A Thesis for the Degree of Ph.D. in Engineering

Speed-up of Biologically Inspired Image Categorization for Information Retrieval and Object Localization

February 2014

Graduate School of Science and Technology
Keio University

Takuya Minagawa

Abstract

It is a relatively easy task to understand "what" in an image not for a machine but for a human. Primates can recognize a category of image contents very rapidly: between 100 and 130 msec. A feedforward path of ventral stream in the visual cortex is known to be activated for this "rapid categorization" task. HMAX model has been invented to approximate the activity of the feedforward path of ventral stream, however it is so slow to compute because of the different architecture between machines and brain.

This thesis aims to speed up HMAX feature computation for image retrieval and object localization.

HMAX was originally applied to object categorization tasks, thus we first tried to reduce the processing cost for these tasks. Our approach reduced computing cost of the HMAX with simplified features, elimination of processing area, and reduction of number of feature patches. We demonstrated an image based information retrieval system with this speed-up HMAX model as an application example.

Then more difficult application that is object localization was addressed. This task is more time consuming because object categorization must be executed on every sliding window. Because HMAX feature is extracted by computing similarities between an image and image patches, our approach searches the similar regions to the shapes of parts that target object has from coarse to fine resolution. This approach reduced redundancies of HMAX and sliding window enormously.

The proposed object categorization method was evaluated with Caltech-101 and scene image datasets, and the proposed object localization method was evaluated with UIUC car and FDDB datasets. They showed that our categorization method is about 37 times faster than the original HMAX, and our localization is

250 times faster or more than the sliding window approach with little reduction of recognition rate.

Acknowledgements

During my Ph.D. program, I have met many people those who influenced me in any sense. Without them, I could not have finished my Ph.D. work.

First of all, I would like to thank my thesis adviser, Prof. Hideo Saito, for the liberty he gave me and his encouragement and strong support on my way. He advised me to apply Ph.D. program while working when I lost my job and considered the next career. That was a catalyst for me to back to the laboratory. After I had become independent as a freelance engineer, he introduced me some jobs which helped me and my family to survive. For his support, I did what I wanted to study during the program.

I would like to sincerely thank the Ph.D. alumni in Ozawa and Saito lab: Mitsuru Ambai, Tomoaki Teshima, Yujin Oyamada, and Hideaki Uchiyama. Dr. Ambai, in Denso IT Lab, Inc, gave me helpful advice and encouragement when we debated whether or not to give up the program. He also made time for me on weekday to consult me about my thesis. Dr. Teshima advised me about my career and the program too. The materials he provided were useful references for me. I might have done a terrible mistake of procedure to apply public hearing without Dr. Oyamada's notice. He also gave me a template of thesis that is used in this document. Dr. Uchiyama told me his experience of public hearing that let me notice the importance and difficulty to create good story for thesis.

I don't know how to express my appreciation to Takuya Miyata, who was a founder of J-Magic, Inc and is a CEO of mixi America. In spite of my study in Keio University during working, he provided me a great job opportunity in his company, J-Magic. I've never done such exciting experiences in my career before. I would also like to thank the co-workers in J-Magic who understood my situation and supported me a lot.

I am always grateful to Yoichi Miyazawa and Takeo Miyazawa who hired me in Kizna Corporation. It was a trigger of my career to start learning computer vision for purpose of business. They gave me stimulating moments to manage projects with great researchers in universities abroad.

Let me express my gratitude to the customers, the friends, and the members of computer vision study group. As a freelance engineer, the job offers from the customers have made it possible not only to feed myself and my family but to afford the time to research. The friends of mine have provided me psychological support by hearing my grouch, and sometimes treated me a dinner. With the cooperation of the study group members, I have been encouraged to study new knowledge.

My greatest gratitude goes to my family. My father, mother, and brother always understand and back up my life and career. My father and mother in law in Hokkaido have kindly accepted my rudeness to write papers during our homecoming visit. They have always given me fine hospitality. It is no exaggeration to say that my wife, Akiko, is the biggest supporter. At all times, she is the nearest person in my life and understanding what I do. She have been taken care to make my time for study and work. Without her patience encouragement, and support, I would never have achieved what I have ever done. Words can't express how grateful I am to her.

Finally, I would like to dedicate this thesis to our coming baby.

January 15, 2014
Takuya Minagawa

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding contents of images is the final goal of the research area called *computer vision*. It is a difficult task not for humans but for computers because an image is just an array of pixel values for machines (Figure 1.1). The pixel array must be categorized into a certain pattern to identify "what" is in the image. Humans also capture light rays by retinas and translate them into electrical signals like cameras and computers do; it means visual cortices of humans have algorithms to recognize the patterns from these simple signals.

For this reason, understanding the mechanism of visual perception in brains is important study to make computer understand image contents. Some might criticize that understanding brains did not mean building intelligent computers because of differences between their architectures. That criticism does not consider "recognition" as information processes whose logics are independent from any architectures. Marr proposed to understand the brain's work as information processes and defined three layer to be addressed: computational theory, representation and algorithm, and hardware[Marr, 1982]. "Computational theory" defines problems to solve, "representation and algorithm" defines input/output formats and how to transfer inputs into outputs, and "hardware" defines circuits. Marr thought that these three layers can be treated individually.

The research area of computer vision addresses the "computational theory" and the "representation and algorithm" layer. At first, the problem to recognize "what" is in images should be defined as the computational theory; it is called

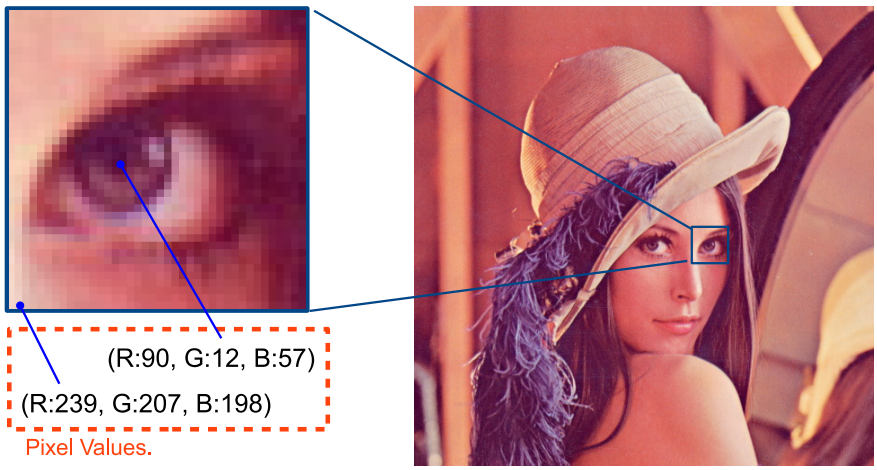


Figure 1.1: What a computer knows about an image is only a sequence of pixel values.

generic object recognition or *generic image recognition*. Generic object recognition aims to answer names of objects in input images as shown in Figure 1.2. This task is much challenging for reasons such as enormous numbers of generic terms of object, appearance changes caused by view point, scale or illumination changes, and occlusions by other objects. The generic object recognition had been divided into some sub tasks to make the problem easier. *Object categorization* or *image categorization* is a task to label a whole image with a generic term of scenes or objects (building, mountain, ocean, animal, face, etc.).[1] *Object localization* is a task to detect positions of certain categories of objects (face, pedestrian, vehicle, etc.). *Specific object recognition* is a task to recognize unique nouns of objects in images (Basilique de Notre Dame de la Garde, Toyota Prius, the CD album "Abbey Road" of The Beatles, etc.).

These object recognition tasks have been active research area for more than a decade; however, most of major approaches have not been developed based on biological evidences. They have been developed just to achieve better performance. These efforts have succeeded in bringing some applications to commercial usages. Face recognition is one of the most successful applications which has been imple-

[1]Some papers also call this task *generic object recognition*; however, this paper uses the term *object categorization* or *image categorization* to distinct from the task of Figure 1.2

Figure 1.2: An example of generic object recognition: each object in the image is labeled with its generic term by the algorithm.

mented into digital cameras[Canon U.S.A. Inc, 2012; Sony Corporation, 2012], smart phones[FaceLock.mobi, 2012], web services[Facebook.com, 2012; Google Inc, 2012b], and other applications. Specific object recognition for rigid textured image (such as CD jackets, magazines, etc.) is another example used in commercial applications[A9.com Inc, 2012; Google Inc, 2012a; kooba, 2012]. Nevertheless, they are still inferior to the human ability; therefore, it is still important to build recognition algorithms based on biological studies. Remember the history that many progresses of technologies have been caused by learning cases in the nature: for instance, the invention of airplanes were much inspired by observing birds[Wikipedia, 2013]. In fact, some major techniques of computer vision were influenced by the biological knowledge [Lowe, 1999; Wiskott et al., 1997].

Computational models of primates' visual cortex have been developed in a few decades by a lot of scientific efforts of such as anatomy, neurophysiology, psychophysics, and artificial intelligent. These knowledge have been applied to object recognition algorithms and improved better and better[Bengio, 2009; Fukushima,

1980; LeCun et al., 1989; Serre et al., 2005]; however, they usually require huge computing resources, training data, and processing time. Thus, it must be valuable to address the problem of computational cost for purpose of practical usages of these algorithms.

Some might say that we should wait for evolutions of hardware because the processing capability of computers, according to Moore's law [Wikipedia, 2014], has improved day by day. However, it is difficult to expect computing resources every time because selection of hardware depends on its application. Popular architectures have also been and will be changing: personal computers, servers, smart phones, wearable and embedding devices, and other types of computers. Speed-up of algorithm is generally versatile to every hardware, thus it is valuable challenge in the present and the future.

HMAX[2] model is one of the successful computational models to approximate a *feedforward path of ventral stream*[Serre, 2006; Serre et al., 2005, 2007] which is activated in object recognition tasks. The model could predict human performance extremely well in rapid animal vs. non-animal recognition task[Serre, 2006]. Machines usually take long time to compute the HMAX model, whereas brains process recognition tasks very rapidly (around 120 ms) in the feedforward path of ventral stream. Therefore, its computational cost is a bottleneck in developing applications. It may mainly be caused by differences of the architectures: brains process visual signal parallel through distributed neural modules. We aim to develop applications of the HMAX model by reducing its computational cost.

1.1 Objective

The objective of this thesis is to speed up the computation of the HMAX model, which was inspired from biological evidences, for applying it for image based information retrieval and object localization.

First, we try to reduce the computing cost of the HMAX for object categorization tasks, because the HMAX was originally designed for these tasks. The target response time is determined to satisfy requirements of visual search system which

[2]"HMAX" stands for *Hierarchical Model And X*; "X" represents *maXimum* operation.[Tarr, 1999]

receives query image and returns search results related to image contents. The contents are recognized in object categorization tasks with the HMAX. Our modifications to the original HMAX features significantly reduced its computational cost, which enabled the system to work in reasonable time.

Second, we try more challenging task, i.e., object localization tasks. We modify the HMAX for speeding-up of object localization tasks which usually spend much more time than categorization tasks. Of course the speeding-up of categorization tasks also improves the speed of localization tasks; however, some redundancies still remain in sliding window approaches. We succeeded in reducing the computing cost by eliminating the redundancies which are specific to object localization tasks.

The HMAX has been used mainly to study biologically inspired object recognition models from scientific interests, because the HMAX model was invented from the studies of visual cortex. The speed-up of computing HMAX might also contribute to speeding up evaluations of the biological hypotheses.

1.2 Organization of the Thesis

This thesis is organized as follow. This chapter states the motivation and the main interest of this thesis. Chapter 2 describes the architecture of Serre's HMAX model[Serre et al., 2005, 2007] in detail and reviews its related work. Then, we review existing feature descriptors and speed-up approaches for object recognition in Chapter 3. Chapter 4 describes the way to reduce processing times of HMAX in image categorization tasks and evaluates visual search system that uses our modified HMAX model. Chapter 5 describes the way to reduce processing time in object localization tasks by eliminating redundancies of sliding window approach with HMAX. Finally, Chapter 6 concludes this thesis with a summary of the contributions and discussions.

Chapter 2

Ventral Stream and HMAX Model

This section reviews the HMAX model and its related work. The review consists of overview and its history, the architecture, and applications and extensions.

2.1 Overview

HMAX can be regarded as a feature descriptor inspired by a primate's visual cortex. The visual cortex has two pathways to proceed visual stimuli: the dorsal stream and the ventral stream. The dorsal stream is known as "where" pathway which is used *for action*, such as stretching arm and grabbing an object. It finds a spatial position of an object. The ventral stream is known as "what" pathway which is used *for recognition* of an object category. Primates can categorize an image in rapid serial visual presentation (RSVP) that does not allow sufficient time for eye movements or shifts of attention (10 images / sec). This *rapid categorization* is proceeded in the feedforward path of ventral stream. When an image is presented in enough time for patients to be aware, primates pay attention to recognize it finely and activate a feedback path. The architecture of HMAX was designed to be similar to the feedforward path of ventral stream.

Figure 2.1 illustrates the pathway from the retina via ventral stream to action. At first, light achieving the eye is projected onto the retina where the optic signals are translated to electrical signals. Then the signal is proceeded through lateral geniculate nucleus (LGN), primary visual cortex (V1), V2, V4, inferior posterior

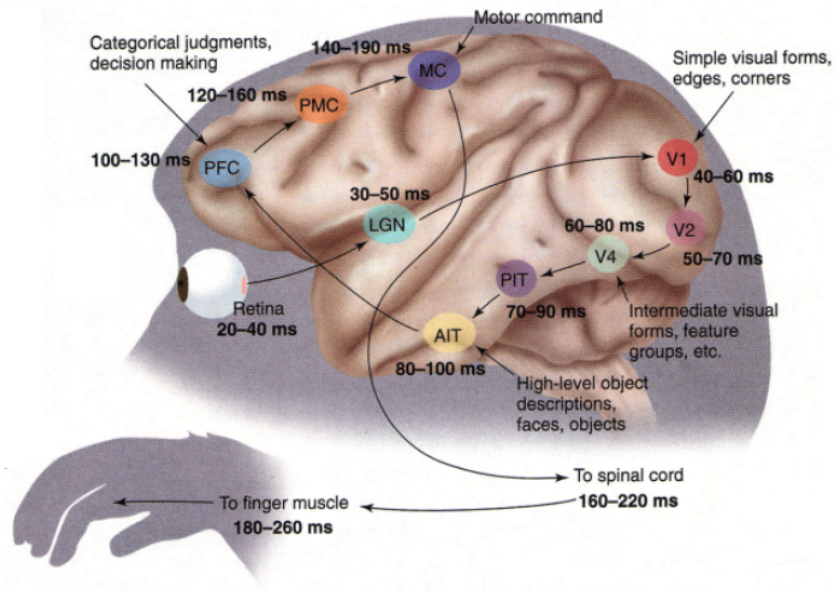


Figure 2.1: Flow of visual information from the retina via ventral stream to action. The timing for visual stimuli to arrive at each cortical area in the monkey's brain are explained. This image is reproduced from [Thorpe and Fabre-Thorpe, 2001].

and anterior inferior temporal cortex (PIT and AIT), and prefrontal cortex (PFC). It takes 100 - 130 ms for visual stimuli to reach PFC which is related to decision making. Then the signal for action transmits through premotor cortex (PMC), primary motor code (MC), and spinal code to muscle. On the ventral stream from retina to PFC, the cells at the upper area respond to more complex signals and cover larger receptive field. For instance, V1 units respond to edge like features, on the other hand AIT units respond to object like features such as faces, and the latter cells have more invariance to transformations such as position and scale changes.

Selectivity and invariance to visual stimuli are important features of visual cortex. Hubel and Wiesel found that simple cells and complex cells in V1 area respectively have natures of selectivity and invariance[Hubel and Wiesel, 1962, 1965]. The simple cells fire against edge-like features which have certain position, orientation, and scale patterns. On the other hand, the complex cells also react to the edges of certain orientations but are not so sensitive to small changes

of position and scale. The cells that react to more complex shapes have also been studied. For instance, Logothetis et al. trained the neurons in the AIT area of monkeys to respond to a novel 3D object (e.g., paper clips)[Logothetis et al., 1995]. The trained neurons reacted to a certain angle of view and reduced its response rapidly when the angle changed.

The HMAX model that was originally presented by Riesenhuber and Poggio[Riesenhuber and Poggio, 1999] was derived to achieve this selectivity to object types and invariance to position and scale by hierarchical architecture. The architecture was inspired by Hubel and Wiesel's work about the simple and complex cells[Hubel and Wiesel, 1962, 1965] and by other computational models such as Neocognitron[Fukushima, 1980]. Serre et al. extended the HMAX model mainly by introducing dictionaries of shape-components[Serre et al., 2005, 2007]. They demonstrated that the model could predict human performance extremely well in rapid animal vs. non-animal recognition task[Serre, 2006]. The following parts of this chapter mention about the Serre's HMAX model.

The HMAX processes input signals alternately through the *simple* S layer and the *complex* C layer to achieve both selectivity and invariance: the S layers react selectively, and the *complex* C layers add invariance. The behavior of V1 simple cells is known to be approximated by Gabor filter[Daugman, 1980, 1985; Mardelja, 1980], thus the first S layer is designed to respond to a certain edge pattern by using the Gabor filter as shown in Figure 2.2. Upper S layer responds to more complex shapes which are represented as a combination of edges. The response to the shapes is measured with Radial Basis Function (RBF). The RBF reduces its response with a (Gaussian-like) bell-shape curve[Logothetis et al., 1995] as a stimulus changed from a trained shape.

The invariance is implemented as *max-pooling* in the HMAX model. Figure 2.3 illustrates how max-pooling achieves the invariance to the small changes of position and scale: the process passes the highest response value in cells of neighbor positions and scales.

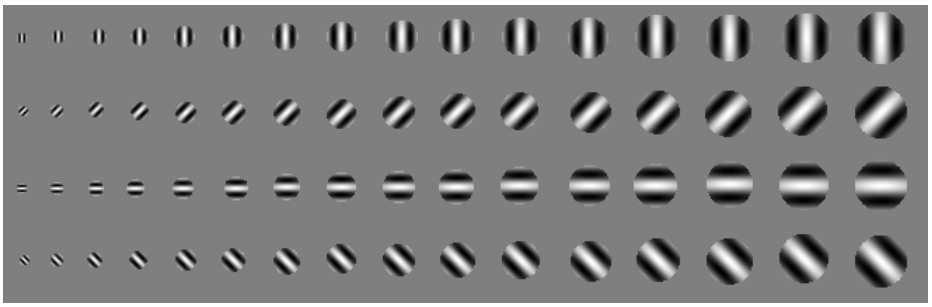


Figure 2.2: Gabor filter of 4 orientations and 16 scales.

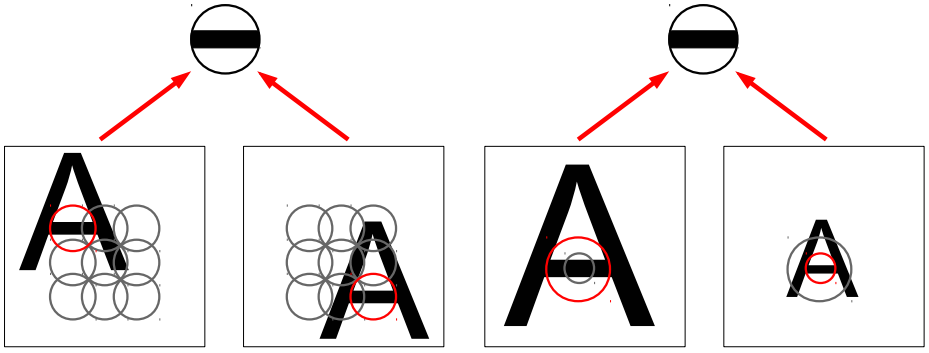


Figure 2.3: Max-pooling.

2.2 Architecture

The architecture of the HMAX model[Serre et al., 2007] is illustrated in Figure 2.4. It is a hierarchical structure of four-layers (S1, C1, S2, and C2), in which S layers represent selectivity and C layers represent invariance. The selective and the invariant layers appear alternately. An input image is proceeded from the S1 to the C2 layer and an output of each layer is an input to the next layer. Finally, an output of the C2 layer is a feature vector for classification.

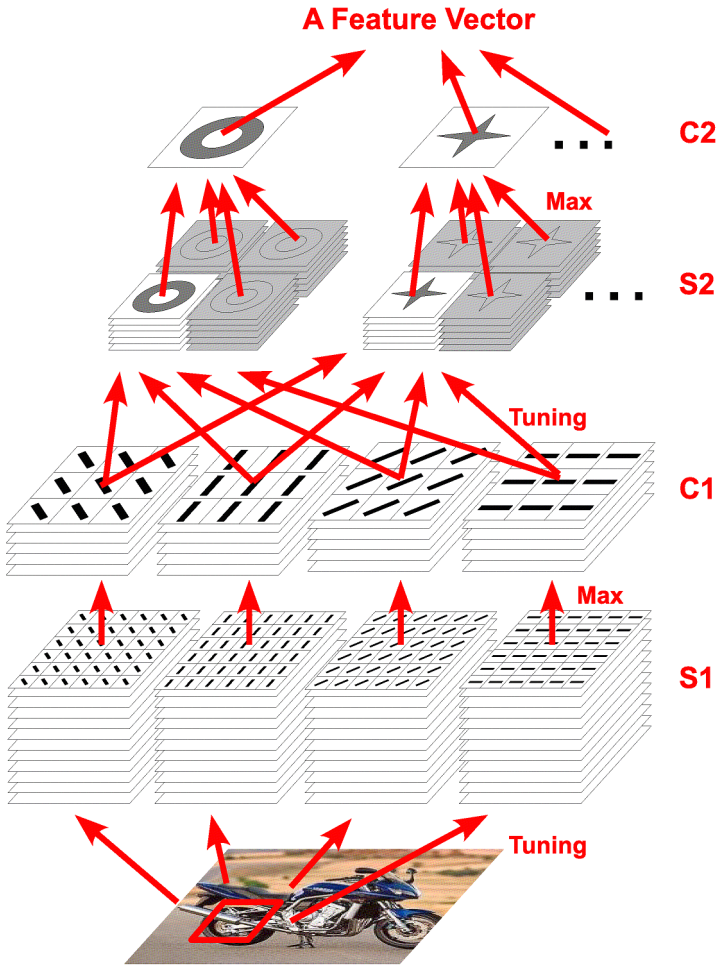


Figure 2.4: The architecture of HMAX model.

S1 layer

An input image is processed at a unit on the S1 layer by a 2D Gabor filter with orientation θ_o, wavelength λ_s, and effective width σ_s:

$$F(x, y, s, o) = \exp\left(-\frac{x_0^2 + y_0^2}{2\sigma_s^2}\right) \cos\left(\frac{2\pi x_0}{\lambda_s}\right) \tag{2.1}$$
$$x_0 = x\cos\theta_o + y\sin\theta_o, y_0 = -x\sin\theta_o + y\cos\theta_o$$

Arguments s and o correspond to an index of a scale band and an orientation. If 16 scale bands and four orientations ($\boldsymbol{\theta} = 0^o, 45^o, 90^o, 135^o$) are used, then 64 S1

values, which correspond to one pixel, are generated.

C1 layer

The C1 layer adds invariance of scale and position to signals from the S1 layer by max pooling as:

$$r_{c1}(\boldsymbol{x}_{c1}) = \max_{\boldsymbol{x}_{s1} \in R(\boldsymbol{x}_{c1})} r_{s1}(\boldsymbol{x}_{s1}) \tag{2.2}$$

where $r_{s1}(\boldsymbol{x}_{s1})$ is a response value of the S1 unit at position $\boldsymbol{x}_{s1}$ and $r_{c1}(\boldsymbol{x}_{c1})$ is that of the C1 unit at position $\boldsymbol{x}_{c1}$. Position vector $\boldsymbol{x}$ consists of coordinates (x, y) and scale s. Each processing unit on the C1 layer passes the maximum signal value from the S1 layer in region $R(x_{c1})$ that is defined by the $N_{c1}(s_{c1}) \times N_{c1}(s_{c1})$ area in neighbouring ΔS_{c1} scale bands at each orientation. Max-pooling area $N_{c1}(s_{c1})$ are overlapped with ratio ΔN_{c1}.

S2 layer

An S2 unit reacts to a signal similar to the pre-defined feature patch that is a randomly selected region of C1 output from a training image. These N_p patches are defined as a *feature dictionary* here. The response of S2 unit is calculated from distance between an input and each patch by using the following radial basis function (RBF):

$$r_{s2}^{n} = \exp(-\beta ||\boldsymbol{X} - \boldsymbol{P}_n||^2), \tag{2.3}$$

where $\boldsymbol{X}$ is a part of a signal generated by the C1 layer, $\boldsymbol{P}_n$ is the n_{th} feature patch, and β is the sharpness of reaction. $\boldsymbol{X}$ and $\boldsymbol{P}_n$ are vectors that have $N_{s2} \times N_{s2} \times D$ elements when D is the number of orientation o and N_{s2} is the patch size.

The S2 layer is the most time consuming process because RBF must be computed for all patches at each position and scale.

C2 layer

The C2 layer integrates the S2 outputs of all position and scale by taking the maximum one. Therefore the response of the C2 layer is an N_p-dimensional vec-

tor in which each element represents the signal most similar to each feature patch over all scales and positions. N_p is the number of feature patches

Finally, this vector is used for training and discrimination by a machine learning algorithm in order to recognize the object category.

2.3 Application and Expansion of HMAX

2.3.1 Simulation of biological model

HMAX has been applied to explain some biological hypotheses because it was also designed for the biological hypotheses of ventral stream. For instance, HMAX model was successfully used to explain quantitatively the behavior of neurons in macaque monkey area V4 that have selectivity for complex stimuli and invariant to spatial translations[Cadieu et al., 2007]. Jhuang et al. modified HMAX model for action recognition. They tried to simulate dorsal stream from V1 to MT and MST by capturing the spatio-temporal motion from image sequence[Jhuang et al., 2007].

Chikkerur et al. proposed the Bayesian model that found "what" object and "where" it were in an image[Chikkerur et al., 2010]. The ventral stream deals "what" and the dorsal stream deals "where" as described in Section 2.1, and therefore this model simulated interactions between the parietal cortex on the dorsal stream and the ventral stream mediated by feedforward and feedback connections. It used the HMAX features to create a saliency map to simulate top-down attention (feedback path) by making use of shapes in an image.

The HMAX model was also used to explain "why the brain separates face recognition from object recognition". It is well known that there are neurons that react selectively to human face but not to other objects. Leibo et al. found that the dedicated circuits recognized faces with much better accuracy by simulating the circuits for recognition with the HMAX model[Leibo et al., 2011]. That is because 2D appearance of face changes significantly by 3D viewpoint and illumination changes; and therefore the brain needs to separate the circuit for important objects such as face.

Temporal association methods are hypotheses that explain the ability of the

brain to track temporally changing object like moving one. The HMAX was used to simulate not only the temporal association learning but "invariance disruption"[Isik et al., 2012].

2.3.2 Application

The HMAX model was not used only for biological simulations but for application purposes. An example of the applications is object localization tasks which have simply been implemented as an image categorization on each sliding window[Bileschi and Wolf, 2005; Huang et al., 2011; Mutch and Lowe, 2006]. We address this application in Chapter 5 by focusing on reducing redundancies of the sliding window approach.

Bileschi and Wolf expanded the HMAX model to recognize whole contexts of street scene images[Bileschi and Wolf, 2005]. Figure 2.5 illustrates the data flow of their method. Their method divided regions in an image to shape-based and texture-based objects. The shape-based objects represented car, pedestrian, and bicycle, which were recognized with C1 features on sliding windows. The texture-based objects represented tree, sky to recognize a scene, building, and road, which were recognized with customized C2 features on segmented regions. Finally, these recognized objects were merged to explain the image.

