative sliding-window approach is proposed for simultaneous processing of parallel FV construction and object detection. Two schemes, employing cell-based non-normalized FVs and block-based normalized FVs, respectively, demonstrate the window-level FVs construction, which quantifies the similarity of contained objects to reference windows for the recognition process. The detection operation synchronizes with the serial output of local partially computed FVs from multiple simultaneously processed scan windows. Due to the efficiency of the multi-window parallel processing, a significant reduction of power and storage consumption is achieved in this research.

To reduce the computation complexity without a significant decrease in detection accuracy, the proposed block-based FV-normalization scheme and the dimensionality reduction employing partial least squares (PLS) regression, turn out to complement each other and are thus effective for achieving the task of high detection performance. The PLS regression scheme significantly reduces the FV dimensionality as well as the computation cost, while the normalization scheme enhances the robustness along with the texture and illumination variation. The reconfigurable architecture is adapted to variable cell and window sizes, both of which are suitable for enabling multi-scale image search

and multi-object detection.

Finally, detection-performances evaluation of ASCI and FPGA implementations with different classifiers are reported, which employ the nearest neighbor search (NNS) or the support vector machine (SVM) as object classifiers The properties of the designated architecture include efficient resource utilization, high processing speed and classification accuracy, low computational cost, low power consumption as well as high compatibility and versatility for extensive mobile applications and handheld devices.

6.2 Future work

The architecture proposed in this research can be further integrated together with an image sensor and the essential peripheral circuits, to manufacture a complete on-chip vision system. The manageable image resolution is confined by the image width so that higher resolutions such as Full HD (1920×1080 pixels) still cannot be handled with the developed coprocessor design. The storage capacity limits the processing capacity of the architecture while the memory requirements occupy a large portion of the area of the bare chip. It is possible to enhance the processing performance by improving the allocation strategy of the memory space and the logical scheme of the algorithm. The sliding-window approach in raster scan manner, which results in mass computation of multiple overlapping windows, is another place that can be considered for improvements. It is further possible to apply the developed vision-based feature representation in the fields of deep learning and neural networks for achieving still better performance in the future.

Acknowledgement

First of all, I would like to thank my mentor Professor Hans Jürgen Mattausch for giving me a great opportunity to work with his laboratory. I really appreciate his supervision during my entire research journey at Hiroshima University. His piercing insight and laudable humility make being his student a delight. But even more exceptional are his integrity and his unwavering devotion to his students. I offer my heartfelt thanks to all his help on the check of my articles. I also would like to express my sincere appreciation to Professor Minoru Fujishima and Professor Shin Yokoyama for their invaluable and essential comments from various perspectives.

I want to thank all my research collaborators, the ones who worked with me on the projects described in this dissertation deserve specific commendations. Thanks to Dr. Fengwei An for the instruction and help on my research work; And thank Dr. Lei Chen, without you this journey would never have started. Both Xiangyu Zhang and Jungang Guan gave me a lot of help on the research and daily life in Japan; thank you very much! I also want to thank Yuuki Fujita, Ikki Nakashima, Dai Suzuki, and Hiroaki Gau for their help to collect testing images around the university and take verification for the architecture design, etc. Thanks to the group members again; they make my research journey fulfilled and inspire me to learn more knowledge of digital circuit designs in the field of machine vision.

I am forever grateful to everyone who has made my time in the Graduate School of Advanced Sciences of Matter at the Hiroshima University the fulfilling and intensely meaningful experience it has been. I also would like to express thanks to the financial support from the Japanese government, Hiroshima University, and other civil organizations to make me complete my study in Japan.

Lastly, and most importantly, I want to thank my family for their unconditional support and encouragement. Thanks my parents for their indispensable spiritual supports and giving me the respect to my own opinions and full freedom to choices for my life. I also want to thank my other family members for their unconditional help and positive suggestions.

List of Publications, Awards and Patents

1. International conference presentations related to this thesis

First author

(1) Real-Time Haar-like Feature Extraction Coprocessor with Pixel-Based Pipelined Hardware Architecture for Flexible Low-Power Object Detection and Recognition

Aiwen Luo, Fengwei An, Yuki Fujita, Xiangyu Zhang, Lei Chen, Hans Jürgen Mattausch;

International Conference on Solid State Devices and Materials, Tsukuba, Japan, September 2016: Oral Presentation.

