OBJECT DETECTION FOR BIG DATA

A Dissertation presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

GUANG CHEN

Dr. Tony X. Han, Dissertation Supervisor

May 2014

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

OBJECT DETECTION FOR BIG DATA

presented by Guang Chen,

a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Tony X. Han

Dr. Zhihai He

Dr. Michela Becchi

Dr. Wenjun Zeng

Dr. Jianchao Yang

ACKNOWLEDGMENTS

Thank you to my advisor Prof. Tony X. Han. It is my honor to work with you in the past five years. My research benefits a lot from discussions with you. With your great help, we successfully published some papers in top conferences and built up several detection/recognition systems. Cheers!

Thank you to my committee members: Dr. Zhihai He, Dr. Michela Becchi, Dr. Wenjun Zeng and Dr. Jianchao Yang. I learned a lot from you regarding the research attitude. I appreciate the suggestions and comments during the comprehensive exam and dissertation defense.

Thank you to my colleagues at Epson Research and Development Inc.: Dr. Jing Xiao, Dr. Yuanyuan Ding, Dr. Jinjun Wang, Dr. Chenyu Wu. with your help I had two productive and delightful internships in Epson.

Thank you to my colleagues at Adobe Research: Dr. Jianchao Yang, Dr. Hailin Jin, Dr. Jonathan Brandt, Dr. Eli Shechtman, Dr. Aseem Agarwala. It is exiting to build a very effective fine-grained recognition system. Hope we could publish the bussiness service very soon!

Thank you to my lab-mates: Dr. Xiaoyu Wang, Dr. Xutao Lv, Wei Gong, Shuai Tang, Yan Li, Miao Sun, Guobin Chen, Arthur Chen and etc. I will never forget the great time we spent together not only for research discussions, but also for the very happy experiences we hung out together:-)

Thank you to my father Mr. Yanfei Chen, my mother Mrs. Shuping Lu(Lv). Without your strong support, I would not be able to conduct the cutting edge research and always be so happy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

We have observed significant advances in object detection over the past few decades and gladly seen the related research has began to contribute to the world: Vehicles could automatically stop before hitting any pedestrian; Face detectors have been integrated into smart phones and tablets; Video surveillance systems could locate the suspects and stop crimes. All these applications demonstrate the substantial research progress on object detection. However learning a robust object detector is still quite challenging due to the fact that object detection is a very unbalanced big data problem.

In this dissertation, we aim at improving the object detector's performance from different aspects. For object detection, the state-of-the-art performance is achieved through supervised learning. The performances of object detectors of this kind are mainly determined by two factors: features and underlying classification algorithms. We have done thorough research on both of these factors. Our contribution involves model adaption, local learning, contextual boosting, template learning and feature development. Since the object detection is an unbalanced problem, in which positive examples are hard to be collected, we propose to adapt a general object detector for a specific scenario with a few positive examples; To handle the large intra-class variation problem lying in object detection task, We propose a local adaptation method to learn a set of efficient and effective detectors for a single object category; To extract the effective context from the huge amount of negative data in object detection, we introduce a novel contextual descriptor to iteratively improve the detector; To detect object with a depth sensor, we design an effective depth descriptor; To distinguish the object categories with the similar appearance, we propose a local feature embedding and template selection algorithm, which has been successfully incorporated into a real-world fine-grained object recognition application. All the proposed algorithms and features have achieved state-of-the-art performances on extensive object detection and recognition benchmark datasets.

Chapter 1

Introduction

Among various approaches to object detection, the sliding window approach [7, 8] dominates due to its good performance [9, 10, 11, 12, 13, 14], efficiency [7, 15], parallelizability, and easy implementation. The sliding-window-based detectors treat the object detection as a classification problem: The whole image is densely scanned from the top left to the bottom right with rectangular scanning windows of different sizes. For each possible scanned rectangle, certain features such as edge histogram, texture histogram, or wavelet coefficients, are extracted and fed to an offline trained classifier using labeled training data. The classifier is trained to classify any rectangle bounding an object of interest as a positive sample and to classify all other rectangles as negative samples.

The performances of sliding-window-based detectors are mainly determined by two factors: the feature and the underlying classification algorithm. In this dissertation, we aim at improving the performance of object detectors by researching on both of these factors. We start from investigation on the aspect of the underlying classification algorithm. Many supervised learning algorithms such as various boosting algorithms [7, 16, 17], SVM of different flavors including linear, kernel, multi-kernel, latent, structured, etc. [8, 12, 14, 9, 13], and Convolutional Neural Networks (CNN) [18], have been applied to object detection during the past decade.

However the performance of supervised learning algorithms heavily depends on the labeled training dataset. For a specific classification task, a generic classifier trained with the data collected from various environments may only achieve fair performance because it has to accommodate the extensive dataset. Whereas the classifier trained using only the data sampled from the testing environment tends to overfit the training data and perform poorly if the variation of the testing dataset is big. This is the trade-off we have to balance in practical applications; we can adapt a generic classifier to a specific task, but we have to tune the adaptation rate according to different application scenarios.

Although the performance of an offline-trained classifier can be improved on-site by adapting the classifier, the performance gain is substantially affected by the adaptation rate. Poor selection of the adaptation rate may worsen the performance of the original classifier. Therefore, automatic model adaptation for classification tasks is an important problem with great application value.

Various model adaptation approaches [19, 20, 21, 22, 23, 24, 25] have been proposed under the scenarios of online learning for detection/tracking tasks. Zhang *et.al.* [26] elegantly formulate the model adaption problem as an optimization problem on the combined cost function of the old dataset and the new dataset. The cost function of the old dataset is approximated with its second order Taylor expansion to alleviate the storage burden and computational complexity. However, in their work, the adaption rate is an empirical parameter affects the adaptation performance significantly and demands careful fine tuning.

To avoid tuning the important adaptation rate parameter, we propose a conservative model adaptation method [27] by considering the worst case during the adaptation process. We first construct a random cover of the set of the adaptation data from its partition. For each element in the cover (*i.e.* a portion of the whole adaptation data set), we define the cross-entropy error function in the form of logistic regression. The element in the cover with the maximum cross-entropy error corresponds to the worst case in the adaptation. Therefore we can convert the conservative model adaptation into the classic min-max optimization problem: finding the adaptation parameters that minimize the maximum of the cross-entropy errors of the cover. Taking the object detection as a testbed, we implement an adapted object detector based on binary classification. Under different adaptation scenarios and different datasets including PASCAL, ImageNet, INRIA, and TUD-Pedestrian, the proposed adaption method achieves significant performance gain and is compared favorably with the state-of-the-art adaptation method [26] equipped with the fine tuned adaptation rate. Without the need of tuning the adaptation rates, the proposed conservative model adaptation method can be extended to other adaptive classification tasks.

With the help of proposed Min-Max model adaptation algorithm, we move on to address another learning problem: striking a good balance between model complexity and learning capacity.To ensure the detector has enough learning capacity to learn from training data and can be generalized well, people frequently resort to the Occam's razor principle [28, 29] to select underlying classifiers: we want to pick up a classifier, as simple as possible, with good performance on training data. With a spectrum of classifiers with different model complexity, is it possible to automatically pick up a classifier with appropriate complexity and to learn the corresponding model parameters? When the distribution of data in the input space is uneven, local learning algorithms can adjust the learning capacity locally to improve the overall performance. Zhang *et.al.* [30] proposed SVM-KNN that successfully tackles the problem of high variance of the data complexity in the input space, at the expense of high computational complexity. Similar idea was introduced in [31]. These local learning algorithms are superior in adjusting the learning capacity according to the local data distribution.

However, the local learning algorithms have three difficulties in real world applications: **First**, probing the local data distribution is very expensive. For example, both of the local learning algorithms [31, 30] rely on the K-Nearest Neighbor (KNN) algorithm to guide the local classifier. This probing procedure limits the application of the local learning algorithms in large scale learning practice such as object detection. **Second**, the localities depend on data distributions: a region with a simple distribution should be covered with a relatively small number of local classifiers whereas a region with a complicated distribution should be covered with a large number of local classifiers. The "*K*" in KNN algorithm is a constant and cannot fulfill such an adaptive task. **Third**, the performance of a local classifier relies on the population of the cluster. Complicated distributions may lead to low-population clusters, making the corresponding local classifier under trained.

To tackle the difficulties above, we developed a hybrid learning algorithm combining the global classification and the local adaptations, which automatically adjusts the model complexity according to the data distribution. As sketched in Figure 3.1, we divide the data samples into two groups, easy samples and ambiguous samples, using a learned global classifier. A local adaptation approach based on spectral clustering and Min-Max model adaptation is then applied to further process the ambiguous samples. The idea behind the proposed hybrid algorithm is straightforward: 1) Easy regions do not need local learning; 2) The local classifier can leverage on global classifier to avoid under-training; 3) Data in hard regions can be automatically clustered based on their affinity matrix using accelerated spectral clustering. Taking human detection as a testbed, under different scenarios and datasets, including Caltech pedestrian dataset [3], self-collected large pedestrian dataset, and INRIA dataset [8], the proposed hybrid learning method achieves significant performance gain. Compared with 11 state-of-the-art algorithms [3] on Caltech, the proposed approaches achieves the highest detection rate, outperforming the deformable part based algorithm [9] 17% at FPPI=1. Additionally, without the need of tuning parameters, the proposed algorithm automatically generates the optimum group (verified by brute force enumeration) of local classifiers for different scenarios, which makes the algorithm easily be extended to different object detection tasks.

Besides the great effect of algorithm design in object detection task, the feature design also plays an important rule. Currently context has been playing an increasingly important role to improve the object detection performance, especially the context information from the surrounding area of a target object. However the different surrounding scenarios make it hard to extract useful contextual feature. To handle this, we propose an effective representation [32], Multi-Order Contextual co-Occurrence (MOCO), to implicitly model the high level context using solely detection responses from a baseline object detector. The so-called (1^{st}-order) context feature is computed as a set of randomized binary comparisons on the response map of the baseline object detector. The statistics of the 1^{st}-order binary context features are further calculated to construct a high order co-occurrence descriptor. Combining the MOCO feature with the original image feature, we can evolve the baseline object detector to a stronger context aware detector. With the updated detector, we can continue the evolution till the contextual improvements saturate. Using the successful deformable-part-model detector [33] as the baseline detector, we test the proposed MOCO evolution framework on the PASCAL VOC 2007 dataset [10] and Caltech pedestrian dataset [34]: The proposed MOCO detector outperforms all known state-of-the-art approaches, contextually boosting deformable part models (ver.5) [33] by 3.3% in mean average precision on the PASCAL 2007 dataset. For the Caltech pedestrian dataset, our method further reduces the log-average miss rate from 48% to 46% and the miss rate at 1 FPPI from 25% to 23%, compared with the best prior art [35].

While color image based detection systems are very popular, depth sensor based detection systems have many advantages such as easy foreground segmentation, insensitivity to illumination variation and texture/background clutter. So we also investigate the depth sensor based object detection problem. For object detection with a depth sensor, feature design is also a key problem. We thus propose an effective feature, *Normal Vector Pattern* (NVP), designed exclusively for object detection with a depth sensor. The trade-off that a feature can achieve between the discriminative power and the invariance is essential for the classification performance: a superior feature should be discriminative enough to detect inter-class variations, whereas the feature invariance minimizes the disturbance caused by the intra-class variations. This feature trade-off depends on both the sensor type and the application. For the depth sensor based object detection task, the proposed NVP is a successful attempt due to its two components: the cross products of neighboring normal vectors induce the discriminative power and the inner products of the normal vectors lead to the view-angle invariance. The superiority of the proposed NVP is evident: with a simple linear SVM model, the proposed framework outperforms the state-of-the-art features based on more complicated models such as the deformable parts model on standard Washington RGB-D [6] and NYU benchmark datasets [36].

Finally for object detection in big data, it is hard but important to distinguish an object category from similar categories. This problem belongs to fine-grained object categorization problem. If not tackle carefully, these similar object categories will affect the detection accuracy substantially. We research this problem with a real-world application: font recognition. Since fonts appear quite similar and have thousands of categories, it is a perfect problem to evaluate the fine-grained object categorization problem. We addresses the large-scale visual font recognition (VFR) problem, which aims at automatic identification of the typeface, weight, and slope of the text in an image or photo without any knowledge of content. Although visual font recognition has many practical applications, it has largely been neglected by the vision community. To address the VFR problem, we construct a large-scale dataset containing 2,420 font classes, which easily exceeds the scale of most image categorization datasets in computer vision. As font recognition is inherently dynamic and open-ended, *i.e.*, new classes and data for existing categories are constantly added to the database over time, we propose a scalable solution [37] based on the nearest class mean classifier (NCM). The core algorithm is built on local feature embedding, local feature metric learning and max-margin template selection, which is naturally amenable to NCM and thus to such open-ended classification problems. The new algorithm can generalize to new classes and new data at little added cost. Extensive experiments demonstrate that our approach is very effective on our synthetic test images, and achieves promising results on real world test images.

Our contributions are in five-folds: 1) we propose a model adaptation algorithm that without the need of tuning the adaptation rates, the algorithm achieves significant performance gain and is compared favorably with the state-of-the-art adaptation method with the fine tuned adaptation rate. 2) we develop a hybrid learning algorithm that enables the application of local learning algorithm in large-scale tasks; 3) we propose a contextual feature which enhances the detection performance by iterative evolution. 4) we propose an effective depth descriptor to improve the object detection with a depth sensor. 5) we propose a local feature embedding and template selection algorithm to distinguish object categories sharing similar appearance.

Chapter 2

Object Detection by Adaptation: Min-Max Model Adaptation

2.1 Overview

Object detection is an unbalanced problem, in which positive examples are hard to collect and label. Thus in this chapter, we focus on adapting a general object detector for a specific scenario with a few positive examples. As we know, the performance of an offline-trained classifier can be improved on-site by adapting the classifier towards newly acquired data. However, the adaptation rate is a tuning parameter affecting the performance gain substantially. Poor selection of the adaptation rate may worsen the performance of the original classifier. To solve this problem, we propose a conservative model adaptation method [27] by considering the worst case during the adaptation process. We first construct a random cover of the set of the adaptation data from its partition. For each element in the cover (*i.e.*a portion of the whole adaptation data set), we define the cross-entropy error function in the form of logistic regression. The element in the cover with the maximum cross-entropy error corresponds to the worst case in the adaptation. Therefore we can convert the conservative model adaptation into the classic min-max optimization problem: finding the adaptation parameters that minimize the maximum of the cross-entropy errors of the cover. Taking the object detection as a testbed, we implement an adapted object detector based on binary classification. Under different adaptation scenarios and different datasets including PASCAL, ImageNet, INRIA, and TUD-Pedestrian, the proposed adaption method achieves significant performance gain and is compared favorably with the state-of-the-art adaptation method with the fine tuned adaptation rate. Without the need of tuning the adaptation rates, the proposed conservative model adaptation method can be extended to other adaptive classification tasks.

2.2 Introduction

The performance of supervised learning algorithms including various flavors of SVM [28, 25], boosting [38, 39, 40, 7], neural networks [41, 18], heavily depends on the labeled training dataset. For a specific classification task, a generic classifier trained with the data collected from various environments may only achieve fair performance because it has to accommodate the extensive dataset. Whereas the classifier trained using only the data sampled from the testing environment tends to overfit the training data and perform poorly if the variation of the testing dataset is big. This is the trade-off we have to balance in practical applications; we can adapt a generic classifier to a specific task, but we have to tune the adaptation rate according to different application scenarios.

Although the performance of an offline-trained classifier can be improved on-site by adapting the classifier, the performance gain is substantially affected by the adaptation rate. Poor selection of the adaptation rate may worsen the performance of the original classifier. Therefore, automatic model adaptation for classification tasks is an important problem with great application value.

To solve this problem, we propose a conservative model adaptation method by considering the worst case during the adaptation process. We first construct a random cover of the set of the adaptation data from its partition. For each element in the cover (*i.e.* a portion of the whole adaptation data set), we define the cross-entropy error function in the form of logistic regression. The element in the cover with the maximum cross-entropy error corresponds to the worst case in the adaptation. Therefore we can convert the conservative model adaptation into the classic min-max optimization problem: finding the adaptation parameters that minimize the maximum of the cross-entropy errors of the cover. Taking the object detection as a testbed, we implement an adapted object detector based on binary classification. Under different adaptation scenarios and different datasets including PASCAL, ImageNet, INRIA, and TUD-Pedestrian, the proposed adaption method achieves significant performance gain and is compared favorably with the state-of-the-art adaptation method with the fine tuned adaptation rate. Without the need of tuning the adaptation rates, the proposed conservative model adaptation method can be extended to other adaptive classification tasks.

The remainder of the chapter is organized as follows. Sec. 2.3 formulates the adaptation cost as the cross-entropy error function. Converting the adaptation in the worst case into an classical min-max optimization problem is discussed in details in sec. 2.4. Experimental results are presented and discussed in Sec. 2.5. Finally we draw the conclusion in Sec. 5.6.

2.3 Cross-Entropy Error as the Adaptation Cost

To train a classifier, positive and negative examples are given as a training dataset $\mathcal{S} = \{(\mathbf{x}_k, y_k), k = 1, \ldots, K, (\mathbf{x}_k, y_k) \in \mathcal{X} \times \mathcal{Y})\}$, in which y_k is the label of the example x_k. If $\mathbf{x}_k$ is a positive example, $y_k = 1$ and $y_k = 0$ if it is a negative example. A parametric learning algorithm is then applied on $\mathcal{S}$, to find the decision function:

$$\mathcal{Z} = F(\mathcal{X}|\mathbf{w}), \tag{2.1}$$

where $\mathbf{w}$ is the parameter vector of the trained classifier. In Eq.(2.1), each $z \in \mathcal{Z}$ is a mapped label of the corresponding example $\mathbf{x} \in \mathcal{X}$. Usually, $\mathbf{w}$ should be optimized according to a cost function defined to measure the classifier performance over the training dataset:

$$C(\mathcal{Z}, \mathcal{S}) = C(F(\mathcal{X}|\mathbf{w}), \mathcal{S}). \tag{2.2}$$

During the model adaptation, the parameter vector of the classifier trained on *old* dataset $\mathcal{S}^{(o)}$ is used as the initial parameter vector, denoted as $\mathbf{w}^{(o)}$. With a labeled dataset $\mathcal{S}^{(n)}$ (also called as adaptation dataset) collected from a new environment, we want to obtain an adapted classifier with the updated parameter vector $\mathbf{w}^{(n)}$, which performs better in the new environment. That is to find the parameter vector minimizing the cost function on the new dataset $\mathcal{S}^{(n)}$:

$$\mathbf{w}^{(n)} = \arg\min_{\mathbf{w}} C^{(n)}(\mathbf{w}) \tag{2.3}$$

There exist many formulations to define the cost function, among which we choose *logistic regression* [42] for its good performance, applicability, and popularity. Specifically, for a data set $\mathcal{S} = \{(\mathbf{x}_k, y_k), k = 1, \dots K\}$, the likelihood of an example x_k being a positive example is:

$$p_k = \frac{1}{1 + exp\{-Score(\mathbf{x}_k, \mathbf{w})\}}, \tag{2.4}$$

where the $Score(\mathbf{x}_k, \mathbf{w})$ is the confidence score of the example $\mathbf{x}_k$ computed by the classifier with the parameter vector $\mathbf{w}$. Therefore, the likelihood function of the whole data set can be written as:

$$P = \prod_{k=1}^{K} p_k^{y_k} (1 - p_k)^{1 - y_k}. \tag{2.5}$$

We define the cost function by taking the negative logarithm of the likelihood; this definition leads to the *cross-entropy* error function:

$$C = -\frac{1}{K} ln\, P = -\frac{1}{K} \sum_{k=1}^{K} \{y_k\, ln\, p_k + (1-y_k)\, ln(1-p_k)\}. \tag{2.6}$$

The entries of the gradient vector $\nabla C(\mathbf{w})$ and the Hessian matrix $\mathbf{H}_C(\mathbf{w})$ of the cost function can be computed as:

$$\frac{\partial C}{\partial w_j} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{x}_k(j)(p_k - t_k), \tag{2.7}$$

and

$$\frac{\partial^2 C}{\partial w_i \partial w_j} = \frac{1}{K} \sum_{k=1}^{K} p_k(1-p_k)\mathbf{x}_k(i)\mathbf{x}_k(j), \tag{2.8}$$

where $\mathbf{x}_k(i)$ is the ith entry of the feature vector $\mathbf{x}_k$.

2.4 Model Adaptation as Min-Max Optimization

The model adaptation problem aims at adapting a previously trained detector, to fit new data with a distribution different from the one of the data used to train the original detector. The distribution of the new data determines the adaptation rate. If the variation within th new data is small, *i.e.* the new data are relatively similar to each other, we should set a high adaptation rate. On the other hand, if the variation within th new data is big, that means fast adaptation to the new data is risky since we cannot estimate the new data distribution robustly; we have to set a low adaptation rate to avoid overfitting. But how do we estimate the variation of new data? We can randomly split the whole new data in to many overlapped subsets and watch the differences among these subsets: big differences indicate large variation and vice versa.

2.4.1 Constructing a random cover from the adaptation dataset

To estimate the data variation of the adaptation set $\mathcal{S}^{(n)}$, we randomly divide it into M small partitions:

$$\mathcal{S}^{(n)} = \bigcup_{i=1}^{M} \mathcal{S}^{(n)_i}, \tag{2.9}$$

constrained by the non-intersection condition:

$$\mathcal{S}^{(n)_i} \cap \mathcal{S}^{(n)_j} = \emptyset, \qquad \text{for} \quad i \neq j, \tag{2.10}$$

where $i, j = 1, \ldots, M$.

With this random partition, we construct an N-element random cover of $\mathcal{S}^{(n)}$, denoted as $\{G_1, G_2, \ldots, G_N\}$, as illustrated in Figure 2.1:

$$G_j = \bigcup_{i \in E_j} \mathcal{S}^{(n)_i}, \tag{2.11}$$

for $j = 1, \ldots, N-1$, and

$$G_N = \mathcal{S}^{(n)} - \bigcup_{j=1}^{N-1} G_j, \tag{2.12}$$

where $E_j \subset \{1, \ldots, N\}$ is a random subset.

This random cover construction avoids the complete enumeration of the power set of the partition (*i.e.*, the uniform sampling), which leads to huge computation.

2.4.2 Min-Max objective function

Since direct evaluate the data distribution on each element (*i.e.*, each G_j) of the random cover is very difficult, we resort to evaluate the cost function on the G_j. We denote the cost functions on the old dataset and the adaptation dataset as $C_{(o)}(\mathbf{w})$ and $C_{(n)}(\mathbf{w})$ respectively. For each G_j, the corresponding cost function is denoted as $C_j(\mathbf{w})$. Their logistic regression

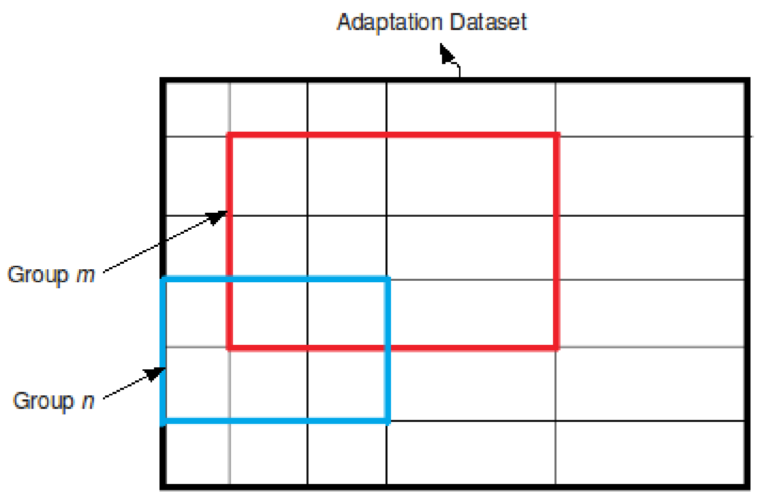


Figure 2.1: The construction of the random cover of the adaptation dataset. Each small block represents $S^{(n)_i}$, an element of the partition. The G_j's could have different size and overlap with each other.

formulation is given in Eq. (2.6).

Different from previous methods which update parameter vector $\mathbf{w}$ through minimizing the $C_{(n)}(\mathbf{w})$, we propose a *conservative adaptation approach*, to guarantee that the adapted detector performs relatively well even when the adaptation dataset has scattered data distribution. During the adaptation, we focus on updating the parameter vector w to improve the detector's performance on the G_j with the largest cost function, *i.e.*, the biggest cross-entropy error. Therefore, we formulate the conservative adaptation into a *min-max* problem:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \left(\max_j \left[C_j(\mathbf{w}) \right] \right). \tag{2.13}$$

Do solve this min-max problem, at each adaptation iteration, we define the cost function on the adaptation dataset as

$$E(\mathbf{w}) = \max_j \left[C_j(\mathbf{w}) \right]. \tag{2.14}$$

2.4.3 Min-Max optimization

Noticing the positivity of $C_j(\mathbf{w})$'s in Eq.(2.13), we can compute $\max[C_j(\mathbf{w})]$ as the infinity norm of the cost function vector, $\mathbf{C}(\mathbf{w}) = (C_1(\mathbf{w}) \quad C_2(\mathbf{w}) \quad \dots \quad C_N(\mathbf{w}))^T$. Therefore, we have

$$\max_j[C_j(\mathbf{w})] = ||\mathbf{C}(\mathbf{w})||_\infty = \lim_{q\to+\infty} \Big(\sum_{j=1}^{N} \big[C_j(\mathbf{w})\big]^q\Big)^{\frac{1}{q}}. \tag{2.15}$$

Therefore Eq. (2.13) can be approximated with a large q:

$$\mathbf{w} = \arg\min_{\mathbf{w}} (E(\mathbf{w})) \simeq \arg\min_{\mathbf{w}} \Big(\sum_{j=1}^{N} \big[C_j(\mathbf{w})\big]^q\Big). \tag{2.16}$$

To update the parameter vector $\mathbf{w}$, we use *Newton-Raphson* [42] method to compute the minimizer of the cost function above:

$$\mathbf{w}^{[i+1]} = \mathbf{w}^{[i]} - \mathbf{H}_E^{-1}(\mathbf{w}^{[i]}) \cdot \nabla E(\mathbf{w}^{[i]}), \tag{2.17}$$

where $\nabla E(\mathbf{w})$ and $\mathbf{H}_E(\mathbf{w})$ are the gradient and the Hessian matrix of the cost function $E(\mathbf{w})$. They can be computed as:

$$\nabla E(\mathbf{w}^{[i]}) = q \sum_{j=1}^{N} \Big(\big[C_j(\mathbf{w}^{[i]})\big]^{q-1} \cdot \nabla C_j(\mathbf{w}^{[i]})\Big) \tag{2.18}$$

$$\begin{aligned}\mathbf{H}_E(\mathbf{w}^{[i]}) = q \sum_{j=1}^{N} \Big(&\big[C_j(\mathbf{w}^{[i]})\big]^{q-1} \cdot \mathbf{H}_{C_j}(\mathbf{w}^{[i]}) \\ &+(q-1)\big[C_j(\mathbf{w}^{[i]})\big]^{q-2} \cdot [\nabla C_j(\mathbf{w}^{[i]})]^2\Big).\end{aligned} \tag{2.19}$$

We use the parameter vector $\mathbf{w}^{(o)}$ of the classifier trained on the old dataset to initialize the iterative optimization process: $\mathbf{w}^{[0]} = \mathbf{w}^{(o)}$. The iterative optimizing process terminates at a certain threshold ξ:

$$\sqrt{\nabla E(\mathbf{w}^{[i]})^T \cdot \mathbf{H}_E^{-1}(\mathbf{w}^{[i]}) \cdot \nabla E(\mathbf{w}^{[i]})} < \xi. \tag{2.20}$$

Our algorithm is summarized in Algorithm(1).

Algorithm 1: The Min-Max Based Adaptation

```
Input: Old detector with parameter vector $\mathbf{w}^{(o)}$ , adaptation dataset $\mathcal{S}^{(n)}$, and
       threshold $\xi$.
Output: $\mathbf{w}^{(n)}$ for adapted detector.
Based on Eq.(2.10), divide $\mathcal{S}^{(n)}$ into $M$ small subsets:
$\mathcal{S}^{(n)} = \mathcal{S}^{(n)_1} \cup \mathcal{S}^{(n)_2} \cup \ldots \cup \mathcal{S}^{(n)_M}$;
form $N$ data covers $G_1, G_2, \ldots, G_N$ refer to Eq.(2.11, 2.12), and get the cost
functions $C_1(\mathbf{w})$, $C_2(\mathbf{w})$,..., $C_N(\mathbf{w})$;
define cost function $E(\mathbf{w})$ on $\mathcal{S}^{(n)}$ using Eq.(2.13);
approximate $E(\mathbf{w})$ as shown in Eq.(2.15);
set $\mathbf{w}^{[0]} = \mathbf{w}^{(o)}, T = \infty, i = 0$;
while $T >= \xi$ do
    compute $\mathbf{H}_E(\mathbf{w}^{[i]})$, $\nabla E(\mathbf{w}^{[i]})$ as Eq.(2.18, 2.19);
    compute $\mathbf{w}^{[i+1]}$ refer to Eq.(2.17);
    $T = \sqrt{\nabla E(\mathbf{w}^{[i]})^T \cdot \mathbf{H}_E^{-1}(\mathbf{w}^{[i]}) \cdot \nabla E(\mathbf{w}^{[i]})}$;
    $i = i + 1$;
$\mathbf{w}^{(n)} = \mathbf{w}^{[i]}$;
return $\mathbf{w}^{(n)}$;
```

2.5 Experiments

We now present experiments to evaluate the proposed model adaptation algorithm. The model adaptation algorithm is test for two group of adaptation scenario: the first three experiments are designed to adapt an original human detector trained on the same old dataset to three different adaptation datasets; the last experiment emphasizes on the adapting a general object detector. The good performance of the proposed adaptation algorithm is validated by these experiments.