Meyers and Wolf modified the HMAX features for face recognition, which was named S2 facial features (S2FF)[Meyers and Wolf, 2007], by adding center-surround processing to handle illumination changes, and combining C1 features based on a kernelized and regularized version of the relevant component analysis transformation that is capable of handling high dimensional data. The S2FF features showed good or better face image representations than other popular ones.

Smile recognition is another example of applications which use the HMAX. Lihua applied his modified HMAX features [Lihua, 2011] into smile recognition. The modified features were expanded to be invariant to rotation changes by max-pooling around orientations of the C1, and optimized the feature space of patches by online dictionary learning technique of sparse coding[Mairal et al., 2009].

Huang et al. proposed scene classification application for video surveillance by using the HMAX[Huang et al., 2011]. They selected the highest energy regions

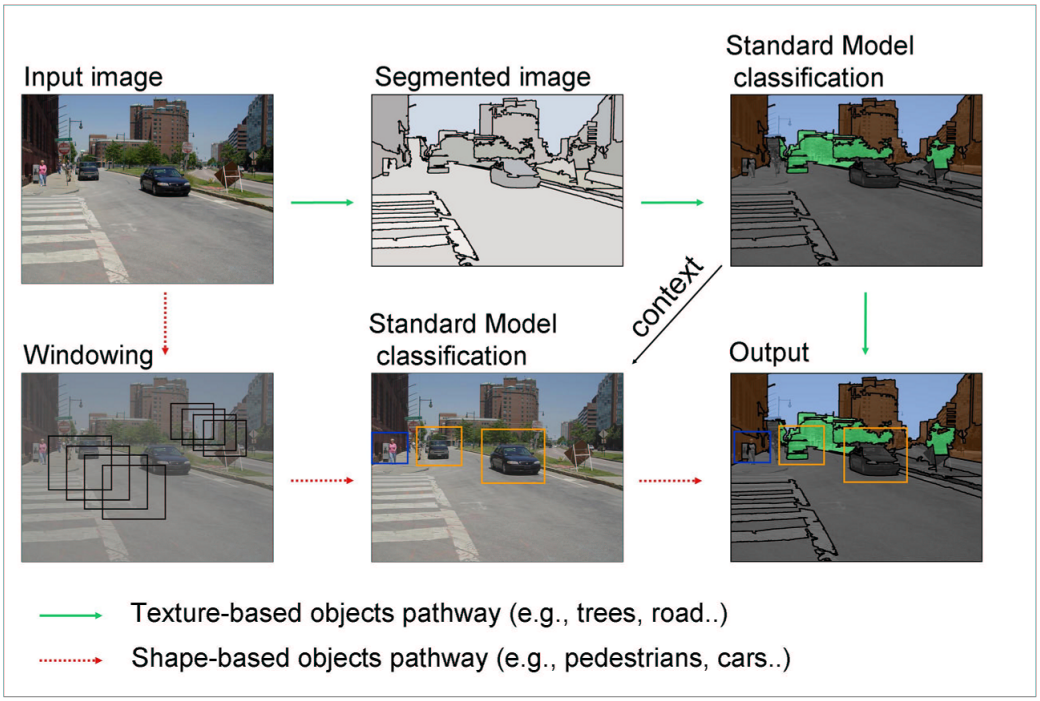


Figure 2.5: Data flow diagram of context analysis with HMAX. This image is reproduced from [Bileschi and Wolf, 2005].

for robust and selective scene recognition.

Some applications have made use of output signals from the C1 layer, not from the C2 layer. For instance, Mu et al. employed the C1 features for face recognition tasks by reducing the dimension of concatenated features with manifold learning[Mu et al., 2009]. They also applied the similar approach to gait recognition[Mu and Tao, 2010] and, amazingly, crater detection[Mu et al., 2011].

Age estimation of faces was also examined with the modified C1 features[El Dib and El-Saban, 2010; Guo et al., 2009]. Guodong et al. achieved the performance over the state-of-the-art method by replacing max-pooling over space to standard deviation (STD) pooling [Guo et al., 2009] in the age estimation tasks. El Dib and El-Saban also extended the C1 features to represent facial features for age estimation[El Dib and El-Saban, 2010]. Their method was superior to the state-of-the-art ones by incorporating fine detailed facial features, automatic initialization using active shape models and analyzing more complete facial areas by

including the forehead details.

For scene recognition, Song et al. added information of intensity and color of image to C1 feature[Song and Tao, 2010]. Cheng applied the similar approach for detecting peripapillary atrophy (PPA) which is an atrophy of pre-existing retina tissue[Cheng et al., 2012].

Most of these studies above have mentioned applications of the HMAX but not systems, which means the processing speeds have not been discussed. However, in practical usages, requirements of speed must be considered to build systems.

2.3.3 Extended HMAX models

The HMAX model has been improved to include other biologically inspired mechanisms or to achieve better performance. Lateral inhibition is well-known phenomenon as a 'winner-take-all' strategy of neurons; the strongest response of neuron among neighbors inhibit responses of the lateral neurons so that the strongest signal is finally emphasized. Because this mechanism help neurons to be more selective, salient regions or edges of images can be captured clearly by humans.

Mutch and Lowe modified the HMAX model by implementing the concept of lateral inhibition[Mutch and Lowe, 2006, 2008]. They used 12 orientations at the S1 layer and only the strongest responses were kept in the orientations of the S2 units as illustrated in Figure 2.6. In addition, non-characteristic responses, for instance the responses of all orientations were almost the same, were set to zero. Their method selected important feature patches, which had the highest weights trained by support vector machine (SVM)[Cortes and Vapnik, 1995], for efficient calculation of multi-class recognition. Finally, they achieved much better performance than the original HMAX model[Serre et al., 2005].

Huang et al. proposed Enhanced Biologically Inspired Model (EBIM)[Huang et al., 2011] that was an expansion of the HMAX model as illustrated in Figure 2.7. They also applied the idea of lateral inhibition and feature selection into the HMAX model. Different from the approach of Mutch and Lowe[Mutch and Lowe, 2006, 2008], they inhibited weak responded regions, not orientations, for efficient calculation. Important feature patches to recognition were selected

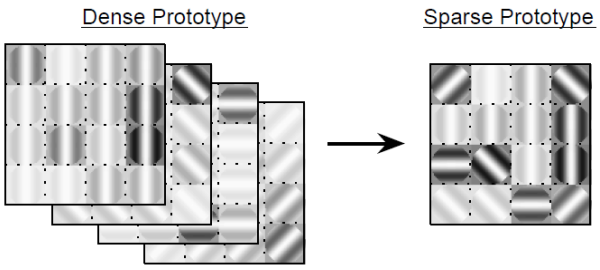


Figure 2.6: Dense to sparse S2 features. This image is reproduced from [Mutch and Lowe, 2006].

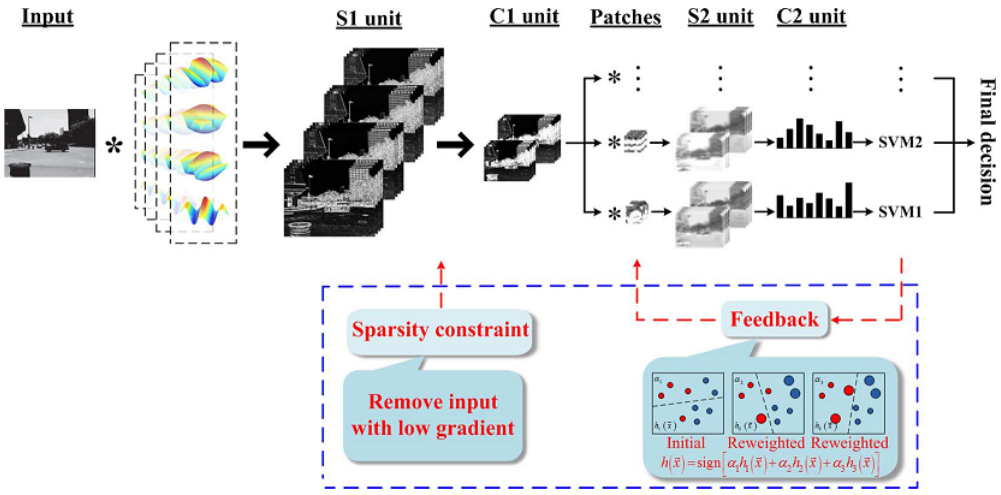


Figure 2.7: Framework of Enhanced Biologically Inspired Model (EBIM). This image is reproduced from [Huang et al., 2011].

by AdaBoost[Schapire and Singer, 1999] with SVM as a weak classifier. This method successfully achieved better performance and reduced processing time.

These extended HMAX models achieved not only better performance but also faster processing. We review and categorize these methods in speed-up point of view in Chapter 4. In the next chapter, we review some speed up techniques for object categorization and localization tasks, then discuss how to apply these techniques to HMAX model.

Chapter 3

Feature Descriptors and Speed-up Techniques for Object Recognition

In this chapter, we review other feature descriptors and speed-up methods for object recognition tasks. The previous chapter explains the HMAX model in biological points of view. Besides, this chapter explains it in computer vision points of view. First, we briefly explain other feature descriptors to clarify the characteristics of the HMAX. Then, we categorize existing speed-up methods into four approaches: faster feature descriptors, reductions of feature dimensions, restrictions of processing areas, and parallel processing. Finally, existing researches that tried to speed up the HMAX features are referred.

3.1 Object Recognition

As discussed in Chapter 1, *specific object recognition*, *object categorization*, and *object localization* are tasks included in *generic object recognition* tasks. These tasks have generally been processed through *training* and *recognition* phases. In the training phase, models of a certain type of object are created from training images. In most cases, the training images should be prepared with their labels that indicate what (and where) objects are in the images. Face detection tasks, for instance, usually requires face image samples and other objects' (backgrounds') samples. These training images are translated from pixel values into *feature vec-*

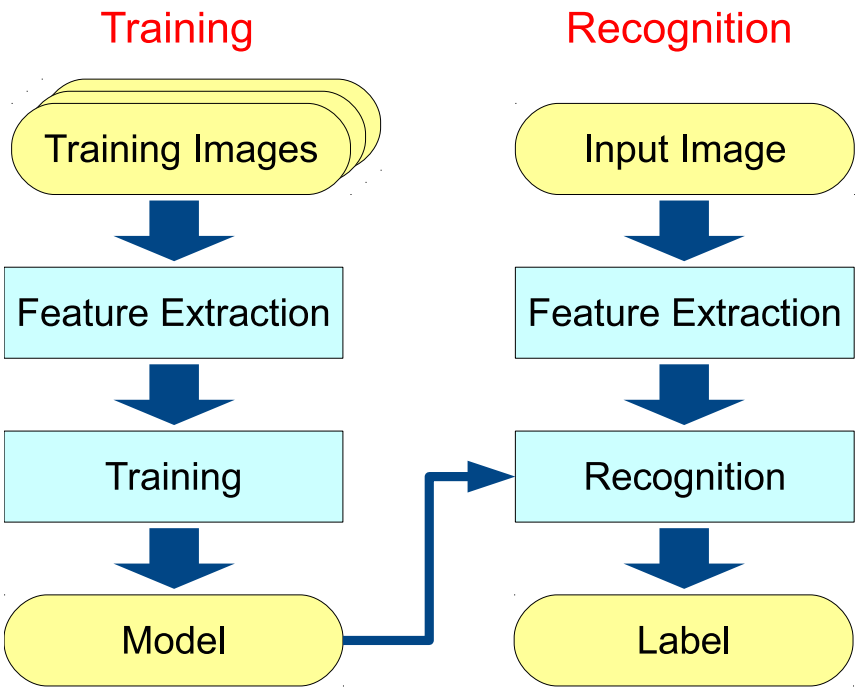


Figure 3.1: The general flow of generic object recognition. At the training phase, feature vectors are extracted from training images and used to create the model of certain object categories. At the recognition phase, a feature vector is extracted from an input image and labeled by using the trained model.

tors that are more suitable representation of models than pixels. The model of the object is created from the feature vectors with such as statistical training, indexing, etc. In the recognition phase, an input image is also translated into a feature vector in the same manner as the training phase. Finally, the label(s) of the input image is determined from the feature vector and the model.

3.1.1 Feature descriptors

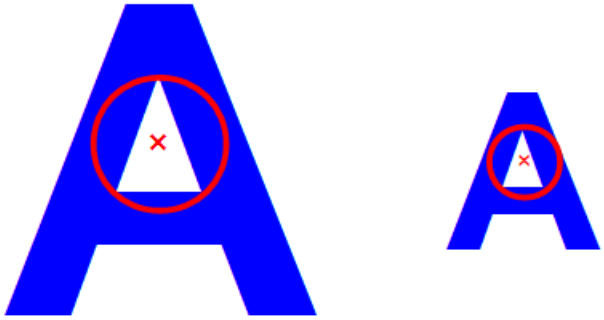
Image feature descriptors are the most popular ways to create feature vectors which are commonly used in both training and recognition phases. A feature descriptor is generally used to represent object texture or shape as a vector. The vector is usually created by computing gradient, difference of brightness between pixels, or spatial frequency, from an image patch. Appropriate ways to generate

features from images depend on types of tasks; therefore, each feature descriptors have been designed for each task.

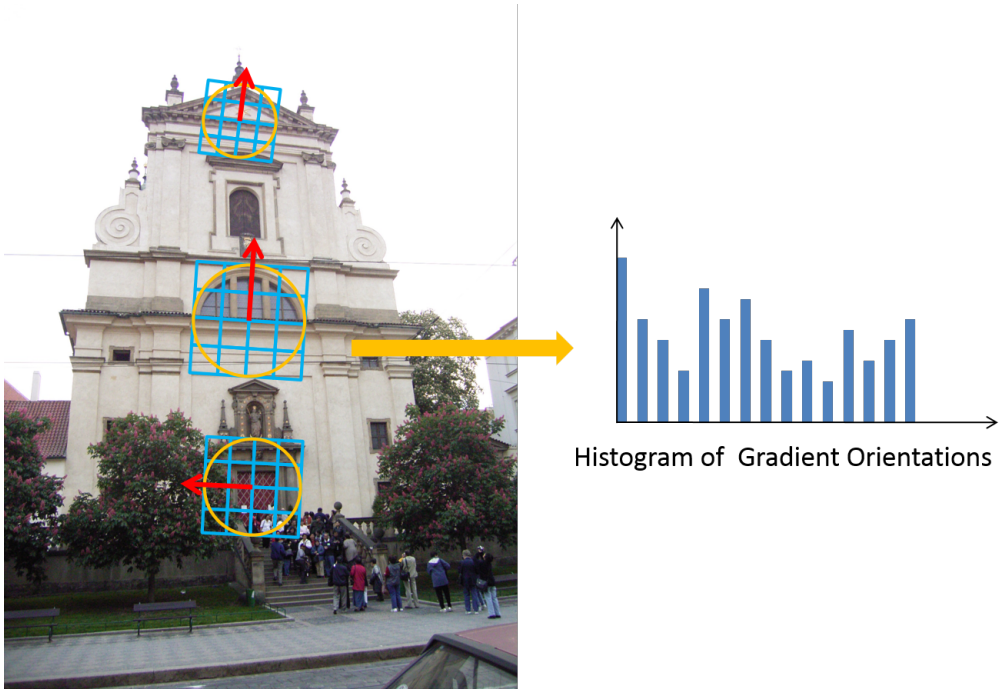
"Scale invariant feature transform"(SIFT)[Lowe, 2004] is the most popular feature descriptor that was originally introduced for specific object recognition tasks. It detects points which are robust to scale changes, and creates descriptions which are invariant to rotation (see Figure 3.2). SIFT is a very powerful technique for an image matching so that a lot of expansions of SIFT have been proposed.[Ambai and Yoshida, 2011; Bay et al., 2006; Leutenegger et al., 2011; Mikolajczyk and Schmid, 2005]

Scale and rotation invariance of SIFT-like features is very useful nature so that the descriptor had become major not only for specific object recognition but for image categorization tasks. For those tasks, these SIFT-like feature descriptors are frequently transformed into other kinds of "high level" descriptors. The well-known example of a "high level" descriptor is called "bags-of-features"(BoF) [Csurka et al., 2004], which is the vector that represents an image as a histogram of visual words that are created from training images in an unsupervised manner (See Figure 3.3). The flow of creating BoF is that: (1) image feature descriptors are extracted from training images, (2) the feature descriptors were clustered into a certain number of clusters to generate visual words which are generally their centroids, and (3) a histogram is created by counting the number of each visual word appeared in an image. The histogram is used as a new feature vector to recognize object. BoF summarized one image to just one vector so that it has also been used in specific object recognition tasks for large scale database[Sivic and Zisserman, 2003]. Other BoF-like feature representations have also been proposed: weak clustering using Gaussian mixture model[Perronnin et al., 2006], Fisher vector[Perronnin and Dance, 2007], Vector of Locally Aggregated Descriptors (VLAD)[Jegou et al., 2010], super-vector[Zhou et al., 2010], and others.

However, different types of feature descriptors have been used in object localization tasks. Haar-like features are popular ones to detect positions of faces with cascaded classifiers[Lienhart and Maydt, 2002; Viola and Jones, 2001, 2002]. Haar-like features simply calculate differences between contiguous square regions by comparing sums of pixel values in each square region. Figure 3.4 illustrates an example of Haar-like features in face detection tasks. Histogram of oriented gra-



(a) Example of a feature point that is invariant to scale and position change



(b) Feature descriptor that is invariant to rotation

Figure 3.2: The way how SIFT achieves scale and rotation invariance.

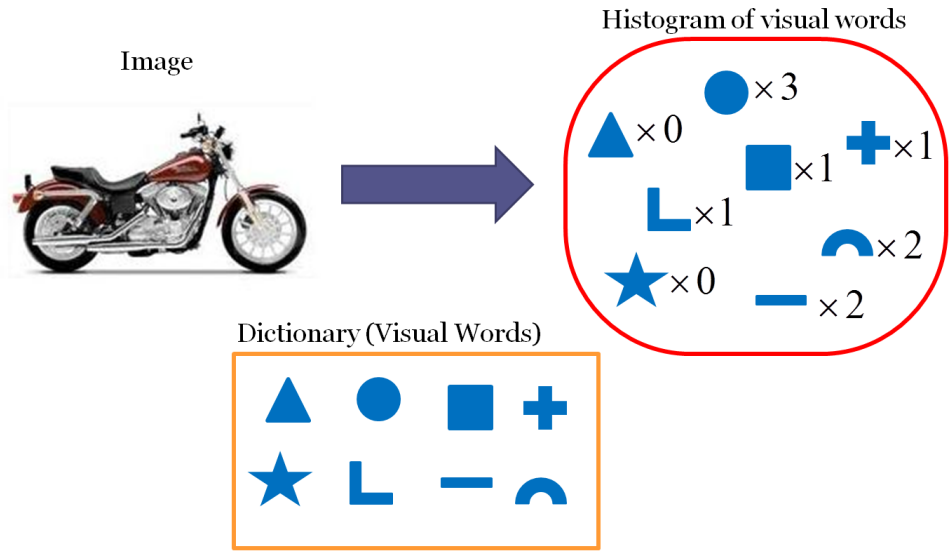


Figure 3.3: Bags-of-features (BoF) which translate an image into a histogram of visual words.

dients (HOG) descriptors[Dalal and Triggs, 2005] are also major features which were originally applied to detection of pedestrians. HOG descriptors are created by concatenating histograms of gradients' orientations in sub-regions which are called *cell*s of an object area as illustrated in Figure 3.5. A lot of other feature descriptors have been proposed to localize objects such as sparse edge of orientation histograms (EOH)[Levi and Weiss, 2004], local binary patterns (LBP)[Ahonen et al., 2006], granular features (SGF)[Huang et al., 2007], Edgelet[Wu and Nevatia, 2007], and others. These features are generally calculated from an area in a sliding window, and then a machine learning algorithm determines whether the target object is in the region or not.

"High level" features for object localization have also been studied, which combine the feature descriptors at several positions. These "high level" features are trained by taking advantage of co-occurrences among "low level" features of target objects. Sabzmeydani and Mori[Sabzmeydani and Mori, 2007] presented *Shapelet* features that combine gradient orientations in sub-regions of a sliding window with AdaBoost[Schapire and Singer, 1999]. Mita et al. combined several Haar-like features into *Joint Haar-like* features[Mita et al., 2008]; the im-

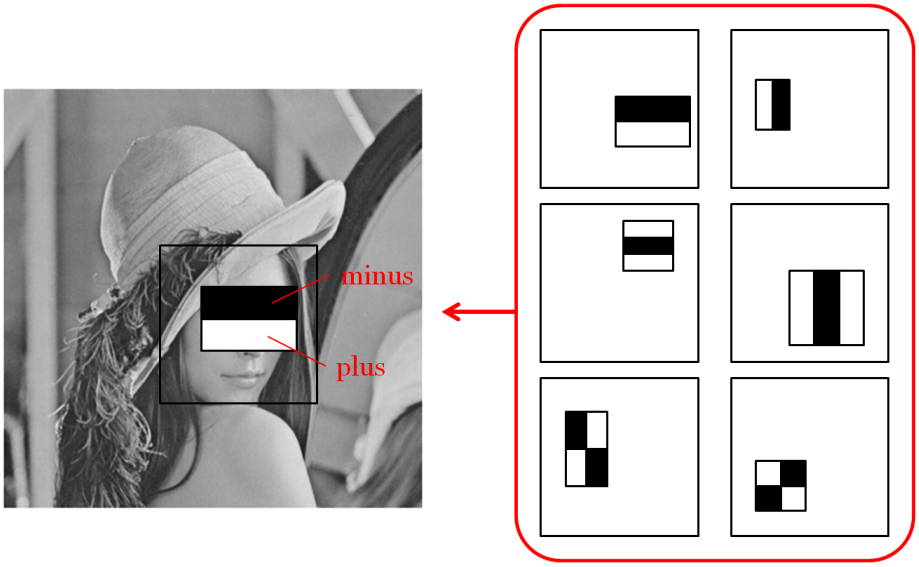


Figure 3.4: Haar-like features which compare sums of pixel values in adjacent square regions. Six example features are shown in the right red box. The suitable features are selected from a huge amount of candidates by AdaBoost.

portant combinations to recognition were selected by AdaBoost. Combinations of HOG descriptors have also been studied such as Joint HOG features[Mitsui and Fujiyoshi, 2009], CoHOG[Watanabe et al., 2010], Co-occurrence Probability Feature (CFP)[Yamauchi et al., 2010] and others. Felzenszwalb et al. presented features that not only included the appearance of object parts but also deformation[Felzenszwalb et al., 2009], which is called *deformable part models*. Figure 3.6 illustrates an example of deformable part models which consists of root HOG filter and part HOG filters with deformation costs.

Contrary to BoF, these "high level" features for object localization are trained in a supervised way. Although these features using labeled data can obtain better quality to detect objects, they never be used for other categories of objects. In contrast, the visual words used in BoF can commonly be used for other categories. Additionally, they do not need labeled data for training.

Of course, the objectives of high level features between categorization and localization tasks are different: to translate the content of one image to one vector for categorization and to create a richer expression of an object for localization.

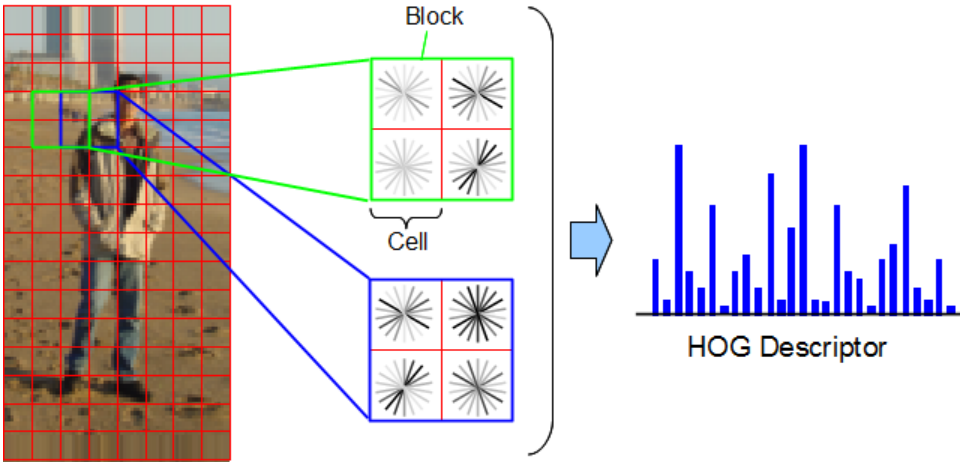


Figure 3.5: Histogram of oriented gradients (HOG) descriptor.

However, this generality and lower cost of training are advantageous in the analysis of multimedia content because it is difficult to predict what kind of category is required.

Another different concept between BoF and the features for object localization is how a model of an object is created. Joint features or deformable part models describe an object as having a similar appearance to their shape model. These similarity based approaches compute likelihoods of a target existence with distances between model shape and image appearances. On the other hand, BoF representation is not similarity but a histogram that describes an object with how many visual words appear in an image. BoF basically disregards where that feature point is. The information on each visual word's position in an image is lost. Although spatial pyramid matching[Lazebnik et al., 2006] recovered the information, it still used a histogram in each sub-region. Histogram based approaches have often been used for object localization tasks with sliding windows[Lampert et al., 2009], where dense sampling of features is needed to create histograms.

HMAX features represent an image as a vector whose elements are similarity values to shape patches in a dictionary. As same as BoF, this HMAX dictionary is created in unsupervised manner from a random set of natural images unrelated to any categorization tasks; therefore, the dictionary that was previously created can be applied to other categories. In addition, similarities to patches in a dictionary

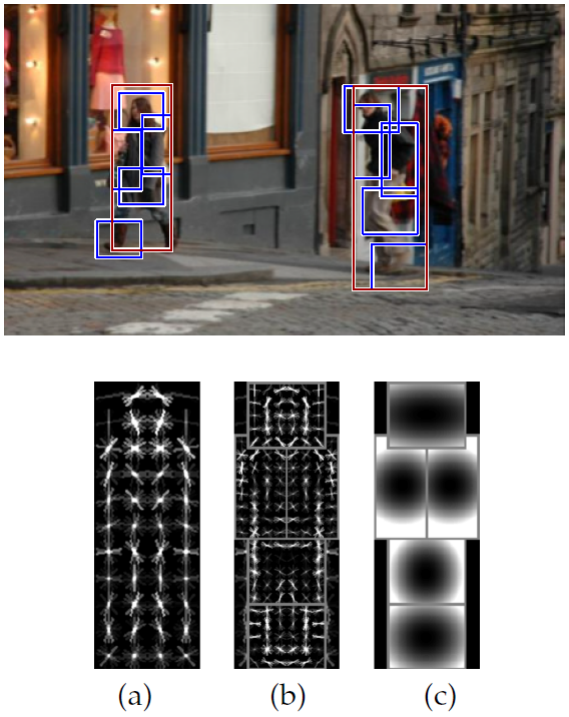


Figure 3.6: The example of deformable part model. Top row illustrates detections obtained with a single component person model. The model is defined by a coarse root filter (a), several higher resolution part filters (b) and a spatial model for the location of each part relative to the root (c). This image is reproduced from [Felzenszwalb et al., 2009].

can be utilized to search object positions; we describe it in Chapter 5.