(2) Reconfigurable Block-based Normalization Circuit for On-chip Object Detection

Aiwen Luo, Fengwei An, Xiangyu Zhang, Lei Chen and Hans Jürgen Mattausch

International Conference on Solid State Devices and Materials, Sendai, Japan, September 2017: Poster Presentation.

Coauthor

(1) Pixel-based Pipeline Hardware Architecture for High-performance Haar-like Feature Extraction

Yuki Fujita, Fengwei An, Aiwen Luo, Xiangyu Zhang, Lei Chen, Hans Jürgen Mattausch;

IEEE Asia Pacific Conference on Circuits and Systems, Korea, 2016: Poster Presentation.

2. Awards

丸文財団　交流研究助成賞、一般財団法人　丸文財団、平成２９年３月８日

3. Patents

(1)【発明の名称】ピクセルベース特徴量抽出回路

【発明者】マタウシュ ハンスユルゲン、 安豊偉、陳蕾、張湘イク、駱愛文

【特許願番号】2016-166568

【出願日】2016 年 8 月 29 日(Applied in Japan)

(2)【発明の名称】画像認識装置
【発明者】マタウシュ ハンスユルゲン、安豊偉、陳蕾、張湘イク、駱愛文
【特許願番号】2017-030253
【出願日】2017 年 2 月 21 日 (Applied in Japan)

公表論文
（Articles）

(1) Low-power Coprocessor for Haar-like Feature Extraction with Pixel-based Pipelined Architecture

Aiwen Luo, Fengwei An, Yuki Fujita, Xiangyu Zhang, Lei Chen and Hans Jürgen Mattausch

Japanese Journal of Applied Physics, vol.56, no.4S, pp.04CF06:1-9, 2017.

(2) Resource-efficient Object Recognition Coprocessor with Parallel Processing of Multiple Scan Windows in 65 nm CMOS

Aiwen Luo, Fengwei An, Xiangyu Zhang, Lei Chen, Hans Jürgen Mattausch

IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 3, pp. 431-444, March 2018, DOI: 10.1109/TVLSI.2017.2774813.

(3) Flexible Feature-space-construction Architecture and its VLSI Implementation for Multi-scale Object Detection

Aiwen Luo, Fengwei An, Xiangyu Zhang, Lei Chen, Zunkai Huang and Hans Jürgen Mattausch

Japanese Journal of Applied Physics, vol.57, no.4S, pp.04FF04:1-8, 2018.

参 考 論 文
（Thesis Supplements）

(1) Real-time Haar-like Feature Extraction Coprocessor with Pixel-based Pipelined Hardware Architecture for Flexible Low-power Object Detection and Recognition
Aiwen Luo, Fengwei An, Yuki Fujita, Xiangyu Zhang, Lei Chen, Hans Jürgen Mattausch
Extended Abstracts of the International Conference on Solid State Devices and Materials, Tsukuba, 2016, pp.457-458.

(2) Reconfigurable Block-based Normalization Circuit for On-chip Object Detection
Aiwen Luo, Fengwei An, Xiangyu Zhang, Lei Chen and Hans Jürgen Mattausch
Extended Abstracts of the International Conference on Solid State Devices and Materials, Sendai, 2017, pp.819-820.

(3) A Hardware Architecture for Cell-based Feature-Extraction and Classification Using Dual-feature Space
Fengwei An, Xiangyu Zhang, **Aiwen Luo**, Lei Chen, Hans Jürgen Mattausch
IEEE Transactions on Circuits and Systems for Video Technology, vol.PP, no.99, pp.1-13, 2017.

(4) A Hardware-oriented Histogram of Oriented Gradients Algorithm and its VLSI Implementation
Xiangyu Zhang, Fengwei An, Ikki Nakashima, **Aiwen Luo**, Lei Chen, Idaku Ishii and Hans Jürgen Mattausch
Japanese Journal of Applied Physics, vol.56, no.4S, pp. 04CF01:1-7, 2017.

(5) Pixel-based Pipeline Hardware Architecture for High-performance Haar-like Feature Extraction
Yuki Fujita, Fengwei An, **Aiwen Luo**, Xiangyu Zhang, Lei Chen, Hans Jürgen Mattausch
IEEE Asia Pacific Conference on Circuits and Systems, Korea, 2016, pp.611-612.