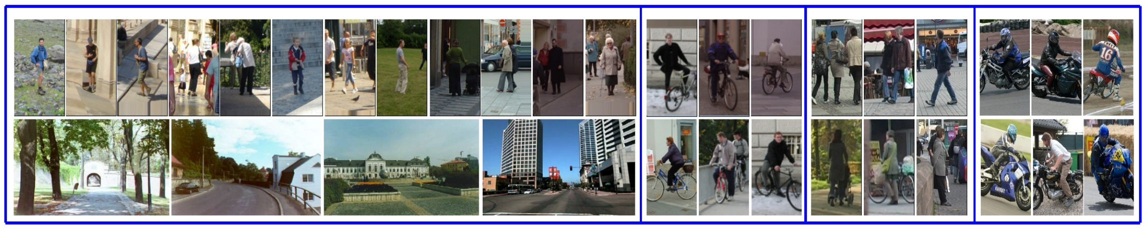


Figure 2.2: From left to right, the first box gives the examples we use to train our generic human detector , all coming from INRIA dataset; the second to the fourth boxes give positive examples sampled from three adaptation dataset: the INRIA dataset, the TUD dataset, and the PASCAL dataset.

The rest of this section consists of seven subsections: the first subsection describes the features we use, and details the parameters used; the second subsection presents the comparison of the adaptation methods; the third subsection introduces the generic human detector we use and evaluates our method on part of the INRIA dataset [8]; the fourth subsection tests our algorithm on the TUD-pedestrian dataset [43]; the fifth subsection adopts our method on adapting a "motorcyclist" detector, with data sampled from the PASCAL dataset [44], and with the analysis on the effects of the related parameters; the sixth subsection focuses on regular object detector: adapting a Parrot detector from a generic bird detector, in which training and test images come from ImageNet dataset [45]; the last subsection gives the computational complexity and a short experimental discussion.

2.5.1 Image representation and experimental setup

Shape and Texture descriptors. In all of the following experiments, we use integrated HOG-LBP features [2]. HOG has been widely accepted as one of the best features to capture the edge or local shape information, whereas LBP is an exceptional texture descriptor.

Parameters setting. For all of the positive and negative examples, we fix their sizes as 64×128 pixels. Thus, the HOG-LBP feature is a 5668 dimensional vector, which makes the Hessian matrix in Eq.$(2.17, 2.19)$ a 5668×5668 matrix. Considering the huge computational cost in computing the Hessian matrix, we use a small positive number $\alpha =$

$0.1 \cdot \mathbf{I}$ to replace the Hessian matrix in Eq.(2.20). To construct the random cover, we firstly split the adaptation set into 10 small subsets, and then form 5 data covers. In the third experiment, we analyze the effect of data cover number N using PASCAL dataset. As mentioned before, the q in Eq.(2.15) affects the approximation of our proposed cost function. Theoretically, bigger q leads to better approximation. In our experiments, we assign 3 to q and achieve quite good performance. The iteration threshold ξ is set 0.04.

2.5.2 Performance comparison between different adaptation methods

Taylor expansion adaptation with optimizing adaptation rate.Taylor expansion adaptation method proposed in [26] gives us a good way to adapt a generic classifier in a new environment. The method uses Taylor expansion of the cost function on the old data as an approximation, and then combines it with the cost function on the adaptation dataset:

$$\begin{aligned} J(\mathbf{w}) \quad \approx \quad & C_{(o)}(\mathbf{w}^{(o)}) + \nabla C_{(o)}(\mathbf{w}^{(o)} \cdot (\mathbf{w} - \mathbf{w}^{(o)}) + \\ & \frac{1}{2}(\mathbf{w} - \mathbf{w}^{(o)})^T \cdot \mathbf{H}_{C_{(o)}}(\mathbf{w}^{(o)}) \cdot (\mathbf{w} - \mathbf{w}^{(o)}) \\ & \lambda \cdot C_{(n)}(\mathbf{w}) \end{aligned} \tag{2.21}$$

where $J(\mathbf{w})$ is the overall cost function, and λ is the parameter controlling the adaptation rate.

The adaptation rate λ is a very important parameter and requires careful tuning in [26]. The poor setting of λ will make the adapted detector perform worse than the original detector. So we improve this algorithm using *cross-validation* to find the optimal λ. Shown in Algorithm.(2), the improved version of [26] is used as a baseline to compare with our conservative adaptation algorithm based on min-max.

Algorithm 2: Taylor Expansion Adaptation With Optimizing Adaptation Rate

```
Input: Old detector with parameter vector $\mathbf{w}^{(o)}$ , and adaptation dataset $\mathcal{S}^{(n)}$
Output: $\mathbf{w}^{(n)}$ for adapted detector
divide $\mathcal{S}^{(n)}$ into K same size partitions $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_K$;
$i = 1$;
while $i <= K$ do
    take $\mathcal{S}_i$ to validate, the rest parts to adapt;
    for $\lambda = 0 \to \infty$ do
        use Eq.(2.21) to adapt. validate on $\mathcal{S}_i$, find the optimal $\lambda_i$
    $i = i + 1$;
$\bar{\lambda} = \frac{1}{K} \sum_{i=1}^{K} \lambda_i$;
set $\lambda = \bar{\lambda}$ in Eq.(2.21), do adaptation and get $\mathbf{w}^{(n)}$;
return $\mathbf{w}^{(n)}$;
```

2.5.3 INRIA dataset

The INRIA person dataset [8] now has become a standard to evaluate human detector on, which provides 2416 positive samples and 1218 negative images for training, 1132 positive samples and 453 negative images for testing. All the positive samples are set into 64×128 pixels.

Looking at INRIA dataset, we find there exists one interesting point that the positive samples from training and test set can be divided into two parts: pedestrian images and cyclist images. So they could be separately used to train a generic pedestrian detector and to adapt a cyclist detector. We pick up 246 cyclist positive examples from the whole dataset, and take all the 453 negative images from test set, to build up the cyclist dataset. Then using 116 positive examples and 79 negative images in it for adaptation, the rest positive and negative for evaluation. In the INRIA dataset, the rest 3302 positive pedestrian samples together with 1218 negative images from training set are used to train a generic pedestrian detector. This detector is treated as the old classifier in experiment 1-3.

Fig. 2.5(a) shows that the old detector already works very well in the cyclist dataset. When false alarm rate is 0.02, detection rate is already 99.3%. In certain point, it proves our generic pedestrian detector performs well on norm person detection task, suitable to be used as old detector in experiment 1-3. Also, we could see our proposed adaptation algorithm could further improve the detector rate, even when the old one already performs well. However, experiment 1 is not quite able to show the good performance of our algorithm, when comparing with the rest experiments.

2.5.4 TUD-pedestrian dataset

TUD-pedestrian dataset [43] is a new benchmark for pedestrian detection. In this dataset, images are sampled from a long video sequence, which is collected by an on-board camera from a driving car in urban environment. So this dataset is a realistic and challenging pedestrian dataset for human detector. It contains 3552 positive samples with mirroring and 192 negative image pairs for training and 508 test images with 1269 evaluation positive examples.

As we mentioned before, the INRIA pedestrian images train an old detector. From TUD dataset, we use 1000 positive samples with mirroring from training dataset and 1620 negative examples extracted from 81 frames(20 negative examples per image), to adapt the old detector. Then we test our adaptation algorithm on the whole TUD testing dataset(1269 positive examples, 10160 negative examples from 508 images). From Fig. 2.5(b), it is easy to observe that the detector adapted by our algorithm outperforms the old detector by $15\% - 20\%$ detection rate in FPPW, and $7\% - 10\%$ higher comparing with the Taylor expansion algorithm result with optimal parameter λ.

2.5.5 PASCAL dataset

PASCAL VOC [44] is probably one of the most difficult and widely used reference datasets in computer vision. The PASCAL VOC 2007 is the latest version having labels for both training and test datasets. For the human detection specific category, there are $2000+$ images for training and $2000+$ images for testing. Using our generic human detector on these,

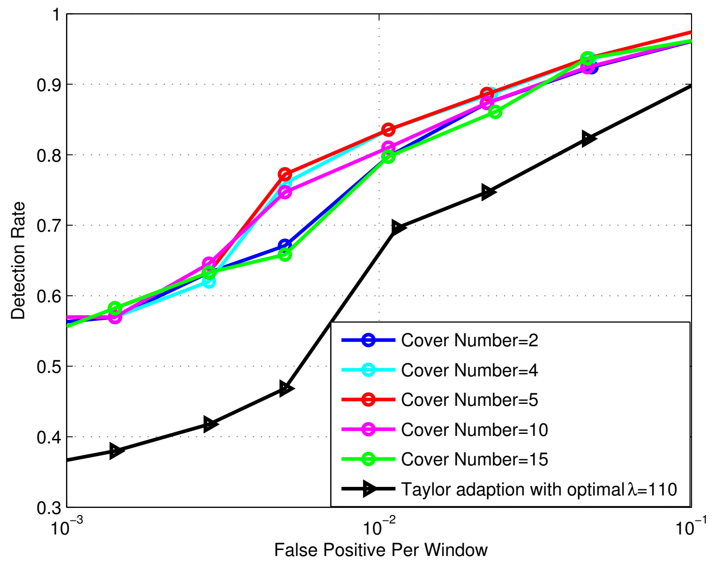


Figure 2.3: Evaluation of the effect of data cover number N in our algorithm, experiment is on PASCAL dataset. We randomly set N, and draw the curve when $N = 2, 4, 5, 10, 15$, all comparing with Taylor adaptation method with optimal λ.

we find the performance in "motorcyclist" images is bad. So we extract these images out, constructing a motorcyclist dataset, containing 600 positive examples, and 200 negative images randomly selected from images which not contain person in PASCAL dataset. From this small dataset, we take 92 positive samples, and 1440 negative samples extracted from 72 images(20 negative examples per image) as our adaptation dataset, the rest positive and negative are all considered as test dataset.

Fig. 2.5(c) illustrates how badly the old classifier works on this motorcyclist dataset. If using Taylor expansion method, we can improve it a lot, about 30% detection rate in FPPW. While using our adaptation algorithm, the performance can be further improved by $10\% - 15\%$, and now the adapted detector turns out to be a good detector on this new dataset.

Effect of data cover number N In Fig. 2.5(c), there exists a *STD bar*(standard deviation bar) in the curve of our proposed adaptation algorithm. It represents the effect of different adaptation data cover number N to our algorithm. It is obvious that this effect is small and tolerable. Tuning N, the performance of our adaptation algorithm is always good with slight fluctuation. This means, our algorithm is stable to parameters. So it doesn't have much troubles in tuning parameters. Fig. (2.3) presents more details.

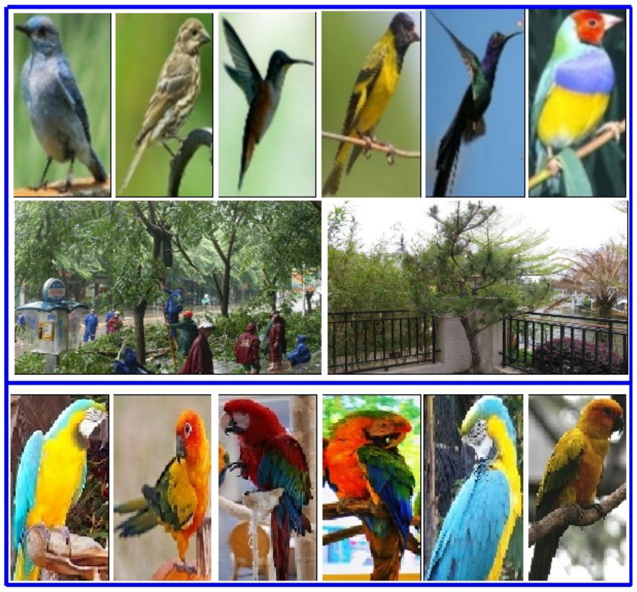


Figure 2.4: First two rows show positive and negative examples to train our bird detector, all extracted from ImageNet dataset; The third row gives some positive examples we labeled for our parrot dataset

2.5.6 ImageNet dataset

ImageNet [45] is a huge image dataset designed according to the WordNet [46] hierarchy(recently only with the $8000+$ nouns), in which each node of the hierarchy is described with hundreds and thousands of images.

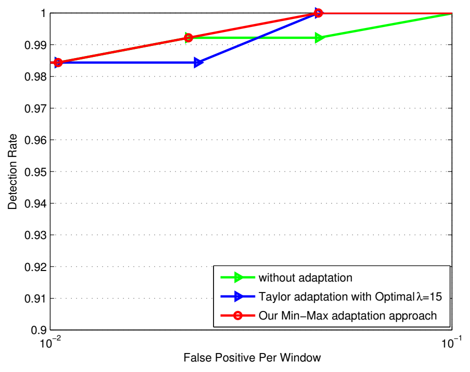
This dataset follows tree-like structure, while images of each concept in it own good quality and provide some annotations. These characteristics make it able to train tons of object detectors. Considering the time and labor consumption, training a generic detector for node in low level and adapting this for node in high level(root node is treated as level 0) might be an optimal way to get detectors for different categories.

Our proposed adaptation algorithm provides an efficient, good and non-expensive method to complete this job. We demonstrate this in bird category. From the ImageNet dataset, we extract 2762 bird positive examples and 749 negative images from different other categories to train a generic bird detector, which is regarded as old detector in this experiment. Then we labeled a small *parrot* dataset, including 400 parrot positive examples, and 100 negative images selected from other categories. 20% of the parrot dataset are used for adaptation, the rest for test. In Fig. 2.5(d), we could see the adapted detector obtained by our algorithm outperforms the old bird detector by $10\% - 20\%$, the Taylor's adapted detector by $5\% - 10\%$ in FPPW. So it proves our algorithm could make a norm generic detector adapt into a good detector for certain specific category.

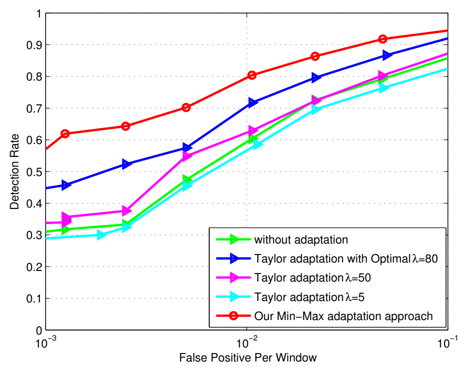
2.5.7 Experiments discussion

Computational Complexity On a Core 2 Duo 2.8 GHz computer, when we adapt with 2000 examples which generate 5 data covers, the average adaptation time of our proposed algorithm is 3 hours for 10000 adaptation iterations, which usually returns near-optimal adapted detector.

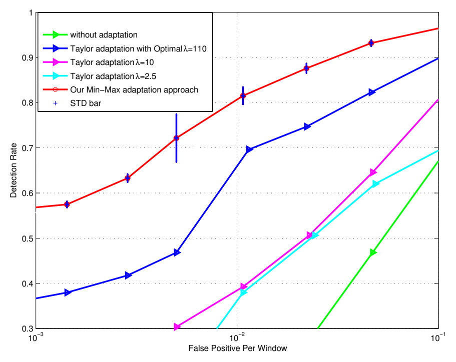
These experiments indicate, firstly the adapted detector generated by our algorithm works well in different detector adaptation tasks and outperforms the old detector and the Taylor expansion adaptation detector. Even when the old detector already performs very well in certain environment, as shown in experiment 1, our adapted detector also further improves the performance. Secondly, Comparing with the Taylor expansion adaptation method, our adaptation method has much less "tuning parameter" troubles, which means it's easy and efficient to apply for detector adaptation tasks. Finally, from the PASCAL experiment, we find the adaptation data cover number N during division of adaptation dataset affects less to our algorithm's performance. In Figure(2.6), we sample some detection results from our experiments. Comparing with old detector and Taylor's adapted detector, the results from our adapted detector are mostly better. The positive and negative examples are separated further away.



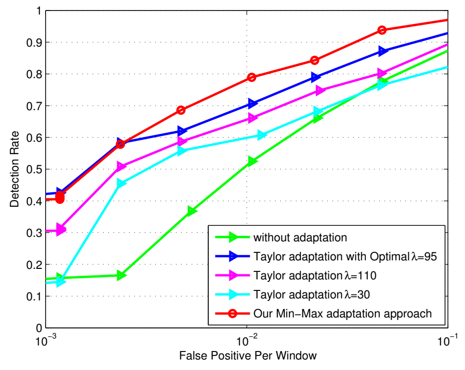
(a) INRIA dataset



(b) TUD dataset



(c) PASCAL dataset



(d) ImageNet dataset

Figure 2.5: Evaluation of our method on multiple datasets, compared with other adaptation method. In the curve of PASCAL dataset, the standard deviation bar represents the effect of data cover number N on the performance of our algorithm.

2.6 Conclusions

We have proposed an object detector adaptation algorithm which has little worry about tuning many parameters. This idea is to generate several adaptation data covers from the adaptation dataset according to certain rules, and during each adaptation iteration, always consider the worse data cover. The adaptation problem then changes into a Min-Max problem which could be solved by approximation of infinite norm method. This algorithm has the ability to train performance-guaranteed detectors for different categories in various environments. This characteristic of our algorithm is also demonstrated in experiments.

0.426	0.440	0.028	-4.280	-3.307	-4.033
0.434	0.437	-0.201	-5.264	-3.917	-3.925
1.060	0.927	0.301	-9.897	-5.321	-8.376

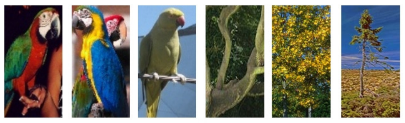
(a) INRIA dataset

-2.351	-0.422	0.414	-2.561	-1.338	-2.686
1.767	2.057	2.119	-1.510	-0.992	-3.992
6.646	3.682	6.047	-2.695	-5.29	-10.495

(b) TUD dataset

-2.718	-1.728	-2.101	-3.392	-3.505	-2.719
-0.714	-0.172	1.649	-2.908	-2.837	-1.890
0.142	0.975	3.018	-4.121	-3.082	-1.712

(c) PASCAL dataset



1.031	1.520	1.222	-2.191	-1.523	-0.808
4.362	4.517	1.432	-2.643	-1.743	-1.380
7.505	7.434	2.673	-2.717	-2.175	-3.566

(d) ImageNet dataset

Figure 2.6: Performance comparison between detectors, all the number are detection scores.The first row is from old detector; the second row is from adapted detector generated by Taylor expansion adaptation method with the optimal adaptation rate; the third row is from detector adapted by our proposed algorithm. In each dataset, we samples 3 positive and 3 negative examples. The score difference between positive and negative examples is larger, the performance is better.

Chapter 3

Object Detection by Local Learning: Divide with Global Classifier, Conquer with Local Adaptation

3.1 Overview

In previous chapter, we introduce a novel Min-Max model adaptation algorithm. Starting from it, we move on to propose a local learning algorithm to handle large intra-class variation problem in object detection, which would be detailed in this chapter. Observing the fact that classifiers used for object detection are task dependent and data driven, we develop a hybrid learning algorithm combining global classification and local adaptations, which automatically adjusts the model complexity according to the training data distribution. Using a learned global classifier, we divide the data samples into two groups, the group of easy samples and the group of ambiguous samples. A local adaptation approach based on classification-accuracy-driven clustering and Min-Max model adaptation is then applied to further process the ambiguous samples. The proposed algorithm automatically determines model complexity of the local learning algorithm according to the distribution of ambiguous samples. By striking a balance between model complexity and learning ca-

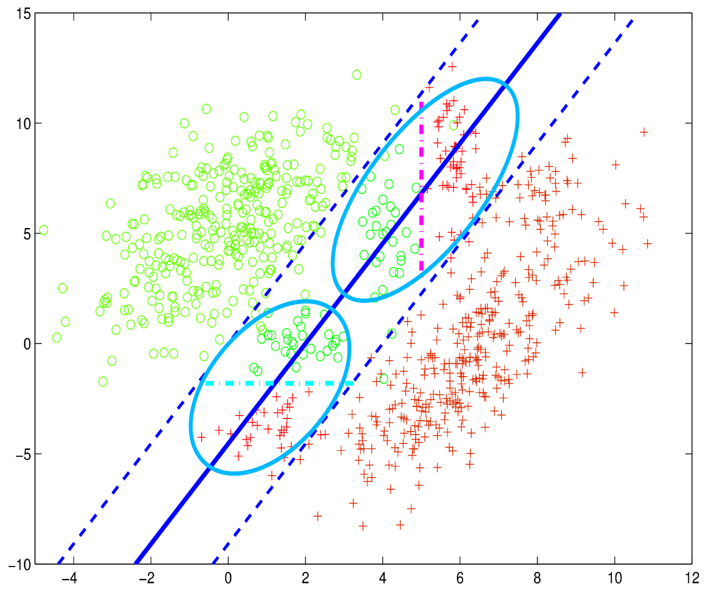


Figure 3.1: A Toy Example of Two-class ("o" vs "x") Classification Using Our Approach: A global classifier (blue solid line) and its boundaries (blue dotted lines) divide the data space into easy regions and hard regions. The ambiguous data in hard regions are clustered according to the data distribution, which automatically adjusts the model complexity. Each cluster of samples are classified using locally adapted classifier that avoids under training. The hybrid learning algorithm autonomously strikes a balance between model complexity and learning capacity.

pacity, the proposed hybrid learning algorithm incarnates a human detector outperforming the state-of-the-art algorithms on a couple of benchmark datasets [8, 3] and a self-collected pedestrian dataset. Compared with lots of state-of-the-art algorithms benchmarked on the Caltech dataset [3], the proposed approaches achieves the lowest Log-average miss rate.

3.2 Introduction

Among various approaches to object detection, the sliding window approach [7, 8] dominates due to its good performance [9, 10, 11, 12, 13, 14], efficiency [7, 15], parallelizability, and easy implementation. The sliding-window-based detectors treat the object detection as a classification problem: The whole image is densely scanned from the top left to the bottom right with rectangular scanning windows of different sizes. For each possible scanned rectangle, certain features such as edge histogram, texture histogram, or wavelet coefficients, are extracted and fed to a offline trained classifier using labeled training data. The classifier is trained to classify any rectangle bounding an object of interest as a positive sample and to classify all other rectangles as negative samples.

The performances of object detectors of this kind are mainly determined by two factors: features and underlying classification algorithms. In this work, we aim at improving the performance of object detectors from the aspect of classification algorithm. Observing the fact that classifiers used for object detection are task dependent and data driven, we developed a hybrid learning algorithm combining global classification and local adaptations, which automatically adjusts model complexity according to data distribution. We divide data samples into two groups, easy samples and ambiguous samples, using a learned global classifier. A local adaptation approach based on spectral clustering and Min-Max model adaptation is then applied to further process the ambiguous samples. The proposed algorithm automatically determines model complexity of the local learning algorithm according to the distribution of ambiguous samples. By autonomously striking a balance between model complexity and learning capacity, the proposed hybrid learning algorithm incarnates a human detector outperforming the state-of-the-art algorithms on a couple of benchmark datasets [8, 3] and a self-collected pedestrian dataset. Compared with 11 state-of-the-art algorithms [3] on the Caltech dataset, the proposed approaches achieves the highest detection rate, outperforming the seminal and successful deformable model approach [9] by 17% at FPPI=1.

Our contributions are in three-folds: 1) we develop a hybrid learning algorithm that enables the application of local learning algorithm in large-scale tasks; 2) the proposed hybrid learning method can automatically adjust the model complexity according to the distribution of the training data; 3) the proposed scheme of local adaptation from global classifier avoids the common under-training problem for local classifier: we gain significant performance enhancement in human detection over traditional algorithms, with very little

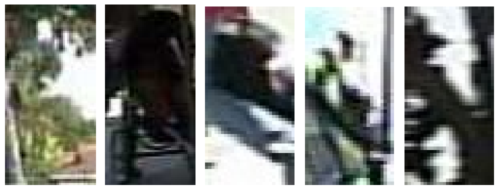
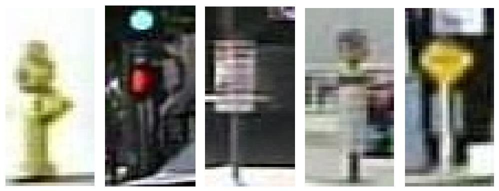
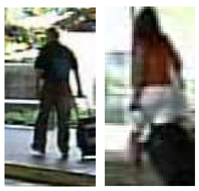
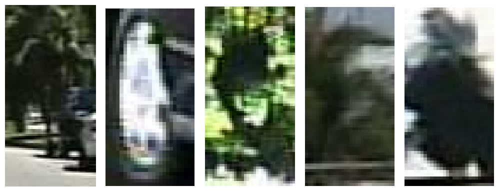
Postive	Negative
	
	
	

Figure 3.2: Sample results of our automatic clustering on ambiguous sample data. Each row corresponds to a particular cluster, showing similar shape, background, and appearance.

increment in computational cost.

The rest of the chapter is organized as follows. Sec. 3.3 describes our global classification process for dividing the candidates into easy and ambiguous cases. Sec. 3.4 details our clustering method for balancing model complexity and learning capacity. Sec. 3.5 presents our local adaptation algorithm to further enhance learning capacity. Experimental results are shown in Sec. 3.6 followed by conclusion and future work in Sec. 3.7.

3.3 Divide by Global Classification

Our approach starts with a global classifier learned using all of the training data. The classified training data are then divided into two groups: easy samples and ambiguous samples. The ambiguous samples are further processed using a local adaptation algorithm.

The learned global classifier partitions the input space into easy regions and hard regions. Only the *ambiguous* data in hard regions will be passed into the next stage and handled by more discriminative local classifiers. Various general global learning algorithms [47, 48, 9, 49] are suitable to this task. Since one role of global classifier is a filter to select hard regions for local learning/adaptation, we require the global classifier to be efficient and highly generalizable with a relative satisfactory performance. Linear SVM meets our requirements. In order to locate the hard regions of *ambiguous* data, we set up the upper bound Θ_1 and lower bound Θ_2 based on the classification scores of the global classifier. The data bounded inside are ambiguous data, requiring local learning.

3.4 Clustering - Adjusting Model Complexity

After filtering by the global classification, the remaining data (ambiguous data) are then processed using an automatic clustering algorithm. This provides an efficient and effective way to probing the local data distribution for each sample. The number of clusters and the population of each cluster are automatically adjusted. Together with a follow-up local adaptation, it strikes a balance between model complexity and learning capacity. Specifically we use a tailored spectral clustering algorithm to automatically divide the ambiguous data into local regions in feature space. This essentially adjusts the model complexity autonomously according to the data distribution.

3.4.1 Distance Metrics

In order to effectively cluster the ambiguous data, we first define the distance metric between a pair of samples in the feature space. Different distance metrics may lead to different clustering on the data. Here for popular shape descriptor and texture descriptor, we introduce several frequently used distance measures.

Crude distance Several simple and yet good measures are frequently used for computing distance in feature space, such as L_1-*sqrt* distance, L_∞-*Norm* distance, and *Euclidean* distance. These crude distance measures have been widely adopted as meaningful solutions to distance computation.

Accurate distance Alternatively, more costly "accurate" distance measures were developed. [50] proposes *shape context distance* that matches the point sets of two shapes and scores how different the shapes are; χ^2 *distance* [51] maps the texture of each example to a histogram of "textons", then defines distance as the Pearson's χ^2 test statistic between the two histograms; Adapted from [51] , *marginal distance* [52] sums up the distances between response histograms to measure the texture distance.

All these metrics may be used for clustering. In our experiments, crude distance already yields reasonable results with low computational complexity. Specifically we adopt the Euclidean-like distance. For each sample, the features are normalized according to their L_2 norm, then the Euclidean distances with others are computed. The normalization is critical for finding and setting proper clustering parameters.

3.4.2 Clustering Algorithm

Many clustering algorithms can be adopted for clustering the ambiguous data. One straightforward method is k-*means* with a given number of clusters k. However inappropriate k may drastically deteriorate the performance of the system: if k is too small, certain local clusters would contain too many samples resulting in over-complicated models; On the contrary, if k is too big, most local clusters may be sparsely populated and inevitably suffer from under-training. Thus care must be taken to choose an appropriate k, which is usually unknown beforehand. To find the appropriate value of k, one may need to exhaustively search all possible k and check the associated performance, demanding formidable computations.

To solve this problem, we adopt spectral clustering to effectively find appropriate value of k and compute the clustering. Inspired by [53, 54] we search for certain drop in the

Algorithm 3: Spectral Clustering with Eigen-Selection

```
Input: ambiguous data points $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$; eigen-selection parameters $\alpha, \beta$
        $(0 < \alpha,\ \beta < 1)$.
Output: $k$ partitions of the input data.
1. Form the affinity matrix $A$ with elements:
        $a_{ij} = exp(-\frac{||Dis(x_i,x_j)||^2}{2{\sigma_{ij}}^2}), \quad i, j = 1, \ldots, n$ ;
2. Compute diagonal matrix $D$ with elements $d_i$=$\sum_{j=1}^n a_{ij}$;
3. Compute Normalized Laplacian matrix $L$:
        $L = D^{-\frac{1}{2}} \cdot (D - A) \cdot D^{-\frac{1}{2}}$;
4. Compute eigenvalues of $L$ and sort in descending order:
        $\lambda_i \geq \lambda_{i+1}, i = 1, 2, \ldots, n-1$
5. Get $k$ by eigenvalue selection:
for $i = 2 \to n$ do
    if $\lambda_i \leq \alpha \cdot \lambda_{i-1}$ $or$ $\lambda_i \leq \beta$ then
        break;
  $k = i - 1$;
6. Form normalized matrix $S$ using $k$ largest eigenvectors;
7. Treating $S$'s row as point, cluster points by $k$-means;
8. Assign original data $x_i$ to cluster $j$ if and only if row $i$ of
   the matrix $S$ was assigned to cluster $j$;
```

magnitude of the eigenvalues to decide the number of clusters. The clustering algorithm is summarized in Algorithm (3).