As well as HMAX, the techniques called *deep learning* are also unsupervised and similarity based approaches[Bengio, 2009], which have been applied to object recognition tasks. Contrary to *hand-crafted features* such as SIFT, SURF, HOG, and others, deep learning statistically learns image features from enormous sets of training data. These approaches have achieved superior performance in competitions not only for object recognition[Deng et al., 2012; Krizhevsky et al., 2012] but speech recognition[Hinton et al., 2012], data mining[Kaggle Inc, 2012a,b], etc. For object recognition tasks, *deep convolutional neural networks* is popular in deep learning approaches. This architecture is similar to the HMAX model, however the deep neural networks learn features of all layers completely in unsu-

pervised way. By contrast, the HMAX learns not all features because Gabor filter, which is hand-crafted feature, is used in the S1 layer. It might be one of reasons that deep learning requires much more training samples than the HMAX does [Le et al., 2011; Serre et al., 2005]. Deep learning also requires huge computational resources for training. In addition, it convolves a lot of trained features for recognition. For these reasons, most of these methods have been implemented on *graphics processing unit* (GPU)[Krizhevsky, 2012] or on a great number of servers[Le et al., 2011]. Because of the similar hierarchical architecture among these methods, the idea of speeding up HMAX is expected to be applied to deep learning too.

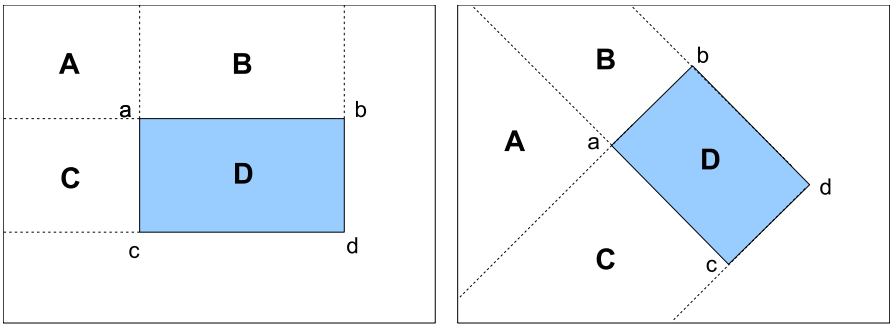
3.2 Speed-up Approaches

This section reviews existing speed-up techniques in object categorization and localization tasks. The techniques are segmented into four approaches: make a feature descriptor faster, eliminate redundant features, restrict regions to calculate, parallel its processing on hardware.

3.2.1 Faster feature calculation

The *integral image* is a popular technique to make descriptors faster[Viola and Jones, 2001, 2002]. Integral image is an intermediate representation for an image, which makes computation of rectangle feature very fast. For example, as shown in Figure 3.7(a), the pixel value at 'a' in the integral image is the sum of pixel values in the rectangle region 'A' of the original image, value 'b' is the sum of values in the region 'A+B', and value at 'd' is the sum in 'A+B+C+D'. Thus the sum of pixel values in the rectangle region 'D' can be easily calculated as $D = a - b - c + d$. Integral image technique was expanded by Lienhart and Maydt to calculate sum of pixels in a skewed rectangle region[Lienhart and Maydt, 2002] as illustrated in Figure 3.7(b); therefore, the area can be calculated by the same means.

The integral image technique is often used to speed up computations of feature descriptors. Computing Haar-like features is the most popular example to use the integral image technique[Lienhart and Maydt, 2002; Viola and Jones, 2002].



(a) integral image (b) skewed integral image

Figure 3.7: An integral image is created by setting each pixel value to the sum of a pixel values in the square region of an original image. For instance, the pixel value at 'a' in the integral image is the sum of pixel values in the region 'A' of the original image. Thus the sum of pixel values in the region 'D' of the original image is calculated very quickly by using four pixel values at 'a', 'b', 'c', and 'd' of the integral image.

Zheng and Liang introduced *image stripe features* to detect cars rapidly[Zheng and Liang, 2009], which were skewed lines and arcs shape features similar to Haar-like features. These shapes were approximated by combinations of rectangular regions in order to use the integral image technique. *Speed Up Robust Feature (SURF)*[Bay et al., 2006, 2008], which is a scale and orientation invariant keypoint detector and feature descriptor like SIFT, can be calculated rapidly with the integral image technique too.

The integral histogram is another technique to compute histograms in rectangular regions rapidly[Porikli, 2005]. Some feature descriptors, such as HOG and SIFT, consist of histogram of gradients in a small block of image, which can use integral histogram for speed-up. An integral histogram is created with almost the same manner as an integral image is: number of pixels which have a certain orientation of gradient are accumulated at each position so that counts in each bin of histogram in rectangular areas can easily be computed with the values at the four corners.

Dollár et al. studied the *integral channel features* that rapidly compute features of multiple channels by using the integral image or histogram techniques[Dollár et al., 2009]. These multiple channels were generated by various transformations

of an input image, such as LUV decomposition, Gabor filtering, edge extraction, creation of gradient histograms, thresholding, and others. Then, important channel features for recognition were selected by AdaBoost in training phases.

Although multi-scale Haar-like features can be calculated without creating image pyramid using integral image, multi scale HOG descriptors cannot be calculated straight forward without image pyramid. Dollár et al. proposed the method to approximate the calculation of multi-scale HOG with smaller number of image pyramid layers[Dollár et al., 2010]. Benenson el al. expanded Dollár's method to calculate HOG descriptors without image pyramid by creating multi-scale model template instead[Benenson and Mathias, 2012].

Another popular approach to create faster descriptors is to represent feature vectors with *binary patterns*. The binary pattern consists of only 0 or 1; therefore, the descriptor of N elements needs only N bits, which occupies very small memory storage. As its binary value is computed, in most cases, by comparing values of pixel pair, it is extracted very fast. Moreover, the distance between two binary features is computed by Hamming distance which simply counts the number of different bits so that feature matching is processed very fast. The binary pattern techniques have been used in many tasks such as corner detection[Rosten and Drummond, 2005], specific object recognition[Alahi et al., 2012; Ambai and Yoshida, 2011; Calonder et al., 2010; Leutenegger et al., 2011; Rublee et al., 2011], object tracking[Lepetit and Fua, 2006; Ozuysal et al., 2010], object localization[Ahonen et al., 2006], and others.

3.2.2 Reduction of redundant features

Number of feature dimension also affects processing times such as extracting feature vectors and matching between features. Ke and Sukthankar proposed *PCA-SIFT*[Ke and Sukthankar, 2004] that reduced the dimension of SIFT descriptors with principal component analysis (PCA); this features achieved faster matching speed.

Sparse feature vectors have a small number of non-zero elements, which can be computed fast because the elements of zero are ignored. This effect is similar to the reduction of feature dimension. Song et al. proposed *sparselet*[Song

et al., 2012] to speed up multi-class object localization with deformable part models[Felzenszwalb et al., 2009]. The sparselet is a dictionary of object parts bases. The dictionary is trained to represent each object parts with combinations of a small number of the bases. Thus, most of the processing times of their method depend on the number of bases but not on the number of object categories. They achieved real-time multi-category object detection with GPU implementation.

AdaBoost[Freund and Schapire, 1997] is one of the most popular methods of eliminating redundant features in object localization tasks[Lienhart and Maydt, 2002; Mita et al., 2008; Mitsui and Fujiyoshi, 2009; Sabzmeydani and Mori, 2007; Viola and Jones, 2002; Wu and Nevatia, 2007; Yamauchi et al., 2010]. AdaBoost is a machine learning algorithm which selects important features to recognition iteratively. As illustrated in Figure 3.8, the strong classifier of AdaBoost outputs weighted sums of outputs of weak classifiers which are bound to corresponding features. The weak classifiers and their weights are selected and trained one by one with positive and negative samples. Viola and Jones applied AdaBoost to select effective Haar-like features for recognition and its weight from huge number of features[Viola and Jones, 2002]. Then, only the selected Haar-like features were calculated in recognition phases, which made much reduction of computing time.

3.2.3 Elimination of processing areas

In most cases, elimination of processing regions have been studied in object localization tasks because the tasks are much time consuming to estimate existence of target object at every position and scale. Most of regions in an image may be background (non-target) area; therefore, rejecting the background at early stage is a key to speed up. Viola and Jones implemented *attentional cascade* which concatenated several classifiers[Viola and Jones, 2002] as illustrated in Figure 3.9. The attentional cascade is trained to reject the obvious background-like regions with classifiers of earlier stages, then regions which are passed from the first to the last classifiers are assumed as target object locations[Viola and Jones, 2002; Wu and Nevatia, 2007; Wu et al., 2004; Zheng and Liang, 2009]. Bourdev and Brandt proposed the *soft cascade* that combined the ideas of the AdaBoost and

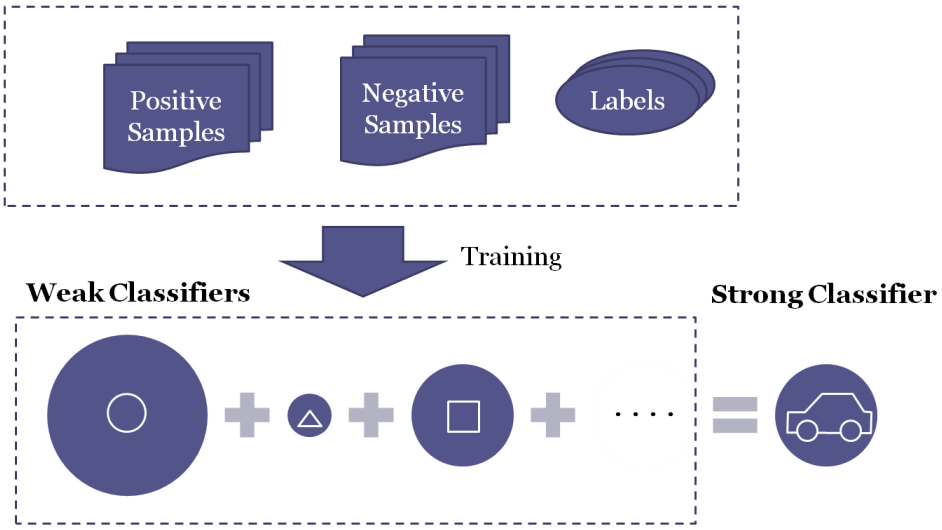


Figure 3.8: Illustration of the AdaBoost: outputs of strong classifier are equal to weighted sums of outputs of weak classifiers. The weights of weak classifiers are trained with positive and negative samples.

the attentional cascade[Bourdev and Brandt, 2005]. Instead of training sequence of stages, the soft cascade is trained as one long stage of weak classifiers with the corresponding thresholds. The regions are rejected at stage t if the sum of outputs from 1_{st} to t_{th} weak classifiers is under the t_{th} threshold. Felzenszwalb et al. applied the cascade approach into deformable part models[Felzenszwalb et al., 2010], in which dot product of input HOG features and each part filter were computed sequentially. If the dot product with earlier part filter was under threshold, then computing with later part was canceled.

Coarse-to-fine approach is also a type of cascade classifier which reject obvious background-like regions at lower resolution. Pedersoli et al. proposed coarse-to-fine object localization approach with HOG descriptors[Pedersoli et al., 2010] and with deformable part models[Pedersoli et al., 2011].

There is another approach to reduce search areas, which is called the efficient subwindow search (ESS)[Lampert et al., 2008, 2009]. ESS defines a parameter space of bounding boxes, and then takes advantage of the *branch-and-bound*

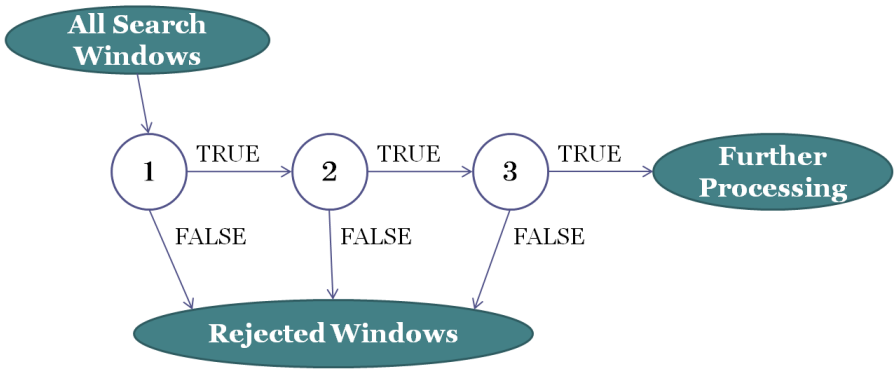


Figure 3.9: Illustration of the attentional cascade: image regions on search windows are processed from earlier stage to later one. The windows which are rejected by classifiers of earlier stages are never processed by later classifiers.

technique to ignore bounding boxes which have low scored parameters.

Key point detectors were used to find salient regions and to restrict number of points on which image feature descriptors are extracted[Ambai and Yoshida, 2011; Bay et al., 2006; Harris and Stephens, 1988; Leutenegger et al., 2011; Lowe, 2004; Matas et al., 2004; Rosten and Drummond, 2005; Smith and Brady, 1997]. For instance, SIFT and other SIFT-like approaches try to detect scale-invariant salient points[Ambai and Yoshida, 2011; Bay et al., 2006; Leutenegger et al., 2011; Lowe, 2004] in specific object recognition tasks. Instead of key point detectors, dense sampling has often been used in object categorization and localization tasks for better performance[Nowak et al., 2006]. However, we applied this idea into object categorization tasks to eliminate computing cost of HMAX in Chapter 4.

3.2.4 Parallel processing

Another idea of speed-up is using a parallel processing architecture. Graphical processing unit (GPU) is designed for computer graphics to render 2D textures or 3D models; however it has also been used in image processing and computer vision tasks. For instance, some major feature descriptors, such as SIFT[Wu, 2007], SURF[Astre, 2012; Schulz et al., 2012],and HOG[Prisacariu

and Reid, 2009], have been implemented on GPU. There are also a lot of computer vision[Eyetap Personal Imaging Lab at University of Toronto, 2012; Institute for Computer Graphics and Vision at Graz University of Technology, 2012; opencv.org, 2012] and machine learning[Lopes et al., 2012; Multimedia Group of Information Technologies Institute (CERTH-ITI), 2012; Srinivasan et al., 2010] algorithms implemented on GPU.

GPU helps to build real time applications. Benenson et al. built pedestrian detection system which worked in more than 100 fps with GPU[Benenson and Mathias, 2012]. Deformable part model (DPM) is the high performance object localization algorithm that requires to convolve several HOG filters[Felzenszwalb et al., 2009, 2010]. Song et al. implemented both the DPM and their modified one using *sparselet*, and then achieved real time recognition[Song et al., 2012].

GPU is also useful to reduce training time of object recognition. Deep convolutional neural network is known as very time consuming algorithm to be trained. Krizhevsky et al. released their GPU implementation of deep neural network implementation named "cuda-convnet"[Krizhevsky, 2012; Krizhevsky et al., 2012].

3.3 Speed-up of HMAX

In this section, we reviewed speed-up approaches applied to computations of HMAX features. These approaches can also be categorized into 1.faster feature calculation (Section 3.2.1), 2. reduction of redundant features (Section 3.2.2), 3. elimination of processing areas (Section 3.2.3), and 4. parallel processing (Section 3.2.4). As seen in the Chapter 2, HMAX model convolves shape filters at each scale and position in order to obtain similarity to each shape, which causes much calculation time. It takes more than 10 seconds to proceed a QVGA size image with 1000 feature patches, which is shown later in Chapter 4. A few works addressed this problem to reduce processing time.

We directly addressed to this problem in 2009[Minagawa and Saito, 2009a,b] by combining some speed-up techniques reviewed in Section 3.2. This approach approximated 2D Gabor filter as Haar-like features for *faster feature calculation*, restricted computation on local peaks for *elimination of processing areas*, and reduced number of feature patterns with clustering for *reduction of redundant*

features. These modifications improved the processing speed of HMAX about 30 times faster. We review the detail procedures of this approach in Chapter 4.

The report of Chikkerur and Poggio[Chikkerur and Poggio, 2011] was also direct work to address the problem of processing speed. They tried several approximation methods for each layer (from S1 to C2) of HMAX, and measured its performance and timing especially for S layers. For speed-up of the S1 layer, they surveyed three approximation approaches. First approach decomposed the 2D Gabor filters into two 1D separable filters. Second one decomposed the 2D filters to row rank 2D filters by singular value decomposition (SVD). Third one approximated the Gobor filters as box filters (Haar-like features) which can be calculated rapidly with the integral image technique. All these three approaches are categorized into the "faster feature calculation". The 1D separable filters and SVD kept its accuracy in the experiments. They reduced its processing time in lower image sizes (less than 256x256) tests, but not in higher ones. In contrast, the box filters reduced processing time to around 0.3-0.4 times, however it also reduced accuracy a little (around 0.5%). For speed-up of the S2 layer, they surveyed two approximation approaches. First approach simply computed the responses at regular grid or fixed number of random positions, which was categorized into "elimination of processing areas". Second one decomposed the shape patches into a certain number of bases with the principal component analysis (PCA), which was categorized into "reduction of redundant features". The sub-sampling approach was better than the PCA a little (less than 3%) in accuracy. The PCA approach was faster in large image size (more than 256x256) tests, and the sub-sampling was faster in small size tests.

The method proposed by Mutch and Lowe[Mutch and Lowe, 2006, 2008] inhibited orientations of weak responses at S1 layer (see Figure 2.6), which means "reduction of redundant features". It also inhibited the feature patches which have small weight trained by support vector machine (SVM)[Cortes and Vapnik, 1995]. The system improved the speed; however "it takes several seconds to process and classify an image on a 2GHz Intel Pentium server"[Mutch and Lowe, 2008].

Huang et al, achieved great result to improve speed of HMAX about 20 times faster with two ideas[Huang et al., 2011] (see Figure 2.7). One is to eliminate processing areas to where gradients of S1 response is over a threshold. Another is

to reduce redundant feature patches by AdaBoost.

Brosch and Neumann applied HMAX into object localization tasks with coarse-to-fine approach to eliminate processing areas[Brosch and Neumann, 2012]. They computed responses in lower resolution by using sliding windows, and then restricted computing regions in higher resolution to peaks in the lower resolution.

We also tried to improve processing time of HMAX for localization tasks [Minagawa and Saito, 2014]. This method focused on "HMAX specific" redundancies of sliding window with coarse-to-fine approach. We review this method in Chapter 5.

HMAX also can be implemented with the "parallel processing". Mutch et al. implemented 'Cortical Network Simulator'(CNS) that is a framework on GPU for cortex like model including HMAX[Mutch et al., 2010].

In this chapter, we reviewed the four speed-up approaches: 1 faster feature calculation, 2 reduction of redundant features, 3 elimination of processing areas, and 4 parallel processing. From next chapter, we reviewed our methods to speed up HMAX model in object categorization and localization tasks. We applied from the first to third approaches into categorization tasks[Minagawa and Saito, 2009a,b], and the second and third ones into localization tasks[Minagawa and Saito, 2014]. Though parallel processing with GPU is so powerful, it has some limitations: implementation difficulty, unsuitable for some types of algorithms, and hardware dependency. Because of these limitations, we avoided this approach.

Chapter 4

Speed-up in Categorization Tasks

The main interest of this chapter is to eliminate processing time of HMAX features for object categorization tasks. In addition, an image based search system is presented to show an example of its application using our modified HMAX.

4.1 Overview

The HMAX features were originally invented for object categorization tasks, which well emulated *rapid categorization* of humans[Serre, 2006]. The HMAX features, however, cannot be rapidly computed by machines. Thus, speed-up techniques of the HMAX have been studied as we reviewed in Section 3.3. We categorized these speed-up techniques into four groups as shown in Table 4.1: faster feature calculation, reduction of redundant features, elimination of processing areas, and parallel processing.

Our work was the first proposal which was directly dedicated to speed-up in computation of the HMAX[Minagawa and Saito, 2009a]. The works of Mutch and Lowe[Mutch and Lowe, 2006], and Huang et al.[Huang et al., 2011] focused on achieving better performance of HMAX and speed-up was ancillary benefit. Our work covered three categories of speed-up, except for parallel processing. It achieved great reduction of processing time into 3.3% and was successfully implemented in web server. The work of Chikkurur and Poggio studied several approximation approaches in each layer of the HMAX[Chikkerur and Poggio,

Table 4.1: Speed-up Methods for HMAX

	Faster Feature Calculation	Reduction of Redundant Features	Elimination of Processing Areas	Parallel Processing
[Mutch and Lowe, 2006]		✓		
[Minagawa and Saito, 2009a]	✓	✓	✓	
[Mutch et al., 2010]				✓
[Huang et al., 2011]		✓	✓	
[Chikkerur and Poggio, 2011]	✓	✓	✓	

2011], which complemented other approaches we did not try.

This chapter reviews how we achieved speed-up in computation of HMAX model for categorization tasks[Minagawa and Saito, 2009a]. Huang et al, revealed that feature selection by AdaBoost was great benefit for speed-up[Huang et al., 2011]; therefore, we also studied the effect of AdaBoost in this chapter.

4.1.1 Preliminary experiment

We first implemented the HMAX method[Serre et al., 2005, 2007] with C++ / OpenCV [opencv.org, 2012] as the "base method" and measured its processing speed in the following environment:

- CPU: Intel Core2 Duo 2.26GHz
- RAM: 3G Bytes
- Image Size: QVGA(240×320)
- Programming Language: C/C++ with OpenCV 1.0

Table 4.2 represents processing time on S1, C1, S2, and C2 layers of the HMAX (see Fig. 2.4), and time to predict by machine learning (ML). The table shows that the base method was much slow: it took about 30 sec, which is not acceptable for applications such as web services. Resizing an image is one of the easiest ways to reduce processing time because the time is usually proportional to

Table 4.2: Processing Speed (sec)

Process	Our Method (AdaBoost)	Our Method (SVM)	Base Method
S1	0.268 ± 0.014	0.268 ± 0.012	0.941 ± 0.016
C1	0.195 ± 0.016	0.195 ± 0.016	0.177 ± 0.014
S2	0.303 ± 0.122	0.524 ± 0.056	28.814 ± 0.167
C2	0.051 ± 0.014	0.053 ± 0.015	0.549 ± 0.480
ML	0.000 ± 0.000	0.001 ± 0.008	0.001 ± 0.009
Total	0.817 ± 0.123	1.040 ± 0.057	30.483 ± 0.563

image size. But even if the image were resized to half (i.e., to 120×160), the base method would still take about 7.5 ($= 30/4$) sec. In addition, to keep hierarchical architecture, there is the lower limit of image size for processing. We therefore tried to increase the processing speed of the algorithm.

4.2 Proposed Method

This section describes how we modified the base method to speed it up. Table 4.2 shows that the slowest processes of the base method are the S1 and S2 processes of the base method. Thus, it was reasonable to concentrate our effort on improving these two processes.

We improved the base method in the following five ways:

1. For decreasing the number of feature patches in S2, summarize features by vector quantization.

2. For restricting the area of calculation in S2, do the calculation only for signals from the points where C1 takes maxima in the local area.

3. For simplifying the S1 processing, replace the Gabor filter of with the Haar wavelet feature.

4. For improving accuracy, retain the position information of each feature of C2.

5. For decreasing the number of feature patches in S2, choose important features by AdaBoost.

1, 2, and 5 of the above is reducing the computation time of S2, and 3 reduces the time of S1 calculation.

4.2.1 Reduction of feature patches

As explained in Section 2.2, all distances between input signals and trained feature patches are calculated in the S2 layer. Accordingly, the number of patches has much influence on processing time for S2. These feature patches are obtained by "imprinting" way: local area in a training image is selected randomly and calculated C1 signals are just saved as a feature patch. Therefore, these patches might include several similar features that are redundant for recognition. To avoid this redundancy, we tried to cluster feature patches using the Linde-Buzo-Gray (LBG) algorithm[Linde et al., 1980] and combined similar ones into one representative. The set of the representative vectors was selected to have the smallest L2 errors in five trials of clustering. Each clustering is done by k-means algorithm[Bishop, 2006] with 10 iterations. In our system, the number of feature patches was reduced from $N_p = 4000$ to $N_p = 200$.

4.2.2 Restriction of S2 calculation area

In the C2 layer, most signals from the S2 layer other than the maximum response of each feature are ignored. Since the outputs from neighbor S2 units are assumed to be similar, it is reasonable to eliminate S2 calculation around some salient points. Like Bags of Features, a lot of other local-feature-based object categorization methods use interest point detectors (e.g., Harris operator, difference of Gaussian)[Kim et al., 2008; Lowe, 1999]. We did not use these well-known detectors, however, because it seemed better to reuse the information already calculated instead of adding a new process to the system. We therefore used the positions in which C1 took local maxima on each orientation (illustrated in Figure 4.1). These positions are represented by the equation below:

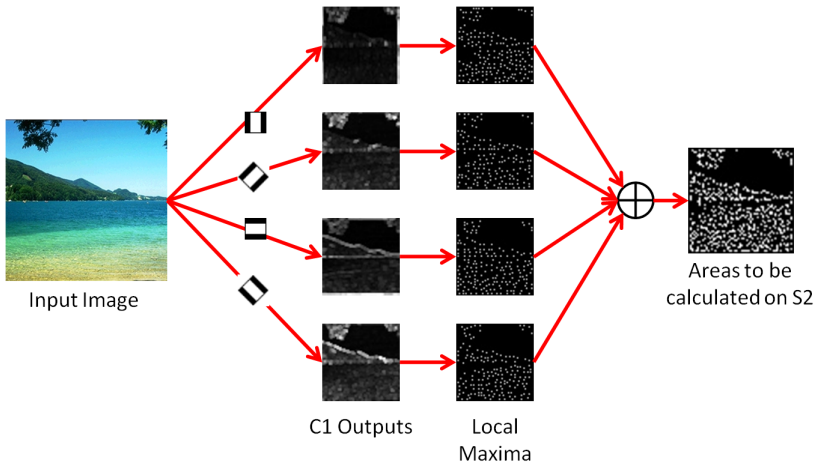


Figure 4.1: S2 calculating area.

$$\bigcup_{\theta_o \in \boldsymbol{\theta}} \{\boldsymbol{x}_{c1} : \boldsymbol{x}_{c1} = \underset{\boldsymbol{x'} \in R(\boldsymbol{x}_{c1})}{\operatorname{argmax}} I_{C1}^{\theta_o}(\boldsymbol{x'})\} \tag{4.1}$$

$\boldsymbol{x}_{c1}$ is a position in C1output, $\boldsymbol{\theta}$ is a group of 2D Gabor filter orientations, $I_{C1}^{\theta_o}$ is the output of C1 at orientation θ_o, and $R(\boldsymbol{x}_{c1})$ are eight neighborhoods of $\boldsymbol{x}_{c1}$.

4.2.3 Simplification of S1 process

The Haar wavelet is much simpler than the Gabor function. Viola and Jones stated that Haar-like features can be calculated very rapidly by using "integral image" [Viola and Jones, 2001] and Lienhart and Maydt expanded Viola and Jones's approach by adding skewed Haar feature [Lienhart and Maydt, 2002]. We therefore approximated the Gobor function by Haar features.

Examples of Haar features are shown in Figure 4.2. The output of the Haar filter is calculated by subtracting the sum of the pixel values in the black areas from the sum of the pixel values in the white areas.