In Algorithm (3), parameters α and β define the criterion for selecting number of clusters, and $Dis(x_i, x_j)$ is the distance between data points x_i and x_j. The scaling parameter σ_{ij} is a measure to decide whether two examples are similar, which can be specified by self-tuning [55]:

$$\sigma_{ij} = \sqrt{\sigma_i \cdot \sigma_j} \tag{3.1}$$

where,

$$\sigma_i = Dis(x_i, x_{k_{th}}) \tag{3.2}$$

In Eq.(3.2), $x_{k_{th}}$ represents the k'th nearest neighbor of point x_i, typically $k=7$. Although

Algorithm (3) produces high-quality clustering result, the computational complexity of $O(n^3)$ limits its application to large-scale data. Note that usually the number of ambiguous data is huge, so a fast approximation of spectral clustering should be applied, such as KASP [56], explained in step 3–6 of Algorithm (4).

3.4.3 Parameter Selection & Fast Approximation

The spectral clustering algorithm helps effectively avoid exhaustive search for optimal model complexity. To balance learning capacity, we determine the parameters of spectral clustering, α and β, based on the accuracy of corresponding locally learned classifiers. Specifically, we randomly partition the ambiguous training data into M (typically M=10) subsets for cross validation to find the optimal parameters. For each fold, we first apply step 1–4 of Algorithm (3) on training set and sort the eigenvalues in descending order. We search for drops between consecutive eigenvalues that are bigger than half of the prior eigenvalue. The corresponding $\{\alpha, \beta\}$ are marked as candidate parameter sets. For each candidate set, we then apply step 6–8 of Algorithm (3) to construct clusters. Local learning algorithm (Section 3.5) is applied for each cluster and the detection rates are evaluated. Among all candidate parameter sets, we use the one with the best detection rate as the optimal parameters.

Given the clustering parameters α and β, we can apply Algorithm (3) to do clustering. However as explained in Sec. 3.4.2, when the data set is large, directly applying Algorithm (3) is computationally prohibitive. We use a fast approximation KASP to speed up this process. Specifically, for the desired k clusters, we first apply k_0-means and compute the centroid $\{y_j\}_{j=1}^{k_0}$ of each cluster (k_0 defined in step 1–3 of Algorithm (4)). We then run Algorithm (3) on $\{y_j\}_{j=1}^{k_0}$. The cluster membership of each sample x_i is recovered using a table of correspondences with y_j.

The complete proposed clustering algorithm with parameter selection and fast approximation is summarized in Algorithm (4). It not only speeds up the original clustering

Algorithm 4: Accelerated Automatic Clustering

```
Input: ambiguous data points $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$.
Output: same as Algorithm (3)
1. Randomly split $\{\mathbf{x}_i\}_{i=1}^n$ into partitions $\{\mathbf{P}_i\}_{i=1}^M$ for
   $M$-fold cross validation to find optimal parameters $\alpha, \beta$:
for $m = 1 \rightarrow M$ do
    1) Define training set $\mathbf{P}_{tr}$ and validation set $\mathbf{P}_{va}$
         $\mathbf{P}_{tr} = \{\mathbf{x}_i\}_{i=1}^n - \mathbf{P}_m, \quad \mathbf{P}_{va} = \mathbf{P}_m$;
    2) Apply step. 1-4 of Algorithm (3) on $\mathbf{P}_{tr}$ and get all
       eigenvalues $\{\lambda_i^{'}\}_{i=1}^{n_1}$, where $n_1 = |\mathbf{P}_{tr}|$;
    3) Search candidate eigenvalue drops. Initialize $T = 0$:
    for $i = 2 \rightarrow n_1$ do
        if $\lambda_i^{'} \leq 0.5 \cdot \lambda_{i-1}^{'}$ then
            $T = T + 1$;
            $k_T^{'} = i - 1, \ \ \alpha_T' = \lambda_{i-1}^{'}/\lambda_i^{'}, \ \ \beta_T' = \lambda_i^{'}$;
    4) Cluster and check performance for each candidate:
    for $t = 1 \rightarrow T$ do
        Apply step. 6-8 of Algorithm (3) with $k = k_t^{'}$ on
           $\mathbf{P}_{tr}$ to get clusters $\mathbf{C}_t^{'}$;
        Learn local classifiers $\mathbf{F}_t'$ on each cluster of $\mathbf{C}_t^{'}$;
        Evaluate $\mathbf{F}_t'$ on $\mathbf{P}_{va}$ and get detection rate $\varepsilon_t^{'}$;
    5)          $\{\alpha_m, \beta_m, k_m\} = \arg\min_{\{\alpha_t', \beta_t', k_t'\}} \{\varepsilon_t^{'}\}_{t=1}^T$
2. Set optimal parameters:
     $\alpha = \frac{1}{M} \cdot (\sum_{m=1}^M \alpha_m), \ \ \beta = \frac{1}{M} \cdot (\sum_{m=1}^M \beta_m)$
3. Fix a $k_0$, where $k_0 \geq 20 \cdot \frac{1}{M} \cdot (\sum_{m=1}^M k_m)$ and
   $k_0 << n$, then perform $k_0$-means on $\{\mathbf{x}_i\}_{i=1}^n$ to get the
   cluster centroids $\{\mathbf{y}_j\}_{j=1}^{k_0}$ ;
4. Build a correspondence table to associate each $x_i$ with the
   nearest cluster centroid $y_j$;
5. Run Algorithm (3) on $\{\mathbf{y}_j\}_{j=1}^{k_0}$ to obtain the k cluster
   membership for each of $y_i$;
6. Recover the cluster membership for each $x_i$ using that of
   the associated $y_j$ by looking up the correspondence table.
```

method in Algorithm (3), but also simplifies the query. During test, for each sample we simply compute its nearest neighbor in the k_0 centers, then use the correspondence table to find its cluster membership. A sample of clustering results by our algorithm is shown in Figure (3.2), where different rows correspond to different clusters. As we can see, the proposed method works well: images from each row shares similar shape, background, and appearance.

3.5 Local Adaptation - Enhancing Learning Capacity

After the ambiguous data being appropriately clustered, local learning is used to enhance the learning capacity.

A straightforward local learning approach is to train a general classifier [49, 9] directly using the data from each local cluster. Considering speed and performance, linear SVM seems to be a good choice. However the disadvantage of direct learning is obvious: it only uses limited samples in a local cluster and discards the information from the whole dataset that are usually beneficial. Hence, the performance of generated local classifier heavily relies on the population and data distribution of the cluster, and often suffers from under-training.

To address this issue, we propose a model adaptation strategy that leverages on global classifier for effective local learning. Though the global classifier F_0 trained in the first stage may not behave perfectly on each local cluster, it should not be too far from the optimum classification boundary. Furthermore, it contains non-negligible information about global data distribution. Therefore, we treat the local learning problem as utilizing a *coarse* global classifier F_0 to adapt into different *fine* local classifiers. This effectively enhances the learning capacity of our classification algorithm in each local cluster.

Here we adopt Min-Max model adaptation [27] in our algorithm. Comparing with other state of the art adaptation methods, such as [21, 25, 26, 24], the Min-Max is free of tuning parameters(e.g., the adaptation rate) and able to adapt from general parametric classifier. Thus F_0 could be trained by various complicated methods such as [49, 9]. We summarize our local adaptation approach in Algorithm (5).

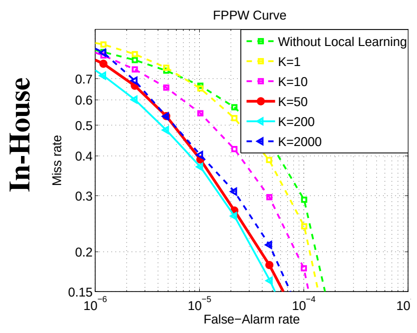
Algorithm 5: Local Learning by Min-Max Adaptation

```
Input: Pre-learned global classifier $F_0$ with parameters $\mathbf{w}_0$; $K$ clusters $\{S_k\}_{k=1}^{K}$
        obtained using Algorithm (4); Min-Max adaptation parameters.
Output: Adapted local classifiers $\{F_k^{'}\}_{k=1}^{K}$
for $k = 1 \to K$ do
    1. Form $N$ data covers$\{G_j\}_{j=1}^{N}$ from data in $\mathcal{S}_k$;
    2. Build the cost functions $\{C_j\}_{j=1}^{N}$ based on
       logistic regression;
    3. Define cost function $E(\mathbf{w})$ on $\mathcal{S}_k$:
          $E(\mathbf{w}) = \Big( \max_j \big[C_j(\mathbf{w})\big] \Big)$
    4. Approximate $E(\mathbf{w})$ as $\infty$-*Norm*:
          $E(\mathbf{w}) = ||\mathbf{C}(\mathbf{w})||_{\infty} \approx \Big( \sum_{j=1}^{N} \big[C_j(\mathbf{w})\big]^q \Big)^{\frac{1}{q}}$
    5. Set $\mathbf{w}^{[0]} = \mathbf{w}_0, T = \infty, i = 0$;
    while $T >= \xi$ do
        compute the gradient and Hessian matrix $\nabla E(\mathbf{w}^{[i]})$, $\mathbf{H}_E(\mathbf{w}^{[i]})$, and update $\mathbf{w}$:
        $\mathbf{w}^{[i+1]} = \mathbf{w}^{[i]} - \mathbf{H}_E^{-1}(\mathbf{w}^{[i]}) \cdot \nabla E(\mathbf{w}^{[i]})$;
        $T = \sqrt{\nabla E(\mathbf{w}^{[i]})^T \cdot \mathbf{H}_E^{-1}(\mathbf{w}^{[i]}) \cdot \nabla E(\mathbf{w}^{[i]})}$;
        $i = i + 1$;
    6. Set parameter for classifier $F_k^{'}$: $\mathbf{w}_k^{'} = \mathbf{w}^{[i]}$;
```

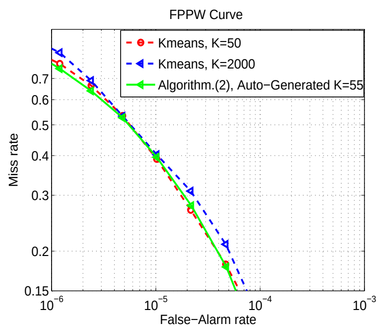
3.6 Experiments

We evaluate our algorithm on pedestrian detection from images that is important and yet challenging in practice. We compare our algorithm with state of the art single layer (non-cascaded) detectors and demonstrate that our algorithm greatly improves the detection rates. We also compare our approach with state of art using the same evaluation methodology used in [3]. We show both qualitative and quantitative results on several challenging datasets to confirm the advantages of our approach.

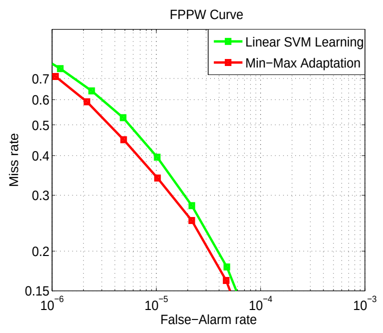
The rest of this section consists of five parts: the first one explains the experiment design and implementation details; the second part tests our algorithm on a large challenging pedestrian dataset "In-House" collected by ourselves; the third and fourth parts respectively evaluate our algorithm on two popular benchmarks: Caltech [3] and INRIA [8] datasets;



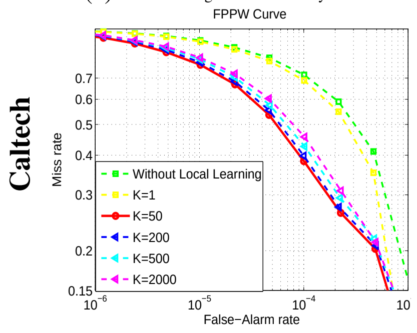
(a) Local learning with K-means by linear SVM



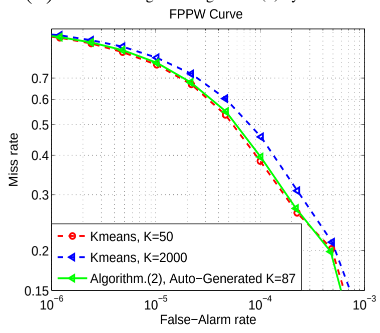
(b) Local learning with Algorithm (4) by linear SVM



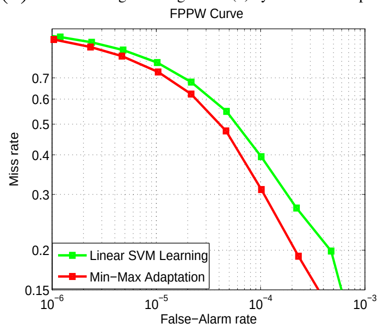
(c) Local learning with Algorithm (4) by Min-Max adaptation



(d) Local learning with K-means by linear SVM



(e) Local learning with Algorithm (4) by linear SVM



(f) Local learning with Algorithm (4) by Min-Max adaptation

Figure 3.3: Evaluation of the proposed algorithm on In-House (1^{st} row) and Caltech (2^{nd} row) datasets. All results are plotted as miss-rate w.r.t. false-alarm-rate in FPPW. The first column details performances of local linear SVM learning with k-means clustering, where k varies from 1 to 2000; The second column shows performance achieved by proposed clustering methods, compared with the best results in the first column; The final column compares performances achieved by 2 different local learning methods.

and finally the algorithm efficiency is discussed.

3.6.1 Experimental Setup

Parameter setting. In Algorithm (3) the eigenvalue-selection parameters α, β are chosen as discussed in Sec. 3.4.3. In our experiments, α=0.2 and β=0.01 yields the best performance. This holds true over cross-validation experiments with the In-House dataset. Later experiments confirm that the same setting also applies for Caltech and INRIA datasets. As stated before, the scaling parameter σ_{ij} could be automatically decided by self-tuning, which requires extra computation on neighbor searching to ensure good performance. In practice we found the local-learning performance is similar to self-tuning when σ_{ij} is between 2 and 5 and the performance is insensitive to different values of σ_{ij} in that range. Thus to speed up, σ_{ij} can be simply set as a constant, e.g. 3. The Min-Max adaptation parameters in Algorithms (5) are the same as [27].

Image representation. We test both shape (HOG [8]) and texture (LBP [2]) descriptors with our learning framework, since HOG has been widely accepted as one of the best features to capture the edge or local shape information and the LBP is an exceptional texture descriptor. We use only the HOG descriptor for In-House and Caltech datasets, and HOG together with LBP for INRIA dataset. Experimental results confirm that our algorithm can robustly find the clusters that yield good detection rates.

Experiments design and evaluation measure. First, to show the effect of clustering on detection accuracy, we cluster every dataset by simple k-means with different values of k. We then construct local classifiers for individual clusters by linear SVM and compare the overall performance with respect to different values of k. Second, to show the effectiveness of our clustering algorithm, we compare the performance of our clustering method with the best result achieved by k-means approach. Finally, we compare the performances improvement gained by two different local learning methods: directly learning by linear SVM and local adaption with Min-Max, not only on detection rate but on speed. For all three types of experiments, we plot detection curves in terms of FPPW instead of FPPI to evaluate the performance of classifiers, since FPPW allows a better assessment of the learning algorithm and isolates the impact of post-processing such as non-maximal suppression. Additionally, in order to compare with 11 algorithms [3] presented in Caltech dataset [3], we also present the accuracy of our detection system in FPPI for Caltech dataset.

3.6.2 In-House dataset

The In-House dataset is collected and labeled by ourselves, containing 5691 images with fixed resolution at 320×240. Performing detection on this dataset is challenging due to the fact that many of the pedestrians are small, partly occluded, and hardly identifiable from background even by human eyes. We randomly divide the dataset into 3 partitions for 3-fold cross-validation.

Firstly, we directly apply k-means clustering followed by local learning using linear SVM with k varying from 1 to 2000. The results with different k are shown in Figure 3.3(a). $k = 1$ means considering all ambiguous data as a single cluster and training for one classifier, similar to a re-training on the whole dataset. As we can see, when k increases from 1 to 200, the performance improves logarithmically. However, if k exceeds 200, the performance starts to drop. The results at k=50 or 200 are similarly the best. Since smaller number of clusters leads to better efficiency in the testing stage, we take k=50 as the optimal clustering. The results confirm that an appropriate local learning would greatly improve the detection performance. Compared with the traditional approach without local adaptation, the miss rate at k=50 is significantly reduced by 25% at 10^{-5} and 16% at 10^{-4} False-Alarm rate in FPPW.

Secondly, we evaluate our local learning algorithm with the proposed clustering and compare with the best results in the first experiment. With a proper k_0=2000, the number of clusters computed by Algorithm (4) is 55. We then perform local learning with linear SVM on individual clusters. As shown in Figure 3.3(b), our method achieves the results as good as the best on in the first experiment. It confirms that our method autonomously strikes a good balance between model complexity and learning capacity.

Finally, we compare the two local classification methods. For fair comparison, we learn classifiers based on the proposed clustering in the second experiment. The detection performance of the two types of local learning algorithms are shown in Figure 3.3(c). As we can see, the Min-Max adaptation method performs better, further reducing the miss rate by 5% at 10^{-4} FPPW. Training the linear SVMs is much faster, 55 local classifiers taking only half an hour to train, while Min-Max adaptation takes 1 day. Overall, the proposed local adaptation algorithm achieves the best detection rates, reducing the miss rate by 30% at 10^{-5} and 21% at 10^{-4} FPPW compared with the single global classifier.

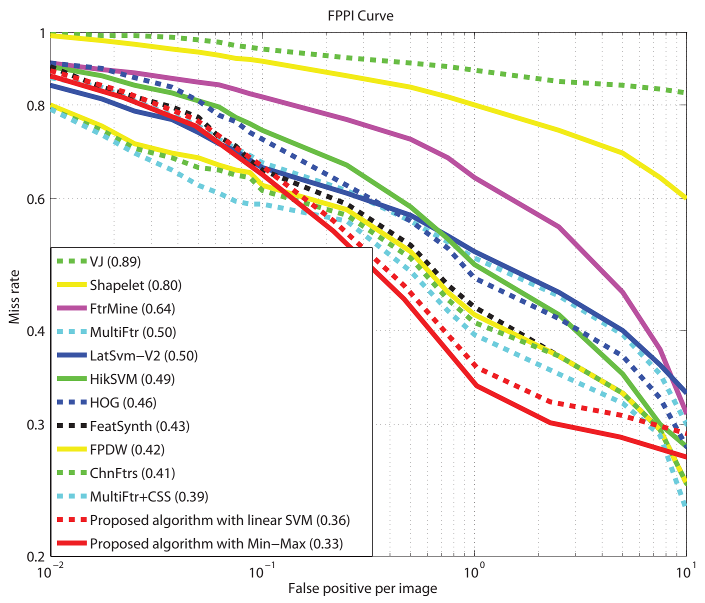


Figure 3.4: Comparison between proposed algorithm and state of the art on Caltech *training* dataset, adopting the same experimental setting and evaluation methodology as [3], plotted as miss-rate w.r.t. FPPI. The precision values at FPPI=1 are shown for easy comparison.

3.6.3 Caltech dataset

Caltech dataset is currently one of the most challenging pedestrian datasets, since pedestrians appear from multiple viewpoints and with a wide range of scales. Additionally, lots of cars and buildings make the background very cluttered. The labeled Caltech *training* dataset contains six sessions (S0-S5), each with multiple videos taken from a moving vehicle. We follow the same 6-folder-cross-validation evaluation methodology as [3] and only consider the “Reasonable” [3] pedestrians.

Following the same step as In-House dataset, we first evaluate the performance with k-means where k varies from 1 to 2000. Figure 3.3(d) show our experiment results. We can see that clusters with k=50 achieve the best result, outperforming the traditional methods by 15% at 10^{-5} and 35% at 10^{-4} in FPPW. Then we test our method with the proposed clustering, which automatically clusters data in 87 classes. The detection rate is similar to the best case in the previous experiment, detailed in Figure 3.3(e). Finally, we compare our local adaptation with direct local learning, as shown in Figure 3.3(f). Again Min-Max

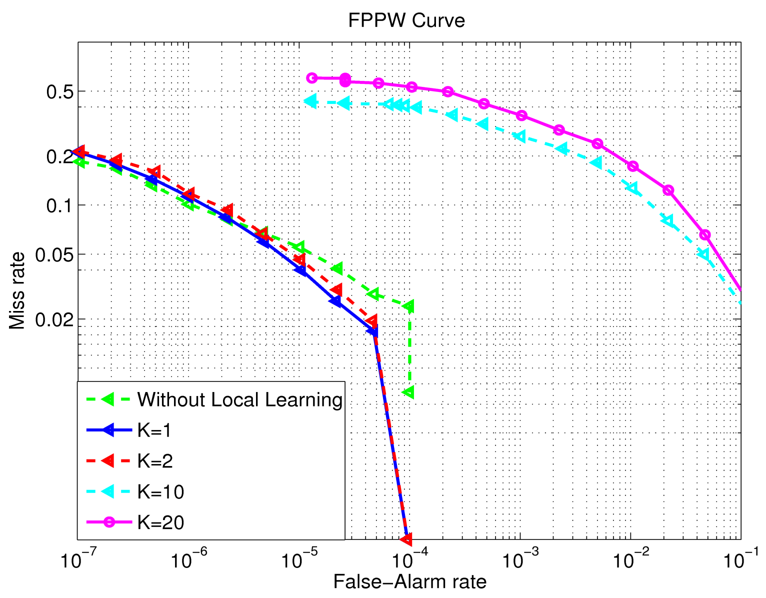


Figure 3.5: Evaluation of local SVM learning after k-means clustering on INRIA dataset. When $k = 1$ or 2 the performance increases slightly. However, further increasing k deteriorates the performance quickly.

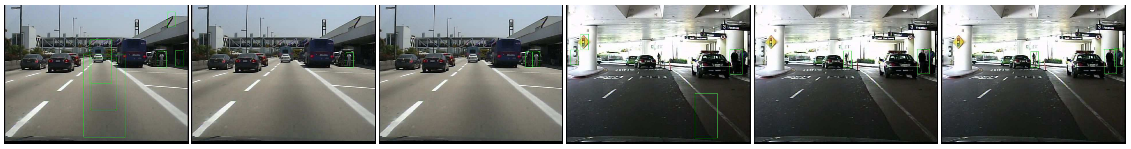
model adaptation achieves about 10% higher detection rate at 10^{-4} FPPW while taking longer time to train. Figure (3.6) show some example images.

Comparison with state-of-the-art Taking the same evaluation procedures as [3], we show the comparison with 11 algorithms [3] on Caltech *training* dataset.From the FPPI curves in Figure (3.4), our algorithm achieves the lowest miss rate at 1 FPPI, 6% lower than the best in [3]. Our method only uses the HOG feature, while many of the 11 algorithms combine several different descriptors. It tells more about the advantage of our method when only comparing with methods using similar features, such as HOG [8], LatSVM-V2 [9], HikSVM [12]. As shown in Figure (3.4), our method outperforms them by 13%, 17%, and 16% at 1 FPPI respectively.

3.6.4 INRIA dataset

INRIA [8] dataset is also popular for researchers to test human detectors. We evaluate on it to show the advantage of our method and its robustness on handling complex backgrounds and using different kinds of features.

Again, we start with testing the performance of k-means clustering. From Figure (3.5), we can see that when k=1 or 2, the performance has a little improvement than that without local adaptation. However, when the cluster number increases, the performance drops quickly. When k=10, the performance drastically decreases by 50% at 10^{-4} FPPW . Inappropriately divide them into smaller clusters will lead to under-training. It also confirms that carefully balancing between model complexity and learning capacity is critical for local learning algorithm to achieve desired performance improvement. The reason is that, for INRIA dataset, the traditional method (HOG+LBP) has already achieved an impressively high detection rate, and the few remaining ambiguous data are difficult to be clustered in feature space (only 2% miss detection in 10^{-4} FPPW). Therefore, instead of gaining improvement in performance, forcing clustering and local learning would make it worse. In such a case, the proposed clustering method is able to automatically identify the proper number of cluster is 1, warning us over-increasing model complexity would break the balance with learning capacity and deteriorate the detection performance. It thus ensures reliable local learning.


(a) Without local learning (b) Local learning by linear SVM (c) Local learning by adaptation (d) Without local learning (e) Local learning by linear SVM (f) Local learning by adaptation

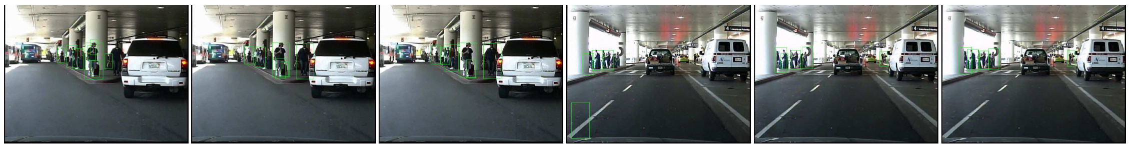

(g) Without local learning (h) Local learning by linear SVM (i) Local learning by adaptation (j) Without local learning (k) Local learning by linear SVM (l) Local learning by adaptation

Figure 3.6: Comparison of detection results on Caltech dataset, between one-stage global classifier without local learning, local learning by linear SVM, and local learning by MinMax adaptation. The last approach achieves both the best detection rate and the lowest false alarm rate.

3.6.5 Computational Complexity in Testing

Compared with one-stage learning methods, the only extra computation for our algorithm is cluster query and classification by corresponding local classifier. Since we use hierarchical k-means with cluster number k_0 before fast spectral clustering, the cluster query computes $\log(k_0)$ times of distance, while SVM-KNN [30] needs at least $\log(n_{train})$ times (n_{train} is the number of samples in training set, where $n_{train} \gg k_0$), and more cost on training kernel SVM during test. From our experiments, adding the local-adaptive stage only takes less than 10% extra time during test.

3.7 Conclusion and Future Work

We developed a hybrid learning algorithm combining global classification and local adaptations, which automatically adjusts model complexity according to data distribution. The proposed adaptation algorithm automatically determines model complexity of the local learning algorithm according to the distribution of the training samples.

In term of classification algorithm, we have shown that our method effectively improves the performance compared with the state-of-art methods, especially when using similar features. On the other hand, the recently proposed features such as Integral Channel Features [57] and Multi-Resolution Features [58] have been successfully used for pedestrian detection detection and achieved highly competitive results. We plan to incorporate such features into our hybrid learning algorithm and believe this can further improve the detection performance.

Chapter 4

Object Detection by Context Learning: Detection Evolution with Multi-Order Contextual Co-occurrence

4.1 Overview

In previous two chapters, we introduce novel algorithms to learn effective detectors. Actually in object detection task, the feature design also plays an important rule. In this chapter, we would like to introduce a novel context feature for object detection. Recently context has been playing an increasingly important role to improve the object detection performance. However the different types of noise and clutter background make it hard to extract useful contextual information. To handle this, we propose an effective representation [32], Multi-Order Contextual co-Occurrence (MOCO), to implicitly model the high level context using solely detection responses from a baseline object detector. The so-called (1^{st}-order) context feature is computed as a set of randomized binary comparisons on the response map of the baseline object detector. The statistics of the 1^{st}-order binary context features are further calculated to construct a high order co-occurrence descriptor. Combining the MOCO feature with the original image feature, we can evolve the baseline object detector to a stronger context aware detector. With the updated detector, we can continue the evolution till the contextual improvements saturate. Using the successful deformable-part-model detector [33] as the baseline detector, we test the proposed MOCO evolution framework on the PASCAL VOC 2007 dataset [10] and Caltech pedestrian dataset [34]: The proposed MOCO detector outperforms all known state-of-the-art approaches, contextually boosting deformable part models (ver.5) [33] by 3.3% in mean average precision on the PASCAL 2007 dataset. For the Caltech pedestrian dataset, our method further reduces the log-average miss rate from 48% to 46% and the miss rate at 1 FPPI from 25% to 23%, compared with the best prior art [35].

4.2 Introduction

Detecting objects from static images is an important and yet highly challenging task and has attracted many interests of computer vision researchers in the recent decades [17, 11, 59, 33, 14, 60, 61]. The difficulties originate from various aspects including large intra-class appearance variation, objects deformation, perspective distortion and alignment issues caused by view point change, and the categorical inconsistency between visual similarity and functionality.

According to the recent results of the standards-making PASCAL grand challenge [10], The detection approach based on sliding window classifiers are presently the predominant method. Such methods extract image features in each scan window and classify the features to determine the confidence of the presence of the target object [62, 7, 63]. They are further enriched to incorporate sub-part models of the target objects and the confidences on sub-parts are assembled to improve detection of the whole objects [64, 59].

One key disadvantage of the approaches above is that only the information inside each local scanning window is used: joint information between scanning windows or information out of the scanning window are either thrown away or heuristically exploited through

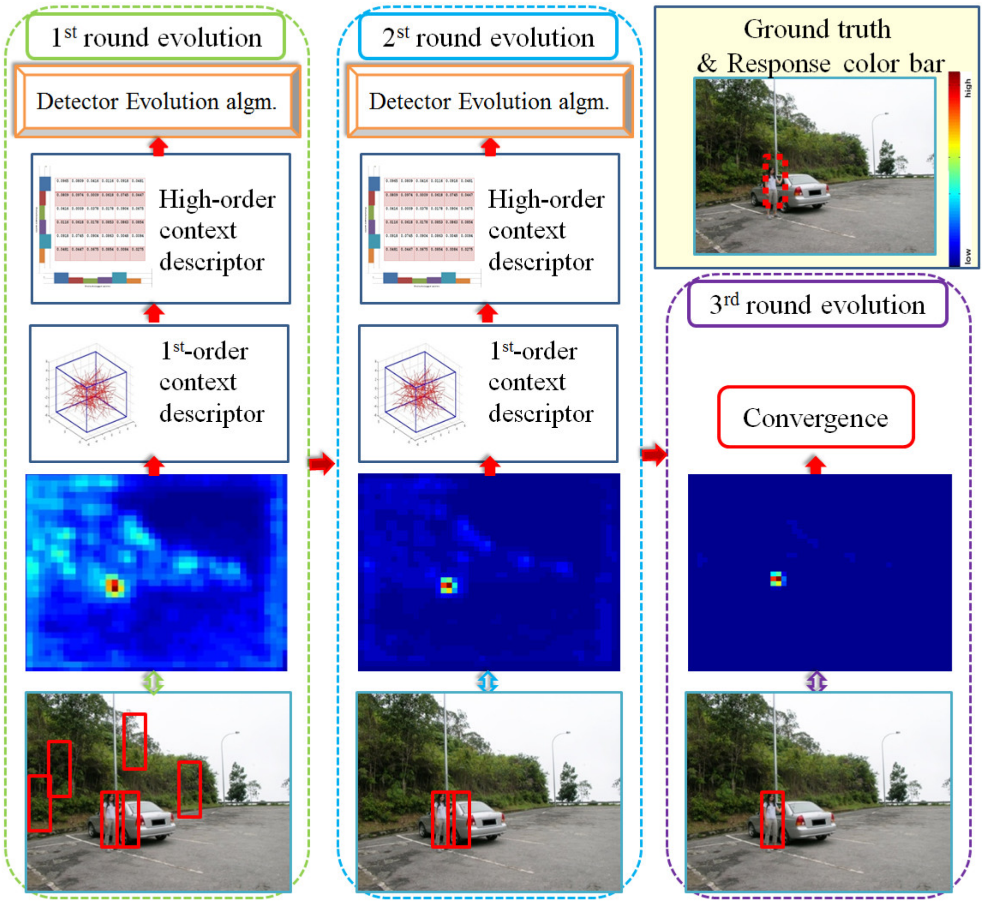

1st round evolution
Detector Evolution algm.
High-order context descriptor
1st-order context descriptor
2st round evolution
Detector Evolution algm.
High-order context descriptor
1st-order context descriptor
Ground truth & Response color bar
high
low
3rd round evolution
Convergence

Figure 4.1: **The proposed MOCO Detection Evolution.** The input image with ground truth label (red dotted rectangle) is shown at top-right corner. The framework evolves the detector using high-order context till the convergence. At each iteration, response map and 0^{th}-order context is computed using the initial baseline detector (for the 1^{st} iteration) or the evolved detector from the prior iteration (for later iterations). Then the 0^{th}-order context is used for computing the 1^{st}-order context, upon which high order co-occurrence descriptors are computed. Finally context in all orders are combined to train a evolving detector. The iteration stops when the overall performance converges. The evolution eliminates many false positives using implicit contextual information, and fortifies the true detections.

post-processing procedures such as non-maximum suppression. Naturally, to improve detection accuracy, context in the neighborhood of each scan window can provide rich information and should be explored. For example, a scanning window in a pathway region is more likely to be a true detection of human than the one inside a water region. In fact, there have been some efforts on utilizing contextual information for object detection and a variety of valuable approaches have been proposed [65, 66, 67]. High level image con-

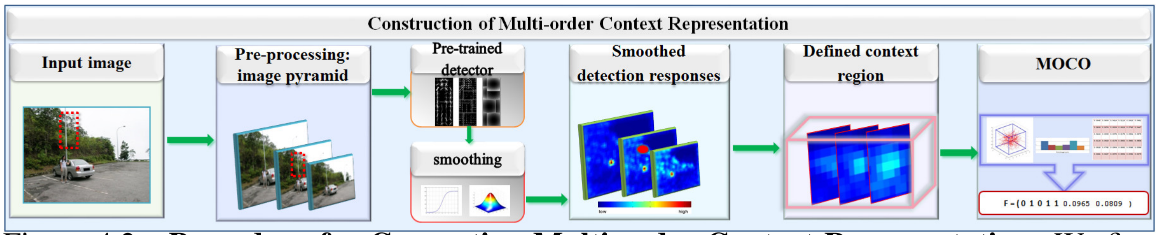

Construction of Multi-order Context Representation
Input image
Pre-processing: image pyramid
Pre-trained detector
smoothing
Smoothed detection responses
Defined context region
MOCO
F=(0 1 0 1 1 0.0965 0.0809)

Figure 4.2: **Procedure for Computing Multi-order Context Representation.** We first build image pyramid and smooth the corresponding detector response map as discussed in Sec. 4.3.1. For each detection candidate (red dotted rectangle), we locate its position (red dotted rectangle) in the image pyramid and its position (red solid area) in the smoothed detection responses map. We define its context structure $\Omega(p)$ (0^{th}-order) as in Sec. 4.3.1. Finally we compute the 1^{st}-order binary comparison based context features, upon which we further extract high order co-occurrence descriptor detailed in Sec. 4.3.3.They are combined as the proposed MOCO descriptors.

.

texts such as semantic context [68], image statistics [66], and 3D geometric context [69], are used as well as low level image contexts, including local pixel context [8] and shape context [70].

Besides utilizing context information from the original image directly, another line of works including Spatial Boost [71], Auto-Context [72], and the extensions elegantly integrate the classifier responses from nearby background pixels to help determine the target pixels of interest. These works have been applied successfully to solve problems such as image segmentation and body pose estimation. Inspired by these prior arts, Contextual Boost [35] was proposed to extract multi-scale contextual cues from the detector response map to boost the detection performance. Contextual information directly from the responses of multiple object detectors has also been explored. In [73, 74, 75] the co-occurrence information among different object categories is extracted to improve the performance in various classification tasks. Such methods require multiple base object classifiers and generally necessitate a fusion classifier to incorporate the co-occurrence information, making them expensive and sensitive to the performance of individual base classifiers.

In this chapter we aim at developing an effective and generic approach to utilize contextual information without resorting to the multiple object detectors. The rationale is that, even though there is only one classifier/detector, higher order contextual information such as the co-occurrence of objects of different categories can still be implicitly and effectively used by carefully organizing the responses from a single object detector. Since only one classifier is available, the co-occurrence of different object types cannot be explicitly encoded as the multi-class approaches. However, the difference among the responses of the single classifier on different object regions implicitly conveys such contextual information. An example is illustrated in Fig.(4.1). The responses of a pedestrian detector to various object regions such as the sky, streets, and trees, may vary greatly, but a homogeneous region of the response map corresponds to a region with semantic similarity. Actually, the initial response map in Fig.(4.1) can lead to a rough tree, sky and street segmentation. This reasoning hints a possibility to encode higher order contextual information with single object detection response. Therefore, if we treat the single classifier response map as an "image", we can extract descriptors to represent high order contextual information.

Our multi-order context representation is inspired by the recent success of randomized binary image descriptors [76, 77, 78]. First we propose a series of binary features where each bit encodes the relationship of classification response values for a pair of pixels. The difference of detector responses at different pixels implicitly captures the contextual co-occurrence patterns pertinent to detection improvements. Recent research also shows that image patches could be more effectively classified with higher-order co-occurrence features [79]. Accordingly we further propose a novel high order contextual descriptor based on the binary pattern of comparisons. Our high order contextual descriptor captures the co-occurrence of binary contextual features based on their statistics in the local neighborhood. The context features at all different orders are complementary to each other and are therefore combined together to form a multi-order context representation.

Finally the proposed multi-order context representations are integrated into an iterative classification framework, where the classifier response map from the previous iteration is further explored to supply more contextual constraints for the current iteration. This process is a straightforward extension of our contextual boost algorithm in [35]. Similar to [35], since the multi-order contextual feature encodes the contextual relationships between neighborhood image regions, through iterations it naturally evolves to cover greater neighborhoods and incorporates more global contextual information into the classification process. As a result our framework effectively enables the detector evolving to be stronger across iterations. We showcase our "detector evolution" framework using the successful deformable part models [33] as our initial baseline detector. Extensive experiments confirm that our framework achieves better accuracy monotonically through iterations. The number of iterations is determined in the training stage when the detection accuracy converges. On the PASCAL VOC 2007 datasets [10], our method outperforms all state-of-the-art approaches, and improves by 3.3% over the deformable part models (ver.5) [33] in mean average precision. On the Caltech dataset [34], compared with the best prior art achieved by contextual boost [35], our method further reduces the log-average miss rate from 48% to 46% and the miss rate at 1 FPPI from 25% to 23%.

4.3 Multi-order Context Representation

Fig.(4.2) summarizes the flow chart for constructing the multi-order context representation from an image. First, the image is densely scanned with sliding windows in a pyramid of different scales. For each location of scan window, image features are extracted and a pre-trained classifier is applied to compute the detection response. The detection response maps for each scale are smoothed as in Sec. 4.3.1. We define the context region in terms of spatial and scale for each candidate location. We then compute a series of binary features using randomized comparison of detector responses within the context region, as detailed in Sec. 4.3.2. Finally, we compute the statistics of the binary comparison features and extract high order co-occurrence descriptors as shown in Sec. 4.3.3. They together construct the proposed Multi-Order Contextual co-Occurrence (MOCO).

4.3.1 Context Basis (0^{th}-order)

Intuitively, the appearance of the original image patch containing the neighborhood of target objects provides important contextual cues. However it is difficult to model this kind of context in original image because the neighborhood around target objects may vary dramatically in different scenarios [61]. A logical approach to this problem is: firstly convolve the original image with a particular filter to reduce the diversity of the neighborhood of a true target object as foreground with various backgrounds; then extract context feature from the filtered image. For object detection tasks, we prefer such a filter to be detector driven. Given the observation from Fig.(4.1) that the positive responses cluster densely around humans but occur sparsely in the background, we simply take the object detector as this specific filter and directly extract context information from the classification response map, denoted as $\mathcal{M}$.

Since the value range of the classification response is $[-\infty, +\infty]$, we first adopt logistic regression to map the value at each pixel s into a grayscale value $s^{'} \in [0, 255]$.

$$s^{'} = \frac{255}{1 + exp(\alpha \cdot s + \beta)}, \tag{4.1}$$

where $\alpha = -1.5$, $\beta = -\frac{\eta}{\alpha}$, and η is the pre-defined classifier threshold. Eq. (4.1) turns the response map into a "standard" image, denoted as $\mathcal{M}^{'}$.

The detection responses are usually noisy. To construct context feature from $\mathcal{M}^{'}$, Gaussian smoothing with kernel size 7*7 and std value 1.5 is performed to reduce noise sensitivity, as shown in Fig(4.1, 4.2). In the smoothed $\mathcal{M}^{'}$, each pixel $\dot{P}$ represents a local scan window in the original image and its intensity value indicates the detection confidence in the window. Such a response image thus conveys context information, which we denote as 0^{th}-order context.

We define a 3D lattice structure centered at $\dot{P}$ in spatial and scale space. We set $\dot{P}$ as the origin of the local 3-dimensional coordinate system, and index each pixel **a** by a 4-dimension vector $[x, y, l, s]$. Here $[x, y]$ refers to the relative location with respect to $\dot{P}$; l represents the relative scale level with respect to $\dot{P}$; s means the value of the pixel **a** in the smoothed response image $\mathcal{M}'$, *e.g.* $[2, 3, 2, 175]$ means the pixel **a** locates in the 2-level higher than $\dot{P}$, $(2, 3)$ in (x, y)-dimensions relative to $\dot{P}$, with pixel value 175. The context structure $\Omega(\dot{P})$ around $\dot{P}$ in the spatial and scale space is defined as:

$$\boldsymbol{\Omega}(\dot{P}; W, H, L) = \left\{ (x, y, l, s) \;\middle|\; \begin{array}{rcl} |x| & \leq & W/2 \\ |y| & \leq & H/2 \\ |l| & \leq & L/2 \end{array} \right\}, \tag{4.2}$$

where (W, H, L) determines the size and shape of $\Omega(\dot{P})$. For example, $(1, 1, 1)$ means the context structure is a $3 \times 3 \times 3$ cubic region.

4.3.2 Binary Pattern of Comparisons (1^{st}-order)

Given the 0^{th}-order context structure, we propose to use comparison based binary features to incorporate the co-occurrence of different objects. Although we only have a single object detector, the response values at different locations indicate the confidences of the target object existing. Therefore, each binary comparison encodes the contextual information of whether one location is more likely to contain the target object than the other.

Comparison of Response Values

Specifically, we define the binary comparison τ in the 0^{th}-order context structure $\Omega(\dot{P})$ of size $W \times \; H \times L$ as:

$$\tau(\mathbf{s};\mathbf{a},\mathbf{b}) := \begin{cases} 1 & if \;\; \mathbf{s}(\mathbf{a}) < \mathbf{s}(\mathbf{b}) \\ 0 & otherwise \end{cases}, \tag{4.3}$$

where $\mathbf{s}(\mathbf{a})$ represents the pixel value in $\Omega(\dot{P})$ at $\mathbf{a} = [\mathbf{x_a}, \mathbf{y_a}, \mathbf{l_a}]$. Naturally selecting a set of n $(\mathbf{a}, \mathbf{b})$-location pairs inside $\Omega(\dot{P})$ uniquely defines a set of binary comparisons. Similar to [77], we define the n-dimensional binary descriptors $\mathbf{f_n} = [\tau_\mathbf{1}, \tau_\mathbf{2}, \ldots, \tau_\mathbf{n}]$ as our 1^{st}-order context descriptor. However, care needs to be taken for selecting the n specific pairs for the descriptor.

Randomized Arrangement

There are numerous options for selecting n pairs of binary comparisons in Eq. (4.3). As shown in Fig.(4.3), two extreme cases of selection are:

(i) The locations of each test pair $(\mathbf{a_i}, \mathbf{b_i})$ are evenly distributed inside $\Omega(\dot{P})$ and binary comparison τ_i can occur far from the origin point: $\mathbf{x_{a_i}}, \mathbf{x_{b_i}} \sim U(-\frac{W}{2}, \frac{W}{2})$,i.i.d
$\mathbf{y_{a_i}}, \mathbf{y_{b_i}} \sim U(-\frac{H}{2}, \frac{H}{2})$,i.i.d; $\mathbf{l_{a_i}}, \mathbf{l_{b_i}} \sim U(-\frac{L}{2}, \frac{L}{2})$,i.i.d;

(ii) The locations of each test pair $(\mathbf{a_i}, \mathbf{b_i})$ concentrate heavily surrounding the origin: $\forall i \in (1, n)$, $\mathbf{a}_i = [0, 0, 0]$, and $\mathbf{b_i}$ lies on any possible position on a coarse 3D polar grid.

Type (i) ignores the facts that the origin of $\Omega(\dot{P})$ represents the location of the detection candidates and thus the context near it might contain more important clues; while type (ii) yields too sparse samples at the boarders of $\Omega(\dot{P})$ to stably capture the complete context information. To address these issues, we adopt a randomized approach:

(iii) $\mathbf{a_i}, \mathbf{b_i} \sim Gaussian(\mu, \Sigma)$, i.i.d. $\mu = [0, 0, 0]$, and $\Sigma = \begin{vmatrix} \epsilon_1 \cdot W^2 & 0 & 0 \\ 0 & \epsilon_2 \cdot H^2 & 0 \\ 0 & 0 & \epsilon_3 \cdot L^2 \end{vmatrix}$. So Σ is correlated with the size of context structure $\Omega(\dot{P})$ and the scaling parameters $[\epsilon_1, \epsilon_2, \epsilon_3]$ are

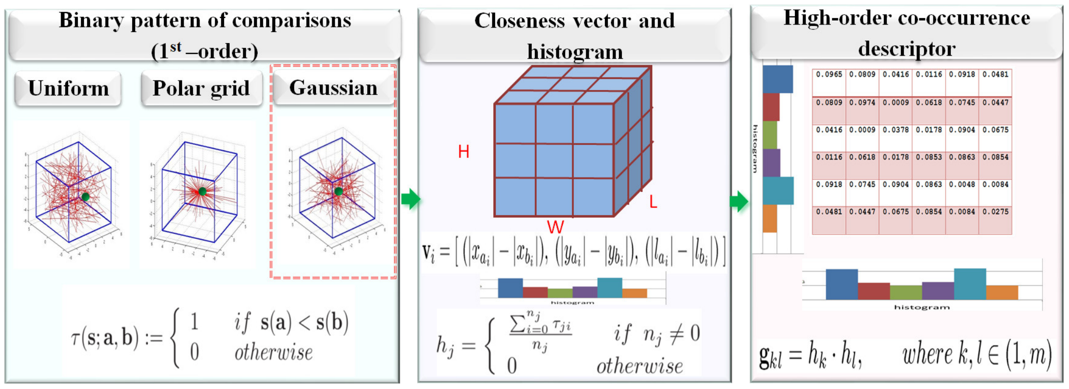

Binary pattern of comparisons (1st –order)
Uniform
Polar grid
Gaussian
$\tau(s;a,b) := \begin{cases} 1 & if\ s(a) < s(b) \\ 0 & otherwise \end{cases}$
Closeness vector and histogram
H
W
L
$v_i = [(|x_{a_i}| - |x_{b_i}|), (|y_{a_i}| - |y_{b_i}|), (|l_{a_i}| - |l_{b_i}|)]$
$h_j = \begin{cases} \frac{\sum_{i=0}^{n_j} \tau_{ji}}{n_j} & if\ n_j \neq 0 \\ 0 & otherwise \end{cases}$
High-order co-occurrence descriptor
histogram
$g_{kl} = h_k \cdot h_l, \quad where\ k,l \in (1,m)$

Figure 4.3: **Multi-order Context Representation.** In the context structure $\Omega(\dot{P})$ with size $W \times H \times L$ around a position $\dot{P}$ (green dot), we first define binary pattern of randomized comparisons (1^{st}-order) based on certain distributions shown on left, described in Sec. 4.3.2 and 4.3.2. We then define the closeness measure $\mathbf{v}_i$ and divide each dimension into t intervals yielding $m = t^3$ subregions (bounded by the solid and dotted red lines), upon which we compute the histogram h_j using Eq. (4.5,4.4) as the high-order co-occurrence descriptor.

set empirically as $[0.15, 0.15, 0.15]$ that give the best detection rate in our experiments.

The randomized binary features compare the 0^{th}-order context in a set of random patterns and provides rich 1^{st}-order context. The patterns of comparisons capture co-occurrence of classification responses within the context structure $\Omega(\dot{P})$. We can then construct the high order context descriptor using the 1^{st}-order context.

4.3.3 High Order Co-occurrence Descriptor

It has been shown that higher-order co-occurrence features help improve classification accuracy [79]. Inspired by it, we exploit higher order context information based on the co-occurrence and statistics of the 1^{st}-order context.

Denote $\mathbf{f}_n = [\tau_1, \tau_2, \ldots, \tau_n]$ the randomized co-occurrence binary features, where τ_i corresponds to a comparison between two pixels $\mathbf{a}_i = [x_{a_i}, y_{a_i}, l_{a_i}]$ and $\mathbf{b}_i = [x_{b_i}, y_{b_i}, l_{b_i}]$. For each pair of pixels $\mathbf{a}_i$ and $\mathbf{b}_i$, we define a closeness vector $\mathbf{v}_i = [\ |x_{a_i}| - |x_{b_i}|,\ |y_{a_i}| - |y_{b_i}|,\ |l_{a_i}| - |l_{b_i}|\ ]$ to measure the absolute difference of the locations of $\mathbf{a}_i$ and $\mathbf{b}_i$ in x-dimension, y-dimension, l-dimension. For example, $|x_{a_i}| - |x_{b_i}| > 0$ implies that in x-

dimension, $\mathbf{a}_i$ is closer to the origin $\dot{P}$ than $\mathbf{b}_i$. Thus $\mathbf{v}_i$ measures whether $\mathbf{a}_i$ or $\mathbf{b}_i$ is closer to $\dot{P}$. This is an important measure as it can be easily observed that stronger detection responses occur in regions closer to the true positive locations. Accordingly the distribution of τ_i w.r.t. $\mathbf{v}_i$ contains important context cues. To compute a stable distribution that is robust against noise, we evenly divide each dimension into t intervals yielding $m = t^3$ subregions, and compute a histogram $\mathbf{h}_m = [h_1, \ldots, h_m]$, as shown in Fig.(4.3).

Specifically, suppose n_j co-occurrence tests fall into the j-th subregion and their values are $\{\tau_{j_1}, \tau_{j_2}, \ldots, \tau_{j_{n_j}}\}$, the corresponding histogram value h_j is calculated as

$$h_j = \begin{cases} \frac{\sum_{i=0}^{n_j} \tau_{j_i}}{n_j} & if \;\; n_j \neq 0 \\ 0 & otherwise \end{cases} \tag{4.4}$$

The high order co-occurence descriptor is then constructed as follows,

$$\mathbf{f}_p = \{g_{kl} \mid g_{kl} = h_k \cdot h_l, \;\; {\scriptstyle (k,l=1,\ldots,m)}\}, \tag{4.5}$$

While the 1^{st}-order co-occurrence features $\mathbf{f}_n$ describes the direct pair-wise relationships between neighborhood positions in a local context, the high order co-occurrence features $\mathbf{f}_p$ capture the correlations among such pair-wise relationships in the local context. Complementarily they provide rich context cues and are combined into the Multi-Order Contextual co-Occurrence (MOCO) descriptor, $\mathbf{f}_c = [\mathbf{f}_n, \mathbf{f}_p]$.

4.4 Detection Evolution

To effectively use the MOCO descriptor for object detection, we propose an iterative framework that allows the detector to evolve and achieve better accuracy. Such a concept of detection "evolution" had been successfully used for pedestrian detection in Contextual Boost [35]. In this chapter, we straightforwardly extend the MOCO based evolution framework to integrate with deformable-part models [59, 33] for general object detection tasks.

4.4.1 Feature Selection

Our detector uses the MOCO descriptor together with the non-context image features extracted in each scan window in the final classification process. The image features can further consist of more than one descriptors that are computed from different perspectives, e.g., the FHOG descriptors for different parts in the deformable-part-model [59, 33]. As a result, the dimension of the combined feature descriptor can be very high, sometimes more than $10,000$ dimensions. Feeding such features to a general classification algorithm can be unnecessarily expensive. Therefore a step of feature selection is employed when constructing the classifiers at each iteration of detection evolution. Many popular feature selection algorithms have been proposed, such as Boosting [80, 40] or Multiple Kernel Learning [14, 81]. Either of them can be used for our purpose. In our experiments boosting [40] is used for feature selection.

4.4.2 General Evolution Algorithm

The iterative process of the detector evolution framework is similar to Contextual Boost [35]. Given an initial baseline detector, the iteration procedure for training a new evolving detector is as follows. First, the baseline detector is used to calculate the response maps. Then, the MOCO as well as the image features are extracted on all the training samples. Bootstrapping is used to iteratively add hard samples to avoid over-fitting. Next, feature selection is applied to select the most meaningful features amongst the MOCO and image features. Finally, the selected features are fed into a general classification algorithm to construct a new detector, which will serve as the new baseline detector for the next iteration. As our MOCO is defined in a context region, the iteration will automatically propagate context cues to larger and larger regions. As a result, more and more context will be incorporated through the iterations, and the evolved detectors can yield better performance. The iteration process stops when the performances of the evolving detectors converge. In the testing stage, the same evolution procedure is applied using the learned detectors respectively.

4.4.3 Integration with Deformable-Part-Model

The deformable-part-model approach [59, 33] has achieved significant success for general object detection tasks. The basic idea is to define a coarse root filter that approximately covers an entire object and higher resolution part filters that cover smaller parts of the object. The relationship between the root and the parts is modeled in a star structure as,

$$s_f = s_r + \sum_{i=1}^{N_p} (s_{p_i} - d_i), \tag{4.6}$$

where s_r is the detection score of the root filter, s_{p_i} and d_i respectively represent the detection score and deformation cost of the i-th part filter, and N_p is the number of part filters. The star-structural constraints and the final detection are achieved using a latent-SVM model.

From the viewpoint of context, the deformable-part-model essentially exploits the intra context inside the object region, e.g., various arrangements of different parts. In contrast, the proposed MOCO deals with the co-occurrence of scanning windows that cover the object region and its neighborhood. Therefore it exploits the inter context around the object region. Clearly these two kinds of context are exclusive and complementary to each other. This encourages us to combine them together to provide more comprehensive contextual constraints.

Note that Eq. (4.6) consists of both the final detection response s_f and the detection responses s_{p_i} from the N_p part filters. Since each response s corresponds to a response map, we calculate the MOCO descriptors using each of the response maps. We follow the same procedure of computing the MOCO descriptors $\mathbf{f}_c$ for the root filter from s_f, to obtain the MOCO descriptors $\mathbf{f}'_{c_i}$ for parts on s_{p_i} . Furthermore, to effectively evolve the baseline deformable-part-model detector using the calculated MOCO, we apply the iterative framework not only on the root filter but also on part filters and detectors for every component. The detailed training procedure for integrating our MOCO and the deformable-part-model is summarized in Algm. (6). The input to the algorithm includes the training dataset S_{train} and the deformable-part-model Ψ_0 as the initial baseline detector. In each iteration, we first adopt the same iteration process as in Sec. 4.4.2 for part filters and the model for each component, and evolve the component model accordingly for the next iteration. This step is shown as step 2 in Algm. (6). Then we use the latent-SVM to fuse the N_c components and retrain an evolved detector for the next iteration. Bootstrapping is again used to avoid over-fitting. The iteration process stops when we observe that the detection accuracy rate converges.

Algorithm 6: Detection Evolution

```
**Input**: Pre-trained deformable-part-model $\Psi_0$ with $N_c$ components, each containing $N_p$ part
    filters; training data set $\mathbf{S}_{train}$; detection accuracy rate (*e.g.* average precision) $\delta_0$ of
    $\Psi_0$ on $\mathbf{S}_{train}$; convergence threshold $\xi$.
**Output**: Iteratively evolved detectors $\Psi_1, \ldots, \Psi_{N_d}$
Set $R = 0$
**Do**
    1. $R = R + 1$, $N_d = R$.
    2. **for** $i = 1 \rightarrow N_c$ **do**
        1). Extract the image feature $\mathbf{f}_I$ according to the $i_{th}$ component of $\Psi_{(R-1)}$ on $\mathbf{S}_{train}$.
        2). Compute the detector response maps on $S_{train}$ using $\Psi_{(R-1)}$.
        3). For each detection candidate $\dot{P}$, compute the $1^{st}$-order and high-order context
        descriptors on $\Omega(\dot{P})$ according to Eq. (4.3, 4.4, 4.5) for each of the $N_p$ part filter
        responses, resulting multiple MOCOs as $[\mathbf{f}_c, \mathbf{f}'_{c_1}, \ldots, \mathbf{f}'_{c_{N_p}}]$
        4). Do feature selection using Boosting [40] on $[\mathbf{f}_I, \mathbf{f}_c, \mathbf{f}'_{c_1}, \ldots, \mathbf{f}'_{c_{N_p}}]$, to learn the
        informative features $\mathbf{f}_{L_i}$ for the $i_{th}$ component.
        5). Bootstrap and retrain the evolved detector for the $i_{th}$ component.
    3. Bootstrap and retrain the evolved detector $\Psi_R$ via latent-SVM [59, 33] for fusing the
    responses from the $N_c$ evolved component detectors.
    4. Evaluate the detection rate $\delta_R$ on $\mathbf{S}_{train}$ using $\Psi_R$.
**While** $\delta_R - \delta_{(R-1)} > \xi$;
```

4.5 Experiments and Discussion

We have conducted extensive experiments to evaluate the proposed MOCO and the detection evolution framework. To demonstrate the advantage of our approach, we adopt the challenging PASCAL VOC 2007 dataset [10] with 20 categories of objects, which are widely acknowledged as one of the most difficult benchmark datasets for general object detection. We use the deformable-part-model [33] with default setting (3 components, each with 1 root and 8 part filters) as our initial baseline detector. First, to demonstrate the advantage of the MOCO, we compare the performance achieved by using different orders of context information. We show performances with various parameter settings to demonstrate the characteristics of the MOCO. Second, we compare the performance at different iterations as the detector evolves to show that the detectors quickly converge in about 2$\sim$3 iterations. Third, we compare the performance of our method with those of state-of-the-art approaches and show substantial improvement. Furthermore, we also experiment on Caltech pedestrian dataset [34], which was used as the main evaluation benchmark for Contextual Boost [35]. The comparisons demonstrate the advantages of our approach.

4.5.1 Multi-order Context Representation

We first evaluate the MOCO representation and experiment with different parameters settings. We use 5 categories (*plane, bottle, bus, person, tv*) from PASCAL VOC 2007 and experiment on "train" and "val" set for various parameters. All experiments in this section only run 1-iteration of detection evolution. We compare the mean Average Precisions (mAP) to show how the performance varies with different parameter settings.

Context Parameters. Two important parameters that directly affect the computation of context descriptors are the size of Ω_p and the number n of binary comparisons. Since the binary comparisons $\{\tau_1, \tau_2, \ldots, \tau_n\}$ are randomly sampled inside the 3D context structure $\Omega(\dot{P})$, the comparison number n is chosen proportional to the size of $\Omega(\dot{P})$, $W \times H \times L$.