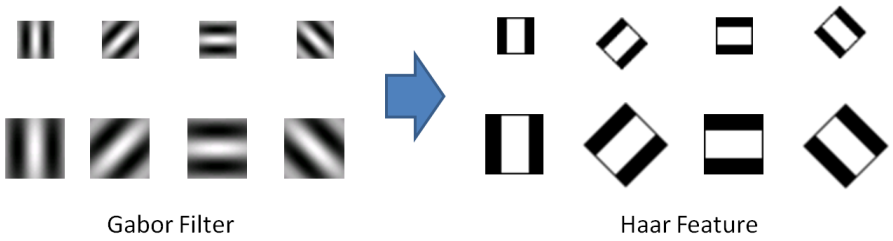


Figure 4.2: Gabor filter and Haar feature.

4.2.4 Retention of feature position

In the C2 layer, all S2 signals are combined to N_p elements, which means that feature location information is lost. Mutch et al. expanded the hierarchical model of [Serre et al., 2005] to retain feature positions in order to improve recognition rate [Mutch and Lowe, 2006]. This approach is effective because it keeps a certain level of position invariance in the C layer. In our approach, C2 signals are divided to $N_{c2} \times N_{c2}$ areas, and integrated N_p-dimensional vectors are calculated at each area. The overlap ratio of contiguous region is $\Delta N_{c2} \in [0 - 1]$. Finally, an output of the C2 layer is a vector which has $N_{c2} \times N_{c2} \times N_p$ elements.

4.2.5 Selecting important features by AdaBoost

As described in section 4.2.1, the S2 process takes a long time to compute the distances between input image and all feature patches. If the calculation of distances can be limited to only "important" features, it might be possible to reduce processing time without decreasing accuracy. For instance, Viola and Jones's selected the Haar-like features from enormous feature pools effectively by using AdaBoost [Viola and Jones, 2001]. For this reason, we implemented the feature selection approach of AdaBoost.

AdaBoost is a machine learning algorithm in which weak classifiers and their weights are selected by training a strong classifier [Freund and Schapire, 1997].

$$H(x) = sign\left[\sum_{t=1}^{T} \alpha_t h_t(x)\right] \tag{4.2}$$

In Eq. (4.2), each weak classifier $h_t(x)$ discriminates input signal x and outputs the result as 1 or -1. A strong classifier $H(x)$ sums these outputs of $h_t(x)$ with weight α_t.

In the training stage, weak classifiers and their weights are chosen one by one iteratively. Viola and Jones[Viola and Jones, 2001] bound each weak classifier to a Haar-like feature, and selected features effective for object detection.

In our approach, a weak classifier is bound to each element of the C2 output vector. We used "classification and regression tree (CART)" [Bishop, 2006] as a weak classifier and trained it as a depth-1 decision tree. Training steps are:

1. Prepare $N_{c2} \times N_{c2} \times N_p$ classifiers of CART, which are trained by each element of the C2 output vector.

2. Select classifiers and their weights through AdaBoost training.

3. Decide which feature patches should be used in the S2 layer.

The number of features are reduced in this manner.

4.3 Experiments

4.3.1 Implementing the base method

We first implemented HMAX model as a base method with C++ and OpenCV [opencv.org, 2012]. The parameters of S1, C1, S2, and C2 in the base method are the same as in [Serre et al., 2005, 2007], and linear support vector machine (SVM)[Cortes and Vapnik, 1995] was used as a machine learning algorithm. As explained in Section 2.2, feature patches were created from the Caltech-101 image set[Fei-Fei et al., 2004].

Table 4.3 shows the recognition rates of our base method, the recognition rates reported in Serre's original paper [Serre et al., 2005, 2007], and the recognition rates reported in Fergus's paper [Fergus et al., 2003]. In this experiment, we created ten image sets for training and testing, by selecting randomly from each Caltech-101 image category. Each training image set contained 40 positive and 50 negative samples, and each test set contained 50 positives and 50 negatives.

Table 4.3: Recognition rate of base method (%)

Category	Base method	[Serre et al., 2005]	[Fergus et al., 2003]
Cars	97	99.8	88.5
Faces	93.1	98.1	94
Airplanes	97.8	94.9	90.2
Motorcycles	95.7	97.4	92.5

The number of training samples was also determined to be the same as in [Serre et al., 2005, 2007]. Serre et al., studied the influence of the number of training samples on the performance of the HMAX model and reported that the HMAX model requires relatively the small number of training samples[Serre et al., 2005, 2007]. These images were resized to a height of 140 pixels. The difference between the recognition rates obtained with the base method and those obtained with the method described in the Serre's paper might be due to factors such as the different training images, the random selection of test images, the different program environments, and unknown parameters.

In the following section, our approach is compared to this base method in terms of recognition rate and processing speed. The experimental parameters of our approach are listed in Table 4.4.

4.3.2 Comparison with base method

Processing speed

We measured the process time of the base method and our modified method by using 25 QVGA(240×320) images in each of four categories (airplane, car, face, motorbike). The results listed in Table 4.2 show that our method reduced the processing time from 30 sec to about 1 sec at SVM case and about 0.8 sec at the AdaBoost case. If input image size is the same, the number of calculation steps of the base method is the same for any kinds of images. In our methods, in contrast, the number of calculation steps of the S2 layer can be affected by the number of interest points. When the AdaBoost is used, process time for the S2 layer is also

Table 4.4: Experimental Parameters

C1 layer			S1 layer
Scale band	Spatial pooling grid ($N_{c1} \times N_{c1}$)	Overlap ΔN_{c1}	Haar filter size
1	8×8	0.5	3×3 5×5
2	10×10	0.5	7×7 11×11
3	12×12	0.5	14×14 17×17
4	14×14	0.5	20×20 22×22
5	16×16	0.5	25×25 28×28
6	18×18	0.5	31×31 34×34
7	20×20	0.5	36×36 39×39
8	22×22	0.5	42×42 45×45

C2 layer		S2 layer	
Output grid size ($N_{c2} \times N_{c2}$)	Overlap ΔN_{c2}	Model feature size	Model feature number N_p
5×5	0.25	8	200

affected by the number of selected weak classifiers. Hence, coefficient of variation (= standard deviation / average) is largest (=0.075) for the AdaBoost case and smallest (=0.009) for the base method. The fluctuation time of our method, however, is only 0.12 ms. Our approaches are therefore sufficiently fast and stable.

Recognition rate

In this experiment, we used four categories in Caltech-101 database[Fei-Fei et al., 2004] to evaluate tradeoff between speeds and recognition rates. Caltech-101 was one of the popular datasets in image categorization tasks, which has 101 different object categories and one background/clutter category. The dataset totally consists of 9146 images, which have roughly 300 x 200 pixels. The base method had been evaluated by using four categories (airplanes, cars, faces, motorbikes) of Caltech-101[Serre et al., 2005]. Thus, we adopted the same categories to evaluate our implementation. Example images of these categories are shown in Fig. 4.3. Four object categories were trained with 40 positive images and 50 negative ones in the "background" category of Caltech-101. These trained classifiers were tested on 50 positives and 50 negatives. Training and testing were repeated five times with randomly selected images. All images used in this experiment were normalized to a height 240 pixels (the size most often used in mobile phones).

The results listed in Table 4.5 show that our approaches decrease the overall accuracy of the basic method only slightly and that this decrease is due largely to the results for images in the "faces" category. This reduction was caused by accumulating the errors of approximation of S1 features, reduction of processing area on the S2, and divide of C2 area; it is explained in detail at later of this section.

Comparing to the AdaBoost case with the SVM one, one sees that the feature selection of AdaBoost did not only improve processing time for 0.2 sec (see Table 4.2) but also increase the recognition rate slightly.

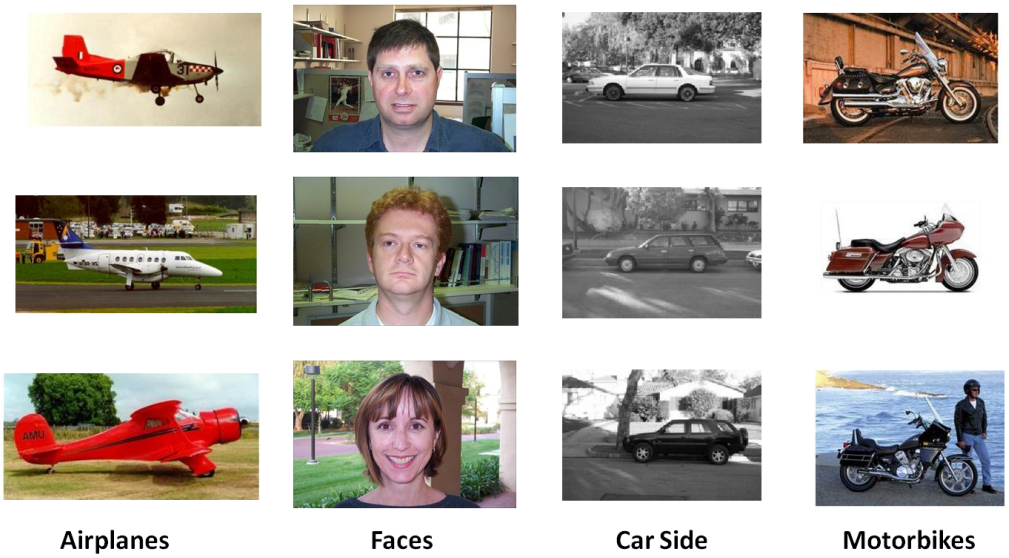


Figure 4.3: Examples of Caltech-101 image sets.

4.3.3 Evaluation of each modification

Process time and recognition rate

We tried the methods in which each modification was eliminated, to evaluate contributions of our improvements to the base method. Recognition rates obtained in this experiment are listed in Table 4.6 and process times are listed in Table 4.7. "Without VQ" means that feature patches were created without vector quantization. The vector quantization is not mentioned in Table 4.7 because it has no

Table 4.5: Recognition Rate (%)

Category	Our Method (AdaBoost)	Our Method (SVM)	Base Method
Cars	99.0	99.2	96.6
Faces	89.6	88.0	94.3
Airplanes	97.4	98.4	97.9
Motorcycles	97.2	95.6	96.2
Average	95.8	95.3	96.3

effect on the processing steps in the recognition phase. "Without POI of S2" is the method in which RBF distance of feature patches are calculated at all positions on the S2 layer, not using interest points described in Section 4.2.2. "Gabor Filter" means the method that uses the Gabor filter instead of the Haar feature. Trade-off between recognition rate and processing speed of each change can be seen by comparing Table 4.6 with Table 4.7.

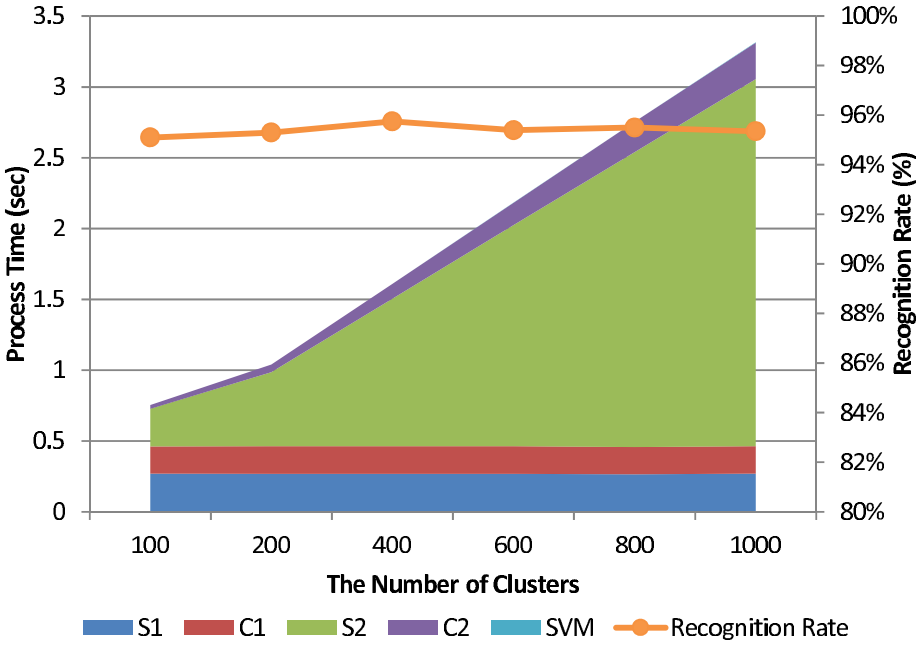
The vector quantization in our SVM and AdaBoost cases has improved the overall recognition rate slightly. According to [Serre et al., 2007], clustering of feature vectors might decrease the recognition rate; but that was not the case in this experiment. As mentioned in Section 4.2.1, vector quantization might combine similar features and improve efficiency when the number of features is small. In our case, the number of features must be small in order to improve speed. Therefore the vector quantization approach worked well.

Restriction of the calculation area on the interest points reduced the S2 layer processing time by about 50% in both the SVM case and the AdaBoost case but reduced the recognition rate by only 0.6 and 0.3% in each case. Replacing the Gabor filter with the Haar-like feature sped the S1 layer processing by about 30% in both cases with reducing the recognition rate by only 0.5% in the SVM case and with increasing by 0.2% in the AdaBoost case.

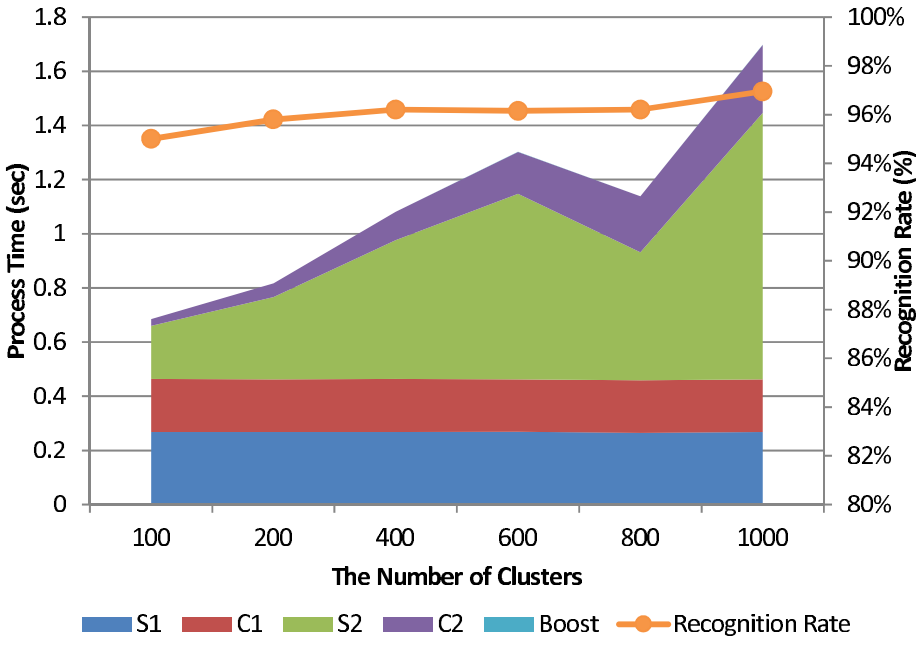
The number of clusters

The relations between the number of clusters(=the number of feature patches) and the recognition rates and processing times are shown in Figure 4.4. In the SVM case, the S2 process time increases almost linearly with increasing numbers of features (Fig. 4.4(a)). The speed in the S2 layer is also affected by the number of features in the AdaBoost case (Fig. 4.4(b)), but less than it is in the SVM case. This might be because the S2 process time in the AdaBoost case is not directly affected by the number of clusters but is affected by the number of selected weak classifiers.

Considering the trade-off between recognition rate and processing speed (Fig. 4.4), we used a relatively the small number of features ($N_p = 200$).



(a) SVM case



(b) AdaBoost case

Figure 4.4: Process times and recognition rates for each number of clustering. The process times are represented by stacked line graph.

Table 4.6: Evaluation of our improvements in recognition rate (%)

(a) SVM case

Category	Our Method (SVM)	Without VQ	Without POI of S2	Gabor Filter
Cars	99.2	99.4	99.6	98.6
Faces	88.0	88.0	89.6	90.8
Airplanes	98.4	98.2	98.0	98.0
Motorcycles	95.6	92.2	96.2	95.6
Average	95.3	94.5	95.9	95.8

(b) AdaBoost case

Category	Our Method (AdaBoost)	Without VQ	Without POI of S2	Gabor Filter
Cars	99.0	99.0	99.4	99.2
Faces	89.6	88.4	91.0	90.0
Airplanes	97.4	97.6	97.2	97.4
Motorcycles	97.2	96.2	96.8	95.6
Average	95.8	95.3	96.1	95.6

The number of C2 segments

Figure 4.5 shows the relations between the size of C2 segmentation (See Sec. 4.2.4) and the recognition rate and process time. The segmentation size was changed from 1×1 to 5×5. This graph indicates that C2 segmentation improved recognition rate well at the cost of only a small increase of process time. Although this shows the effectiveness of our improvement, this approach must be applied carefully because segmentation makes recognition sensitive to the positions of objects in images. The recognition rates of "face" decreased in the larger size of segmentation as seen in Fig. 4.5 because the face category has relatively larger variation of positions than the other categories have in this dataset. In addition, the approximation of S1 features (Sec. 4.2.3) and the restriction of S2 calculation area (Sec. 4.2.2) also reduced the recognition rate of face as seen in Table 4.6. The accumulations of these errors caused the overall reduction of recognition rate of face category in Table 4.5.

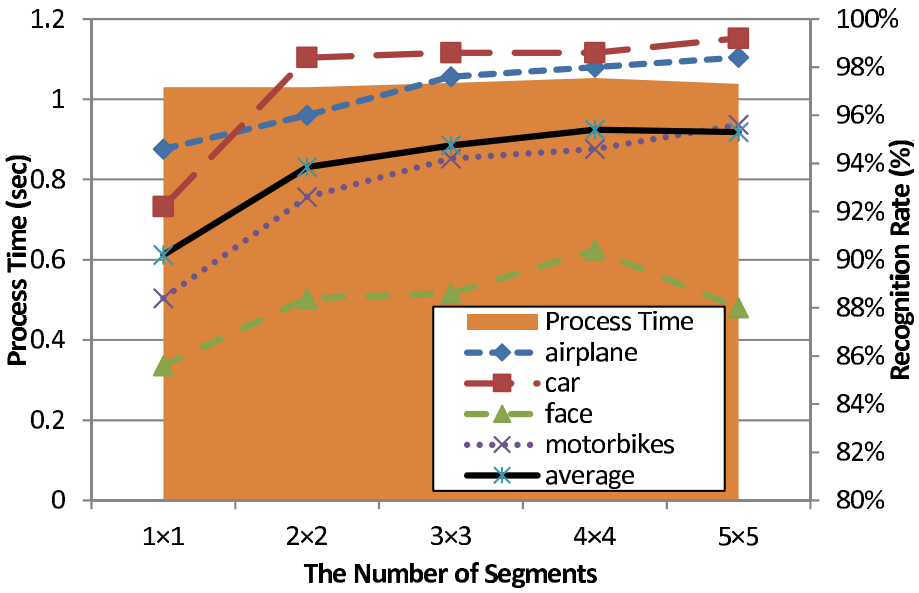
Table 4.7: Evaluation of our improvements in process time (sec)

(a) SVM case

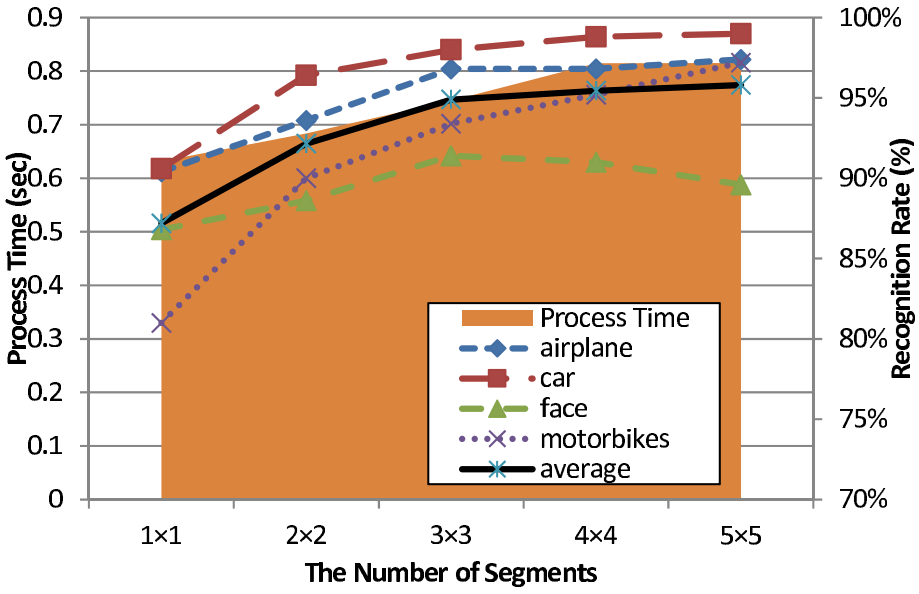
Process	Our Method (SVM)	Full area of S2	Gabor Filter
S1	0.268 ± 0.012	0.266 ± 0.007	0.943 ± 0.025
C1	0.195 ± 0.016	0.172 ± 0.011	0.195 ± 0.024
S2	0.524 ± 0.056	1.190 ± 0.014	0.494 ± 0.072
C2	0.053 ± 0.015	0.035 ± 0.013	0.061 ± 0.032
ML	0.001 ± 0.008	0.001 ± 0.009	0.001 ± 0.009
Total	1.040 ± 0.057	1.665 ± 0.018	1.694 ± 0.092

(b) AdaBoost case

Process	Our Method (AdaBoost)	Full area of S2	Gabor Filter
S1	0.268 ± 0.014	0.267 ± 0.011	0.942 ± 0.019
C1	0.195 ± 0.016	0.175 ± 0.023	0.196 ± 0.036
S2	0.303 ± 0.122	0.528 ± 0.210	0.236 ± 0.073
C2	0.051 ± 0.014	0.036 ± 0.015	0.057 ± 0.016
ML	0.000 ± 0.123	0.000 ± 0.000	0.000 ± 0.000
Total	0.817 ± 0.123	1.006 ± 0.209	1.430 ± 0.080



(a) SVM case



(b) AdaBoost case

Figure 4.5: Process times and recognition rates for each number of C2 area segments.

4.3.4 Scene category

Our method was also evaluated with other categories than objects. Scene images (e.g., mountain, coast, city, etc.) are major categories in internet services like photo-sharing services or social network services (SNS). We therefore tried our approach with eight categories of scene image sets [Oliva and Torralba, 2001], examples of which are shown in Figure 4.6. The scene dataset includes 2688 images classified as 8 categories: 360 coasts, 328 forest, 374 mountain, 410 open country, 260 highway, 308 inside of cities, 356 tall buildings, and 292 streets. The size of each image is 256 x 256 pixels.

Both the SVM and AdaBoost approaches were trained with 80 positive samples and 100 negative samples of each category. Negative samples were composed of 30 background images of the Caltech-101 database and 10×7 images from the other categories. Table 4.8 shows recognition rates of each scene category. We added Oliva and Torralba's work [Oliva and Torralba, 2001] in the table just for a reference, though their way of evaluation was different from ours: they separately evaluated natural scenes (coast, forest, mountain, open country), and man-made scenes (highway, inside city, street, tall building) in confusion matrix with multi-category classifiers. In this experiment a recognition rate of about 85% was obtained in each case. Though required accuracy is up to a type of application, this could be used for some entertainment purposes such as fortune-telling.

Compared with Oliva and Torralba's work, the recognition rate of "mountain" decreased in both the SVM and the AdaBoost cases. Of course, it is not fair to compare our method with theirs because test methods are different. However, we guess that it was caused by combination of some approximations such as the Haar features and the restriction of S2 area because both their and our methods divided images and computed spatial frequency in each region.

Some works of scene category recognition showed that local feature descriptors were powerful tools[Fei-Fei and Perona, 2005; Lazebnik et al., 2006]. In these works, both scene textures described by local features and the positions of these features have been used for recognition. In our methods, feature patches might represent texture information and C2 segmentation might represent the position of that information.

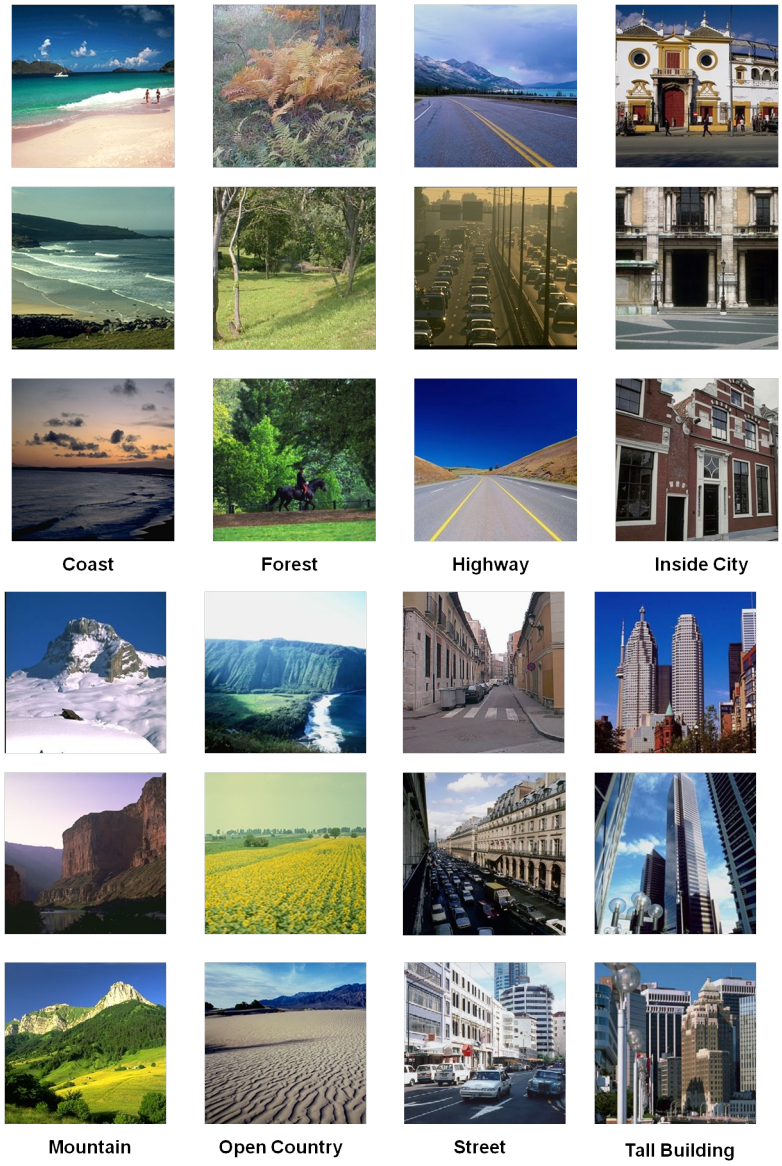


Figure 4.6: Examples of scene category image set.

Table 4.8: Result of scene category image (%)

Category	AdaBoost case	SVM case	[Oliva and Torralba, 2001]
Coast	83.5	83.5	88.6
Forest	89.5	87.5	91.5
Highway	79.5	84.5	91.6
Inside city	84.0	82.5	87.8
Mountain	77.5	78.0	91.2
Open country	85.0	84.5	85.2
Street	89.5	87.5	89.6
Tall building	91.0	91.0	88.0
Average	84.9	84.9	89.2

4.4 Application Example

4.4.1 Application overview

This section describes an example of application using our speed-up HMAX model, i.e., an image based information retrieval system. This system receives images as queries and returns information which is related to contents of query images. This means that a user can obtain information from what he or she sees by sending images which are captured with a camera of mobile phone. For this reason, such systems have been increasingly released because market of mobile phones have been expanded in all these years.

Specific object recognition techniques are the most popular ones that have been used in mobile image search; for instance, users can access to prices, reputation, online stores, and so on by shooting photos of book covers, CD jackets, and other products[A9.com Inc, 2012; Google Inc, 2012a; kooba, 2012]. Face recognition is also used to retrieve images of a specific person[Facebook.com, 2012; Google Inc, 2012b], and optical character recognition is used to add indexes, or tags, to an image database [Evernote, 2008].

We implemented an object category recognition technique in an information retrieval system[Minagawa and Saito, 2009a,b]. This technology would help users

of photo sharing and social network services find people who uploaded the same kind of images and would help service providers make money with advertisements linked to image contents. Object category recognition techniques have also been released into markets in recent years[foo.log Inc., 2013; Google Inc, 2012b, 2013].

It has been known that response time of web services is an important factor to keep users[Zona Research, 2001]. Thus, we implemented our speed-up HMAX model in web system and evaluated it.

4.4.2 System architecture

We assume an image based information retrieval system with our modified HMAX model as shown in Figure 4.7. The flow of this system begins when end user send query image to the system via web or mail interface. Then system forwards it to object category recognition engine, and obtain the category information of image. Finally, this system searches information of its category from database or internet, and returns it to user. This system would help the users of photo sharing and social network services find people who uploaded the same kind of images and would help service providers make money with advertisements linked to image contents.

To make the system more flexible and avoid having to retrain all categories whenever a new image category is added, we adopted several binary classifiers rather than a multi-class classifier and this system shows all result candidates to a user.

4.4.3 Operational test

For operational testing of our approach, we implemented this object category recognition program (SVM version) on a web server, which could be accessed through the internet. Table 4.9 explains an environment of that server.

In this experiment, users could send their images to the server through web form (see Figure 4.9), and receive its result. When user clicked the word on the result page, that page was jumped to Google Image Search. This system could return all candidates because input image was processed by several 2 class object category recognition engines.

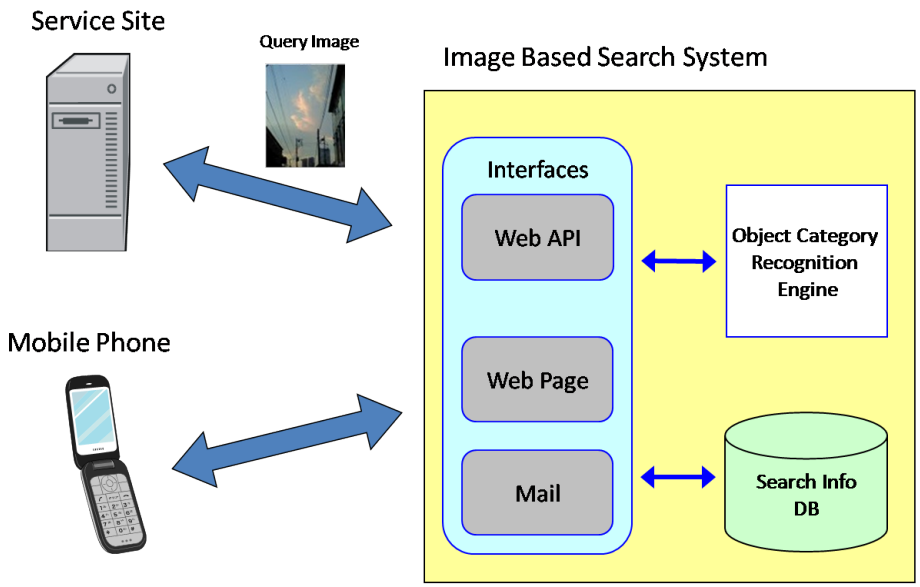


Figure 4.7: Search system overview. The system receives an image as a query, then returns information related to the category of the image.

The recognition engines were trained with 8 scene categories of Oliva and Torralba[Oliva and Torralba, 2001], and dog & cat images were gathered by crawling through the internet (See Figure 4.8). Images in these categories are popular in internet photo sharing services, and so on. An engine of pet category was trained with these 55 dog images, 24 cat images, and 100 negative images.

We opened this web site to some patients, and the system received and processed 40 images for scene category, and 60 images for pet category. Recognition rate is shown in Table 4.10. At scene category test, we assumed it correct if right category name was included in candidates indicated by the system.

In natural scene category test, only 55% of recognition rate is obtained. The most cause of this problem is that the system answered wrong category to non-scene images: if an input image was not belonged to any categories, false positive rate would be tied to the number of engines. In some cases, it is better to reply wrong candidates than nothing, since user can avoid to miss the correct answer. In this situation, object category recognition engine limits the volume of information for user. However, it is concern if a lot of recognition engines might cause user's

Table 4.9: Environment of search system

Server	IBM eServer x306
CPU	Intel Pentium4 3GHz
RAM	2GB
Web App Server	Apache Tomcat 6.0

Table 4.10: Recognition rate of operational test (%)

Category	Our approach (SVM)	Base Method
Natural Scene	54.8	54.8
Dog & Cat	71.4	67.9

confusion by the large number of candidates. In pet category test, it is around 70% of recognition rate. It might be caused by more variety of illumination, object size, and position than Caltech-101 images.

We measured the response time of each request on this system, from reception of request to reply the candidates in Java servlet, thus latency of network was not included. It was average 0.78 sec, and standard deviation was 0.18: this is enough practical processing speed.

4.5 Conclusions

This work increased processing speed of the HMAX model significantly at the cost of only a small decrease of accuracy. We did this by using vector quantization and the concept of interest points, by replacing Gabor filters with Haar features, by retaining the position of C2 signals, and by the feature selection. We confirmed that these approaches could work on scene category as well.

When recognition error is not a critical issue and that the number of object categories and poses can be restricted, this algorithm could be used for some kinds of practical purposes (e.g., entertainment, fortune telling, sales campaign, etc.).

Wider use of our approach will require further increases in processing speed

Figure 4.8: Examples of training image (pet animal).

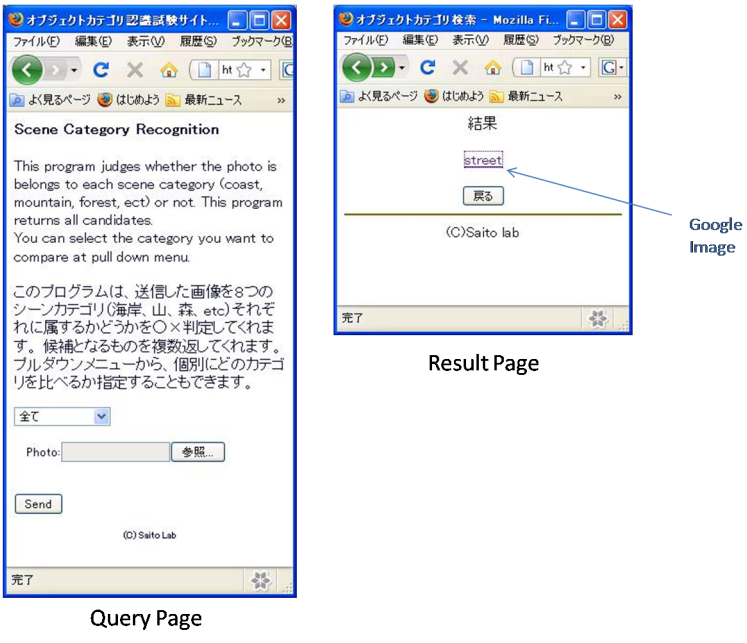


Figure 4.9: An example of search and result page.

and improvement of recognition rate, as well as its extension to the treatment of multiple objects in one image.

Chapter 5

Speed-up in Localization Tasks

5.1 Overview

We reviewed our approach for reducing computational costs of the HMAX in image categorization tasks in the previous chapter. In this chapter, we review our approach for speeding up the HMAX in object localization tasks. The object localization tasks have more difficulties than the categorization tasks because they have to extract more information from images, i.e., objects' positions and scales.

If we aim to apply the HMAX features to object localization tasks, in which the categorization tasks are repeatedly performed by sliding windows, their processing time increases enormously. Of course, the speed-up of categorization tasks also improves the speed of localization tasks; however, some redundancies still remain in the sliding window approach. The redundancies are derived from the natures of HMAX computation: convolutions, max-pooling, and multi-scale features. Our approach focused on eliminating these "HMAX specific" redundancies of the sliding window approach. The sliding window approach estimates existence of target objects at each position in images. On the other hand, our method searches the similar image regions to the shapes that the object has. This idea enabled to narrow the search area from coarse to fine and avoid duplicate processes in overlapping regions between sliding windows and in overlapping scales.

In the next section, we explain what the redundancies of HMAX in the sliding window approach are. A description of our improvements to HMAX for ob-

ject localization is given in Section 5.3. Section 5.4 then describes evaluations of our proposal, where we focus on comparing our method to a sliding window approach with HMAX and clarify the impact of our improvements. We discuss limitations with our approach and present some ideas on improvements in Section 5.5. Finally, Section 5.6 summarizes and draws conclusions on our approach for speed-up of localization.

5.2 Redundancies of HMAX in Sliding Windows

The procedures of the sliding window approach with HMAX follow the steps below:

1. Set windows over all positions of an input image.
2. Trim the image regions in the windows.
3. Process the trimmed images through the S1 to the C2 layer.
4. Predict existences of the object by a machine learning algorithm using the C2 features.

In multi-scale object localization tasks, the input image is resized larger or smaller and processed with the same sliding window size in the same steps above. Notice in step 3 that input images are decomposed to multi-scale information at the S1 layer, and all scale information are gathered into a single scale at the C2 layer. The conventional HMAX based localization approaches have taken this approach[Bileschi and Wolf, 2005; Huang et al., 2011; Mutch and Lowe, 2008].

The method of speeding up HMAX described in the previous chapter is for image categorization tasks. Of course it can also contribute to localization tasks; however, the sliding window approach using HMAX still has three redundancies as seen in Figure 5.1 (see red words). First, the overlapping areas between sliding windows are computed by duplicate processes through the S1 to the C2 layer. Second, as HMAX computes an image by multi-size convolution filter, the process may be duplicated for multi-scale localization. Finally, max-pooling at the C2 layer of HMAX only takes the maximum similarity value to each patch over all

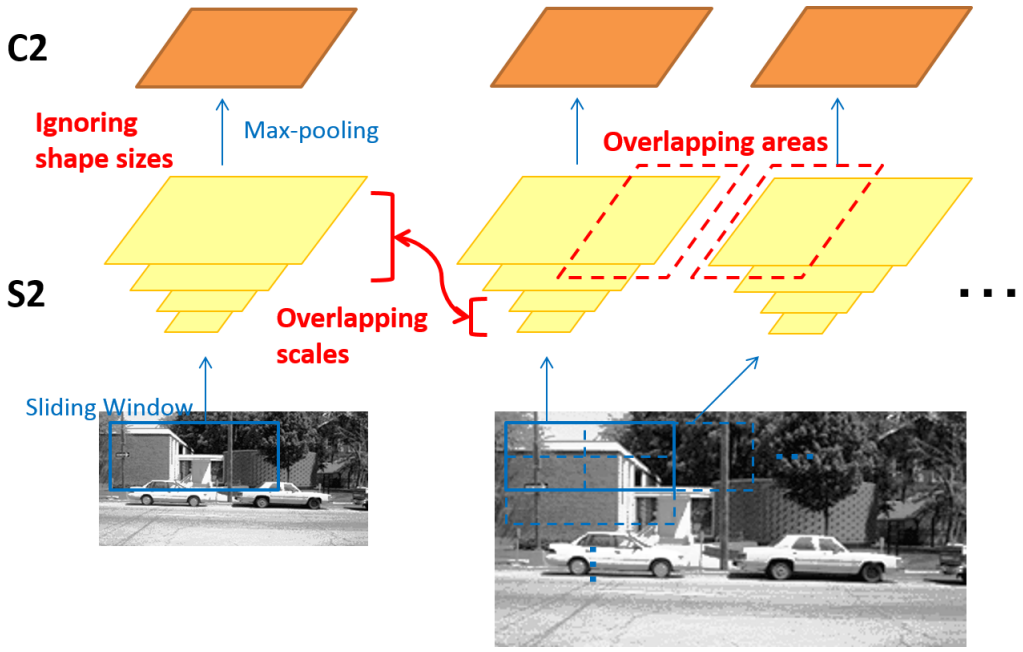


Figure 5.1: Redundancies in sliding window approach using HMAX.

scales and ignores the rest, which means information on patch size is discarded. Unlike the categorization task, a localization task should make use of scale information to recognize an object's size. Thus the max-pooling over scales is unnecessary for localization tasks. These redundancies are derived from natures of the HMAX computation which have convolutions, multi-scales, and max-pooling.

Eliminating overlapping areas and scales in Figure 5.1 contributes to speed-up directly by avoiding duplicate processing. We took a simple approach to eliminating the overlapping areas by filtering a whole image region at one time. The overlapping scales are also omitted by sharing processes of between small filters in larger objects and large filters in smaller objects. We also addressed "ignoring shape sizes" in Figure 5.1 by avoiding the max-pooling over scales in the C2 layer. It does not directly contribute to speed-up because the computing cost of the max-pooling is not expensive. However, the information on shape sizes can be used to estimate a target's position and size in a coarse-to-fine manner to reduce processing times.

Although the "Branch-and-bound" technique[Lampert et al., 2009] is also the major method of eliminating the calculation area without sliding windows, our approaches are suitable for eliminating the redundancies in the HMAX. In addition, the "filtering a whole image region" and the "coarse-to-fine" techniques are also suitable for similarity based features like the HMAX. For instance, the HOG[Dalal and Triggs, 2005] is also similarity based features which compute distances between filters and appearances. The deformable part model[Felzenszwalb et al., 2009] calculates the HOG features and scores of a support vector machine (SVM) classifier in a whole image without sliding windows, and Pedersoli et al. took advantage of these scores for coarse-to-fine object localization[Pedersoli et al., 2011].

Brosch and Neumann also applied a coarse-to-fine approach to reduce the processing region for their combined features of HMAX and HOG[Brosch and Neumann, 2012]; the method computed the likelihood of an object's presence at each sliding window on a coarse scale and then estimated the positions at local peaks of the likelihood on a fine scale. Thus, their coarse-to-fine approach still had redundancies over positions and scales: the overlapping areas in sliding windows and the duplicate filtering between coarse and fine scales in multi-scale object localization tasks (see Figure 5.1).

This chapter explains how to speed up the HMAX model by avoiding these redundancies in the sliding window approach. Because this chapter focuses on localization specific problems with HMAX, our approach is able to be expanded to other HMAX based models [Chikkerur and Poggio, 2011; Huang et al., 2011; Minagawa and Saito, 2009a; Mutch and Lowe, 2008].

5.3 Proposed Method

The main idea of our approach to eliminate the redundancies in HMAX for localization tasks is that the image region of a target object should have high degrees of similarity to the shapes that the object has. The idea was inspired by the work of Chikkerur et al.[Chikkerur et al., 2010] who created a saliency map to simulate top down attention of humans by making use of shape in images. Therefore, our method defines a model of a target as a group of shapes. These

shapes are searched from large to small in an image. In other words, smaller shapes are not searched where larger shapes have not been found. This idea of searching similar parts is in contrast to the idea of the sliding window approach, which is to check an object's existence at each position. Consequently, our model not only avoids overlap redundancies in sliding windows but also restricts the processing area for small shapes.

Figure 5.2 overviews our approach to detect single scale objects. The left side in Fig.5.2 represents the flow to search the large shapes, and the right side does the small shapes. The confidence map $\boldsymbol{I}_{prob}$ expresses probability of target existence at each position, which is computed from the similarities between the image appearance and the shapes. The candidate area $\boldsymbol{M}$ is calculated by thresholding $\boldsymbol{I}_{prob}$. The search of smaller shapes are executed only above the candidate area $\boldsymbol{M}$.

5.3.1 Detection

Object model

The object must be defined as a combination of feature patches $\boldsymbol{P}_n$ (see Equation (2.3)) at each scale band to search an object with its shape; this patch groups of an object are defined as an *object model* in this paper. Figure 5.3 illustrates an example of an object model that has two scale bands. $\boldsymbol{Pg}_i$ is a set of feature patches $\boldsymbol{P}_n$. The shapes of a target are searched from large to small in an image; if similar shapes of large size are not found, then the search for smaller ones is canceled. However, similar areas to feature patches are searched through an image pyramid from low to high resolution in actual implementation while the sizes of patches are fixed, for the purpose of faster processing. Because patches become relatively larger in smaller images, the lower row in Figure 5.3 represents the search of larger shapes more than the higher row does. The bottom and top in Figure 5.3 correspond to the flows on the left and right of Figure 5.2.

The four main differences between the basic HMAX model described in Section 2.2 and the object model defined in this section are summarized as follows:

- The basic model creates multi-scale information with multi-scale Gabor fil-

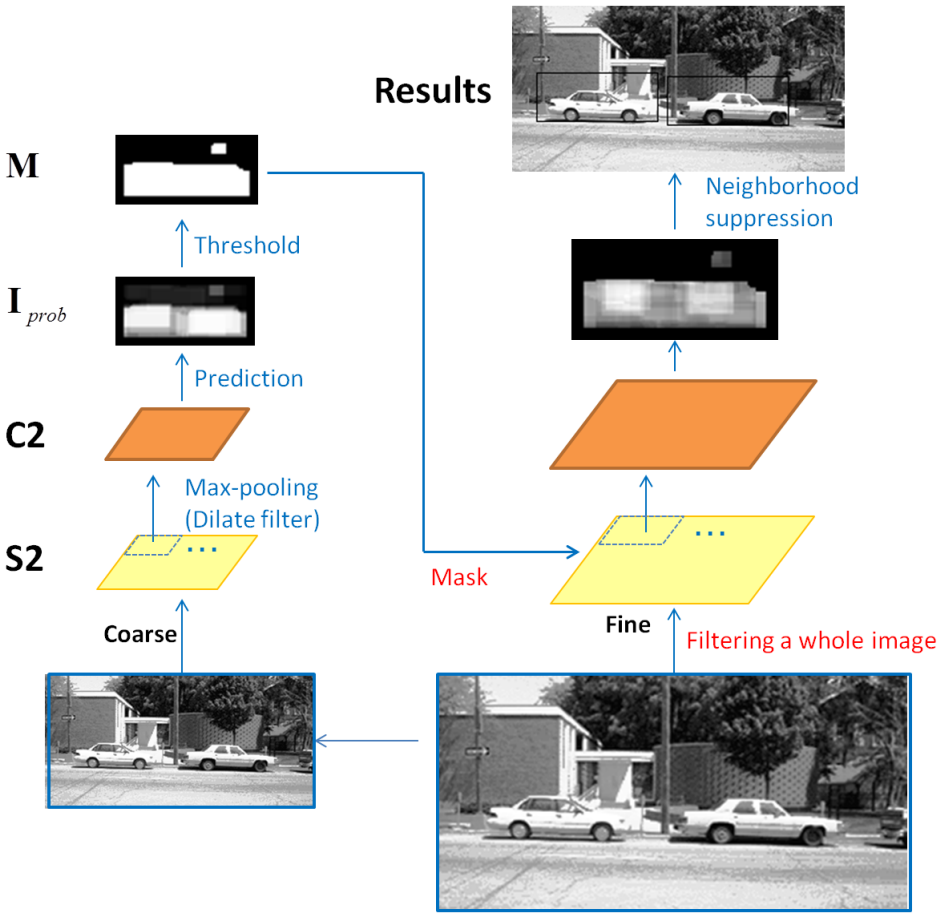


Figure 5.2: Coarse-to-fine localization.

ters at S1; our model creates it with image pyramids.

- The basic model processes each scale band simultaneously; our model processes it from a coarse to a fine resolution.
- The basic model applies the same feature patches to all scale bands; our model applies different ones to each.
- The basic model takes the maximum response of each patch over all scale bands and categorizes it with a classifier; our model has a classifier at each scale for coarse-to-fine processing.

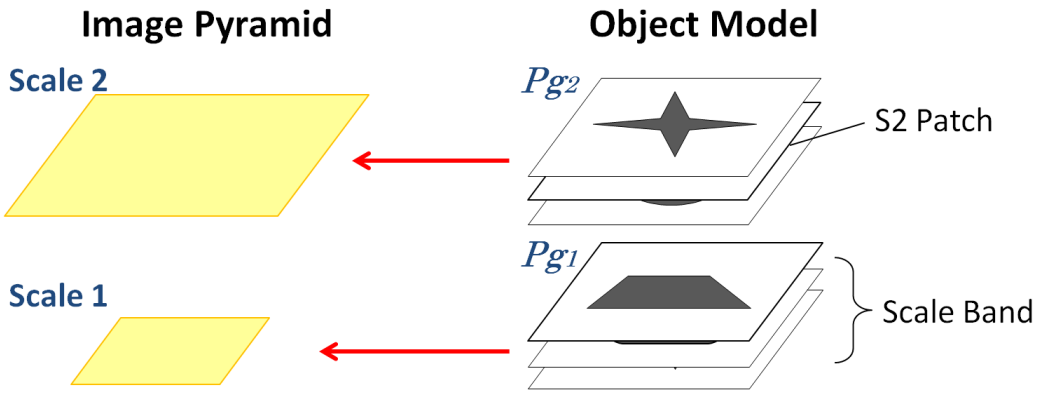


Figure 5.3: Object model and coarse-to-fine search.

As a result, the object model can also be assumed to have a basic HMAX model, which only has a single-scale band at the S2 layer, at each scale band.

Flow of localization

Pseudo code of single scale object detection is figured in Algorithm 1.

Candidate regions $\boldsymbol{M}$ of the target object are narrowed down from coarse to fine, as shown in Figure 5.2. First, an object model that has already been trained (see Section 5.3.2) must be prepared. Here, the object model $\boldsymbol{\Omega}$ is assumed to have l scale bands. The i_{th} scale band of the object model $\boldsymbol{\Omega}$ has group of feature patches $\boldsymbol{Pg}_i$, classifier C_i, and threshold T_i as properties. Then, image pyramids of l scales, $\boldsymbol{I}_1 \ldots \boldsymbol{I}_l$, are created from an input image $\boldsymbol{I}$, which are ordered from small to large. The image pyramids are proceeded from 1_{st} to l_{th} scale band in masked region $\boldsymbol{M}$, which is narrowed down at each scale.

Mask $\boldsymbol{M}$ is updated at the i_{th} scale in three steps. At the first step, similarities between image $\boldsymbol{I}_i$ of pyramids and all patches $\boldsymbol{P}_{i,j}$ in the group $\boldsymbol{Pg}_i$ are computed with Equation (2.3). At the second step, a confidence map $\boldsymbol{I}_{prob}$ that represents the probability of a target existing at each position is estimated by the classifier C_i. Finally, candidate region $\boldsymbol{M}$ is updated by binarizing $\boldsymbol{I}_{prob}$ with the threshold T_i. Only regions over threshold are used for processing at the next $(i+1)_{th}$ scale band. If i is equal to final scale band l, the peaks of the confidence map $\boldsymbol{I}_{prob}$ are extracted by neighborhood suppression[Agarwal et al., 2004]. Finally these peaks

Algorithm 1 Object Localization

Require: an input image $\boldsymbol{I}$

Require: an object model $\boldsymbol{\Omega}$ that has l scale bands: the i_{th} scale band has the following properties

–Group of feature patches: $\boldsymbol{Pg}_i = [\boldsymbol{P}_{i,1} \dots \boldsymbol{P}_{i,N_t}]$

–Classifier: C_i

–Threshold: T_i

Create image pyramids: $\boldsymbol{I} \rightarrow \boldsymbol{I}_1 \dots \boldsymbol{I}_l$

Computing Mask $\boldsymbol{M} \leftarrow \mathbf{1}$

for $i = 1$ to l **do**

  Resize the mask $\boldsymbol{M}$ to the same size of $\boldsymbol{I}_i$

  for all position $\boldsymbol{x}$ where 1 in $\boldsymbol{M}$ **do**

    $\boldsymbol{I}_i \rightarrow \boldsymbol{I}_{S1}$ on the S1 layer by Eq. (2.1).

    $\boldsymbol{I}_{S1} \rightarrow \boldsymbol{I}_{C1}$ on the C1 layer by Eq. (2.2).

    $\boldsymbol{I}_{C1} \rightarrow \boldsymbol{I}_{S2}$ on the S2 layer with $\boldsymbol{Pg}_i$ by Eq. (2.3).

    $\boldsymbol{I}_{S2} \rightarrow \boldsymbol{I}_{C2}$ on the C2 layer by Eq. (5.3).

    $\boldsymbol{I}_{C2} \rightarrow \boldsymbol{I}_{prob}$ by Eqs. (5.1) and (5.2) of the classifier C_i .

    Binarize $\boldsymbol{I}_{prob}$ by threshold T_i then output $\boldsymbol{M}$.

  end for

end for

De-noise $\boldsymbol{I}_{prob}$ by smoothing

Compute peak position $\boldsymbol{X}_{obj}$ of $\boldsymbol{I}_{prob}$ by neighbourhood suppression.

return $\boldsymbol{X}_{obj}$

are assumed to be the positions of target objects.

HMAX features are extracted from resized input image $\boldsymbol{I}_i$ in the similarity calculation step. The whole image area is proceeded through the S1 layer to the C2 layer on the mask $\boldsymbol{M}$ to avoid redundancies in overlapped sliding windows. $\boldsymbol{I}_{S1}, \boldsymbol{I}_{C1}, \boldsymbol{I}_{S2}, \boldsymbol{I}_{C2}$ in Algorithm 1 correspond to the outputs of the S1, C1, S2, and C2 layer. The similarities to each patch are computed at the S2 layer and, as described in the previous chapter, it is the most time consuming process. Only important patches in a *feature dictionary*, which has randomly gathered N_p patches (see Section 2.2), are used to reduce the time for computation. These important patches for recognition are able to be selected with machine learning methods[Huang et al., 2011]. Real AdaBoost[Schapire and Singer, 1999] was adopted in our implementation for two reasons: first, the number of feature patches could be arranged via its number of iterations in training, and second it returned contin-

uous values that were transferred to probabilistic values. However, other types of machine learning algorithm (e.g., sparse linear SVM[Bi et al., 2003]) can be used here too. These important patches are grouped as $\boldsymbol{Pg}_i$ at each scale band.

The combination of real AdaBoost and a logistic sigmoid function is defined as classifier C_i that is used to compute confidence map I_{prob} from HMAX output $\boldsymbol{I}_{c2}$. Equations (5.1) and (5.2) respectively explain real AdaBoost and the logistic sigmoid function.

$$H(I_{C2}(x,y)) = \sum_{j=1}^{N_t} h_j(I_{C2}(x,y)) \tag{5.1}$$

$$I_{prob}(x,y) = \frac{1}{1+\exp(-\frac{\alpha}{N_t}H(I_{C2}(x,y)))} \tag{5.2}$$

The strong classifier, H, of AdaBoost generally returns binary value -1 or 1 for classification tasks by using a sign function; however, the function has been eliminated here to return continuous values from $-\infty$ to ∞. Here, h_j is a weak classifier that has been trained with similarity values of certain feature patch $\boldsymbol{P}_{i,j}$. The N_t is the number of feature patches selected in this scale band. The returned value from the strong classifier in Eq. (5.2) turns to a range from zero to one due to the logistic sigmoid function. The α is the amplifier to strong classifier H, which we set to 2.0.