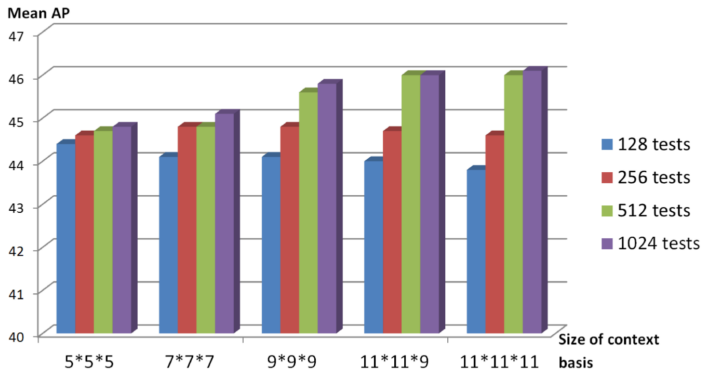


Figure 4.4: Mean AP (mAP) Varies for Different Parameters: the size $W \times H \times L$ of context structure $\Omega(\dot{P})$ and the number n of binary comparison tests. Only 1^{st}-order context feature and the image features is used for evaluation.

As shown in Fig.(4.4), bigger size of $\Omega(\dot{P})$ and number n correspond to richer context information and thus yield better performance, yet requiring more computation. To balance the performance and computational cost, we finally choose $11 \times 11 \times 9$ as $\Omega(\dot{P})$ size, and 512 as the binary comparison test number, where the scale factor is $2^{0.1}$ as in [59] and 9 scales up is about 2 times.

1^{st}**-order Context**. According to the analysis in Sec. 4.3.2, we choose type iii of Gaussian sampling for constructing the 1^{st}-order context descriptor. We compared the detection performances using different Gaussian parameters. As shown in Fig.(4.5), the best accuracy is achieved when the variances in the three dimensions are $[0.15, 0.15, 0.15]$ respectively. Fig.(4.5) also shows the comparison with the sampling methods of type i and type ii, which confirms the advantage of Gaussian sampling.

High Order Context. The most important parameter for computing high order context descriptor is the dimension m of the histogram. Since the high order context descriptor $\mathbf{f}_p$ is complementary to the 1^{st}-order context feature $\mathbf{f}_n$, they are combined when evaluating the detection performance. Table.(4.1) shows the detection accuracy when choosing different values of m, where the best accuracy is achieved when the closeness vector space is divided into $m = 27 (= 3^3)$ subregions.

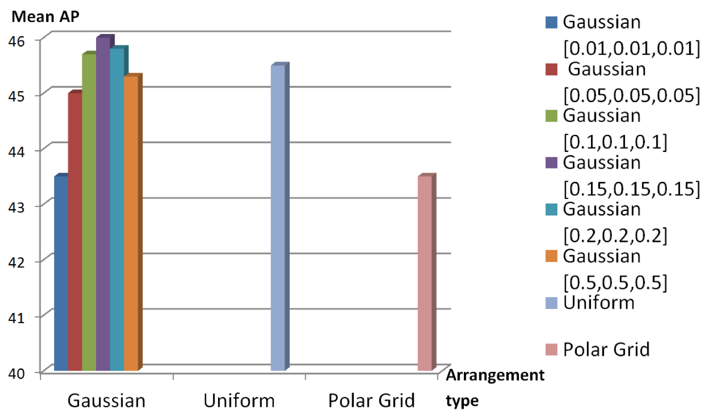


Figure 4.5: Mean AP (mAP) Varies for Different Arrangements. Only 1^{st}-order context features and the image features is used for evaluation.

$m = 0$	$m = 8$	$m = 27$	$m = 64$	$m = 125$
46.0	46.3	**46.7**	46.5	46.1

Table 4.1: Mean AP (mAP) varies with respect to the length of high-order co-occurrence feature $\mathbf{f}_p$. The high order context descriptor together with 1^{st}-order context feature and the image features are used. $m = 0$ refers to not using any high order feature.

Context in Different Orders. To show that different orders of context provide complimentary constraints for object detection, we compared the detection accuracy using different combinations of the multi-order context descriptors. For 0^{th}-order context, we chose the best parameter settings presented in [35]. As shown in Table.(4.2), clearly the MOCO descriptor that combines all orders of context achieves the best detection performance. This confirms that none of the multi-order contexts is redundant. Another way of exploring the 1^{st}-order context is to extract the gradient-based features such as SURF [1] or LBP [2] directly on each scale of the context structure $\Omega(\dot{P})$. However it does not help improve the accuracy in our experiments, as shown in Table.(4.2). This means that the context across larger spatial neighborhood or different scales can be more effective than the context con-

0^{th}	1^{st}	$1^{st} + H$	$0^{th} + 1^{st}$	$0^{th} + 1^{st} + H$	SURF	LBP
45.5	46.0	46.7	46.8	**47.2**	44.7	45

Table 4.2: Mean AP (mAP) varies with the combination of different order context feature, where $0^{th}, 1^{st}, H$ respectively refers to 0^{th}, 1^{st} and high order descriptors. We also compared with SURF [1] or LBP [2] extracted on each level of context structure $\Omega(\dot{P})$.

0	1	2	3(converged)	4	5	6
35.4	37.6	38.3	38.7	38.8	38.7	38.7

Table 4.3: Mean AP (mAP) varies with respect to the proposed detection evolution algorithm, where 0-iteration in the left refers to the baseline without detection evolution.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
Leo [11]	29.4	55.8	9.4	14.3	28.6	44.0	51.3	21.3	20.0	19.3	25.2
CMO [61]	31.5	61.8	12.4	18.1	27.7	51.5	59.8	24.8	**23.7**	27.2	30.7
Det-Cls [60]	38.6	58.7	**18.0**	18.7	31.8	53.6	56.0	**30.6**	23.5	31.1	**36.6**
Oxford [14]	37.6	47.8	15.3	15.3	21.9	50.7	50.6	30.0	17.3	**33.0**	22.5
NLPR [17]	36.7	59.8	11.8	17.5	26.3	49.8	58.2	24.0	22.9	27.0	24.3
Ver.5 [33]	36.6	62.2	12.1	17.6	28.7	54.6	60.4	25.5	21.1	25.6	26.6
Our method	**41.0**	**64.3**	15.1	**19.5**	**33.0**	**57.9**	**63.2**	27.8	23.2	28.2	29.1

	dog	horse	motor	person	plant	sheep	sofa	train	tv		**mAP**
Leo [11]	12.5	50.4	38.4	36.6	15.1	19.7	25.1	36.8	39.3		29.6
CMO [61]	13.7	60.5	51.1	43.6	14.2	19.6	38.5	49.1	44.3		35.2
Det-Cls [60]	20.9	62.6	47.9	41.2	**18.8**	23.5	41.8	**53.6**	45.3		37.7
Oxford [14]	**21.5**	51.2	45.5	23.3	12.4	23.9	28.5	45.3	48.5		32.1
NLPR [17]	15.2	58.2	49.2	44.6	13.5	21.4	34.9	47.5	42.3		34.3
Ver.5 [33]	14.6	60.9	50.7	44.7	14.3	21.5	38.2	49.3	43.6		35.4
Our method	16.9	**63.7**	**53.8**	**47.1**	18.3	**28.1**	**42.2**	53.1	**49.3**		**38.7**

Table 4.4: Comparison with the state-of-the-art performance of object detection on PASCAL VOC 2007 (trainval/test).

veyed by local gradients between adjacent positions.

4.5.2 Detector Evolution

Using the best parameters for the MOCO descriptor obtained using the "train" and "val" datasets, we evaluate the detector evolution process across iterations. The entire PASCAL dataset is used as the testbed, e.g., training on "trainval" and testing on "test" [10]. We run Algm. (6) and compare the detection accuracy through iterations. For most categories, our framework converges at the second or third iteration. To better show the trend of the detector evolution process, we keep it running for 6 iterations. As shown in Table.(4.3), the accuracy is steadily improved through iterations and converges quickly.

4.5.3 Comparison with State of Art

Finally, we compare the overall performance of our approach with the state of art.

PASCAL VOC 2007. We first compare our method with state-of-the-art approaches on PASCAL dataset [10]. As shown in Table.(4.4), our algorithm stably outperforms the baselines [33] in all 20 categories. Especially on the categories of sheep, tv, and monitor, the algorithm achieves significant AP improvements by $6.6\%, 5.7\%$. When compared with all prior arts, our approach outperforms 12 out of 20 categories, and achieves the highest mean AP (mAP) at 38.7, outperforming the deformable model (ver.5) [33] by 3.3%.

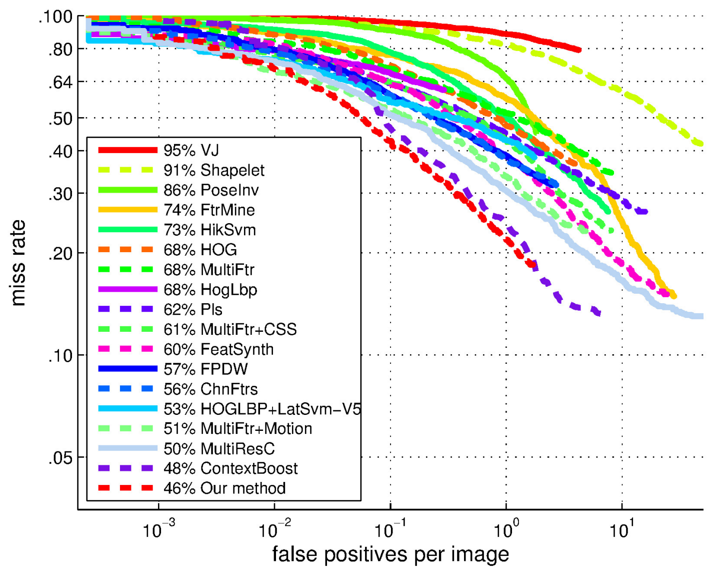


Figure 4.6: The comparison between our algorithm and the state of the arts in Caltech Pedestrian test dataset.

Caltech Pedestrian Dataset. We also experiment our algorithm on Caltech pedestrian dataset [34]. We follow the same experimental setup as [35, 34] for evaluations. We use LBP [2] to capture the texture information and FHOG [59] to describe the shape information, and only consider "reasonable" pedestrians of 50 pixels or taller with no occlusion or part occlusion [35, 34]. We compare our algorithm with the state-of-the-art results surveyed in [34], as shown in Fig.(4.6): the best reported log-average miss rate is 48% [35], while our algorithm further lowers the miss rate to 46%. If we consider the miss rate at 1

FPPI, the best reported result is 25% [35], and our algorithm achieves 23%.

4.5.4 Processing Speed

Our detection evolution framework needs to evaluate each test image N_d times, where N_d is the number of evolved detectors. The experiments show that it generally converges after 2 or 3 iterations and thus the computational cost would be around 2 or 3 times of the deformable part models (ver.5) [33]. On PASCAL dataset [10], for a 500×375 images, it takes about 12 seconds. One way to speed up the detection is to adopt the cascade scheme. In that case most negative candidates can be rejected in early cascades, and the detection could be around 10 times faster [82].

4.6 Conclusion

In this chapter we have proposed a novel multi-order context representation that effectively exploits co-occurrence contexts of different objects, denoted as MOCO, even though we only use detectors for a single object. We preprocess the detector response map and extract the 1^{st}-order context features based on randomized binary comparison and further develop a high order co-occurrence descriptor based on the 1^{st}-order context. Together they form our MOCO descriptor and are integrated into a "detection evolution" framework as a straightforward extension of Contextual Boost [35]. Furthermore, we have proposed to combine our multi-order context representation with the recently proposed deformable part models [33] to supply a comprehensive coverage over both inter-contexts among objects and inner-context inside the target object region. The advantages of our approach are confirmed by extensive experiments. As the future work, we plan to further extend our MOCO to temporal context from videos and contexts from multiple object detectors or multi-class problems.

Chapter 5

Object Detection with Depth Sensor: Depth Sensor Based Object Detection Using Normal Vector Pattern

5.1 Overview

In previous chapters, we propose algorithms and features for object detection with a common RGB camera. While as we know, the depth sensors have many advantages over RGB cameras such as easy foreground segmentation, insensitivity to illumination variation and texture/background clutter. So in this chapter, we investigate object detection with the depth sensor. We propose an effective feature, *Normal Vector Pattern (NVP)* , designed exclusively for object detection with a depth sensor. The trade-off that a feature can achieve between the discriminative power and the invariance is essential for the classification performance: a superior feature should be discriminative enough to detect inter-class variations, whereas the feature invariance minimizes the disturbance caused by the intra-class variations. This feature trade-off depends on both the sensor type and the application. For the depth sensor based object detection task, the proposed NVP is a successful attempt due to its two components: the cross products of neighboring normal vectors induce the discriminative power and the inner products of the normal vectors lead to the view-angle invariance. The superiority of the proposed NVP is evident: with a simple linear SVM model, the proposed framework outperforms the state-of-the-art features based on more complicated models such as the deformable parts model on standard Washington RGB-D [6] and NYU [36] benchmark datasets.

5.2 Introduction

The improvements or the breakthroughs in hardware inevitably impact the successive algorithms, which may revolutionize the existing applications or generate new applications. The quick growth in gaming industry leads to the fast development of GPU, which facilitates the high performance parallel computing based on GPU and fosters the large scale learning on big data [83]; the proliferation of personal mobile devices such as smart-phones and Google glasses catalyzes the research in egocentric vision [84, 85]; the recently emerging low-cost depth sensors, *i.e.*the Kinect depth sensor [86, 87], have fostered researchers to revisit unsolved vision problems such as object recognition/detection [6, 88, 89, 90, 91, 92, 93, 94], motion capture [88], and action recognition [5, 95, 96].

Depth sensors have many advantages over RGB cameras such as easy foreground segmentation and insensitivity to illumination variation and texture/background clutter. For the challenging object detection task, many image features [97, 98, 99] are proposed, presumably taking the RGB or gray level images as the input. A depth image is essentially different from RGB or gray level images, because the intensities of the depth image represent the distances between the depth sensor and object surface points. If we treat a depth image shown in Figure 5.1 as a gray level intensity image and simply apply off-the-shelf vision algorithms originally designed for intensity images such as SIFT [97], HOG [98], and LBP [99], we may only achieve suboptimal results, as noticed in [5, 4]. We therefore propose an effective feature, *Normal Vector Pattern* (NVP), designed exclusively for object

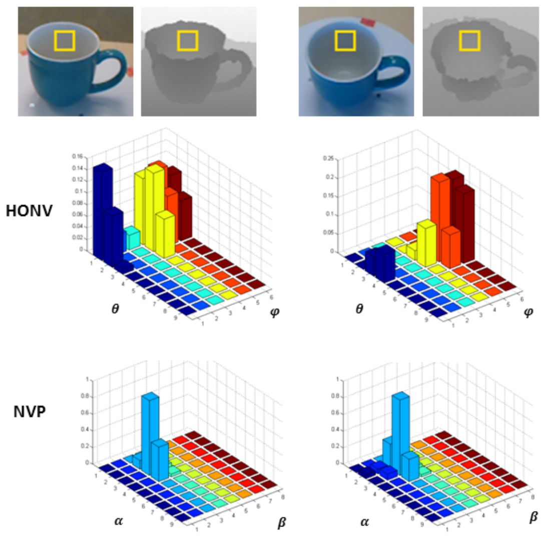


Figure 5.1: **The proposed Normal Vector Pattern (NVP) feature is robust to the viewing angle variation, one of major causes of intra-class variations.** **Top :** Two pairs of RGB and depth images for the same coffee mug. Each pair corresponds to a view angle. The yellow rectangles locate the exact same region of the coffee mug from which we extract the depth feature. **Middle:** the current state-of-the-art depth descriptor—Histogram of Oriented Normal Vector(HONV) [4, 5] on the two depth images. **Bottom:** one histogram of our NVP feature on the two depth images (NVP contains certain number of histogram, each behaves with the view-angle invariant characteristics). The proposed NVP adopts the 2D histogram quantization as HONV, but bases on fundamentally different angles, explained in Section 5.4. Thus as we could see from the figure, the NVP histograms for the two view angles are quite similar. Especially the domain bins almost keep the same. In contrast, the HONV histograms are totally different in these two view angles. So clearly NVP has the advantage of view-angle invariance. With this metric, NVP outperforms the prior art HONV [4] by 12% in Mean Average Precision on standard Washington RGB-D dataset [6].

detection with a depth sensor. The proposed NVP, as illustrated in Figure 5.1 and Figure 5.2, captures the geometric characteristics of an object and tolerates the intra-class variation caused by the viewing angle change. We make the NVP achieve a trade-off between the discriminative power and invariance: In contrast to the state-of-the-art HONV feature, the proposed NVP is quite robust to the viewing angle variation as show in Figure 5.1.

Meanwhile, the proposed NVP is discriminative enough to detect inter-class variations. For example, as illustrated in Figure 5.2, for a concave surface and a convex surface with same surface norm distribution, the HONV based representations are identical whereas the the difference between the corresponding NVP representations are quite obvious: Telling the difference between a baseball hat and a bowl is very difficult for HONV feature but quite easy for NVP feature.

The success of the proposed NVP is mainly due to its two components to be detained in Section 5.4.2: The cross products of neighboring normal vectors induce the discriminative power and the inner products of the normal vectors leads to the view-angle invariance. The superiority of the proposed NVP is evident: with a simple linear SVM model, the proposed framework outperform the state of the arts feature based on more complicated models such as the deformable parts model on Washington RGBD [6] and NYU [36] benchmark datasets.

Our contributions are three folds: 1) For pan-tilt-roll viewing angle changes, we give a thorough analysis on the variations of the inner product and the cross product between the neighboring normal vectors. 2) Based on this analysis, we propose an very effective feature, NVP, achieving a performance superior to the state of the arts. 3) A very efficient implementation of the NVP is given, making the NVP ready for the realtime demo.

5.3 Related Work

Researches have been working on images with depth information or range images for decades [100, 101, 102, 103]. However, the emergence of cost-effective depth sensors, especially the Kinect sensor [86, 87], has catalyzed the research in this direction [89, 90, 91, 92, 93]. Some of these approaches either adapt the existing techniques to the depth image, or combine the RGB and depth information to improve the recognition accuracy, whereas others propose new techniques designed specifically for the depth image. Bo *et*

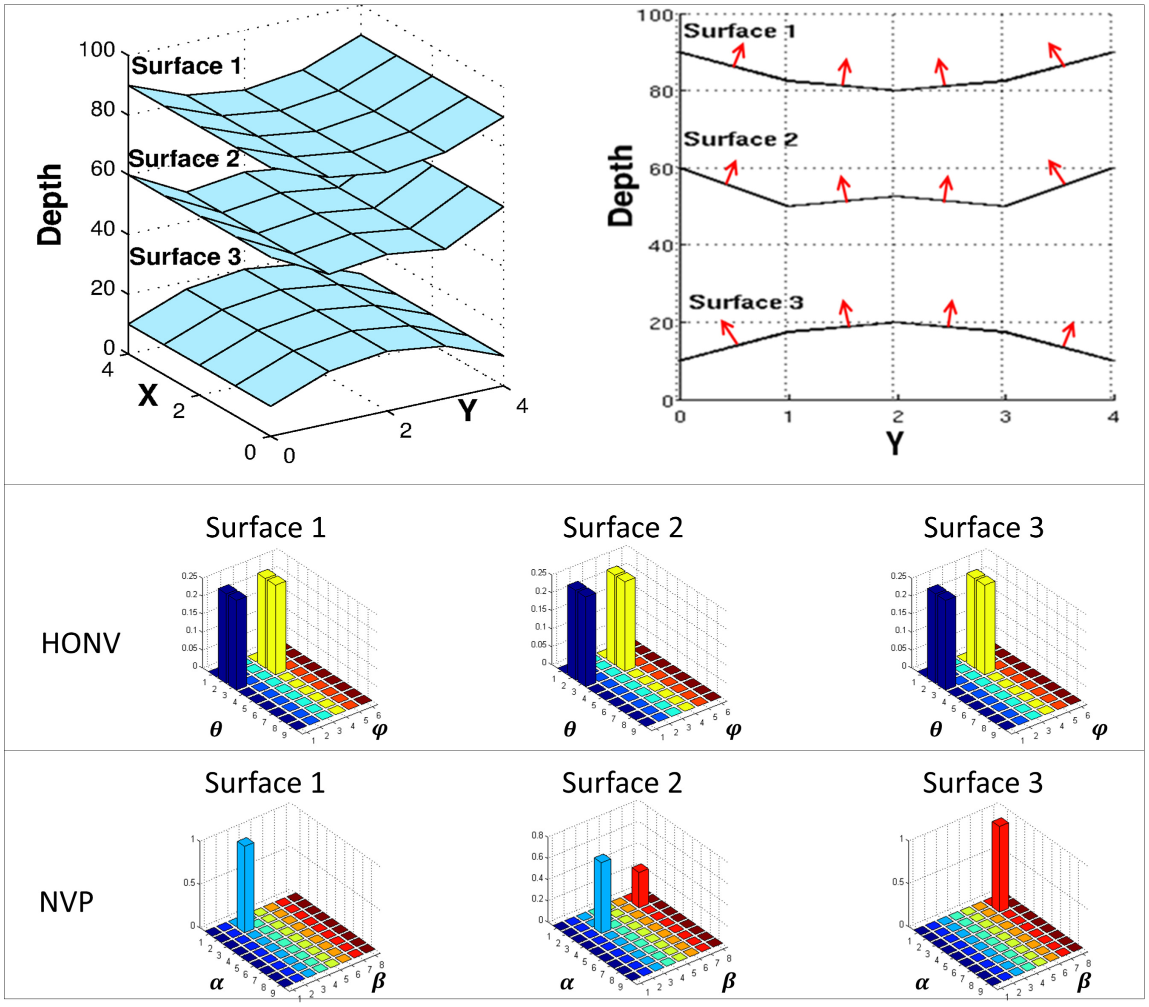


Figure 5.2: **Example showing the discriminative characteristics of NVP.** For better illustration, in this example, we assume that the three different surfaces are all perpendicular to the "Y-D" plane. **Top:** The left image shows the three surfaces in the "XYD" 3D space. While the right image gives out the three surfaces from the viewpoint that is perpendicular to "Y-D" plane. The red arrows here show the orientations of surface normal vectors. **Middle:** The 2D histogram of surface normal HONV [4] on the 3 surfaces. In [4], the normal vector is represented by its two angles, azimuthal angle φ and zenith angleθ. **Bottom:** The 2D histograms of the proposed NVP on the three surfaces. The 2D histogram of NVP is determined by two different angles α and β, explained in Section 5.4.2 & Section 5.4.3. The HONV [4] counts the normal vectors in the surface independently and discards their location information. Therefore it cannot distinguish these three surfaces. In contrast, the proposed NVP not only captures how large the surface varies but also describes the shape of the surface,e.g., concave, convex, etc. Therefore it is able to differentiate the three surfaces.

al. [91] develop a set of kernel features on depth images that model size, 3D shape, and depth edges in a unified framework, which has been demonstrated to have much better performance than spin images [104]. The Relational Depth Similarity Features (RDSF) [89] calculates features derived from a similarity of depth histograms that represents the relationship between two local regions using the Bhattacharyya distance [105]. Lai *et al.* [93] tried to address the joint object category and instance recognition problem in the context of RGB-D (depth) cameras using instance distance learning. Jamie *et al.* [88] proposed a real-time pose recognition method using the Kinect. Cai *et al.* [106] proposed a regularized maximum likelihood deformable model algorithm for 3D face tracking. There is also some work on 3D modeling in indoor environments [107, 108, 109] which takes advantage of the depth information . Xia *et al.* [90] proposed a model based approach which detects humans using a 2-D head contour model and a 3-D head surface model.

A large-scale 3D object dataset, the RGB-D dataset, was recently announced as a benchmark dataset [6] for object recognition algorithms using depth information. The state-of-the-art algorithms such as Depth Kernel Descriptors (DKD) [91], Hierarchical Kernel Descriptors (HKDES) [92] and Instance Distance Learning (IDL) [93], are all evaluated and compared on this benchmark dataset.

Tangent feature has been used for object recognition. Yu and Zhang [110] extended the Local Coordinate Coding (LCC) [111] algorithm by exploring the local tangent directions. In contrast to these approaches, which represents the data manifold through local PCA, the HONV [4] feature represent an object in 3D space as a distribution of the normal vector orientation. Xu and Xu [112] proposes an object description and recognition approach based on the relationship between the arc length and tangent orientation of object contours.

5.4 Normal Vector Pattern (NVP)

The robustness of surface normal vector has been extensively verified in many computer vision tasks, e.g., object detection [4], action recognition [5]. However the previous works treat each surface normal vector independently and consequently perform histogram quantizations over a lattice structure in the region of interests. We notice that a local surface could be represented more effectively by using the relationship between neighboring normal vectors. In this section, we present the proposed 3D depth image descriptor—Normal Vector Pattern (NVP), and show its superiority over direct histogram of surface normal vectors [4].

5.4.1 The surface normal

We represent each pixel p of in a depth image I as a three-dimensional vector $[x, y, d]$, which encodes the X-Y spatial information in the image plane and the distance between the optical center and the reflecting point. Therefore, we can model the depth image I as a function $\mathbb{R}^2 \to \mathbb{R}^1 : d = f(x, y)$, which in the 3D space constitutes a surface: $S(x, y, d) = d - f(x, y) = 0$. For any point (x_0, y_0, d_0), on the surface, (*i.e.*$S(x_0, y_0, d_0) = 0$), we can linearize the function $S(x, y, d)$ locally, with the first order Taylor expansion,

$$S(x,y,d) \doteq S(x_0,y_0,d_0) + \begin{pmatrix} x - x_0 \\ y - y_0 \\ d - d_0 \end{pmatrix}^T \cdot \bigtriangledown S. \tag{5.1}$$

The corresponding tangent plane at the point (x_0, y_0, d_0) is:

$$S(x_0,y_0,d_0) + \begin{pmatrix} x - x_0 \\ y - y_0 \\ d - d_0 \end{pmatrix}^T \cdot \bigtriangledown S = 0. \tag{5.2}$$

Therefore, the surface normal vector at (x_0, y_0, d_0) can be computed as:

$$\mathbf{n} = \bigtriangledown S = (-\frac{\partial d}{\partial x}, -\frac{\partial d}{\partial y}, 1)^T. \tag{5.3}$$

The magnitude of the surface normal of an object is directly related to the physical size of the object and the scale under which the object is observed, which is irrelevant to object categorization or detection. Therefore we normalize the normal vector $\mathbf{n}$ to a unit length normal $\tilde{\mathbf{n}}$:

$$\tilde{\mathbf{n}} = (-\frac{1}{\|\mathbf{n}\|_2} \cdot \frac{\partial d}{\partial x}, \quad -\frac{1}{\|\mathbf{n}\|_2} \cdot \frac{\partial d}{\partial y}, \quad \frac{1}{\|\mathbf{n}\|_2})^T \tag{5.4}$$

After the normalization, the normal vector $\tilde{\mathbf{n}}$ inherently preserves richer information than the 2D gradient orientation $(-\frac{\partial d}{\partial x}, -\frac{\partial d}{\partial y})^T$. This is because, the third dimension of the unit normal $\tilde{\mathbf{n}}$ encodes the magnitude of the gradient $\frac{1}{\|\mathbf{n}\|_2}$, and experimentally this extra dimension enhances the discriminative characteristics of normal vector descriptor [4, 5].

5.4.2 Patterns of neighboring normal vectors represented as inner and cross products

After computing the surface normal vector $\tilde{\mathbf{n}}$ as in Equation 5.4, the previous works [4, 5] simply quantized the normal vector space into specific bins, and accordingly constructed the histogram-based feature.

However several problems exist in these methods: 1) surface normal vector records the absolute orientation information(e.g., zenith angle and azimuth angle of the normal vector) of every pixel in the depth image. Consequently it would be sensitive to the view angle change, as shown in Figure 5.1. 2) the quantization method of surface normal in each local region(e.g., 8×8 pixel image patch [4]) treats the normal vectors independently, and then represents the distribution of normal vectors as a histogram [4]. This procedure ignores to model relative relationship information between each pair of normal vectors. While, in fact the shape of each local surface should be determined by the angles between the neighboring normal vectors. To better illustrate this, considering the three typical surfaces as shown in Figure 5.2, where surface 1 and surface 3 are symmetric to the plane “X-Y” in the “XYD” space. Surface 2 could be represented as a linear combination of surface 1 and surface 3.

Thus the naive histogram quantization of normal vectors [4] for the three surfaces cannot differentiate their types of surface shapes, e.g., concave, convex or sawtooth.

To address this issue, we propose a novel normal vector pattern descriptor, which leverages on the strength of neighboring normal vectors to describe the surface shape robustly. For a normalized normal vector $\tilde{\mathbf{n}}_\mathbf{1}$ computed on local surface $\mathcal{S}_1$, we denote a neighboring/nearby local surface as $\mathcal{S}_2$ and its normal vector as $\tilde{\mathbf{n}}_\mathbf{2}$. The relative relationship between this pair of normal vectors represented as their inner product and cross product describes the surface shape variation from $\mathcal{S}_1$ to $\mathcal{S}_2$, shown in Figure 5.3.