Dilate filter for max-pooling

Our approach computes a whole image from the S1 to the S2 layer to avoid duplicate processing in overlapping regions between sliding windows; however, the same procedure cannot be applied into the C2 layer. The C2 layer takes the maximum similarity value of each S2 feature patch over all positions and scales, hence the information of the positions and scales of the object is lost. Thus, the size of max-pooling must be restricted by the object size W_i to sustain the information. This max-pooling process is equal to a dilate filter in morphological operation, which is written as:

$$I^j_{C2}(x,y) = \max_{(x',y')\in R(W'_i)} I^j_{S2}(x+x', y+y') \tag{5.3}$$

where $R(W'_i)$ is an object region whose center is $(0,0)$ and width is W'_i, which is the object size on the S2 layer at the i_{th} scale. j is the ID of feature patch.

Figure 5.4 illustrates why max-pooling is represented by a dilate filter. The '+' at the top of Figure 5.4 is the position where feature patch A takes the maximum similarity value in the object region of 'c', and '-' is where feature patch B takes the maximum similarity value; therefore, the C2 output at position 'c' is a vector that includes the values of A at '+' and of B at '-' as its elements. The bottom left and the bottom right in Fig.5.4 correspond to the S2 outputs computed from feature patches A and B (see Eq. (2.3)). We can see that the values at position 'c' are equal to those at the '+' of A and at the '-' of B by processing each S2 output by the dilate filter (Eq. (5.3)).

The target position on the input image is sustained by this dilate filter at the C2 layer; however, the positions of feature patches in the object region have still been lost. Some expansions to the HMAX model have divided max-pooling regions on the C2 layer to take into consideration the positions of shapes for categorization tasks. Mutch and Lowe[Mutch and Lowe, 2008], and Chapter 4 revealed that considering the positions of features in an object area improved accuracy. They divided an image into $W_{c2} \times H_{c2}$ areas on the C2 layer, and then calculated the maximum value of similarity to each patch in each area. Therefore, the number of feature vector dimensions was expanded from N_p, which is the number of feature patches in a dictionary, to $N_p \times W_{c2} \times H_{c2}$; then, important feature patches and their positions were also selected by real AdaBoost.

"Divided" max-pooling can also be implemented as dilate filter as well as Eq. (5.3). Figure 5.5 outlines an example of a max-pooling region that has been placed at the top left in a 2×2 divided object region with overlap ratio ΔN_{c2}; the overlap ratio must be $0 \leq \Delta N_{c2} < 1$. The 'c' is the center of the object region and '$\hat{c}$' is the center of the max-pooling region. The cx and cy are the coordinates of c from '$\hat{c}$'. Then, the max-pooling can be assumed to be a dilate filter with size W''_i and anchor point (cx, cy), which is expressed as:

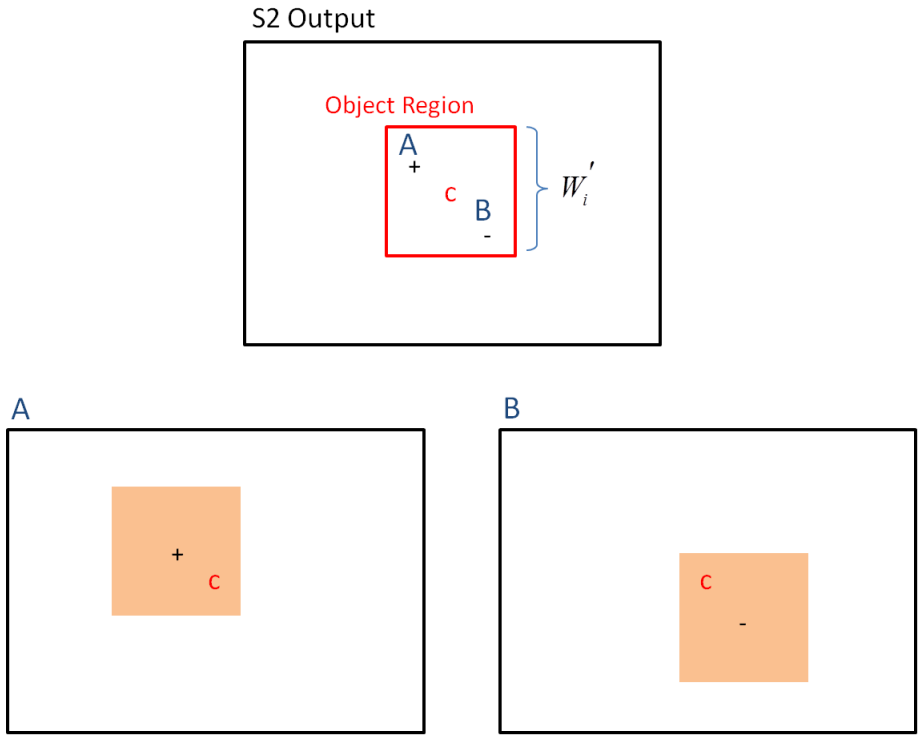


Figure 5.4: Max-pooling with dilate filter. Top: max-pooling on S2 layer in object region 'c' with local maximum similarity points of patches A and B. Bottom: Dilate filter to S2 outputs of patches A and B.

$$I_{C2}^{j}(x,y) = \max_{(x',y')\in R(W_i'')} I_{S2}^{j}(x+x'-cx, y+y'-cy) \tag{5.4}$$

Size W_i'' is calculated as:

$$W_i'' = \frac{W_i'}{W_{c2} + (1 - W_{c2})\Delta N_{c2}} \tag{5.5}$$

Multi-scale localization

Algorithm 1 only explains object detection of fixed size, but it is easy to expand our method to multiple sizes. Figure 5.6 illustrates the detection of two object sizes. An object model has a parameter, R_m, which defines the shape size ratio in scale bands (see Section 5.3.2). The object in Figure 5.6(a) is searched

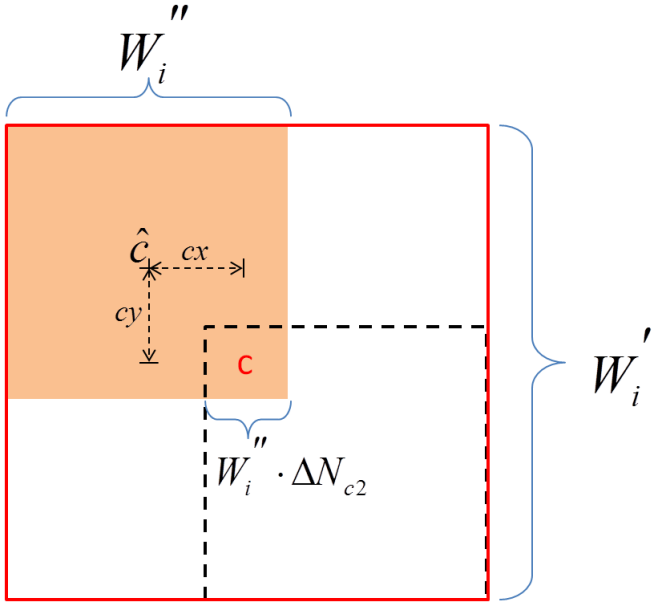
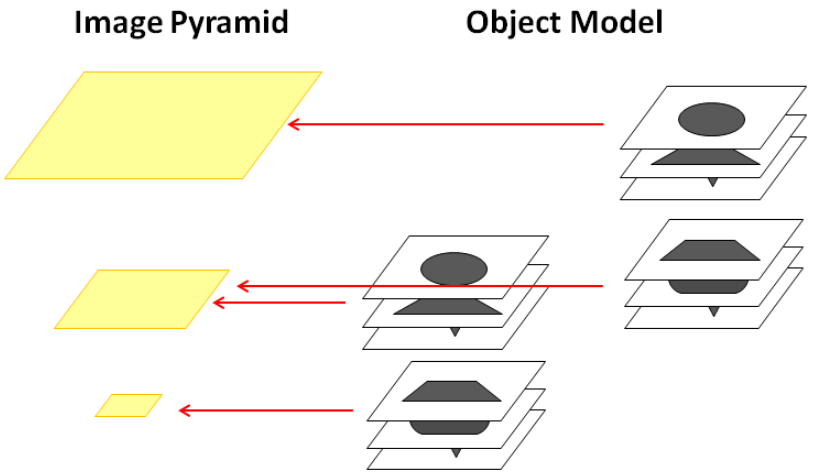


Figure 5.5: Max-pooling for an object region divided by 2×2 with dilate filter.

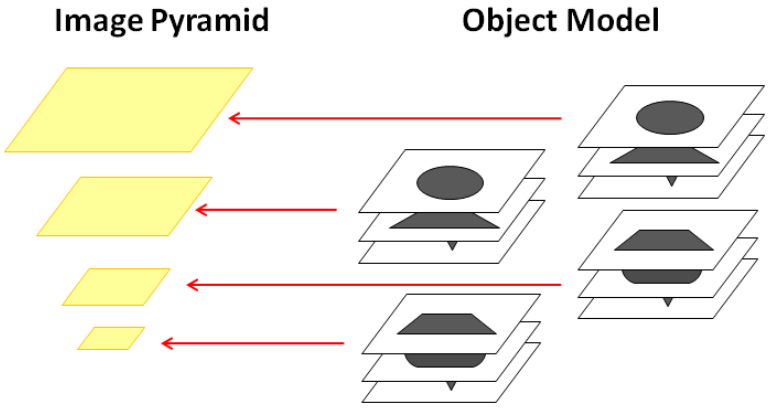
at every scale ratio R_m, which is equal to that of object model; the processing of the second scale of the image pyramid is shared between two sizes of the object. This shared computation between multi-scale objects eliminates the redundancies over scales in Figure 5.1. The calculation cost does not increase linearly for this reason. Figure 5.6(b) illustrates a case where the object is searched at every scale ratio, which is smaller than ratio R_m of the object model. An image pyramid is created in this example by the ratio $\sqrt{R_m}$. When a ratio of image pyramid is set to $\sqrt[3]{R_m}$, $\sqrt[4]{R_m}$, ..., a scale ratio of the target size to be searched becomes increasingly smaller in the same way.

5.3.2 Training

This section explains how an object model was built. We wanted to design an object model to reject background regions at lower resolution stages because rejected regions on lower scales are never processed in higher ones. Smaller images are also less expensive to compute. This approach is similar to that for the cascaded classifier[Viola and Jones, 2002] and the training method is thereby also



(a) Case where size ratio of object model is same as scale ratio of image pyramid for search



(b) Case where size ratio of object model is larger than scale ratio of image pyramid for search

Figure 5.6: Multi scale object detection.

Algorithm 2 Training of Object Model

Require: $P \leftarrow$set of positive training samples
Require: $N \leftarrow$set of negative training samples
Require: $W_1, ..., W_l$: Image pyramid sizes of scale ratio R_m
Require: d: The minimum acceptance detection rate per scale
Ensure: Create the object model $\mathbf{\Omega}$
  Object Model: $\mathbf{\Omega} = [\Omega_1, ...\Omega_l]$
  for $i = 1$ to l **do**
    Resize P to the size $W_i \rightarrow P_i$
    Resize N to the size $W_i \rightarrow N_i$
    $D_i = \{P_i, N_i\}$: set of all training samples on scale i
    $D_i \rightarrow D_{C2}$ by HMAX model through the S1 to C2 layer.
    Train the classifier C_i by D_{C2} and select patch features $\boldsymbol{Pg}_i$.
    Evaluate D_{C2} by trained classifier C_i
    if $i < l$ **then**
      Set threshold T_i to satisfy d
    else
      Set threshold T_i to 0.5
    end if
    Set $W_i, \boldsymbol{Pg}_i, C_i, T_i$ to Ω_i
    $N \leftarrow$set of false positive samples with threshold $T_i - \epsilon$
  end for
  return $\mathbf{\Omega}$

similar in three respects. First, training is started from earlier to later stages and each stage has a classifier and a threshold. Second, the threshold of each stage is determined to satisfy the minimum rate for acceptance detection. Third, only negative training samples that are not classified correctly by a trained classifier of this stage are used for training at the next stage.

A pseudo code is shown in Algorithm 2.

First, image pyramids of training samples that consist of positives P and negatives N must be created with the number of scale l and the ratio of image pyramid R_m. The image pyramids of all training samples must be transformed to the same sizes $W_1, ..., W_l$ at each scale. Then, the classifiers are trained from the smaller W_1 to the larger W_l images. The classifier at each scale is trained to satisfy the minimum rate for acceptance detection, d. Rate d should be high enough (e.g., 0.99) because the entire detection rate of this model turns out to be around d^l.

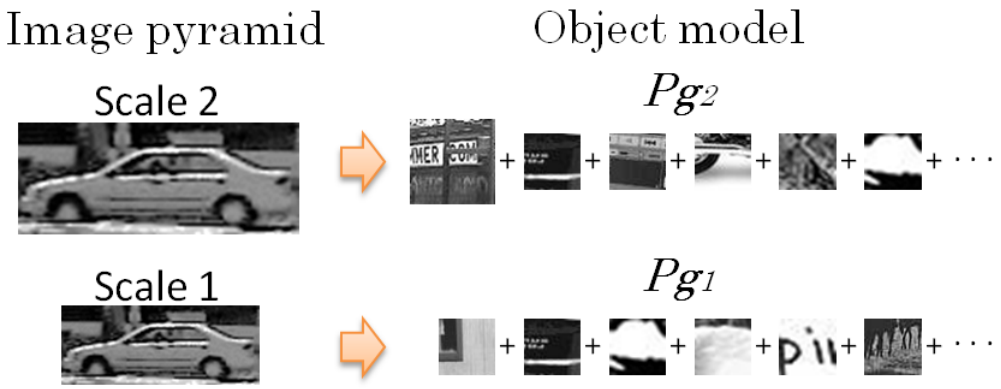


Figure 5.7: Example of trained object model.

HMAX features D_{C2} at scale i are computed from training samples of size W_i. Classifier C_i is trained from D_{C2} by real AdaBoost. The group of patch features $\boldsymbol{Pg}_i$ is simultaneously selected by boosting. The training images are evaluated with Eqs. (5.1) and (5.2) with this classifier C_i to generate the confidence value of each sample. Threshold T_i is determined to satisfy the target detection rate d using the confidence values. Exceptionally, threshold T_l of the last scale, l, is always 0.5. Then, only negative samples over threshold $T_i - \epsilon$ are used for training at the next stage, $i+1$. ϵ is a value to select negative samples near the boundary. If there are few false positive samples, then the number of negative samples over minimum negative sample ratio f are set to N.

Finally, the parameters $\boldsymbol{Pg}_i$, C_i, and T_i are set to the object model, Ω_i.

Figure 5.7 presents an example of an object model of car. The $\boldsymbol{Pg}_1$ and $\boldsymbol{Pg}_2$ are groups of important feature patches, which were selected to discriminate between positive and negative samples by real AdaBoost at each scale. Real AdaBoost combines these important patches with weak classifiers to estimate likelihood of a target existing.

5.4 Experiments

The main focus of the experiments was to clarify what effect our improvements had on the sliding window approach of HMAX. First, the performance

and processing time for these methods had to be measured with two datasets: 1) the University of Illinois-Urbana-Champaign (UIUC) car dataset[Agarwal et al., 2004] and 2) the face detection dataset benchmark (FDDB)[Jain and Learned-Miller, 2010]. Next, the impact of each improvements on our method was evaluated using the UIUC car dataset. Our approach has assumptions that aspect ratio of target object is fixed and that the target is single category. Both UIUC car and FDDB are datasets which can be used under these assumptions.

The parameters for HMAX in the experiments followed the empirical environment of Serre et al.[Serre et al., 2007], i.e., four orientations of the Gabor filter, two scale bands of the S1 layer integrated into one scale band of the C1 layer by max-pooling, and four sizes for the feature patches, $\boldsymbol{P}_n$. We created a dictionary that had 1000 feature patches for each size; therefore, there were a total of 4000 patches in the dictionary. These feature patches were randomly sampled from the Caltech-256 dataset[Griffin et al., 2007]. A total five feature dictionaries were generated in the same way in our experiments.

The criteria for correct detection were the same as the Pascal visual object classes (VOC)[Everingham et al., 2009]; the overlap ratio, a_o, between the predicted bounding box, B_p, and the ground truth bounding box, B_{gt}, had to be over 0.5 according to:

$$a_o = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{5.6}$$

These experiments were carried out in the following environment:

- Machine: Dell Dimension C521
- CPU: AMD Athlon 64 2.20GHz
- RAM: 960 MBytes
- Programming Language: C/C++ with OpenCV2.3.1[opencv.org, 2012]

5.4.1 Evaluation of performance and speed

Parameters

We measured the performance and processing time of our method and the sliding window approach of HMAX using the UIUC car dataset and FDDB. The parameters for our coarse-to-fine model were set for each dataset, as summarized in Table 5.1. The meanings of the nine parameters in the table are as follows:

- R_m: The scale ratio of the image pyramid of the object model
- l: The number of image pyramid
- d: The minimum acceptance detection ratio at each scale for training
- f: The minimum negative sample ratio at each scale for training
- N_t: The number of iterations to train AdaBoost at each scale band
- $W_l \times H_l$: The training sample size
- N_{s2}: The sizes of feature patches on the S2 layer
- $W_{c2} \times H_{c2}$: The number of divided area on the C2 layer
- R_d: The scale ratio of the image pyramid for detection

Sliding window approach

The sliding window approach sampled an image region every four pixel horizontally and vertically, and then the sampled image was resized to the same size as the training samples. This sampling interval was established by considering the C1 max-pooling size and its overlap ratio. Finally, an HMAX feature vector is extracted from the captured image and then scored by real AdaBoost and a logistic sigmoid function. Our implementation of the sliding window approach does also eliminated the computation time for HMAX by using only feature patches selected by AdaBoost, which is the same as that in our coarse-to-fine method.

The parameters for the sliding window approach were set to those listed in Table 5.1 as much as possible to enable comparison with our coarse-to-fine method.

Table 5.1: Object Model Parameters for Each Dataset

	UIUC Car	FDDB
R_m	$\sqrt{2}$	$\sqrt{2}$
l	2	3
d	0.999	0.995
f	0.3	0.45
N_t	100	100
$W_l \times H_l$	100×40	70×70
N_{s2}	4, 8	4, 8, 12, 16
$W_{c2} \times H_{c2}$	1×1	1×1
R_d	$\sqrt{R_m}$	R_m

For instance, training sample size $W_l \times H_l$, the number of C2 layers divided by $W_{c2} \times H_{c2}$, and the scale ratio of the image pyramid to detect R_d were the same for the coarse-to-fine and the sliding window approaches. The number of training iterations for the sliding window was $N_t \times l$, which was equal to the sum of the iterations over all scale bands of the object model.

UIUC car dataset

Our approach in this experiment was compared with the sliding window approach using the UIUC car dataset[Agarwal et al., 2004]. The UIUC car dataset consists of small (100x40) training images of cars and backgrounds, and larger test images in which there is at least one car to be found. There were 550 positive samples and 500 negative samples. There were two sets of test images: a single-scale set in which the cars to be detected were roughly the same size (100x40 pixels) as those in the training images, and a multi-scale set. Because of the size of the training images, only two sizes of feature patches $\{4 \times 4, 8 \times 8\}$ that had 1000 patches for each were used for this dataset.

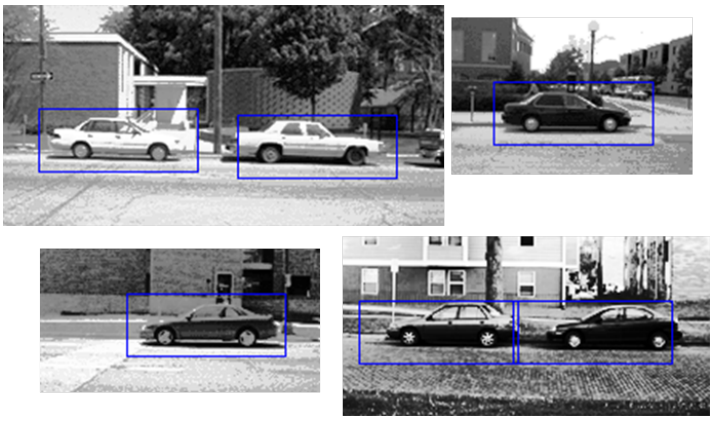
We examined the detection task three times with different feature dictionaries. There are examples of our results in Fig. 5.8. The results at the bottom right in Fig. 5.8(a) seem to have shifted slightly from the correct car position. This could happen because of max-pooling especially at the C2 layer. As we did not divide

the object region for C2 max-pooling (see Sec. 5.3.1) in this experiment, small shift of shape was ignored with dilate filter.

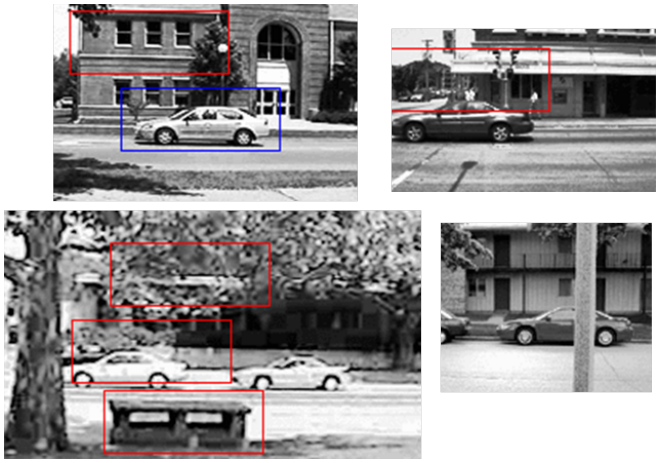
Figure 5.9 plots the recall-precision curves for both methods. These curves were drawn by changing the threshold of the output value represented in Eq. (5.2). Only the confidence map, $\boldsymbol{I}_{prob}$, from the last scale band in the coarse-to-fine approach was used to draw the curve by thresholding. We found that these two curves almost fit each other in each test set. In fact, the recalls at equal-error rates (recall = precision) corresponded to 0.775 and 0.79 for the single-scale data and 0.43 and 0.42 for the multi-scale data for the sliding window and coarse-to-fine approaches (See Table 5.2). Mutch and Lowe[Mutch and Lowe, 2008] in Table 5.2 achieved very high score with the improved version of HMAX with sliding windows. As previously mentioned, our coarse-to-fine approach can be expanded to these modified HMAX models.

Brosch and Neumann modified the Mutch and Lowe's HMAX and combined it with HOG features[Brosch and Neumann, 2012], where they employed the coarse-to-fine approach to also reduce the processing area on the fine scale. The results from their method that only used HMAX features have been listed in Table 5.2 for comparison. As their main interest was not to reduce computational cost, the actual processing time was not mentioned; however, they claimed they decreased the processing time to about 15% on the fine scale (not for the whole process). In contrast, our approach decreased the time for the whole detection process to about 0.33% (calculated from the "UIUC car (multi-scale)" in Table 5.3) because our approach not only eliminated the processing region on the fine scale but also redundancies over positions and scales. That is, our approach processed whole images from the S1 to the C2 layer to get rid of overlapping regions in sliding windows, as described in Sec. 5.3.1 . Furthermore, some processes on coarse and fine scales were shared in multi-scale localization tasks, as described in Sec. 5.3.1 .

Table 5.3 lists the average processing times for both data (single- and multi-scale). The numbers in parentheses "()" in Table 5.3 are the standard deviations. These results indicate that our method reduced the computational cost of sliding windows significantly without reducing accuracy. We achieved 1.5% better recall in 2.36×10^{-3} time in the single-scale test and 1.0% less recall in 1.31×10^{-5} time

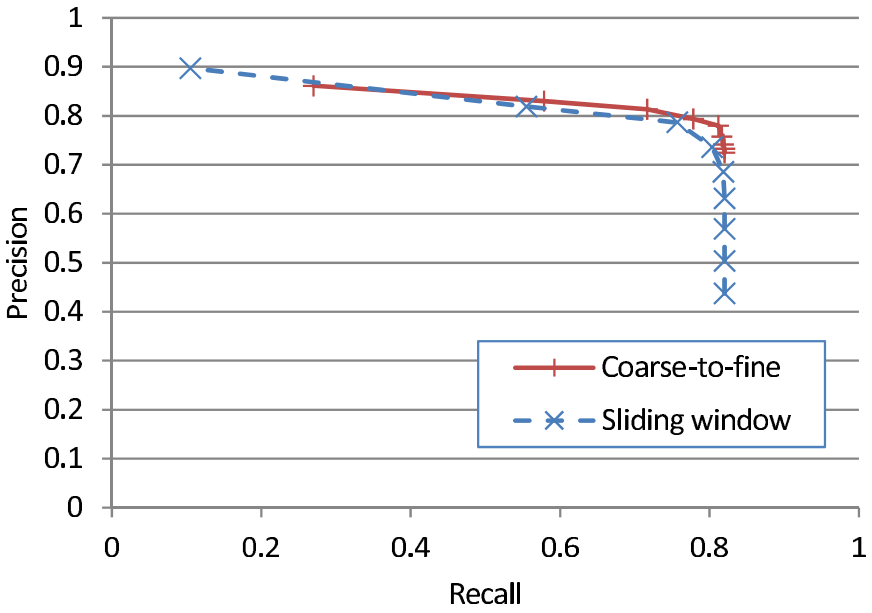


(a) Examples of correct detections from one run on the single scale UIUC car dataset

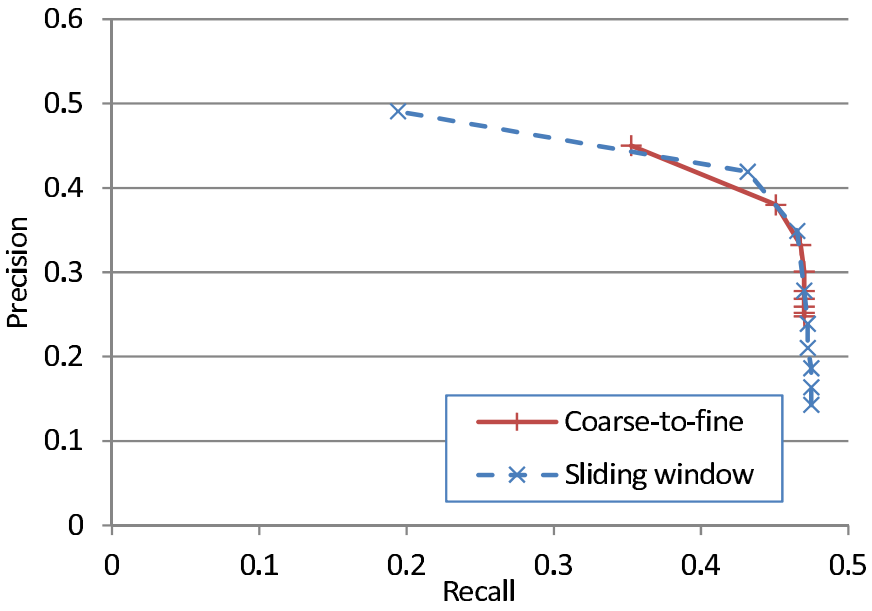


(b) Examples of false detections and miss-detections on the single scale UIUC car dataset

Figure 5.8: Example of results on UIUC car dataset.



(a) Single-scale test



(b) Multi-scale test

Figure 5.9: Recall-precision curves for coarse-to-fine and sliding window approaches.