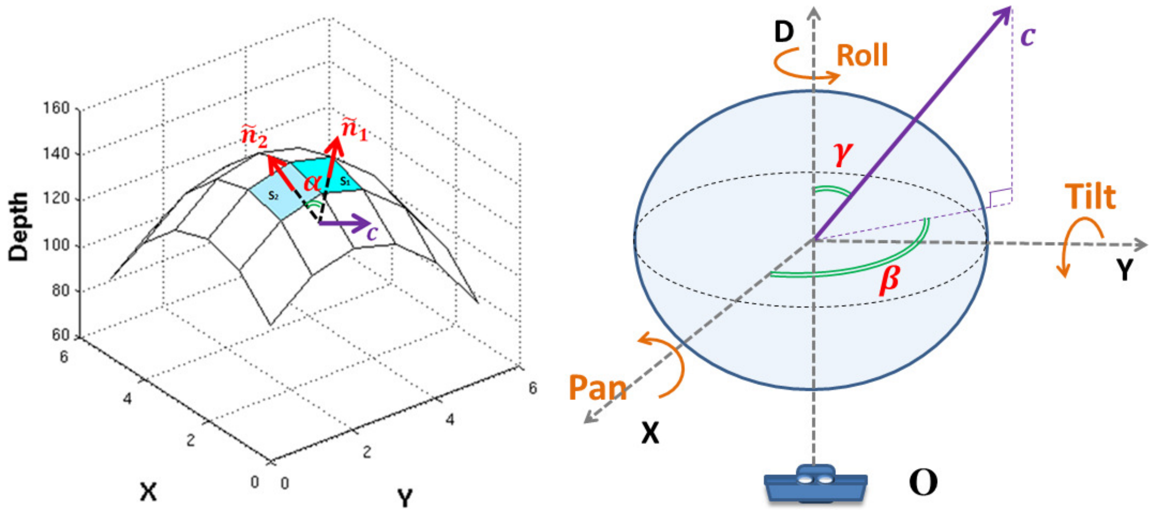


Figure 5.3: **The inner product and the cross product of neighboring normal vectors.** Suppose the local surface are $\mathcal{S}_1$ and $\mathcal{S}_2$. The angle α is the angle between the two neighboring normal vectors $\tilde{\mathbf{n}}_\mathbf{1}$ and $\tilde{\mathbf{n}}_\mathbf{2}$, shown in the left image. While $\mathbf{c}$ represents the cross product vector of $\tilde{\mathbf{n}}_\mathbf{1}$ and $\tilde{\mathbf{n}}_\mathbf{2}$.The right image shows the cross product in the "XYD" coordinate system. The Kinect sensor locates at the view position $\mathbf{O}$, and the "X-Y" plane is always parallel to the sensor plane. The azimuthal angel and zenith angle of $\mathbf{c}$ are denoted by β and γ. Finally we use α, β to define NVP. The yellow arrows in the right image represent the three types of rotation: Pan, Tilt, Roll. Based on the three kinds of rotation, we did the analysis why we keep β instead of γ. The details are explained in Section 5.4.2.

We first compute the inner product of $\tilde{\mathbf{n}}_\mathbf{1}$ and $\tilde{\mathbf{n}}_\mathbf{2}$ to represent how large the surface varies across this pair of local surfaces. The inner product is then converted into the angle $\alpha \in [0, \pi]$:

$$\alpha = cos^{-1}\left(\tilde{\mathbf{n}}_\mathbf{1} \cdot \tilde{\mathbf{n}}_\mathbf{2}\right), \tag{5.5}$$

*i.e.*the angle between neighboring normal vectors. The metric of adopting angle α is quite straightforward: this angle is totally determined by the object shape, not sensitive to the view angle change. However there is also a disadvantage: This view-invariant description only tells how large the local surfaces varies, while it can not distinguish the local surface type, as shown in Figure 5.4. In another word, α is not discriminative enough to describe the shape of local surface. This shortcoming is also proved by experiments, detailed in Section 5.5.3.

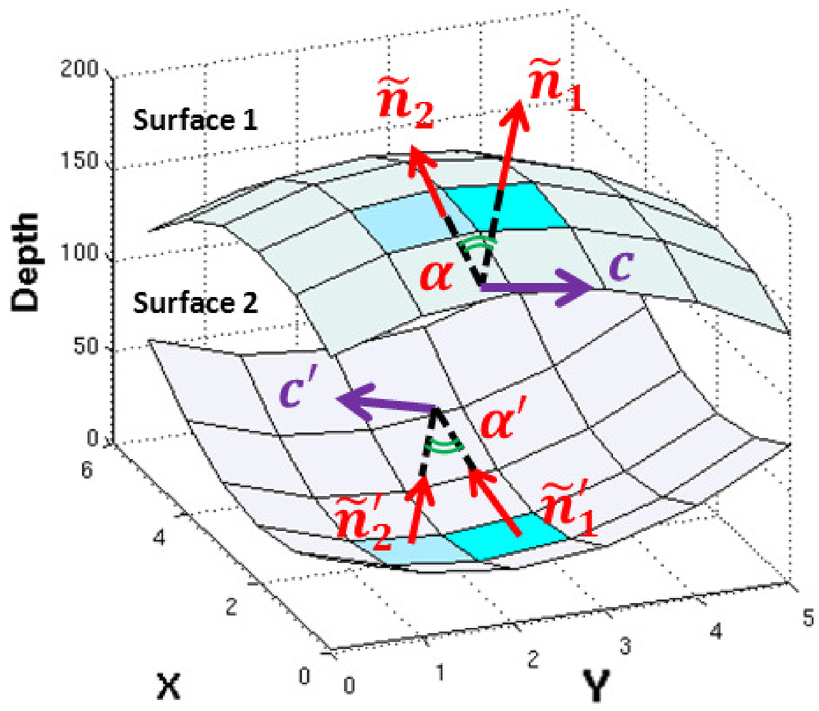


Figure 5.4: **Examples showing the importance of angle β.** The two surfaces are symmetric to the "X-Y" plane. Thus the relative angles α and $\alpha^{'}$ are the same. As a result, you can not distinguish the "concave" surface 1 and "convex" surface 2. Fortunately, with the help of the cross product vector **C** and **C**$'$ (or more specifically, the different β angle as shown in Figure 5.3), we could easily differentiate these two surfaces, explained in Section 5.4.2.

To handle this problem and give a comprehensive representation of local surface, We also compute the cross product between this pair of normal vectors. The magnitude of cross product,which equals to $sin\left(\alpha\right)$, is redundant since it's already captured by the angle between neighboring normal vectors α in Equation 5.5. All we care is the orientation of the cross product, which contains abundant information to describe the surface type(e.g., concave, convex), shown in Figure 5.4. We denote the cross product vector as $\mathbf{C} = [C_x, C_y, C_d]^T$, where

$$\begin{aligned}
&\mathbf{C} = \tilde{\mathbf{n}}_\mathbf{1} \times \tilde{\mathbf{n}}_\mathbf{2}, \\
&\beta = tan^{-1}\left(\frac{C_y}{C_x}\right), \\
&\gamma = \tan^{-1}\left(\frac{\left({C_x}^2 + {C_y}^2\right)^{\frac{1}{2}}}{C_d}\right)
\end{aligned} \tag{5.6}$$

The angles β and γ determine the orientation of the cross product of a pair of neighboring normal vectors. More specifically, we name the angle γ the Zenith Angle of the Cross product (ZAC) and the angle β the Azimuthal Angle of the Cross product (AAC). Here we only keep the angle β and discard the angle γ. This is because in practice when the Kinect captures a depth image, the D axis of the XYD coordinate system is always perpendicular to the Kinect sensor while as a result "X-Y" plane is parallel to the sensor plane. When viewing the $\mathcal{S}_1$ and $\mathcal{S}_2$ with different viewpoints, The angle β with a range $[0, 2\pi)$ is not sensitive to the view angle change variation, as shown in Figure 5.5. But γ does not own this character. Also, the value of angle β could clearly differentiate a surface is concave or convex, etc. Therefore, combining α and β, we are already fully able to describe the surface variation from $\mathcal{S}_1$ to $\mathcal{S}_2$: α tells us how large the variation is, while β distinguishes the surface type, such as convex or concave. Figure 5.2 demonstrates this advantage. In the meanwhile these two angles are neither sensitive to the viewpoint variation, as examples shown in Figure 5.1, overcoming the weakness of naive surface normal vector feature, e.g., HONV [4].

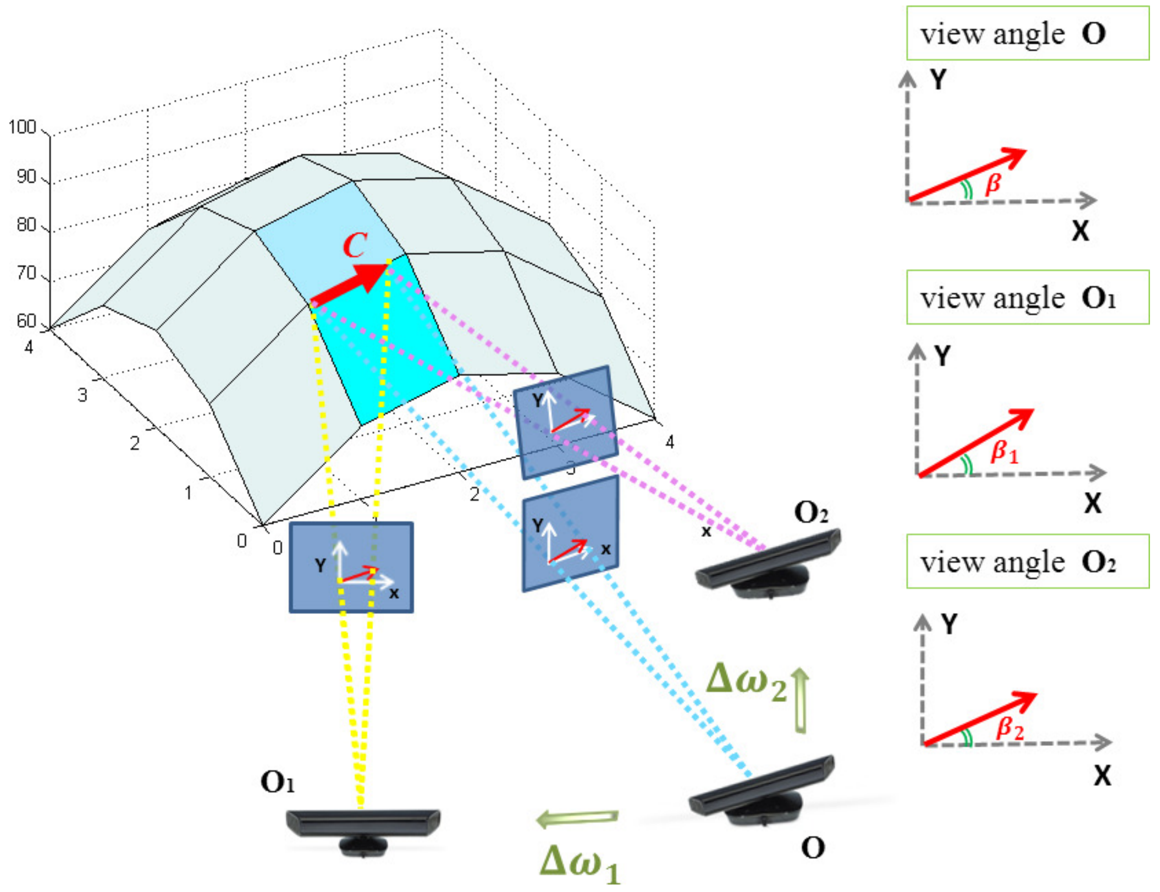


Figure 5.5: **Examples showing the β angle is not sensitive to the view angle change.** In the left image, the cross product vector of the two local surfaces is denoted as red arrow **C** on the 3D surface. Supposing the original view angle of the Kinect sensor is **O**, we could move the sensor horizontally to the position $\mathbf{O_1}$ with view angle change $\triangle\omega_1$, or we could also move the sensor vertically to the position $\mathbf{O_2}$ with view angle change $\triangle\omega_2$. No matter how the sensor rotates, the Depth axis of our "XYD" coordinate system is always perpendicular to sensor while "X-Y" plane is parallel to the sensor plane. According to the three view angles **O**, $\mathbf{O_1}$, $\mathbf{O_2}$, we could compute the angle β, β_1, and β_2 respectively. In the right column, we draw the three angles. We observe that $||\beta_1 - \beta|| \leq || \triangle \omega_1||$, $||\beta_2 - \beta|| \leq || \triangle \omega_2||$. This means the β is not sensitive to the view angle change.

5.4.3 Efficient normal vector pattern (NVP)

After defining the angles α and β, we need to find a way to transform them into descriptors. In the depth image of an object, each pixel p_i corresponds to a local surface $\mathcal{S}_i$. Thus for each pair of pixels$\{p_i, p_j\}$, we could compute the angles $\{\alpha_{ij}, \beta_{ij}\}$ to describe the surface variation from local surface $\mathcal{S}_i$ to local surface $\mathcal{S}_j$. Supposing the depth image has size

$M \times N$, we could get as many as $(M \times N)^2$ relative relationships. The huge number of relationships contains too much redundant and useless information, which inspires us to propose an efficient depth feature. We name it Normal Vector Pattern (NVP). This representation is desired to balance the discriminative characteristics and view-angle invariance. In the meanwhile, it should be efficient enough for the object detection task. Also note that, $\alpha_{ij} = \alpha_{ji}$, but $\beta_{ij} = \pi + \beta_{ji}$. This means the relative position has large impact to the feature representation. Therefore, we need to define NVP in a way that counts the different types of relative location independently.

For a pixel p_i of the depth image, we define a surrounding region centered at p_i. We sample K pixels on the region boundary $\{p_{i_k} | k = 1, 2, \ldots, K\}$. Each pair of $\{p_i, p_{i_k}\}$ corresponds to one type of relative relationship pattern R_k, e.g., pixel p_{i_k} is on left-top of the center pixel p_i. Therefore, according to each R_k, we could get the two-dimensional angle vector $[\alpha_{ii_k}, \beta_{ii_k}]$. In another word, each pixel p_i would have K pairs of two-dimensional angle vectors. Now for each relationship pattern R_k, we could perform the quantization separately based on angle vectors that belong to this pattern R_k.Note that, we only pick up the k pixels on the boundary, because we want the pixels $\{p_i, p_{i_k}\}$ have some distance to each other. This behavior will preserve more context information and resist noise better.

As illustrated in Figure 5.6, following successful cell-structured histogram features such as HOG [98] and LBP [113], firstly we divide the detection window into m by n cells, which could be non-overlapped or overlapped with each other. In each cell, every relationship pattern R_k corresponds to a 2D histogram h_k. Then inside the same cell, all the angle vectors belonging to relationship pattern R_k are voted into the 2D histogram h_k. Assuming I bins and J bins are used for α and β respectively, We will generate a $I \times J$ dimensional feature vector for each relative relationship for each cell. To avoid boundary effects, the angle values are softly voted into multiple neighborhood bins using bilinear interpolation. The final feature representation of the detection window is obtained by concatenating the histogram-based feature from each cell, and each cell has feature length $K \times I \times J$.

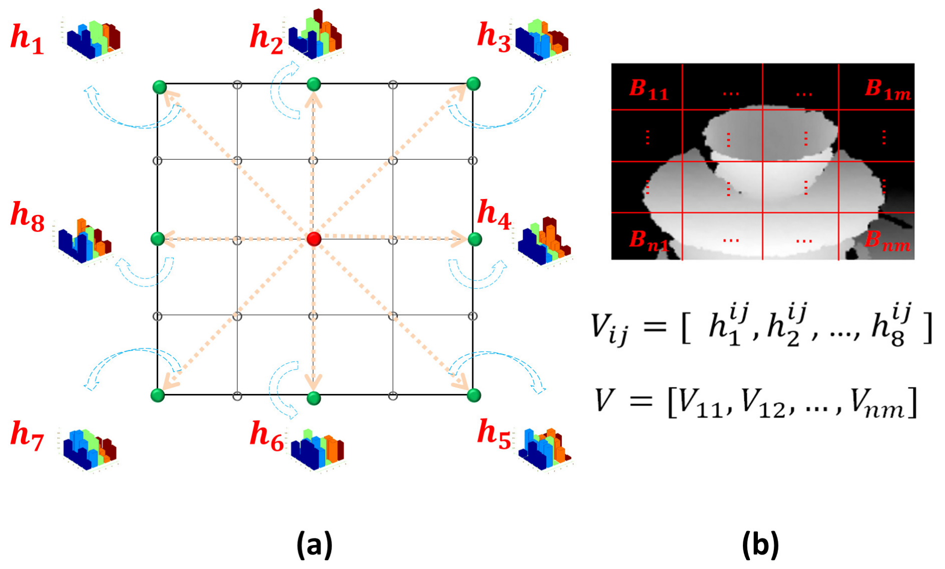

h_1
h_2
h_3
h_8
h_4
h_7
h_6
h_5
B_{11}
...
...
B_{1m}
B_{n1}
B_{nm}
$V_{ij} = [\ h_1^{ij}, h_2^{ij}, \ldots, h_8^{ij}\ ]$
$V = [V_{11}, V_{12}, \ldots, V_{nm}]$
(a)
(b)

Figure 5.6: **Computation of NVP descriptor.** (a) extraction of NVP descriptor for each depth image pixel. For each pixel p (red point), we define a surrounding region and sample K pixels $\{p_1, p_2, \ldots, p_K\}$ (denoted by green points) on the boundary. Here K equals 8. Then for each pixel pair $\{p, p_k\}$, we compute the angle vector $[\alpha_k, \beta_k]$. Finally assign it into 2D histogram h_k according to which relationship pattern R_k it belongs to. (b) extraction of NVP for the whole detection window: split window into cells, get the K histograms for each cell $V_{ij} = [h_1^{ij}, h_2^{ij}, \ldots, h_K^{ij}]$, and finally concatenate all the histogram-based feature $V = [V_{11}, V_{12}, \ldots, V_{nm}]$.

5.5 Experiments

We have conducted extensive experiments to evaluate the proposed NVP. We first introduce the experiment setup in Section 5.5.1, followed with experimental results and analysis. We show both qualitative and quantitative results that confirm the advantages of the proposed NVP.

5.5.1 Experiment setup

Dataset We evaluate NVP on two challenging RGB-D object datasets: Washington RGBD(WRGBD) dataset [6] and the NYU-Depth(NYU) dataset [36]. Among the existing 3D datasets [6, 36, 114], B3DO dataset [114] focuses on office scenes and is very valuable to experiment the context learning algorithm [115] and RGB/depth information joint-learning algorithms [36, 116]. However only relying on depth information, the B3DO dataset is not a good choice. Since many categories appears too small and similar in the depth images. The literature [114] has demonstrated that using only depth image descriptor would perform very bad, achieving less than 5% average precision (Figure 6 in [114]). In contrast, WRGBD [6] dataset contains 250,0000 RGB and depth image pairs, in which both RGB and depth images are captured from three different positions for each instance. It is quite suitable to evaluate the characters of the depth descriptor, such as toleration of the view angle variation, discriminative capacity to different types of surface shape. So we treat WRGBD as our main testbed. Similarly, NYU dataset [36] contains some foreground categories in multi-viewpoints that are good for feature evaluation. We also report the performance on it.

Baselines we compared with several popular descriptors in RGB and depth images, including the HOG [98], HOGD [6], HONV [4], HONV+HOG [4].Also we compared with current state-of-the-art performance achieved by [116] on WRGBD dataset. Besides these baselines, we build up a baseline named "Inner Product" by ourselves: we only keep the angle between neighboring normal vectors α and throw away the AAC β angle. The rest steps are exactly the same as NVP. We evaluate this baseline to show the importance to keep the angle β.

Parameter setting. Each detection window is divided into 12×12 pixel cells, and every cell takes 50% overlapping with surrounding cells. For efficient detector learning, Linear SVM is employed. Besides these setting, there exists several important parameters in NVP: the size of surrounding region defined for each depth image pixel; number of the relative relationship pattern K; The bin numbers of the 2D histogram. For these parameters, we have evaluated them thoroughly on the WRGBD dataset, detailed in Section 5.5.2.

side length	$d = 8$	$d = 12$	$d = 16$
AP	62.8	**67.0**	53.9

Table 5.1: AP(%) varies with respect to the side length d of surrounding region

5.5.2 Parameter study

We first experiment with various different parameter settings of NVP on *soda can* category of the WRGBD dataset. We compare the average precision(AP) to show how the performance varies with different parameters.

Surrounding region size For each pixel p_i, we define a squared surrounding region with side length d and compute the angle vector between p and a boundary pixel p_{i_k}. d here determines how far the local surfaces we are interested in their relationship. If d is too small, the two surface normals would be too similar and easily affected by noise. In the other side, a too large d will lead the two surface normal vectors irrelative and their relationship meaningless. The effect of d is shown in Table 5.1. Experiments shows that $d = 12$ is the best choice.

Number of relative relationship pattern After fixing the surrounding region size, the following important parameter is the relative relationship pattern number K. Supposing the side length of surrounding region is d, thus there are totally d^2 possible relative relationship patterns. We only sample K patterns out of the d^2 candidates instead of using all of them. Because every relationship pattern R_k corresponds to a 2D histogram h_k for each cell in the detection window. Thus large K will result in a very high-dimension descriptor, which is a huge burden for computation. Here we evaluate the effect of the number K to the NVP, the results are shown in Table 5.2. We observe that $K = 16$ achieves similar performance as $K = 8$, while it would double the feature length. So finally we choose $K = 8$.

Bin number of 2D histogram As discussed in Section 5.4.3, for each relative relationship R_k, we build up a 2D histogram of α and β in each cell. So the bin number of the 2D histogram will affect the performance substantially. We denote I and J as bin numbers for angle α and angle β respectively. Too few bins would reduce the discriminative ability of

RR number	$K = 4$	$K = 8$	$K = 16$
AP	63.5	**67.0**	**67.1**

Table 5.2: AP(%) varies with respect to the number of relative relationship pattern (RR number) K.

(I,J)	(5,4)	(5,16)	(10,8)	(10,16)	(20,8)	(20,16)
AP	58.1	59.3	61.4	64.1	**67.0**	66.4

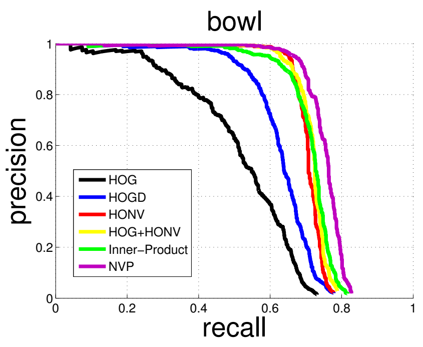
Table 5.3: AP(%) varies with respect to 2D histogram bin number (I, J) for angles α and β.

NVP while too many bins would make NVP too sensitive. Here we test these parameters, shown in Table 5.3. Finally we adopt the best bin numbers: $(I, J) = (20, 8)$.

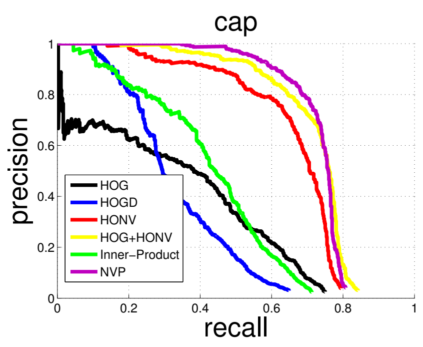
5.5.3 WRGBD dataset

The testing set for the detection task in WRGBD dataset contains 8 clips of video in 8 different scenes, primarily capturing 6 object categories: bowl, cap, cereal box, coffee mug, flashlight and soda can. On this dataset, we compare our NVP with different state-of-the-art methods, including [4, 116]. The precision-recall curves are presented in Figure 5.7. It clearly demonstrates that NVP works best and outperforms other features in five of the six categories by big margins. Note that in these five categories, even combining the state-of-the-art depth feature HONV [4] and HOG on the RGB image together still suffers large gap from our NVP . In the category of "soda can", our NVP performs slightly worse than HOG on color image, but still much better than other depth features. For the reason that feature on RGB image performs better on "sode can" category, previous literature [4] already concluded: it's mainly because the soda can is very small and the Kinect depth sensor could not produce sufficient resolution to extract discriminative depth features.

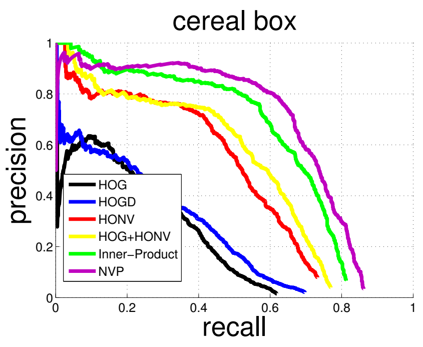
In addition, Figure 5.7 tell that the Inner Product descriptor already works well: except the "cap" category, it stably outperforms the state-of-the-art HONV descriptor. When we investigated the unsuccessful "cap" category, we observed that many instances belonging to



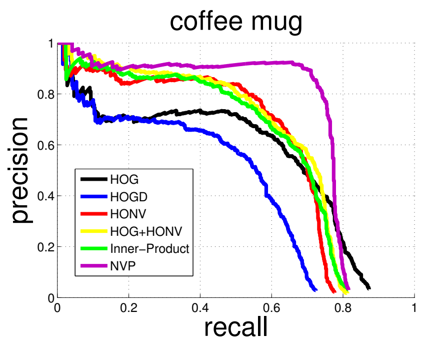
(a) bowl



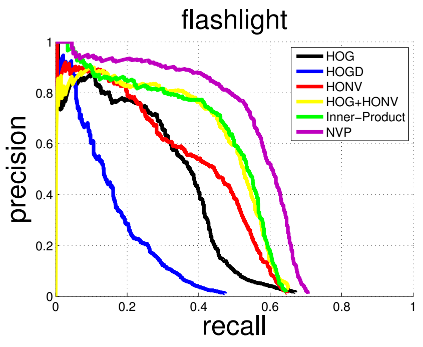
(b) cap



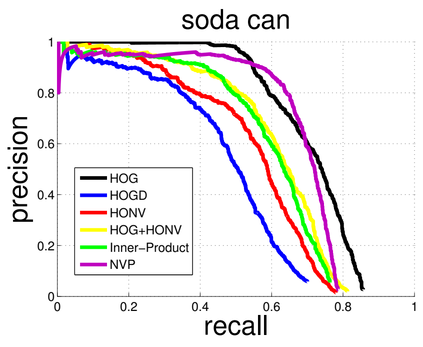
(c) cereal box



(d) coffee mug



(e) flashlight



(f) soda can

Figure 5.7: Precision-recall curves for object detection on the WRGBD dataset.

"bowl" category acted as false alarms even we added many "bowl" images as negative data on purpose during detector learning. This experimentally proves that the Inner Product can not distinguish some different types of surface shape. Fortunately our NVP feature incorporates the AAC β angle information besides the angle α between neighboring normal vectors, and finally overcomes this disadvantage. Therefore comparing with Inner Product feature, NVP gains great performance improvement in all categories.

For quantized comparison we present the average precision obtained in Table 5.4.

FEATURE	HOG	HOGD	HONV	HH	IP	NVP
bowl	51.6	63.1	70.6	71.6	71.3	**75.1**
cap	33.3	33.1	65.3	71.5	42.2	**72.6**
cereal box	21.4	24.3	45.8	50.0	61.3	**66.4**
coffee mug	54.1	43.1	59.3	61.8	59.9	**71.0**
flashlight	32.1	15.3	37.7	44.4	45.8	**54.8**
soda can	**71.0**	46.6	54.3	60.6	58.9	67.0
AVERAGE	43.9	37.6	55.5	60.0	56.6	**67.2**

Table 5.4: Summarization of AP (%) obtained with the six feature configurations on the WRGBD dataset. (HH means the combination of the HOG and HONV feature; HOGD means HOG on depth image; IP means Inner Product.) The best performance is shown in bold with underline. Except the category of *soda can*, the best performances are all achieved by our proposed NVP.

Clearly the NVP outperforms others a lot. Comparing with the state-of-the-art depth descriptor HONV [4], our depth feature NVP significantly outperforms it by 12% mean average precision(mAP) in the WRGBD dataset. Also, We beat the HONV+HOG [4] by 7.2% mAP. Considering the additional RGB information which is complement of depth information is augmented in HONV+HOG setup, it would be easier to observe the good performance of NVP. In addition, the state-of-the-art algorithm in [116] also reported performance of 4 categories in WRGBD dataset: coffee mug, cap, soda can and flash light. With the powerful DPM framework [117], it boosted the mAP of the four categories to 62.1%. Comparing with it, we achieved 66.4% mAP on this four categories, outperforming it by 4.3%. Note that we only adopt the simple linear SVM to learn the detectors.

5.5.4 NYU dataset

The NYU Depth dataset [36] contains a total of 1449 labeled images and we randomly split the set into 800 images for training, the rest for test. Since NYU dataset [36] is originally for scene analysis and provides pixel-level labels, some instances are heavily occluded and

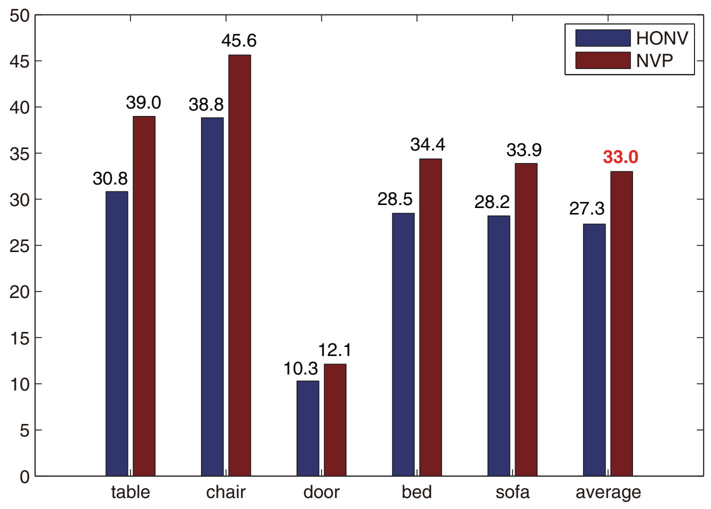


Figure 5.8: Summarization of AP (%) obtained on the NYU dataset.

in low resolution. We refine some labels and ignore the instances occluded more than 50%[1]. We evaluate in 5 foreground categories: bed, sofa, table, chair and door. Comparing with state-of-the-art depth descriptor HONV [4], our NVP improves the mAP from 27.3% to 33.0%, outperforming HONV by 5.7% mAP. The results are summarized in Figure 5.8.

5.6 Conclusion

In this chapter we have proposed a novel depth feature, Normal Vector Pattern (NVP), with application for object detection with a depth sensor. NVP feature has achieved a good balance between discriminative power and view-angle invariance. With a simple linear SVM, our NVP already outperforms the state of the art features based on more complicated models such as Deformable parts model. Extensive experiments on WRGB [6] and NYU depth datasets [36] demonstrate the effectiveness of the proposed NVP feature. Our efficient representation of NVP also making itself possible for the real-time demo in the future.

[1]Prior art [115] did an excellent precise labeling(visible region and context region both labeled) for NYU dataset [36], but unfortunately their label is not available currently, thus we cannot experiment on its configuration

Chapter 6

Fine-grained Object Recognition by Local Feature Embedding and Template Selection

6.1 Overview

Unlike the previous chapters that study the general object detection problems, the following chapter will focus on a specific task in computer vision: fine-grained object recognition. Currently fine-grained recognition has drawn lots of attentions due to its valuable practical usage. Fine-grained recognition has great potential to improve the performance of object detection. For example, if we would like to train a sedan car detector, we need to distinguish all sedan cars from SUVs. This requires us have a good fine-grained recognition algorithm in advance. So in this chapter, we first introduce a scalable and effective algorithm to address a special large-scale fine-grained recognition problem—visual font recognition(VFR) problem. VFR problem aims at automatic identification of the typeface, weight, and slope of the text in an image or photo without any knowledge of the content. Although visual font recognition has many practical applications, it has largely been neglected by the vision community. To address the VFR problem, we construct a large-scale dataset containing

2,420 font classes, which easily exceeds the scale of most image categorization datasets in computer vision. As font recognition is inherently dynamic and open-ended, *i.e.*, new classes and data for existing categories are constantly added to the database over time, we propose a scalable solution [37] based on the nearest class mean classifier (NCM). The core algorithm is built on local feature embedding, local feature metric learning and max-margin template selection, which is naturally amenable to NCM and thus to such open-ended classification problems. The new algorithm can generalize to new classes and new data at little added cost. Extensive experiments demonstrate that our approach is very effective on our synthetic test images, and achieves promising results on real world test images. Then we propose a more scalable hierarchical algorithm for fine-grained object recognition, which shows advantage in both accuracy and speed.

6.2 Introduction

In real world, many object categories appear similar and correlated to each other. A good object detector should be able to distinguish them effectively, otherwise, there would be too many false alarms and accordingly the detection system would crash. To do a thorough research on fine-grained object categorization, we use the font recognition as our testbed due to their high similarity to each other and large-scale charachteristic.Typography is a core design element of any printed or displayed text; graphic designers are keenly interested in fonts, both in their own works, as well as in those of others. Consequently, they frequently encounter the problem of identifying "fonts in the wild." For example, a designer might spot a particularly interesting font on a restaurant menu and would like to identify it for later use. Currently, a designer's best recourse is to take a photo of the text and then seek out an expert to identify the font, such as on an online typography forum[1].

With hundreds of thousands of possible fonts to choose from, it is extremely tedious

[1]E.g. myfonts.com or www.flickr.com/groups/type

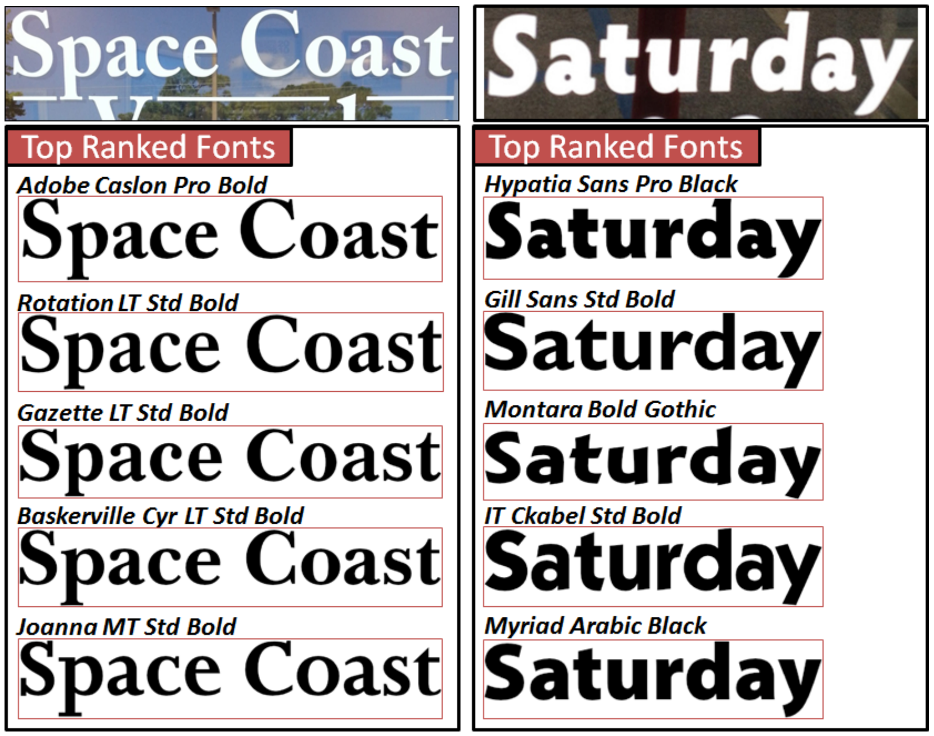


Figure 6.1: Visual font recognition on two real-world test images. The algorithm correctly classifies both (top of the list) and returns four other similar typefaces, despite the high level clutter and noise.

and error-prone, even for font experts, to identify a particular font from an image manually. Effective automatic font identification could therefore greatly ease this problem, and could also facilitate font organization and selection during the design process. To address this need, we propose the application of computer vision methods to automatically identify the typeface, weight and slope of the text in a photo or graphic image. We dub this problem Visual Font Recognition (VFR), in contrast to the well-studied problem of Optical Character Recognition (OCR). (That said, we note that the two problems are coupled in that accurate VFR has the potential to greatly improve OCR accuracy, and vice versa.)

Remarkably, the computer vision research community has largely neglected the VFR problem. The few previous approaches [118, 119, 120, 121, 122, 123, 124] are mostly from the document analysis standpoint, focusing on small number of font classes on scanned document images that typically have high quality, in terms of resolution and contrast, and low geometric distortion. Consequently, tasks such as binarization, noise reduction, char-

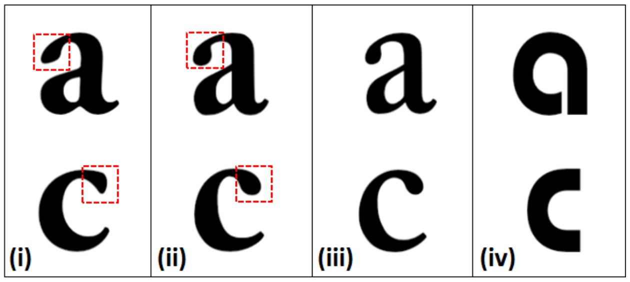


Figure 6.2: Example images of character a and c in different fonts: (i) Adobe Garamond Pro Bold, (ii) Adobe Calson Pro Bold, (iii) Adobe Calson Pro SemiBold, and (iv) BauhausStd-Demi.

acter segmentation, connected component analysis, geometric alignment, and OCR can be robustly applied. Building on these image processing steps, global texture analysis [119, 122], high-order statistical features [124] and typographical features [125, 121, 118] have been exploited for font recognition. However, such scanned image-based techniques are much less applicable to our VFR problem, which needs to be effective even on very short strings or single words from noisy web images and photos taken with mobile devices. In particular, photos in the wild typically suffer from noise, blur, perspective distortions, as well as complex interactions of content with background.

Even though there has been great progress in OCR from photos [126], OCR is generally designed for standard body text fonts and is not very effective for unusual fonts which are often the ones designers wish to identify. In our own experiments with state-of-the-art publicly available OCR engine [127] on our VFR-2420 dataset, which contains 2420 font classes each with 1000 clean English word images, whole words were correctly recognized only 60% of the time. Furthermore, words from more than 13% of our font classes were correctly recognized less than 30% of the time. The poor performance of OCR on wider range of fonts will adversely affect character-based font recognition algorithms. Therefore, current systems such as whatfontis.com, rely on humans in the loop, involving tedious user interactions, to solve the character segmentation and recognition problem for font recognition on real photos.

Even with human intervention, these systems are still far from robust enough for practical use. Equipped with recent advances in machine learning and image categorization [128, 129, 130, 131, 132, 133], we propose to develop an automatic algorithm that does not rely on character segmentation or OCR, and therefore, is applicable to the range of inputs needed for VFR. However, compared with previously studied image categorization and fine-grained recognition problems, visual font recognition has the following additional challenges:

- VFR is an extremely large-scale recognition problem in terms of number of classes. For instance, myfonts.com alone claims that they have more than 100,000 fonts in their collection, which is much larger than most image categorization problems the vision community has thus far addressed. This is dramatically different from previous font recognition works, where, limited to the scope of scanned documents, they have only investigated the problem on a scale of tens of fonts.
- VFR is inherently dynamic and open-ended in that new font classes need to be continually added to the system. It is therefore critical that the algorithm is able to adapt to new classes with low added cost, and to scale sub-linearly with the number of classes.
- VFR is a combination of super fine-grained recognition and coarse-grained image categorization, and it is character-dependent. For example, consider Fig. 6.2, where fonts (i) and (ii) only differ slightly at the letter endings (in red rectangles), (ii) and (iii) only differ on font weight, while all three fonts (i), (ii) and (iii) are visually distinct from font (iv). However, in all cases, without knowing the characters, it is very hard to find a feature space where character a is closer to c from the same font than to character a from another font.

In this chapter, we develop a scalable data-driven approach to VFR that does not depend on character segmentation or OCR to address the above challenges. Therefore, our algorithm does not need to know the context of the word when recognizing the font. Specifically, our contributions are as follows:

1. Inspired by the recent object recognition and fine-grained recognition work [129, 131, 132], we propose an image feature representation called local feature embedding (LFE) that captures salient visual properties to address the simultaneous fine-grained and coarse-grained aspects of VFR.
2. Adopting the nearest class mean (NCM) classifier, we build a scalable recognition algorithm with metric learning and max margin template selection based on the LFE representation. Similar to [133], the new algorithm can generalize to new classes at very little added cost.
3. We have synthesized a large-scale dataset for visual font recognition, consisting of 2,420 font classes each with 1,000 English word images. We also collected a small test set of real world images, each with a known font class label that is one of the classes in the training set. We will release both datasets with publication in the future.

Fig. 6.1 shows our visual font recognition on two real world images, where our algorithm correctly classifies both (top of the list) and returns four similar other typefaces, even though the text images have high levels of clutter and noise, and some perspective distortion (right).

6.3 A Database for Visual Font Recognition

In this work, we focus on visual font recognition for the Roman alphabet. As the most common case for font recognition is to identify the font class of short texts in images, we want to collect images that contain short strings or single English words as the training data. However, collecting real world examples for a large collection of font classes turns out to be extremely difficult because many attainable real world text images do not have font label information. Furthermore, the error-prone font labeling task requires font expertise that is out of reach of most people. Instead, we turn to synthesizing these images for the given fonts to obtain the training data.

In order to get a representative English word set, we randomly select 1,000 English words from the most common 5,000 English words sampled from a large corpus. The English words have variable lengths, resulting in word images of different sizes. To capture the variations caused by letter cases, we randomly divide the selected 1,000 English words into lower and uppercases with equal probability. We collect in total 447 typefaces, each with different number of variations resulting from combinations of different styles, *e.g.*, regular, semibold, bold, black, and italic, leading to 2,420 font classes in the end.

For each font class, we generate one image per English word, which gives 2.42 million synthetic images for the whole dataset. To normalize the text size[2], we add two lower case letters "fg" in front of each word when synthesizing the image. This helps us to find the ascender and descender lines of the text. We then normalize the image size by fixing the distance between the ascender line and descender line. The two letters "fg" are then removed from the synthesized images. After normalization, we obtain all word images with the same height of 105 pixels.

Besides the synthetic data, we also collected 325 real world test images for the font classes we have in the training set[3]. These images were collected from typography forums, such as myfonts.com, where people post these images seeking help from experts to identify the fonts. Compared with the synthetic data, these images typically have much larger appearance variations caused by scale, background, lighting, noise, perspective distortions, and compression artifacts. We manually cropped the texts from these images with a bounding box to normalize the text size approximately to the same scale as the synthetic data.

[2]In font rendering, the same font size may not generate the same text size in pixels for different fonts.

[3]As we mentioned earlier, it is really hard to collect real world test images for the given fonts. We are still growing our real-world dataset. Nevertheless, the current small dataset should be sufficient to give us a teaser of the real problem.

Figure 6.5 in the experiment section shows some real-world test images from different font classes.

6.4 Our Approach

In this section, we first present our image feature representation for visual font recognition, and then we describe our large-scale classification algorithm based on local feature metric learning and max-margin template selection.

6.4.1 Local Feature Embedding

Most of the current state-of-the-art generic image classification systems [134, 135, 129] follow the pipeline of first encoding the local image descriptors (*e.g.*, SIFT [136] or LBP [130]) into sparse codes, and then pooling the sparse codes into a fixed-length image feature representation. With each image represented as a collection of local image descriptors $\{\mathbf{x}_i\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$, the first coding step encodes each local descriptor into some code (typically sparse),

$$\mathbf{y}_i = f(\mathbf{x}_i, T), \tag{6.1}$$

where $T = \{\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_K\}$ denotes the template model or codebook of size K and $\mathbf{t}_i \in \mathbb{R}^d$, f is the encoding function (*e.g.*, vector quantization [134], soft assignment [137], LLC [129], or sparse coding [135]), and $\mathbf{y}_i \in \mathbb{R}^K$ is the code for $\mathbf{x}_i$. Then the pooling step obtains the final image representation by

$$\mathbf{z} = g(\{\mathbf{y}_i\}_{i=1}^n), \tag{6.2}$$

where g is a pooling function that computes some statistics from each dimension of the set of vectors $\{\mathbf{y}_i\}_{i=1}^n$ (*e.g.*, average pooling [134], max pooling [135]), and $\mathbf{z} \in \mathbb{R}^K$ is the pooled feature vector that will later be fed into a classifier.

While the above feature extraction pipeline is effective at distinguishing different categories of objects, it is not sufficient to capture the subtle differences within a object category for fine-grained recognition, *e.g.*, the letter endings in Fig. 6.2. Inspired by the recent fine-grained recognition work [138, 131, 132], we extend the above feature extraction pipeline by embedding the local features into the pooling vector, in order to preserve the details of the local letter parts. Specifically, using max pooling in Eqn. (6.2), we not only pool the maximum sparse coefficients, but also record the indices of these max pooling coefficients:

$$\{\mathbf{z}, \mathbf{e}\} = \max(\{\mathbf{y}_i\}_{i=1}^n), \tag{6.3}$$

where $\mathbf{z}$ contains the max coefficients pooled from each dimension of the set $\{\mathbf{y}_i\}_{i=1}^n$ and $\mathbf{e}$ is its index vector. Denoting $e_k = \mathbf{e}(k)$ and $z_k = \mathbf{z}(k)$, it is easy to see that $z_k = \mathbf{y}_{e_k}(k)$. Instead of using the max pooling coefficients as the final image feature representation [129], we obtain the pooling coefficients together with the local descriptor that fires each of them $\{z_k, \mathbf{x}_{e_k}\}_{k=1}^K$. We construct the final feature representation by concatenating these local descriptors weighted by their pooling coefficients:

$$\mathbf{f} = \left[z_1\mathbf{x}_{e_1}; z_2\mathbf{x}_{e_2}; ...; z_K\mathbf{x}_{e_K}\right]. \tag{6.4}$$

The max pooling procedure introduces a competing process for all the local descriptors to match templates. Each pooling coefficient z_k measures the response significance of $\mathbf{x}_{e_k}$ with respect to template $\mathbf{t}_k$, which is effective at categorizing coarse object shapes [129, 135], while the pooled local descriptor $\mathbf{x}_{e_k}$ preserves the local part details that are discriminative for classifying subtle fine-grained differences when the pooling coefficients are similar [138]. Therefore, our feature representation in Eqn. (6.4) can capture both coarse level object appearance changes and subtle object part changes, and we call this feature representation local feature embedding (LFE).

Our local feature embedding embeds the local descriptors from max pooling into a much higher dimensional space of $\mathbb{R}^{Kd}$. For instance, if we use 59-dimensional LBP descriptors and a codebook size of 2048, the dimension of $\mathbf{f}$ without using SPM is already 120,832. Although embedding the image into higher dimensional spaces is typically amenable to linear classifiers [128, 129, 135], training classifiers for very large-scale applications can be very time-consuming. What's more, a major drawback of training classifiers for large-scale classification is that, when images of new categories become available or new images are add to the existing categories, new classifiers have to be retrained at a very high computational cost [133]. In the following section, we propose a new large-scale classification algorithm based on local feature metric learning and template selection, which can be easily generalized to new classes and new data at very little cost. For this purpose, we modify the LFE feature in Eqn. (6.4) into a local feature set representation:

$$\mathbf{f} = \{(z_k, \mathbf{x}_{e_k})\}_{k=1}^{K}. \tag{6.5}$$

6.4.2 Large-Scale Classification

In our large-scale visual font recognition task, the dataset is typically open-ended, *i.e.*, new font categories appear over time and new data samples could be added to the existing categories. It is, therefore, important for a practical classification algorithm to be able to generalize to new classes and new data at very little cost. Nearest class mean (NCM) together with metric learning [133] has been explored for large-scale classification tasks, where each class is represented by their mean feature vector that is efficient to compute. In this chapter, we generalize this idea to NCM based on pooled local features to form a set of weak classifiers. Then we propose a max-margin template selection scheme to combine these weak classifiers for the final classification.

Within-Class Covariance Normalization

Given the LFE feature $\mathbf{f} = \{(z_k, \mathbf{x}_{e_k})\}_{k=1}^{K}$ for each image, we would like to learn a Mahalanobis distance metric for each pooled local feature space, under which we formulate the NCM classifier using multi-class logistic regression [133], where the probability for a class c given a pooled local feature $\mathbf{x}_{e_k}$ is defined by

$$p(c|\mathbf{x}_{e_k}) = \frac{\exp(-\|\mu_k^c - \mathbf{x}_{e_k}\|_{W_k}^2)}{\sum_{c'=1}^{C} \exp(-\|\mu_k^{c'} - \mathbf{x}_{e_k}\|_{W_k}^2)}, \tag{6.6}$$

where μ_k^c is the class mean vector for the k-th pooled local features in class c, and

$$\|\mu_k^c - \mathbf{x}_{e_k}\|_{W_k}^2 = (\mu_k^c - \mathbf{x}_{e_k})^T W_k^T W_k (\mu_k^c - \mathbf{x}_{e_k}). \tag{6.7}$$

Denoting $\Sigma_k^{-1} = W_k^T W_k$, we can see the k-th pooled feature space (or its projected subspace) is modeled as a Gaussian distribution with an inverse covariance matrix Σ_k^{-1}.

To learn the metric W_k for the k-th pooled feature space, we use a simple metric learning method called within-class covariance normalization (WCCN). First, interpreting z_k as the probabilistic response of $\mathbf{x}_{e_k}$ to template $\mathbf{t}_k$, we can compute the class mean vector μ_k^c by

$$\mu_k^c = \frac{1}{Z^c} \sum_{i \in \mathcal{I}_c} z_k^i \mathbf{x}_{e_k}^i, \tag{6.8}$$

where i is the index for the i-th training image with LFE feature $\mathbf{f}^i = \{z_k^i, \mathbf{x}_{e_k}^i\}_{k=1}^{K}$, $\mathcal{I}_c$ denote the sample index set for class c, and $Z^c = \sum_{i \in \mathcal{I}_c} z_k^i$ is a normalization factor. Then, we compute Σ_k as the expected within-class covariance matrix over all classes:

$$\Sigma_k = E\left[\Sigma_{c'k}\right] \approx \sum_{c'=1}^{C} p(c') \Sigma_k^{c'}, \tag{6.9}$$

where

$$p(c') = \frac{\sum_{i \in \mathcal{I}_{c'}} z_k^i}{\sum_i z_k^i} \tag{6.10}$$

is the empirical probability of class c', and $\Sigma_k^{c'}$ is the within-class covariance for class c' defined as

$$\Sigma_k^{c'} \approx \frac{1}{Z^{c'}} \sum_{i \in \mathcal{I}_{c'}} z_k^i (\mathbf{x}_{e_k}^i - \mu_k^{c'})(\mathbf{x}_{e_k}^i - \mu_k^{c'})^T, \tag{6.11}$$

with $Z^{c'} = \sum_{i \in \mathcal{I}_{c'}} z_k^i$. In practice, empirical estimates of Σ_k may be noisy; therefore, we add a certain amount of smoothness by shrinking it towards the scalar covariance as

$$\hat{\Sigma}_k = (1-\alpha)\Sigma_k + \alpha\sigma^2 I, \ \alpha \in [0, 1), \tag{6.12}$$

where $\hat{\Sigma}_k$ represents a smoothed version of the empirical expected within-class covariance matrix, I is the identity matrix, and σ^2 can take the value of $trace(\Sigma_k)$. Suppose we compute the eigen-decomposition for each $\hat{\Sigma}_k = U_k D_k U_k^T$, where U_k is orthonormal and D_k is a diagonal matrix of positive eigenvalues. Then the feature projection matrix W_k in Eqn (6.6) is defined as

$$W_k = D_k^{-1/2} U_k^T, \tag{6.13}$$

which basically spheres the data based on the common covariance matrix. In the transformed space, nearest class mean can be used as the classifier, which lays the foundation for the multi-class logistic regression in Eqn. (6.6).

To further enhance the discriminative power of W_k, we can depress the projection components with high within-class variability, by discarding the first few largest eigenvalues in D_k, which corresponds to the subspace where the feature similarity and label similarity are most out of sync (large eigenvalues correspond to large within-class variance). In this case, it can be shown that the solution of WCCN can be interpreted as the result of discriminative subspace learning [139].

Max-Margin Template Selection

After we learned the metric for each pooled local feature space, and assuming the templates in T are independent, we can evaluate the posterior of a class c for the input image feature representation $\mathbf{f}$ by combining the outputs of Eqn. (6.6) using a log-linear model:

$$p(c|\mathbf{f}) = \frac{1}{H} \exp\left(a + \sum_k w_k \log p(c|\mathbf{x}_{e_k})\right). \tag{6.14}$$

where H is a normalization factor to ensure the integrity of $p(c|\mathbf{f})$, w_k weights the contribution of each pooled local feature to the final classification, and a is a small constant offset. Here, the weight vector $\mathbf{w} = [w_1, w_2, ..., w_K]^T$, shared by all classes, acts to select the most discriminative templates from the template model $T = \{\mathbf{t}_k\}_{k=1}^K$ for the given classification task. Then classification for $\mathbf{f}$ is simply to choose the class with the largest posterior:

$$c^* = \arg\max_{c'} p(c'|\mathbf{f}). \tag{6.15}$$

Alternatively, we can treat the multi-class logistic regression for each pooled local feature as a weak classifier, and then linearly combine them to obtain a strong classifier:

$$s(c|\mathbf{f}) = \sum_{k=1}^{K} w_k p(c|\mathbf{x}_{e_k}). \tag{6.16}$$

In this way, we can avoid the numerical instability and data scale problem of logarithm in Eqn. (6.14). The score function $s(c|\mathbf{f})$ does not have a probabilistic interpretation any more, but classification is again simply to find the class with the largest score output. In practice, we find this formulation works slightly better than the previous log-linear model, and we adopt this linear model for all the experiments.

Given the training samples $\{\mathbf{f}^i, c^i\}_{i=1}^N$, where $c^i \in \{1, ..., C\}$ is the class label for the i-th data sample, we want to find the optimal weight vector $\mathbf{w}$ such that the following constraints are best satisfied,

$$s(c^i|\mathbf{f}^i) > s(c'|\mathbf{f}^i),\ \forall i, c' \neq c^i, \tag{6.17}$$

which translates to

$$\sum_{k=1}^{K} w_k \left(p(c^i|\mathbf{x}^i_{e_k}) - p(c'|\mathbf{x}^i_{e_k}) \right) > 0,\ \forall i, c' \neq c^i. \tag{6.18}$$

In order to learn $\mathbf{w}$, we define a cost function using a multi-class hinge loss function to penalize violations of the above constraints

$$\mathcal{L}(\mathbf{f}^i, c^i; \mathbf{w}) = \sum_{c' \neq c^i} \max\{0, -\gamma^i(c') + 1\}, \tag{6.19}$$

where

$$\gamma^i(c') = \sum_{k=1}^{K} w_k \left(p(c^i|\mathbf{x}^i_{e_k}) - p(c'|\mathbf{x}^i_{e_k}) \right). \tag{6.20}$$

Then learning $\mathbf{w}$ is simply to solve the following optimization:

$$\min_{\mathbf{w}} \lambda \sum_{i=1}^{N} \mathcal{L}(\mathbf{f}^i, c^i; \mathbf{w}) + \rho(\mathbf{w}), \tag{6.21}$$

where $\rho(\mathbf{w})$ regularizes the model complexity. In this work, we use $\rho(\mathbf{w}) = \|\mathbf{w}\|_2^2$, and Eqn. (6.21) is simply the classical one-class SVM formulation. To see this, denoting

$$\mathbf{p}^i(c) = \left[p(c|\mathbf{x}^i_{e_1}); p(c|\mathbf{x}^i_{e_2}); ...; p(c|\mathbf{x}^i_{e_K}) \right], \tag{6.22}$$

and $\mathbf{q}^i(c') = \mathbf{p}^i(c^i) - \mathbf{p}^i(c')$, Eqn. (6.19) can then translate to

$$\mathcal{L}(\mathbf{f}^i, c^i; \mathbf{w}) = \sum_{c' \neq c^i} \max\{0, -\mathbf{w}^T \mathbf{q}^i(c') \cdot 1 + 1\}, \tag{6.23}$$

where $\mathbf{q}^i(c')$ can be regarded as feature vectors with only positive label $+1$. Therefore, the optimization in (6.21) is the classical SVM formulation with only positive class and thus can be readily solved by many existing SVM packages, *e.g.*., [140]. The regularization term $\rho(\mathbf{w})$ here may also take other forms, such as $\|\mathbf{w}\|_1$, where the ℓ^1-norm promotes sparsity for template selection, which typically has better generalization behavior when the size K of the template model T is very large.

After we learn the WCCN metric for all pooled local feature spaces and the template weights based on LFE, classification for a given $\mathbf{f}$ is straightforward: first compute the local feature posteriors using Eqn. (6.6), combine them with the learned weights $\mathbf{w}$, and then predict the class label by selecting the largest score output $c^* = \max_{c'} s(c'|\mathbf{f})$. When new data or font classes are added to the database, we only need to calculate the new class mean vectors, and estimate the within-class covariances to update the WCCN metric incrementally. As the template model is universally shared by all classes, the template weights do not need to be retrained.[4] Therefore, our algorithm can easily adapt to new data or new classes at little added cost.

6.5 Experiments

We now evaluate our large-scale recognition algorithm on the collected VFR database. We implement and evaluate two baseline algorithms: 1) a representative font recognition algorithm on scanned documents [124]; and 2) a widely used image recognition algorithm LLC [129]. To get the local feature embedding (LFE) representation, we evaluated several state-of-the-art texture or shape descriptors including covariance feature [141], shape context [142], local binary pattern (LBP) [130] and SIFT [136], which have been extensively verified in many computer vision applications. We find that SIFT works the best. For the local descriptor encoding, we use LLC coding [129] to compare fairly with the LLC base-

[4]Same as other supervised learning algorithms, if the new added data or classes change the data distribution substantially, the template model and their weights need to be retrained.

line, although other coding schemes can also be used, such as soft assignment and sparse coding.

Our algorithm has very few parameters. The local descriptors(e.g., SIFT, LBP) are extracted from 12×12 image patches sampled from a regular grid on the image with step size of 6 pixels. The template model T is learned by kmeans with size 2048. To compute the smoothed version of the within-class covariance in Eqn. (6.12), we set $\alpha = 0.1$ as a constant. When calculating the projection matrix W_k in Eqn. (6.13), we throw away the first two largest eigenvalues from D_k to depress the components with high within-class variability.

6.5.1 Dataset Preparation

Our experimental dataset consists of three distinct sets: VFR-447, a synthetic dataset containing 447 typefaces with only one font variation for each typeface; VFR-2420, a large synthetic dataset containing typefaces with all variations; and VFR-Wild, which has 325 real world test images for 103 fonts out of the 2420 classes, each class with the number of images ranging from 1 to 17. Each class in VFR-447 and VFR-2420 has 1,000 synthetic word images, which are evenly split into 500 training and 500 testing. There are no common words between the training and testing images.