Table 5.2: Equal-error rates for UIUC car dataset (%)

	Single Scale	Multi Scale
Sliding window	77.5	43.0
Coarse-to-fine(1×1)	79.0	42.0
Coarse-to-fine(2×1)	80.3	46.5
Mutch & Lowe [Mutch and Lowe, 2008]	99.94	90.6
Brosch & Neumann (only HMAX) [Brosch and Neumann, 2012]	–	84.08

Table 5.3: Average process time per image (sec)

	Sliding window	Coarse-to-fine
UIUC Car (Single Scale)	80.55 (130.67)	0.19 (0.14)
UIUC Car (Multi Scale)	254.32 (229.93)	0.85 (0.47)
FDDB	774.40 (185.34)	3.04 (0.49)

in the multi-scale test. Note that as feature vectors were only computed from the selected patches by AdaBoost, the normal cost of feature calculations increased. We ran the full computation of HMAX feature vector in the single scale tests and it took 167.38 sec per image. That means our approach reduced the processing time to about a total of less than 10^{-3} times.

FDDB dataset

Our coarse-to-fine and the sliding window methods were also tested with FDDB[Jain and Learned-Miller, 2010] to compare them with other categories. The FDDB has image sets for face detection tasks, which have 2845 images with a total of 5171 faces. There are some examples of the faces detected with our approach in Figure 5.10. The relative size of the detected window to the face in the top left image in Figure 5.10 seems larger than that in the other results. That is because the size ratio of the object to search was large, which was around 1.41; hence, the optimal size was omitted. Nevertheless, the max pooling operations in

the C1 and the C2 layer succeeded in covering the gaps.

We have followed the "EXP-1" experimental protocol in this experiment, which was a 10-fold cross-validation of these face images. The negative training samples were randomly created from the Caltech-256 dataset[Griffin et al., 2007]. The results obtained from localization are plotted as receiver operating curves (ROC) in Figure 5.11. The discrete score is the probability of a face existing where the overlap ratio, a_o, in Eq. (5.6) is over 0.5. The continuous score is just defined as a_o. The details of these evaluation protocols are given in Jain and Learned-Miller[Jain and Learned-Miller, 2010]. The results from Viola & Jones face detector[Viola and Jones, 2001] implemented in OpenCV[opencv.org, 2012] are also plotted in Figure 5.11 as a benchmark, even though the parameters, such as detection scale-factors, are not the same as those in our implementation.

The difference in the true-positive rate between the sliding window and coarse-to-fine approaches is only about 0.05 on the discrete score and about 0.03 on the continuous score; however, the processing time for our approach is 255 times faster than that for the sliding window approach as can be seen in Table 5.3.

These results indicate that our approach could reduce the processing speed of HMAX enormously with a small reduction in accuracy not only in car detection but also in face detection tasks.

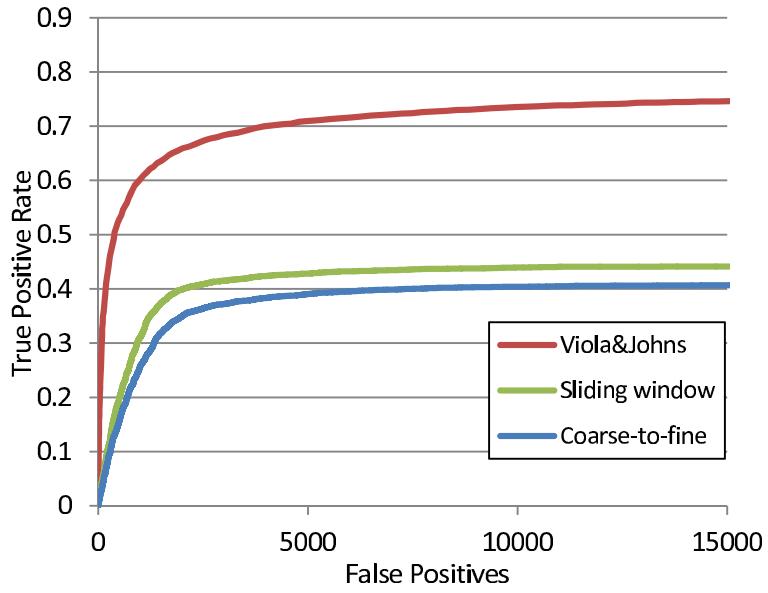
5.4.2 Evaluation of modifications

The main objective of this section is to explain our analysis of what impact our modifications had on performance in our approach. We investigated three points of view: the first was the number of iterations in training, the second was the "coarse-to-fine" approach, and the third was division of the C2 area, which is explained by Eq. (5.4) and Figure 5.5.

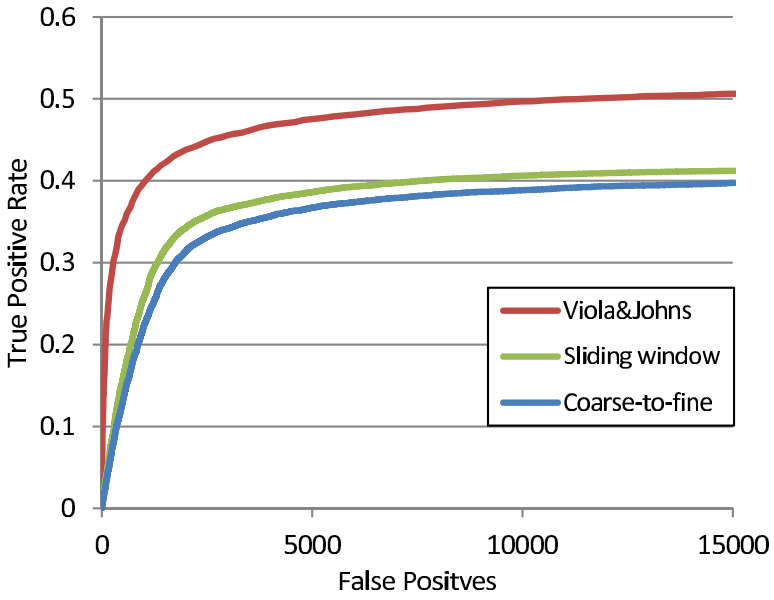
Iterations for training

The number of iterations of real AdaBoost affected the number of feature patches because each iteration selected one patch from N_p patches in the feature dictionary. The number of feature patches had an influence on accuracy and processing speed.

Figure 5.10: Examples of face detection.



(a) ROC curves based on discrete score



(b) ROC curves based on continuous score

Figure 5.11: ROC curves for FDDB.

We plotted the relation between the number of iterations, recall at equal-error rates, and process time in Figure 5.12. Each scale layer of the object model was trained by AdaBoost at 20, 50, 100, 200, and 500 iterations and its processing speed and recall were evaluated.

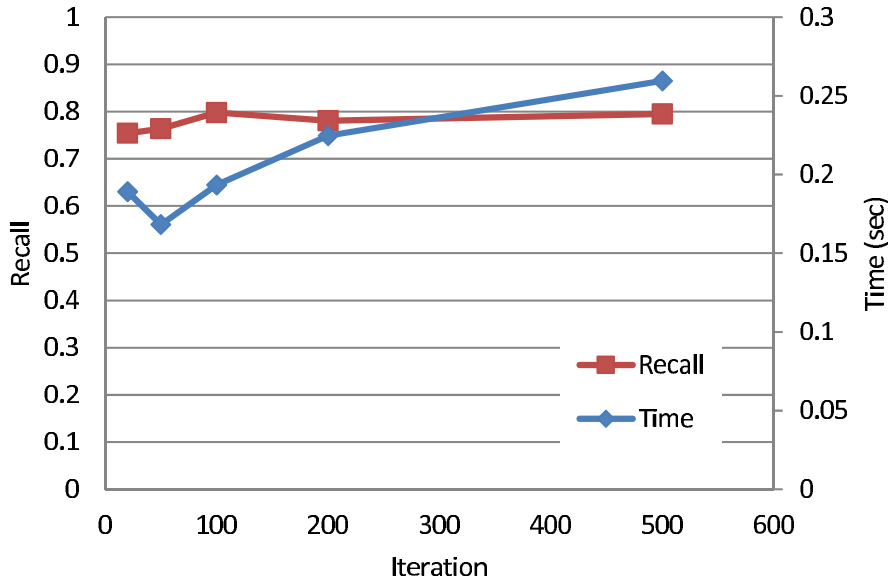
Figure 5.12 plots the process times and recall curves for single-scale and multi-scale tests that have similar shapes: almost all the process times increased following the number of iterations. The recalls were satiated at around 50 - 100 iterations. We selected "100" iterations by considering the trade-off between them.

The process time at 20 iterations is inexplicably higher than that at 50 iterations in the graph. The reason for this is that the first scale band of the object model trained for 20 iterations has less ability to discriminate; thereby, this model rejected a smaller area at an early stage and caused a slight increase in time.

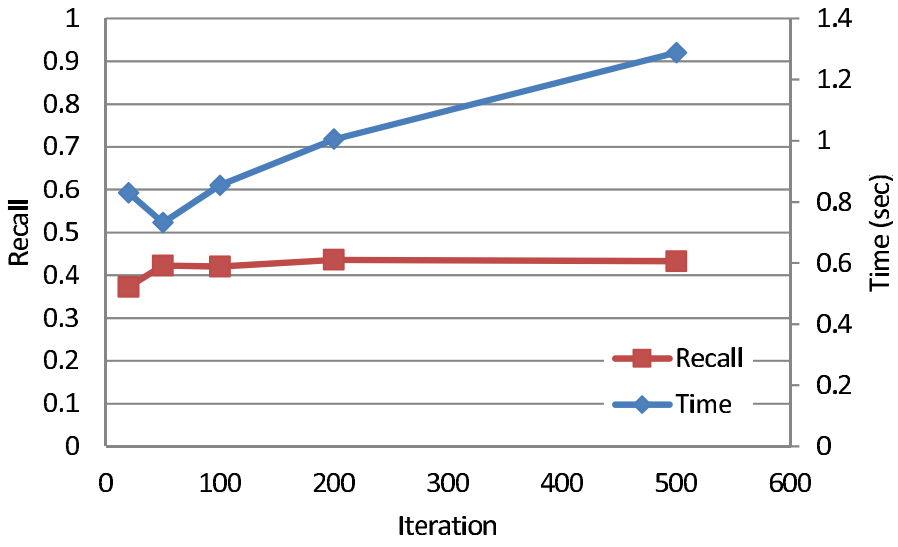
Fine vs coarse-to-fine

The *fine object model* that only had a single-scale band was trained to compare it with the *coarse-to-fine object model* which had two scale bands to study the effects of the coarse-to-fine approach. In other words, the fine model is the case of $l = 1$ in Algorithm 1, and the coarse-to-fine model is the case of $l = 2$ in this experiment. The fine model was trained with 200 iterations of AdaBoost to fit the total number of iterations to the coarse-to-fine model. The recall and process times are summarized in Tables 5.4 and 5.5 .

The coarse-to-fine model did not seem to improve its processing speed while its recall was inferior to that of the fine model in the single scale data test (UIUC car). There are two reasons for this behavior. First, half the number of features in the coarse-to-fine approach were trained at lower resolution because all the features of fine model were trained at higher resolution. In addition, the coarse-to-fine model eliminated the low confidence area at an earlier stage; therefore, its ability to discriminate was reduced. Second, the total number of features was not always equal to the number of iterations because real AdaBoost often chose duplicated features. The fine model in this case chose more duplicated features than the coarse-to-fine model did since it had to select all features on the same scale. As the coarse-to-fine model had more than the fine model for this reason,



(a) Single-scale test



(b) Multi-scale test

Figure 5.12: Processing time, recall, and the number of iterations for training.

Table 5.4: Average process times per image of fine and coarse-to-fine models (sec)

	Single-scale	Multi-scale	FDDB
Fine	0.19 (0.20)	0.95 (0.72)	5.13 (0.82)
Coarse-to-Fine	0.19 (0.18)	0.85 (0.61)	3.05 (0.50)

Table 5.5: Recalls for fine and coarse-to-fine models

	Single-scale	Multi-scale	FDDB
Fine	0.84	0.42	0.49
Coarse-to-Fine	0.80	0.42	0.47

the process time could not be shorter.

The coarse-to-fine model, on the other hand, succeeded in reducing the process time by about 10% while maintaining accuracy in the multi-scale test (UIUC car). The localization tasks of multi-scale dataset were better than those in the single-scale dataset because a small reduction in the process time would have accumulated. Similarly, the process time for the coarse-to-fine approach in the FDDB test that had multi-scale faces was 59% of that for the fine approach with only a 2% decline in recall.

Comparing the fine object model with the sliding window model (see Tables 5.3 and 5.4) reveals that removing overlapping regions between sliding windows had a great effect on computing cost.

Dividing max-pooling area in C2 layer

It has been reported that information on feature patch positions improved the accuracy of classification tasks, as stated in Section 5.3.1. The previous work achieved this simply by dividing the max-pooling area in the C2 layer. We introduced this "dividing" approach and implemented it with our method of localization by using a dilation filter (see Eq. (5.4)).

We wanted to confirm whether the dividing approach worked for localization tasks. The second scale band of the object model was horizontally divided into

two regions in this experiment. The overlap ratio, ΔN_{c2}, between two regions was set to zero. The divided model and the non-divided models were compared.

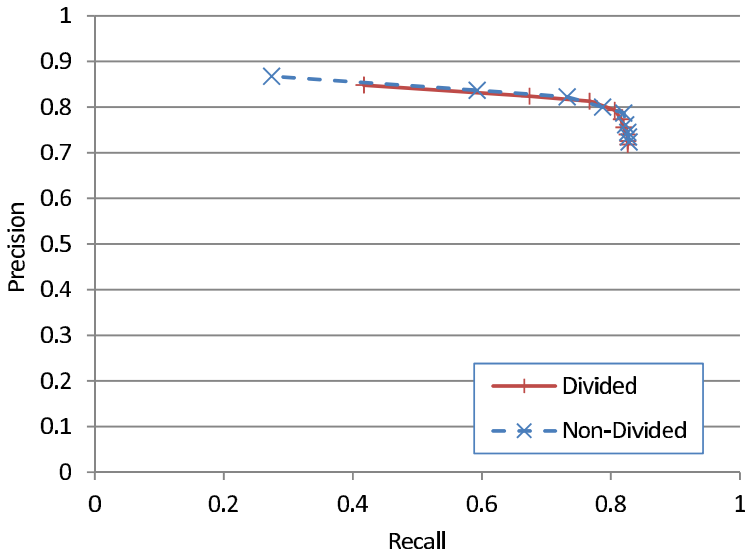
The recall-precision curves for single- and multi-scale tests are in Figure 5.13. Performance has obviously been improved in the multi-scale test but not in that in the single-scale test. The position information of features indicated that it improved performance in more difficult tasks.

5.4.3 Combining with our speed-up techniques of categorization

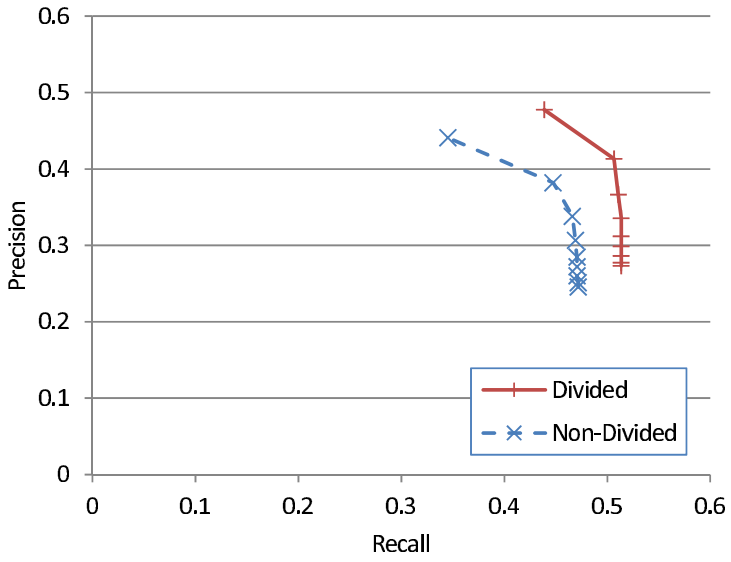
Our method focused on redundancies in sliding window approach of the "basic" HMAX model so that it could be implemented in other "improved" HMAX models[Chikkerur and Poggio, 2011; Huang et al., 2011; Minagawa and Saito, 2009a; Mutch and Lowe, 2008; Mutch et al., 2010] that have been reported to achieve better performance and faster speed than the basic-model. As these improved models mainly focused on categorization tasks, our work might not conflict with these methods.

We therefore tried to combine our localization method with our speed-up method of categorization that was explained in Chapter 4. We implemented two speed-up techniques: "restriction of S2 calculation area"(Sec. 4.2.2) and "simplification of S1 process"(Sec. 4.2.3). Other modifications, "selecting important features by AdaBoost"(Sec. 4.2.5) and "retention of feature position"(Sec. 4.2.4), have already been implemented in our localization method and tested in Sec. 5.4.2. "Reduction of feature patches"(Sec. 4.2.1) was not tested here because it have little effect on speed and performance when AdaBoost is used for the feature selection (see Sec. 4.3.3).

The two techniques were also evaluated with UIUC car dataset and FDDB. Either and both technique have been implemented into our coarse-to-fine localization method. Table 5.6 shows processing times of each technique, Table 5.7 shows equal-error rates with UIUC car dataset, and Figure 5.14 illustrates ROC curves of FDDB tests. The "with LM" means that computation of S2 (Eq. (2.3)) was restricted on local maxima of C1 outputs as represented in Eq. (4.1). The "with Haar" means that Gabor filters on the S1 layer (Eq. (2.1)) were approximated by



(a) Single-scale test



(b) Multi-scale test

Figure 5.13: Recall-precision curves for divided and non-divided C2 areas.

Table 5.6: Average process time per image (sec)

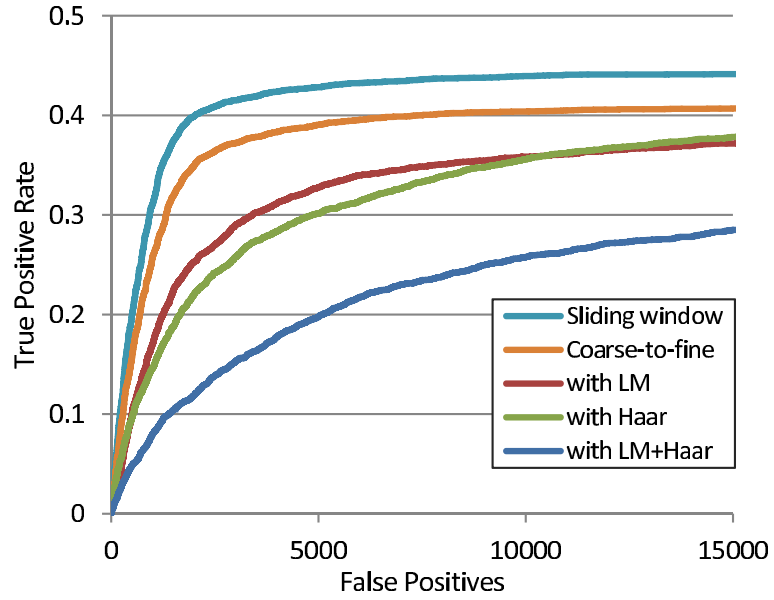
	UIUC Car (Single Scale)	UIUC Car (Multi Scale)	FDDB
Coarse-to-fine	0.19 (0.14)	0.85 (0.47)	3.04 (0.49)
with LM	0.19 (0.14)	0.92 (0.51)	3.07 (0.47)
with Haar	0.16 (0.11)	0.90 (0.49)	3.09 (0.57)
with LM+Haar	0.16 (0.11)	0.90 (0.49)	3.14 (0.55)

Table 5.7: Equal-error rates for UIUC car dataset with speed-up techniques of categorization (%)

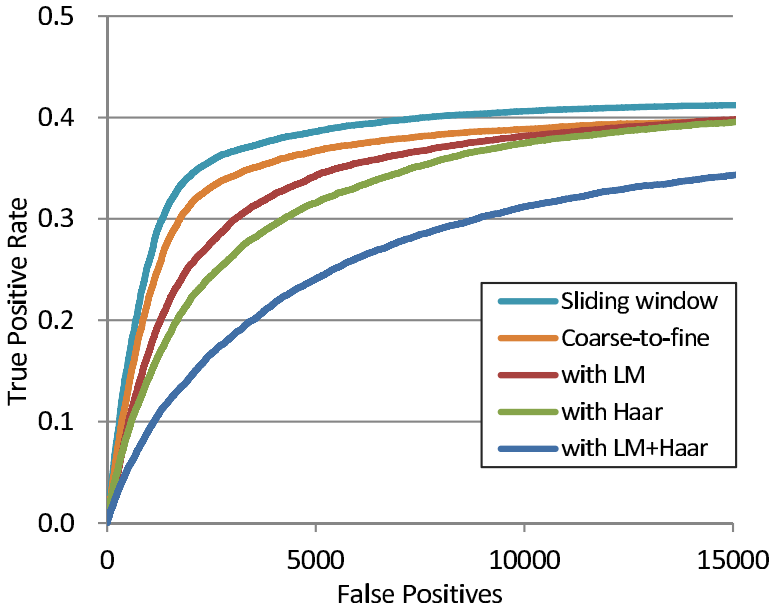
	Single Scale	Multi Scale
Sliding window	77.5	43.0
Coarse-to-fine	79.0	42.0
with LM	83.0	48.0
with Haar	37.2	18.5
with LM+Haar	48.0	25.9

Haar-like features. The "with LM+Haar" means that the both techniques were implemented into our localization method.

Table 5.6 represents that each speed-up technique for categorization tasks did not work in our localization method. The localization method have already restricted processing regions in coarse-to-fine manner and reduced the number of S2 patches by AdaBoost; therefore, C1 local maxima could not reduce procedures more than overhead of computation of the local maxima. Our localization method also sped up computation of multi-scale Gabor filters on the S1 layer by using image pyramids, which did better than the approximation by Haar-like features. The "LM" had slightly improved the performance of car detection as shown in Table 5.7. On the other hand, the "Haar" technique much reduced the performance of car detection. It is well known that performances of object detection are dependent on types of feature descriptors and object. Thus, Haar-like features might be good for face detection[Viola and Jones, 2001], but not for car detection. Figure 5.14 represents that both "LM" and "Haar" techniques decreased continuous



(a) ROC curves based on discrete score



(b) ROC curves based on continuous score

Figure 5.14: ROC curves for FDDB with speed-up techniques of categorization.

scores slightly but discrete scores much in FDDB, which means both techniques detected faces with small misalignment of positions or sizes. “LM” did not occur alignment issue in UIUC car but in FDDB. The reason might be that faces have less texture than cars, which generated less C1 local maxima around faces. It was reported that Gabor filters were robust for alignment[Wiskott et al., 1997]; therefore, approximation by Haar-like features might lose the robustness. As a result, the combination “LM+Haar” lost much scores.

For these results, speed-up techniques for categorization tasks are not always effective to our localization method.

5.5 Discussion

The current implementation of our method did not take into account state-of-the-art performance. However, we focused on redundancies in sliding window approach of the “basic” HMAX model so that it could be implemented in other “improved” HMAX models[Huang et al., 2011; Mutch and Lowe, 2008] that have been reported to achieve better performance than the basic-model. As these improved models mainly focused on categorization tasks, our work might not conflict with these methods. Faster modified HMAX models have also been proposed[Chikkerur and Poggio, 2011; Minagawa and Saito, 2009a]; however, these were not always effective to our localization method as shown in Section 5.4.3.

The current implementation of our method was restricted to the localization of single categories; however, it is not difficult to expand this model to multi-category localization tasks. Since objects can be explained as groups of feature patches in our model, common features in categories can be shared to be searched. Because of this, the process time to detect multi-classes is not expected to linearly increase. We also expect the method proposed by Dean et al. to be applied into the mutli-category localization of HMAX, which can select highly similar patches to image appearances rapidly with locally sensitive hashing[Dean et al., 2013]. In addition, the implementation of multi-category detection can also resolve other limitations. The aspect ratio of training samples must be the same in the current method because it determines the filter size of C2 dilation. The architecture for multi-class detection can treat the same objects with different aspect ratios as dif-

ferent objects. Of course, as the feature patches of these objects are the same, computational costs may increase slightly.

Another idea to attain improvements is to avoid image pyramids for faster processing. For example, Haar-like features are known to calculate multi-scale features without creating image pyramids. Dollar et al.[Dollár et al., 2010] and Benenson and Mathias[Benenson and Mathias, 2012] also reported that other scales of HOG features could also be approximated without image pyramids. It is worth considering whether the same kind of idea can be applied to the Gabor filter in the S1 layer or to patch similarities calculations in the S2 layer.

5.6 Conclusion

We proposed a method that improves the process time of HMAX features for object localization tasks. The previous localization approaches using HMAX simply made use of a sliding windows. We focused on improving three redundancies that were specific to the HMAX model in the sliding window approach; the first was overlapping regions in sliding windows, the second was duplicated filters in multi-scale localization tasks, and the third was omission of an object size by max-pooling in the C2 layer. The main idea in our approach to eliminate the redundancies was that the image region of a target object should have high degrees of similarity to the shapes that the object has. This idea was implemented by three main changes from sliding window approach. First, the whole image region is processed at one time to eliminate overlapping areas in sliding windows. Second, processing of small shapes in larger objects and large shapes in smaller objects are shared to eliminate overlapping scales. Finally, an image region that has high degree of similarity to parts of the target object is searched from coarse-to-fine area.

The experiments proved that these ideas reduced the processing time enormously with negligible reduction in precision. In addition, we evaluated the effect of various parameters and functions such as the number of training iterations, coarse-to-fine approach, and division of the C2 layer, and hence confirmed their optimal settings or benefits.

Chapter 6

Conclusions

6.1 Summary and Contributions

This thesis focused on speeding up HMAX model that had been inspired from biological evidences for two applications: image based information retrieval and object localization.

Chapter 1 explained purpose of this thesis. Object recognition tasks have been active research area in computer vision for more than a decade; however, most of major approaches have been studied just to achieve better performance, but not based on biological models. The HMAX is one of the successful computational models which are biologically plausible. It emulates "rapid categorization" tasks of human which activate a feedforward path of ventral stream in brain; however, its processing costs on machines are very expensive. Therefore, we aimed to speed up the HMAX model for broader usage of applications.

Chapter 2 explained the HMAX model in detail and its related work. Some studies tried to build applications with the HMAX; however, most of them did not mention their processing speeds. There are also some studies which improved HMAX model itself to achieve better performance, which did not mainly focus on speed-up.

Then, we reviewed other image feature descriptors and speed-up techniques for them in Chapter 3. Existing approaches of speed-up can be categorized into four groups: faster feature descriptors, reductions of feature dimensions, restric-

tions of processing areas, and parallel processing. We tried to reduce processing cost of the HMAX with these approaches except for parallel processing.

The HMAX was originally invented for object categorization tasks, thus we first tried to reduce processing costs for the tasks in Chapter 4. It was modified by simplifying Gabor filters to Haar features, inhibiting the area to be proceeded at S2 layer, reducing the number of feature patches, dividing C2 areas, and selecting important features with AdaBoost. The modified model was evaluated with Caltech-101 and scene image datasets and confirmed that the processing time was reduced to 1/30 with little reduction of accuracy. Then the model was implemented into an information retrieval system to show an example of realistic application.

In Chapter 5, more difficult application that is object localization was addressed. This task is usually more time consuming because object categorization must be executed on every sliding window. Because the HMAX features are extracted by computing similarities between input images and patches, our approach searches the similar appearances to shapes of parts which target object has from coarse to fine resolution. This approach reduced redundancies of sliding window with the HMAX enormously. The experiments with UIUC car and FDDB datasets revealed that our localization is 250 time faster or more than sliding window approach with little reduction of accuracy.

Through these studies of speed-up, we have contributed to applying biological models to practical usages.

6.2 Future Perspective

We plan to expand our methods to multi-categories as we discussed in Sec. 5.5. To avoid linear increase of processing time in multi-category recognition tasks, the S2 patches should be shared among different categories. This multi-category recognition can also be applied to detecting objects of different aspect ratios and view angles in localization tasks.

As it is known that biological models achieve better performance with deeper hierarchy, another challenge is to speed up HMAX of deeper hierarchy, i.e., adding S3 and C3 layers or more. Our methods reduced the number of S2 patches with AdaBoost. If we introduce S3 and C3 layers, it is not easy to apply this ap-

proach straightforwardly. Thus, clustering techniques or sparse coding[Mairal et al., 2009] could be used to reduce the number of S2 feature patches as we tried in Sec. 4.2.1. Another way to reduce the number of patches is to use AdaBoost in both S2 and S3 layers to find discriminative combinations of patches on each layer. This hierarchical usage of AdaBoost has been proposed as Shapelet[Sabzmeydani and Mori, 2007], Joint Haar-like features[Mita et al., 2008], Joint HOG features[Mitsui and Fujiyoshi, 2009], and Co-occurrence Probability Feature (CFP)[Yamauchi et al., 2010].

The faster HMAX is expected to be used in many applications: the image based information retrieval system that we built in Chapter 4, for instance, can be applied to photo tagging, image search, and photo related advertisements in social network or photo sharing services. If it became faster enough to run on embedded devices, mobile cameras could configure its settings automatically for recognized scenes. Object localization can be used in broader applications: e.g., surveillance cameras, factory automation, and robots. Face detection had already been implemented in most digital cameras; however, detecting other targets such as cars, bikes, and animals, may help users to shoot objects of their interests.

In addition, these speed-up studies might help researchers who simulate biological models with the HMAX as seen in Sec. 2.3.1. The other applications described in Sec. 2.3.2 might also get benefits from our studies.

Unfortunately, our methods did not achieve state-of-the-art performance because we focused on speed-up of the "original" HMAX model [Serre et al., 2005], after which a lot of image category recognition algorithms have achieved better performance. HMAX models have also been evolved and achieved better performance as we explained in Sec. 2.3.3. Thus, some of our speed-up techniques can contribute to these models. Especially our localization method does not conflict with the extended HMAX models [Huang et al., 2011; Mutch and Lowe, 2006] because it reduces just redundancies of HMAX in sliding window approach, as we discussed in Sec. 5.5. For this reason, the performance of the speed-up HMAX would be improved better by using these extended ones.

This thesis focused on building applications with the biologically inspired model, but not developing a biological model itself. Nevertheless, we can see some similarities between our speed-up techniques and biological evidences. We

clustered S2 feature patches and selected representative one of each cluster to reduce its number (see Sec. 4.2.1). This is similar to the fact that selectivity of simple cells are coded to be sparse for input signals[Olshausen and Field, 1996]. In Sec. 4.2.2, only local peaks of C1 responses were used for computation to reduce processing regions. The similar phenomenon is known in neurobiology as *lateral inhibition*: the strongest signal inhibits neighbor ones. In Chapter 4 and 5, AdaBoost is used to select important feature patches to recognize target objects. Hence, only selected patches were used to find objects in recognition phases. It is known that characteristics of target object is emphasized in brain when a person pays attention to find the object. This phenomenon is called *selective attention*[Fukushima, 1986; Koch, 2004], in which feedback path of ventral stream is activated. The coarse-to-fine approach in Chapter 5 also have similar activities in brains: there are evidences that spatial signals of low frequencies are processed faster than ones of high frequencies[Schyns and Oliva, 1994; Sugase et al., 1999].

These similarities occurred because our methods were also efforts to find "representation and algorithm" which solves object recognition tasks (see Chapter 1). Marr indicated this approach is important to clarify the mechanism of visual perception[Marr, 1982]. In consequence, some algorithms resembled to biological models incidentally even though most of their efforts were just for better performance. Deformable parts model [Felzenszwalb et al., 2009], for instance, is one of the "high-level" features described in Chapter 3. The model consists of several histogram of oriented gradients (HOG)[Dalal and Triggs, 2005], and HOG consists of several orientations of gradients. Moreover, the gradients are also computed from several pixels. You can see this hierarchical architectures are similar to the visual cortex and HMAX (see Chapter 2). Bags-of-features[Csurka et al., 2004], joint Haar-like[Mita et al., 2008], joint HOG[Mitsui and Fujiyoshi, 2009], and other feature descriptors have also hierarchical architectures like visual cortex and HMAX have. Most of these "high-level" image feature descriptors have been developed without biological evidences; however, they have evolved to have hierarchical, from simple to complex, architectures like visual cortex. It implicates that hierarchies of features are important architectures for visual cortex to achieve high performance. For this reason, our approaches might help to understand the process of visual information in brains.

Bibliography

A9.com Inc. Visual search. `http://www.a9.com/whatwedo/visual-search/`, 2012. [Online; accessed 06-Feb-2012]. 3, 52

Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004. 64, 73, 75

Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face description with local binary patterns: application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–41, December 2006. 21, 27

Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517, June 2012. 27

Mitsuru Ambai and Yuichi Yoshida. Card: Compact and real-time descriptors. *International Conference on Computer Vision*, pages 97–104, November 2011. 19, 27, 30

Henri Astre. Gpusurf. `http://www.visual-experiments.com/demos/gpusurf/`, 2012. [Online; accessed 06-Feb-2012]. 30

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, 2006. 19, 26, 30

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008. 26

Rodrigo Benenson and Markus Mathias. Pedestrian detection at 100 frames per second. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2012. 27, 31, 91

Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2009. 3, 24

Jinbo Bi, K Bennett, and M Embrechts. Dimensionality reduction via sparse support vector machines. *The Journal of Machine Learning Research*, 3:1229–1243, 2003. 66

Stanley Bileschi and Lior Wolf. A Unified System For Object Detection, Texture Recognition, and Context Analysis Based on the Standard Model Feature Set. In *Procedings of the British Machine Vision Conference 2005*, pages 83.1–83.10. British Machine Vision Association, Sep 2005. x, 13, 14, 59

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738. 37, 40

Lubomir Bourdev and Jonathan Brandt. Robust Object Detection via Soft Cascade. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 236–243. IEEE, 2005. 29

Tobias Brosch and Heiko Neumann. The Combination of HMAX and HOGs in an Attention Guided Framework for Object Localization. In *International Conference on Pattern Recognition Applications and Methods*, pages 281–288, February 2012. 33, 61, 76, 79

Charles Cadieu, Minjoon Kouh, Anitha Pasupathy, Charles E Connor, Maximilian Riesenhuber, and Tomaso Poggio. A Model of V4 Shape Selectivity and Invariance. *Journal of Neurophysiology*, pages 1733–1750, 2007. 12

Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. In *IEEE Internatinal Conference on Computer Vision*, 2010. 27

Canon U.S.A. Inc. Face detection technology. http://www.usa.canon.com/cusa/consumer/standard_display/PS_Advantage_Ease#d, 2012. [Online; accessed 06-Feb-2012]. 3

Jun Cheng, Dacheng Tao, Jiang Liu, Damon Wing Kee Wong, Ngan-meng Tan, Tien Yin Wong, and Seang Mei Saw. Peripapillary Atrophy Detection by Sparse Biologically Inspired Feature Manifold. *IEEE Transactions on Medical Imaging*, 31(12):2355–2365, 2012. 15

Sharat Chikkerur and Tomaso Poggio. Approximations in the HMAX Model. Technical Report 2011-04-14, MIT Computer Science and Artificial Intelligence Laboratory, 2011. 32, 34, 35, 61, 86, 90

Sharat Chikkerur, Thomas Serre, Cheston Tan, and Tomaso Poggio. What and where: a Bayesian inference theory of attention. *Vision research*, 50(22):2233–47, October 2010. 12, 61

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 15, 32, 40

Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, page 22, 2004. 19, 95

Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:886–893, 2005. 21, 61, 95

John G. Daugman. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision research*, 20(10):847–856, 1980. 8

John G. Daugman. Uncertainly relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of the Optical Society of America*, 2(7):1160–1169, 1985. 8

Thomas Dean, Mark A. Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, Accurate Detection of 100,000 Object

Classes on a Single Machine. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. 90

Jia Deng, Alex Berg, , Sanjeev Satheesh, Hao Su, Aditya Khosla, and Fei-Fei Li. Imagenet large scale visual recognition challenge 2012 (ilsvrc2012). `http://www.image-net.org/challenges/LSVRC/2012/`, 2012. [Online; accessed 20-Oct-2012]. 24

Piotr Dollár, Z Tu, P Perona, and S Belongie. Integral channel features. In *British Machine Vision Conference*, pages 1–11, 2009. 26

Piotr Dollár, S Belongie, and Pietro Perona. The fastest pedestrian detector in the west. In *British Machine Vision Conference*, 2010. 27, 91

Mohamed Y. El Dib and Motaz El-Saban. Human age estimation using enhanced bio-inspired features (EBIF). *2010 IEEE International Conference on Image Processing*, pages 1589–1592, September 2010. 14

Mark Everingham, Luc Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, September 2009. 73

Evernote. Evernote. `http://evernote.com/`, 2008. [Online; accessed 29-Sep-2008]. 52

Eyetap Personal Imaging Lab at University of Toronto. Openvidia: Parallel gpu computer vision. `http://openvidia.sourceforge.net/index.php/OpenVIDIA`, 2012. [Online; accessed 06-Feb-2012]. 31

Facebook.com. Making photo tagging easier. `http://blog.facebook.com/blog.php?post=467145887130`, 2012. [Online; accessed 06-Feb-2012]. 3, 52

FaceLock.mobi. Facelock. `http://www.facelock.mobi/`, 2012. [Online; accessed 06-Feb-2012]. 3

Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *Proc. of Conference on Computer Vision and Pattern Recognition*, volume 2, pages 524–531. IEEE, Jun 2005. 50

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: an incremental baysian approach tested on 101 object categories. In *Workshop on Generative-Model Based Vision, Computer Vision and Pattern Recognition*. IEEE, Jun 2004. 40, 43

Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, September 2009. xi, 22, 24, 28, 31, 61, 95

Pedro F. Felzenszwalb, Ross B. Girshick, and David. McAllester. Cascade object detection with deformable part models. In *2010 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2010. 29, 31

Rob Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271. IEEE, Jun 2003. 40, 41

foo.log Inc. Foodlog app. `http://app.foodlog.jp`, 2013. [Online; accessed 07-Oct-2013]. 53

Yoav Freund and Robert E Schapire. A decision-theoretic generalization and an application to boosting. *Journal of Computer and System Science*, 55:119–139, 1997. 28, 39

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. 3, 8

Kunihiko Fukushima. A neural network model for selective attention in visual pattern recognition. *Biological Cybernetics*, 15:5–15, 1986. 95

Google Inc. Google goggles. `http://www.google.com/mobile/goggles/`, 2012a. [Online; accessed 06-Feb-2012]. 3, 52

Google Inc. Picasa. http://picasa.google.com/, 2012b. [Online; accessed 06-Feb-2012]. 3, 52, 53

Google Inc. Google plus. http://plus.google.com, 2013. [Online; accessed 07-Oct-2013]. 53

Greg Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007. URL http://authors.library.caltech.edu/7694. 73, 80

Guodong Guo, Guowang Mu, Yun Fu, and Thomas S Huang. Human age estimation using bio-inspired features. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 112–119. IEEE, June 2009. 14

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, pages 147–152, 1988. 30

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *Signal Processing Magazine*, pages 1–27, 2012. 24

Chang Huang, Haizhou Ai, Yuan Li, and Shihong Lao. High-Performance Rotation Invariant Multiview Face Detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(4):671–686, 2007. 21

Yongzhen Huang, Kaiqi Huang, Dacheng Tao, Tieniu Tan, and Xuelong Li. Enhanced Biologically Inspired Model for Object Recognition. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 41(6):1668–1680, July 2011. x, 13, 15, 16, 32, 34, 35, 59, 61, 65, 86, 90, 94

David H. Hubel and Torsten N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160:106–154, 1962. 7, 8

David H. Hubel and Torsten N. Wiesel. Receptive Fields and Functional Architecture in Two Nonstriate Visual Areas (18 and 19) of the Cat. *Journal of neurophysiology*, 28:229–89, March 1965. 7, 8

Institute for Computer Graphics and Vision at Graz University of Technology. Gpu4vision. `http://gpu4vision.icg.tugraz.at/`, 2012. [Online; accessed 06-Feb-2012]. 31

Leyla Isik, Joel Z. Leibo, and Tomaso Poggio. Learning and disrupting invariance in visual recognition with a temporal association rule. *Frontiers in computational neuroscience*, 6(June):37, January 2012. 13

Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010. 73, 79, 80

Herve Jegou, Matthijs Douze, Cordelia Schmid, and Patrick Perez. Aggregating local descriptors into a compact image representation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304–3311. IEEE, June 2010. 19

Hueihan Jhuang, Thoms Serre, Lior Wolf, and Tomaso. Poggio. A biologically inspired system for action recognition. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. 12

Kaggle Inc. Job salary prediction. `http://www.kaggle.com/c/job-salary-prediction`, 2012a. [Online; accessed 15-May-2013]. 24

Kaggle Inc. Merck molecular activity challenge. `https://www.kaggle.com/c/MerckActivity`, 2012b. [Online; accessed 15-May-2013]. 24

Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *IEEE Internatinal Conference on Computer Vision and Pattern Recognition*, 2004. 27

Gunhee Kim, Christos Faloutsos, and Martial Hebert. Unsupervised modeling of object categories using link analysis techniques. In *IEEE International Conference on Computer Vision and Pattern Recognition*. IEEE, Jun 2008. 37

Christof Koch. *The Quest for Consciousness: A Neurobiological Approach*. Roberts & Company Publishers, 2004. ISBN 0974707708. 95

kooba. Image recognition and visual search. `http://www.kooaba.com/en/how_it_works`, 2012. [Online; accessed 06-Feb-2012]. 3, 52

Alex Krizhevsky. cuda-convnet. `http://code.google.com/p/cuda-convnet/`, 2012. [Online; accessed 06-Feb-2012]. 25, 31

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1106–1114, 2012. 24, 31

Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *IEEE Conference on Computer Vision and Pattern Recognition*, Alaska, USA, Jun 2008. IEEE. 29

Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Efficient subwindow search: a branch and bound framework for object localization. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2129–42, December 2009. 23, 29, 61

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, Jun 2006. 23, 50

Quoc V. Le, Marc' Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. *CoRR*, abs/1112.6, 2011. URL `http://arxiv.org/abs/1112.6209`. 25

Y LeCun, B Boser, and JS Denker. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541—-551, 1989. 4

Joel Z. Leibo, Jim Mutch, and Tomaso Poggio. Why The Brain Separates Face Recognition From Object Recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9, 2011. 12

Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006. 27

Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. *International Conference on Computer Vision*, pages 2548–2555, November 2011. 19, 27, 30

Kobi Levi and Yair Weiss. Learning object detection from a small number of examples: the importance of good features. In *IEEE Internatinal Conference on Computer Vision and Pattern Recognition*, 2004. 21

Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Proceeding of International Conference on Image Processing*, volume 1, pages 900–903, Sep 2002. 19, 25, 28, 38

Guo Lihua. Smile Expression Classification Using the Improved BIF Feature. In *2011 Sixth International Conference on Image and Graphics*, pages 783–788. IEEE, August 2011. 13

Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Trans.Commun*, COM-28:84–95, 1980. 37

Nikos K. Logothetis, Jon Pauls, and Tomaso Poggio. Shape representation in the inferior temporal cortex of monkeys. *Current Biology*, 5(5):552–63, 1995. 8

Noel Lopes, Ricardo Quintas, and Joao Goncalves. Gpumlib. `http://gpumlib.sourceforge.net/`, 2012. [Online; accessed 06-Feb-2012]. 31

David G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, pages 1150–1157, Sep 1999. 3, 37

David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. 19, 30

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *International Conference on Machine Learning(ICML)*, 2009. 13, 94

S Mardelja. Mathmatical description of the responses of simple cortical cells. *Journal of the Optical Society of America*, 70(11):1297–1300, 1980. 8

David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co. Inc, New York, NY, USA, 1982. ISBN 0716715678. 1, 95

J Matas, O Chum, M Urban, and T Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, September 2004. 30

Ethan Meyers and Lior Wolf. Using Biologically Inspired Features for Face Processing. *International Journal of Computer Vision*, 76(1):93–104, July 2007. 13

Krystian Mikolajczyk and Cordelia Schmid. Performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27 (10):1615–30, October 2005. 19

Takuya Minagawa and Hideo Saito. Image based search system using hierarchical object category recognition technique. In *Proc. of IAPR Conference on Machine Vision Applications*, pages 219–222, May 2009a. 31, 33, 34, 35, 52, 61, 86, 90

Takuya Minagawa and Hideo Saito. Hierarchical object category recognition technique for image based search system(in japanese). *IEEJ transactions on elecronics*, 129(5):947–955, May 2009b. 31, 33, 52

Takuya Minagawa and Hideo Saito. Speed up in computation of hmax features for object localization. *ITE Transactions on Media Technology and Applications*, 2(14), January 2014. 33

Takeshi Mita, Toshimitsu Kaneko, Bjorn Stenger, and Osamu Hori. Discriminative feature co-occurrence selection for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 30(7):1257–69, July 2008. 21, 28, 94, 95

Tomokazu Mitsui and Hironobu Fujiyoshi. Object detection by joint features based on two-stage boosting. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1169–1176. IEEE, September 2009. 22, 28, 94, 95

Yang Mu and Dacheng Tao. Biologically inspired feature manifold for gait recognition. *Neurocomputing*, 73(4-6):895–902, January 2010. 14

Yang Mu, Dacheng Tao, Xuelong Li, and Fionn Murtagh. Biologically Inspired Tensor Features. *Cognitive Computation*, 1(4):327–341, November 2009. 14

Yang Mu, Wei Ding, Dacheng Tao, and T.F Stepinski. Biologically inspired model for crater detection. In *International Joint Conference on Neural Networks(IJCNN)*, pages 2487–2494, 2011. 14

Multimedia Group of Information Technologies Institute (CERTH-ITI). Gpu-accelerated libsvm. `http://mklab.iti.gr/project/GPU-LIBSVM`, 2012. [Online; accessed 06-Feb-2012]. 31

Jim Mutch and David G Lowe. Multiclass object recognition with sparse, localized features. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 11–18. IEEE, Jun 2006. x, 13, 15, 16, 32, 34, 35, 39, 94

Jim Mutch and David G. Lowe. Object class recognition and localization using sparse features with limited receptive fields. *International Journal of Computer Vision*, 80(1):45–57, 2008. 15, 32, 59, 61, 67, 76, 79, 86, 90

Jim Mutch, Ulf Knoblich, and Tomaso Poggio. CNS: a GPU-based framework for simulating cortically-organized networks. Technical Report 2010-02-26, MIT Computer Science and Artificial Intelligence Laboratory Technical Report, 2010. 33, 35, 86

Eric Nowak, Jurie Frederick, and Bill Triggs. Sampling Strategies for Bag-of-Features. In *IEEE Europian Conference on Computer Vision*, pages 490–503, 2006. 30

Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, May 2001. 50, 52, 54

Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996. 95

opencv.org. Open source computer vision library (opencv). `http://opencv.org/`, 2012. [Online; accessed 06-Feb-2012]. 31, 35, 40, 73, 80

Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):448–61, March 2010. 27

Marco Pedersoli, Jordi Gonzàlez, Andrew D. Bagdanov, and Juan J. Villanueva. Recursive coarse-to-fine localization for fast object detection. In *IEEE Europian Conference on Computer Vision*, pages 280–293, 2010. 29

Marco Pedersoli, A. Vedaldi, and J. Gonzalez. A Coarse-to-fine approach for fast deformable object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011. 29, 61

Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 19

Florent Perronnin, Christopher Dance, Gabriela Csurka, and Marco Bressan. Adapted vocabularies for generic visual categorization. In *9th Europian Conference on Computer Vision*, pages 464–475, 2006. 19

Fatih Porikli. Integral histogram: a fast way to extract histograms in cartesian spaces. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 829–836 vol. 1, 2005. 26

Victor Prisacariu and Ian Reid. fasthog - a real-time gpu implementation of hog. Technical Report 2310/09, Department of Engineering Science, Oxford University, 2009. 30

Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–25, November 1999. 8

Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, pages 1508–1515. IEEE, 2005. 27, 30

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, pages 2564–2571, November 2011. 27

Payam Sabzmeydani and Greg Mori. Detecting Pedestrians by Learning Shapelet Features. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2007. 21, 28, 94

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999. 16, 21, 65

Andre Schulz, Florian Jung, Sebastian Hartte, Daniel Trick, Christian Wojek, Konrad Schindler, Jens Ackermann, and Michael Goesele. Cuda surf - a real-time implementation for surf. `http://www.d2.mpi-inf.mpg.de/surf`, 2012. [Online; accessed 06-Feb-2012]. 30

Philippe G Schyns and Aude Oliva. FROM BLOBS TO BOUNDARY EDGES: Evidence for Time- and Spatial-Scale-Dependent Scene Recognition. *Psychological Science*, 5(4):195–200, July 1994. ISSN 0956-7976. 95

Thomas Serre. *Learning a Dictionary of Shape-Components in Visual Cortex: Comparison with Neurons, Humans, and Machines*. PhD thesis, Massachusetts Institute of Technology, Cambridge, April 2006. 4, 8, 34

Thomas Serre, Lior Wolf, and Tomaso Poggio. Object recognition with features inspired by visual cortex. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 994–1000. IEEE, Jun 2005. 4, 5, 8, 15, 25, 35, 39, 40, 41, 43, 94

Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(3):411–426, Mar 2007. 4, 5, 8, 9, 35, 40, 41, 45, 73

Josef Sivic and Andrew Zisserman. Video Google: a text retrieval approach to object matching in videos. In *IEEE Internatinal Conference on Computer Vision*, number 2, pages 1470–147. IEEE, 2003. 19

Sephen M. Smith and J. Michael Brady. SUSAN—A new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997. 30

Dongjin Song and Dacheng Tao. Biologically inspired feature manifold for scene classification. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 19(1):174–84, January 2010. 15

Hyun Oh Song, Stefan Zickler, Tim Althoff, Ross Girshick, Mario Fritz, Chirstopher Geyer, Pedro F. Felzenszwalb, and Trevor Darrel. Sparselet Models for Efficient Multiclass Object Detection. In *IEEE Europian Conference on Computer Vision*, 2012. 27, 31

Sony Corporation. Face recognition technology. http://www.sony.net/SonyInfo/technology/technology/theme/sface_01.html, 2012. [Online; accessed 06-Feb-2012]. 3

Balaji Vasan Srinivasan, Qi Hu, and Ramani Duraiswami. GPUML: Graphical processors for speeding up kernel machines. In *Workshop on High Performance Analytics - Algorithms, Implementations, and Applications, Siam Conference on Data Mining*, April 2010. 31

Y. Sugase, S. Yamane, S. Ueno, and K. Kawano. Global and fine information coded by single neurons in the temporal visual cortex. *Nature*, 400(6747):869–73, August 1999. ISSN 0028-0836. 95

Michael J Tarr. News On Views : Pandemonium Revisited. *Nature Neuroscience*, 2(11):932–935, 1999. 4

Simon J. Thorpe and Michele Fabre-Thorpe. Seeking categories in the brain. *Science*, 291(5502):260–263, 2001. x, 7

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proc of Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Kauai, Hawaii USA, Jun 2001. IEEE. 19, 25, 38, 39, 40, 80, 88

Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2002. 19, 25, 28, 69

Tomoki Watanabe, Satoshi Ito, and Kentaro Yokoi. Co-occurrence Histograms of Oriented Gradients for Human Detection. *IPSJ Transactions on Computer Vision and Applications*, 2:39–47, 2010. 22

Wikipedia. Wright brothers. http://en.wikipedia.org/wiki/Wright_brothers, 2013. [Online; accessed 12-Nov-2013]. 3

Wikipedia. Wright brothers. http://en.wikipedia.org/wiki/Moore%27s_law, 2014. [Online; accessed 9-Jan-2014]. 4

Laurenz Wiskott, Jean-Marc Fellous, Norbert Kruger, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997. 3, 90

Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, January 2007. 21, 28

Bo Wu, Haizhou Ai, Chang Huang, and Shihong Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 79–84. IEEE, May 2004. 28

Changchang Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007. [Online; accessed 06-Feb-2012]. 30

Yuji Yamauchi, Masanari Takaki, Takayoshi Yamashita, and Hironobu Fujiyoshi. Feature co-occurrence representation based on boosting for object detection. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 31–38, June 2010. 22, 28, 94

Wei Zheng and Luhong Liang. Fast car detection using image strip features. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 2703–2710. IEEE, June 2009. 26, 28

Xi Zhou, Kai Yu, Tong Zhang, and Thomas S Huang. Image classification using super-vector coding of local image descriptors. In *European Conference on Computer Vision*, pages 141–154, 2010. 19

Zona Research. The need for speed ii. *Zona Market Bulletin*, (5):1–9, April 2001. 53

Publication list

Journal papers

Takuya Minagawa and Hideo Saito. Speed up in computation of hmax features for object localization. *ITE Transactions on Media Technology and Applications*, 2(14), January 2014. 33

Domestic journal papers

皆川 卓也, 斎藤 英雄. 画像クエリによる情報検索システム構築のための階層型オブジェクトカテゴリ認識手法. **電気学会論文誌** *C*, 129 (5):947–955, May 2009.

Peer-reviewed conference papers

Takuya Minagawa and Hideo Saito. Image based search system using hierarchical object category recognition technique. In *Proc. of IAPR Conference on Machine Vision Applications*, pages 219–222, May 2009. 31, 33, 34, 35, 52, 61, 86, 90

Ruiko Miyano, Takuya Inoue, Takuya Minagawa, Yuko Uemtsu, and Hideo Saito. Camera pose estimation of a smartphone at a field without interest points. In *ACCV Workshop on Intelligent Mobile Vision*, Nov 2012.

Ruiko Miyano, Takuya Inoue, Takuya Minagawa, Yuko Uemtsu, and Hideo Saito. A mobile ar system for sports spectators using multiple viewpoint cameras. In *International Conference on Computer Vision Theory and Applications*, Feb 2013.

Domestic conferences

皆川 卓也, 井原 健喜, 斎藤 英雄. 局所特徴の空間的分布を用いた coarse to fine な物体検出手法. **電子情報通信学会技術研究報告**, pages 259–264, Nov 2009.

皆川 卓也, 斎藤 英雄. 階層型オブジェクトカテゴリ認識手法を用いた画像クエリによる情報検索システム. **電子情報通信学会技術研究報告**, pages 127–134, Sep 2008.

Others

Takuya Minagawa and Hideo Saito. Face-direction estimating system using stereo vision. In *International Conference on Industrial Electronics, Control, and Instrumentation*, pages 1454–1459, Nov 1997.

Eiichiro Momma and Takuya Minagawa. Introducing opencv for developers. In *The 29th Annual Conference of Robotics Society of Japan*, Jul 2011.

Magazines

皆川 卓也. Opencv で学ぶ画像認識（連載）. *gihyo.jp*. 技術評論社, May 2008.

皆川 卓也. 新春特別企画　コンピュータビジョンの業界動向. *gihyo.jp*. 技術評論社, Jan 2010.

皆川 卓也. コンピュータビジョン勉強会@関東. **情報処理**, volume 52, page 449. 情報処理学会, Apr 2011a.

皆川 卓也. Goggles 時代の画像検索（解説記事）. *Software Design*. 技術評論社, Aug 2011b.