To model the realistic use cases, we add moderate distortions and noise to the synthetic data. First, we add a random Gaussian blur with standard deviation from 2.5 to 3.5 to the image. Second, a perspective distortion is added by moving each corner of the image randomly by ± 5 pixels along x and y directions (images are pre-normalized to have the same height of 105 pixels), which defines a perspective transformation. Third, the foreground (text) and background intensities are randomly perturbed between [0, 255] with the constraint that the intensity of foreground is at least 20 intensity levels smaller than the background. Finally, some small Gaussian noise is added to the image, where we assume the noise of a test images will be reasonably reduced with some simply preprocessing.

Codebook	512	1024	1536	2048
LLC + SVM	64.63	73.04	78.13	82.23
LFE + Naive	76.04	80.20	81.77	84.21
LFE + FS	**80.44**	**85.26**	**87.73**	**91.35**
Improvements	18.36%	25.56%	32.69%	45.22%

Table 6.1: The top 1 accuracy of different algorithms with different template model sizes on VFR-447. "Improvements" shows error reduction percentage of LFE+FS against LFE + Naive.

Covariance [141]	Shape C. [142]	LBP [130]	SIFT [136]
44.13	54.13	89.02	**91.35**

Table 6.2: The top 1 accuracies of different local descriptors with our proposed LFE+FS algorithm on VFR-447.

6.5.2 Results on VFR-447

Table 6.1 shows the recognition results on the VFR-447 synthetic dataset under different template model sizes in terms of top 1 accuracy. "LFE+Naive" denotes our method without template selection, *i.e.*, equal weights in Eqn. (6.16), and "LFE+FS" denotes our method with template selection. In all cases, both our methods significantly outperform the baseline algorithm LLC [129]. Table 6.2 lists the recognition results with different local descriptors. We can see that SIFT performs the best. LBP is slightly worse than SIFT, but its efficiency justifies itself as a good alternative to SIFT.

In Table 6.3, we evaluated the top 1, 5, and 10 accuracies on the VFR-447 dataset with 2048 templates, in comparison with the two baseline algorithms [124] (denoted by "STAT") and LLC [129]. In all cases, our algorithm works the best. The previous font recognition algorithm [124] focuses on scanned documents. It depends on a large text sample to extract stable texture statistics for recognition. Therefore, it won't work well in our case where the test images have very short texts with noisy background. Since the VFR-447 dataset does not have font variations within each typeface, *i.e.*, most font classes are visually distinct, coarse-grained techniques such as LLC is still working reasonably well.

Methods	Top 1	Top 5	Top 10
STAT [124]	25.12	30.30	33.41
LLC + SVM [129]	82.23	93.39	95.32
LFE + Naive	84.20	94.14	95.53
LFE + FS	**91.35**	**98.90**	**99.62**

Table 6.3: The top 1, 5, 10 accuracies of different algorithms with template model size 2,048 on VFR-447 synthetic dataset.

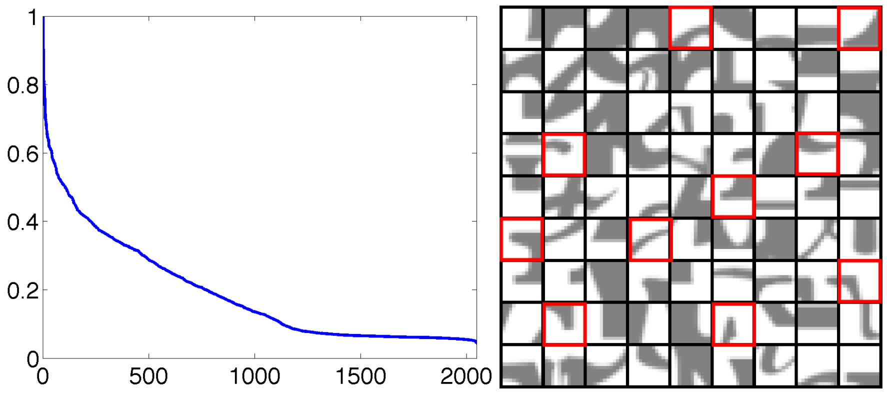


Figure 6.3: Sorted template weights (left) and top 81 selected templates (right) with largest weights. Many of these selected templates correspond to letter endings.

Figure 6.3 depicts the max-margin template selection results on the 2,048 template model. The left figure plots the sorted weights for the 2,048 templates after max-margin template selection using optimization in Eqn. (6.21). Although all templates are selected (using $||\mathbf{w}||_2^2$ regularization), only a small portion of the templates are selected with large weights. The right figure illustrates the top 81 selected templates[5], most of which correspond to letters endings and curvature strokes that are most informative for font recognition.

6.5.3 Results on VFR-2420

In Table 6.4, we report the top 1, 5, and 10 accuracies on the VFR-2420 dataset with template model of size 2,048 and LBP as local descriptor. Compared with the results on

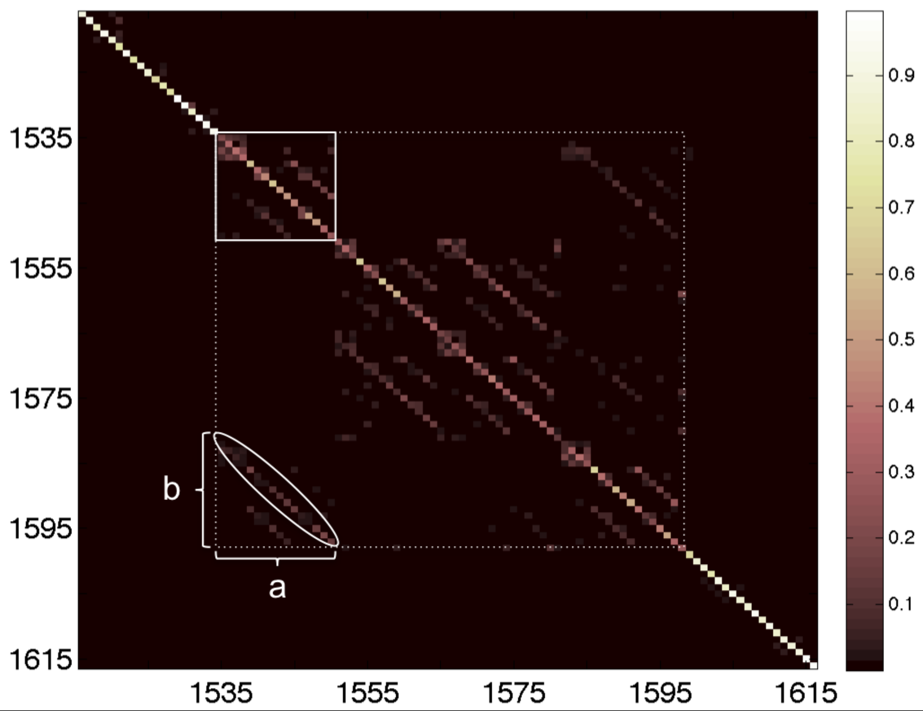
[5]The templates are visualized by image patches whose local descriptors are the nearest neighbors to the selected templates

Methods	Top 1	Top 5	Top 10
STAT [124]	15.25	20.50	26.68
LLC + SVM [129]	50.06	72.48	78.49
LFE + Naive	65.20	85.36	89.93
LFE + FS	**72.50**	**93.45**	**96.87**

Table 6.4: The top 1, 5, 10 accuracies of different algorithms with template model size 2,048 on VFR-2420 synthetic dataset.

VFR-447 in Table 6.1, the top 1 accuracy of our algorithm for VFR-2420 drops notably, from 91.35% to 72.50%, which is expected as the problem becomes super fine-grained with font variations within each typeface. However, the top 5 and 10 accuracies are much better, suggesting that our algorithm is effective at retrieving similar font classes, even though it is confused by subtle font variations for top 1 classification. In contrast, LLC works much worse than our algorithm on this dataset, due to its ineffectiveness in handling fine-grained recognition tasks. Again, STAT [124] performs the worst.

To see that the top 1 accuracy of our algorithm is mainly affected by the similarity of font variations within each typeface, Fig. 6.4 plots a sub-confusion matrix for 100 fonts indexing from 1515 to 1615. The sub-confusion matrix demonstrates a hierarchical block structure, where large blocks correspond to font variations within a typeface, *e.g.*, the large dotted rectangle corresponds to all font variations within typeface Minion Pro, and smaller blocks correspond to font variations within a subfamily of a typeface, *e.g.*, the small rectangle corresponds to variations within Minion Pro Bold. Interestingly, there are also many periodic off-diagonal lines inside the blocks, which are typically caused by one particular font variation. For example, the line structure in the ellipse is caused by the similarity of weight variation between bold and semibold, as indicated by their font names listed in the accompanying table. Such observations suggest that our confusion matrix is a good reflection of font similarity, which may be useful for font organization, hierarchical grouping, selection, and recommendation.



Font list a	Font list b
1534: Minion Pro Bold	1581: Minion Pro SemiBold
1535: Minion Pro Bold Caption	1582: Minion Pro SemiBold Caption
⋮	⋮
1549: Minion Pro Bold Subhead	1596: Minion Pro SemiBold Subhead

Figure 6.4: The sub-confusion matrix for 100 font classes indexing from 1515 to 1615.

Methods	Top 1	Top 5	Top 10	Top 20
STAT [124]	4.13	7.30	9.08	10.17
LLC + SVM [129]	26.46	44.00	51.08	57.23
LFE + Naive	29.54	46.15	54.46	62.46
LFE + FS	**52.61**	**58.40**	**62.14**	**64.16**

Table 6.5: The top 1, 5, 10, and 20 accuracies of different algorithms with 2,048 templates on real world dataset.

6.5.4 Results on VFR-Wild

Table 6.5 shows the performance of our algorithm on the real world test images. As introduced in Section 6.3, all the real-world images were roughly cropped and oriented manually. Then we only did minimum preprocessing of denoising using a bilateral filter with fixed parameters for all images. For heavier distortions, more complicated preprocessing is needed, which we leave as future work. As shown in Table 6.5, compared to the synthetic data, the performance drops notably, which is expected because of the mismatch between training and testing data. Nevertheless, our LFE combined with template selection again significantly outperforms both LLC and LFE without template selection. STAT was developed for scanned documents and performs very poorly on the wild data. Fig. 6.5 shows some example real world test images that are (a) correctly and (b) wrongly classified by our algorithm. Remarkably, our model, although trained on the synthetic dataset, is robust to cluttered background and noise to a large extent shown by (a). In cases of decorated texts, very low-resolution, extremely noisy input, and very cluttered background shown in (b), the algorithm will fail. Better image preprocessing techniques will definitely help in such cases, which we leave as future work. Considering all these challenging factors in real world VFR, a recognition rate of 52.61% at top 1 accuracy is very promising. Note that most of our real world images contain very short strings. In cases where the input may contain long strings of text, we can cut the them into short strings and combine the algorithm's inferences on all of them.

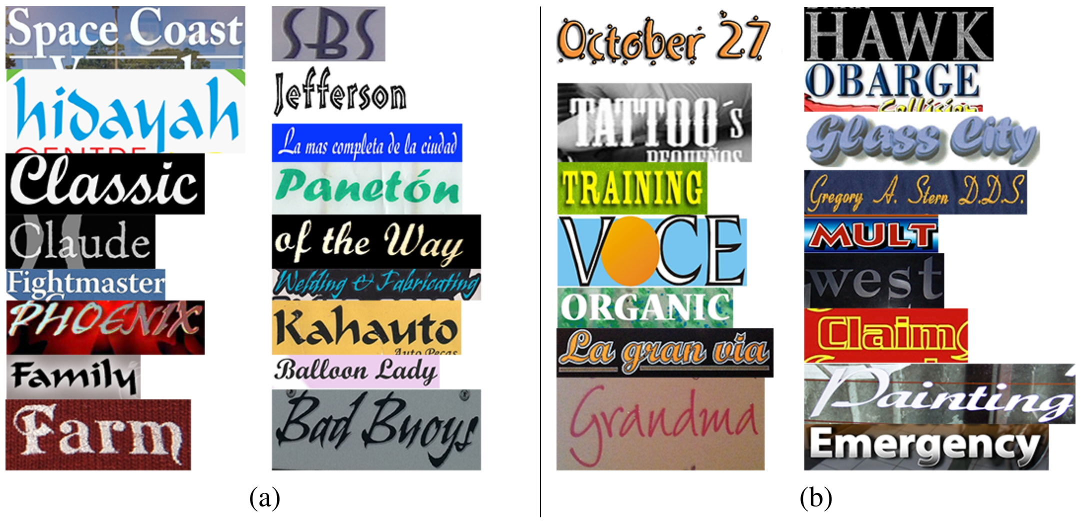


Figure 6.5: (a) Real world images that are correctly classified (rank one). (b) Real world images that are wrongly classified (fall out of top rank 20).

6.6 Extension work

The previous sections introduce our local feature embedding and template selection(LFS) algorithm. The LFS algorithm adopts the nearest class mean (NCM) classifier, which provides a scalable solution to Visual Font Recognition (VFR).In this section, we would like to discuss a new algorithm based on LFS.

We have observed that LFS [37] is very effective on our collected VRF-447 and VFR-2420 synthetic dataset, and it achieved very promising results on the VFR-wild real-world test dataset. However there still exists great potential to achieve improvements over LFS in terms of both recognition accuracy and scalability. Currently, LFS could be recognized as a flat multi-class classification problem, which treats each font class equally and tries to learn classifiers that separate them simultaneously. While clearly, some fonts are sharing a lot (usually they belong to the same font typeface) and only take small variations, such as aspect ratio of character, stroke width, and ending slope. So the differences between them are subtle and classifying these fonts is usually quite different from classifying fonts that share very little. To solve this problem, we prefer to finding a good way to cluster the fonts, which makes fonts inside each cluster are similar but vary dramatically from fonts in other clusters. Then we will be able to learn specific classifiers for each cluster of fonts and brings better classification accuracy. This naturally leads to a hierarchical classification scheme based on a good tree structure. Another equally important aspect of using a tree-based hierarchical classification is that the algorithm is much faster, and thus our new algorithm is even more scalable to large scale problems. Hierarchical algorithms or called tree algorithms are widely used in recognition area. To handle large-scale classification, currently a bunch of “label tree” algorithms [143, 144, 145] have been proposed and achieved state-of-the-art performances in classification benchmark. They usually define a clustering algorithm to split classes inside each node, and then learn classifiers specifically for each node. Following a similar step, we propose a novel node splitting and tree-learning algorithm.

6.6.1 Node hard-splitting: Discriminative classification clustering

During this node splitting process, suppose there are total N font classes in this current node i, we want to assign these N fonts into C child nodes. Here we only allow each font class being assigned into 1 child node. That means, the child nodes contains no same font classes. This step is called node hard-splitting.

Distance between font classes In our method, we use local feature embedding (LFE) to represent each font image: $f = \{(z_k, \mathbf{x}_{e_k})\}_{k=1}^{K}$, where K is the codebook size, z_k is pooling coefficient of the k-th code, $\mathbf{x}_{e_k}$ represents the pooled local descriptor vector. So based on LFE feature, we could compute mean vector μ_k^c for each font class:

$$\mu_k^c = \frac{1}{Z^c} \sum_{i \in I_c} z_k^i \mathbf{x}_{e_k}^i, \tag{6.24}$$

and we could also get the within-class covariance matrix [146] over all font classes, denoted by Σ_k. So now for each font class, we could use $\{(\mu_k^c, \Sigma_k)\}_{k=1}^{K}$ to represent it. After this, we define the distance between each pair of fonts:

$$d(c_1, c_2) = \sum_{k=1}^{K} w_k d_M(\mu_{c_1}^k, \mu_{c_2}^k), \tag{6.25}$$

where $d_M(\mu_{c_1}^k, \mu_{c_2}^k) = \|\mu_k^{c_1} - \mu_k^{c_2}\|_{\Sigma_k}^2$ is the Mahalanobis distance between the template mean vectors $\mu_k^{c_1}$ and $\mu_k^{c_2}$. While w_k is the weight to incorporate the importance of the k-th template. If the k-th template is more effective than other templates in separating the fonts, we would like to give it a larger value. Initially we do not know the importance of the templates, so we set all $w_k = 1/C$ at first.

Sparse affinity matrix After defining distances between font classes, we could build a distance matrix D with element $d_{ij} = d(c_i, c_j)$. and the affinity matrix A with element: $A_{ij} = \exp(-d(c_i, c_j)/\sigma)$, where σ is the scaling parameter. Clearly the affinity matrix A is symmetric and the diagonal elements are all 0. The meaning of matrix A is: The higher value A_{ij} is, the more similar the two font c_i and c_j are.

With the full (non-sparse) affinity matrix A, we could use many classic clustering algorithms to cluster these fonts. Here we use spectral clustering [54] to do this. Suppose we want to cluster these N fonts into K clusters, the steps for spectral clustering are

1. Compute the diagonal matrix T with elements $T_{ii} = \sum_{j=1}^{N} A_{ij}$;
2. Compute the normalized Laplacian matrix: $L = T^{-1/2}(T - A)T^(1/2)$;
3. Compute and sort eigenvalues of matrix L in descending order: $\lambda_i \geq \lambda_i + 1, i = 1, \cdots, n$;
4. Form normalized matrix S using C largest eigenvectors;
5. Treating each row of S as a data point, cluster all the data points by K-means with cluster number C.

However, from experiments, we found that clustering on a full affinity matrix A is usually non-stable and performed badly; Besides, the clustering is usually quite sensitive to parameter σ. Without a carefully-tuned σ, the clustering is often not successful. Consequently, a bad clustering will make the font classification algorithm(LFE+FS) fail. To solve these two problems, we propose a method that returns stable and appropriate clustering results. The basic idea is:

1. Firstly, normalize the distance matrix D by dividing each element d_{ij} by the median value $\bar{d}$ of matrix elements in D, i.e., $\bar{d} = median(d_{ij})$.

2. Only keep the distance values of q-nearest fonts for each font. The distance with far fonts are set as $\inf$. The parameter q is chosen in this way: suppose there are total N font classes, and we want to split them into C clusters, then $q = N/C$.

3. Now the affinity matrix A is a sparse matrix. Note that, the scaling parameter is fixed value $\sigma = 1$. This is due to normalization step 1.

4. Make the affinity matrix A symmetric: $A \leftarrow \frac{1}{2}(A + A^T)$.

5. Finally, perform spectral clustering algorithm [54] on matrix A as before.

From experiment, we found our proposed sparse affinity matrix works pretty well. Comparing with previous self-tuning spectral clustering algorithm [55], our algorithm works much better and stable. In the meanwhile, there are no sensitive parameters and thus avoids tuning. This character is really important for tree construction. Note that, in above step 1, we use median not mean, since in statistic viewpoint, median is usually more stable than mean.

Discriminative classification clustering As mentioned above, we want to introduce the importance weight w_k when computing the font distance $d(c_1, c_2)$ in Equation 6.25. Since our previous work LFE+FS finally performs a template selection step and assigns a weight to each template feature. For the templates good at classifying different fonts, the LFE+FS would give them larger weight. So we could directly use this weight as the importance weight w_k. So, initially we set $w_k = 1/C$ and perform clustering on all fonts. After clustering N fonts into C clusters, we treat each cluster as a new class, and learn LFE+FS to classify these classes and get the weights w_k. With w_k, we re-compute the distance between font classes. Then we get a new sparse affinity matrix and perform clustering again. This procedure can be repeated to get better clustering results. The algorithm steps are as following:

1. Firstly, set all $w_k = 1/C$, perform the clustering algorithm introduced above.
2. Perform LFE+FS and get a set of importance weights $\{w_k\}$. Evaluate current classification accuracy.
3. Based on the new template weights $\{w_k\}$, perform clustering again.
4. Run step 2 and 3 untill classification performance converges.

From experiments, this discriminative classification clustering works well and iteratively improves the classification performance. Typically, it converges in 4 or 5 iterations.

6.6.2 Node soft-assignment: Error-bounded splitting

After node hard-splitting step, we get a good assignment for each font class. However, there still exists one serious problem: error propagation during tree growth. This error propagation is quite straightforward: since each font class in the node i only belongs to 1 child node. Suppose after hard-splitting, we know the way to assign each font into child nodes and thus we train a LFE+FS classifier f_i to judge a test data which child node it belongs to. So if a test data is misclassified by f_i, then it will falls into the wrong child node, then this test data can never find its true font label in the following steps. Here we denote the error of f_i as ϵ_i, then in this node layer, the classification accuracy is upper-bounded by $1 - \epsilon_i$. The problem becomes worse when we have a tree with multiple layers. This character is called error propagation in hierarchical algorithms.

Suppose for a tree, we have M layers, node layer i has upper-bounded classification accuracy $1 - \epsilon_i$. Then the upper-bounded classification rate of the whole tree would be $\Pi_{i=1}^{M}(1 - \epsilon_i)$. Suppose $M = 3$, $\epsilon_i = 0.15$. Then this trees best classification accuracy would be bounded by 0.614. In practice, ϵ_i is usually much larger than 0.15. So this error propagation problem is quite serious.

To solve this problem, we propose the node soft-assignment method, which we called it error-bounded node splitting. Firstly we use node hard-splitting method introduced above to get the initial splitting and LFE+FS classifier for the node i. Then based on classification accuracy of each font class, we allow it to be assigned into multiple child nodes. Imagine that, for font class j, it is supposed to belong to child node c_i, however, during test we find the test data of font j could fall into child nodes $\{c_l, c_{l+1}, c_{l+2}, ..., c_L\}$. Then we could compute the probability of the test data of font class j falls into these child nodes $\{p_l, p_{l+1}, p_{l+2}, , p_L\}$. We select the top R child nodes $\{c_r, c_{r+1}, , c_R\}$ with the highest probability such that the summation of the probability is larger than a pre-set threshold: $\sum_{r=1}^{R} p_r \geq \theta$. Finally we assign this font class into the child nodes $\{c_r, c_{r+1}, , c_R\}$.

With this step, we could make sure the classification accuracy of each font in this node i is at least θ_i. Thus we could bound the error rate of each node less than $1 - \theta_i$. So the entire trees upper-bound classification rate would be $\Pi_{i=1}^{M}\theta_i$. In practice, we usually chose $\theta_i = 0.95$ or higher, so if $M = 3$, the trees upper-bounded classification accuracy would be 0.857, which is much higher than without soft assignment.

Font class soft-assignment time: the average number of child nodes that each font class is softly assigned. Obviously if a font class is assigned into too many child nodes, it will increase the computation complexity. From experiments, we find usually the average assignment time is 2.2 3.5, which is only slightly burden for computation.

6.6.3 Error-bounded tree construction

With step 1 and step 2, we have the error-bounded node splitting method. Then its straightforward that we could build up the error-bounded tree. Suppose totally we get N font classes, and the root node has C child nodes. Then we firstly use step 1 (hard-splitting) to assign the N fonts into C child nodes; Then we use step 2 (soft-assignment) to re-assign the N fonts into C child nodes with certain error bound, denoting the average assignment time for each font as R. Then each child node i contains on average $N_i = RN/C$ font classes. Then for this child node i, we could continue to split it by dividing its N_i font classes into C_i children. Following the same procedure, we could build up the hierarchical error-bounded tree. In our experiments, currently we build a 2-layer tree: first layer is the C child nodes of the root node and each child node has certain number of fonts. The second layer is the leaf nodes, which means each node in the second layer only contains one font class.

For each tree node, we learn a specific codebook for it and perform the template selection by 1-class SVM. Also, for classification at each layer, we keep track of the top probabilities that a test sample is assigned to the child node. Aggregating these probabilities, we can obtain a global ranking.

6.6.4 Performance Evaluation

Speed Comparing with our previous flat classification algorithm(LFS), the error-bounded tree is more efficient. For each test data, we need compare with every font class mean to find the nearest font label.

Suppose we totally have N font classes, the computation complexity of flat classification algorithm would be $O(N)$. If we build up a 2-layer error-bounded tree, 1st layer has C child nodes, the average soft-assignment time is R, then the average font class number in each child node is $N_i = RN/C$. So 1st-layer node on average has RN/C leaf nodes. So

Methods	Top 1	Top 5	Top 10
STAT [124]	15.25	20.50	26.68
LLC + SVM [129]	50.06	72.48	78.49
LFE + Naive [37]	65.20	85.36	89.93
LFE + FS [37]	72.50	93.45	$\mathbf{96.87}$
Error-bounded Tree	$\mathbf{78.41}$	$\mathbf{94.67}$	$\mathbf{96.35}$

Table 6.6: The top 1, 5, 10 accuracies of different algorithms with template model size 2,048 on VFR-2420 synthetic dataset.

the computation complexity for the 1st-layer is $O(C)$ and the computation for the second layer is $O(RN/C)$. So the complexity of the error-bounded tree is $O(C + RN/C)$. In our experiments, we set $C \approx \sqrt{N}$. So the complexity of our error-bounded tree is $O(\sqrt{N})$, which is sub linear to the number of font classes. Comparing with flat classification complexity $O(N)$, the error-bounded tree is much more efficient, especially when font class number N becomes large.

Accuracy The tree algorithm will cluster the fonts that make similar fonts falls into the same child node. Then we could learn specific codebook and pool out image feature that more discriminative in classifying these fonts to each other. Thus the classification performance will be better. From experiment, the error-bounded algorithm works much better than flat classification algorithm. In VFR-2420 dataset, the Top 1 classification accuracy of the error-bounded tree algorithm improves around 6%, increasing from 72% to 78%, as shown in Tab 6.6.

6.7 Conclusion

In this chapter, we focus on the large-scale VFR problem that has long been neglected by the vision community. To address this problem, we have constructed a large-scale set of synthetic word images for 2,420 font classes, and collected a small set of real world images. Experiments on synthetic test data demonstrate the effectiveness of our approach, and experiments on real test images show very promising results. We then introduce a more scalable error-bounded tree algorithm, showing good accuracy and speed. For future work, we will grow the dataset of real test images. We will also explore the effects of different levels of distortions and noise added to synthetic data on the final performance on real test images. Machine learning and computer vision techniques, *e.g.*, transfer learning and robust local descriptors, will be exploited to close the gap between synthetic training and real world testing.

Chapter 7

Conclusion

This dissertation summarizes my research during the PhD study, covering object detection and fine-grained recognition topics. For object detection, we have proposed different algorithms to handle different kinds of problems. For fine-grained object recognition, we introduce a scalable and effective algorithm which is efficient for real-world applications.

To efficiently adapt an object detector for a specific scenario, we propose a Min-Max model adaption algorithm, which avoids tuning the important adaptation rate parameter. It is a conservative model adaptation method by considering the worst case during the adaptation process. Taking the object detection as a testbed, we implement an adapted object detector based on binary classification. Under different adaptation scenarios and different datasets including PASCAL, ImageNet, INRIA, and TUD-Pedestrian, the proposed adaption method achieves significant performance gain and is compared favorably with the state-of-the-art adaptation method [26] equipped with the fine tuned adaptation rate. Without the need of tuning the adaptation rates, the proposed conservative model adaptation method can be extended to other adaptive classification tasks.

To tackle the large intra-class variation difficulty, we developed a hybrid learning algorithm combining the global classification and the local adaptations, which automatically adjusts the model complexity according to the data distribution. Taking human detection as a testbed, under different scenarios and datasets, including Caltech pedestrian dataset [3], self-collected large pedestrian dataset, and INRIA dataset [8], the proposed hybrid learning method achieves significant performance gain. Compared with 11 state-of-the-art algorithms [3] on Caltech, the proposed approaches achieves the highest detection rate, outperforming the deformable part based algorithm [9] 17% at FPPI=1.

To extract the effective context information from the background, we propose an effective context descriptor, Multi-Order Contextual co-Occurrence (MOCO), to implicitly model the high level context using solely detection responses from a baseline object detector. We test the proposed MOCO evolution framework on the PASCAL VOC 2007 dataset [10] and Caltech pedestrian dataset [34]: The proposed MOCO detector outperforms all known state-of-the-art approaches, contextually boosting deformable part models (ver.5) [33] by 3.3% in mean average precision on the PASCAL 2007 dataset. For the Caltech pedestrian dataset, our method further reduces the log-average miss rate from 48% to 46% and the miss rate at 1 FPPI from 25% to 23%, compared with the best prior art [35].

To investigate object detection with a depth sensor, we propose an effective feature, *Normal Vector Pattern* (NVP). The proposed NVP is a successful attempt due to its two components: the cross products of neighboring normal vectors induce the discriminative power and the inner products of the normal vectors lead to the view-angle invariance. The superiority of the proposed NVP is evident: with a simple linear SVM model, the proposed framework outperforms the state-of-the-art features based on more complicated models such as the deformable parts model on standard Washington RGB-D [6] and NYU benchmark datasets [36].

Finally for the fine-grained object recognition problem, we propose a scalable and efficient algorithm to address a special large-scale fine-grained recognition problem—visual font recognition(VFR) problem. We construct a large-scale dataset containing 2,420 font classes, which easily exceeds the scale of most image categorization datasets in computer vision. As font recognition is inherently dynamic and open-ended, *i.e.*, new classes and data for existing categories are constantly added to the database over time, we propose a scalable solution based on the nearest class mean classifier (NCM). The core algorithm is built on local feature embedding, local feature metric learning and max-margin template selection, which is naturally amenable to NCM and thus to such open-ended classification problems. The new algorithm can generalize to new classes and new data at little added cost. Extensive experiments demonstrate that our approach is very effective on our synthetic test images, and achieves promising results on real world test images. Based on this local feature embedding and template learning algorithm, we then introduce a novel error-bounded tree algorithm to further the efficiency and effectiveness of our framework.

Bibliography

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 2008.

[2] Xiaoyu Wang, Xu Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *ICCV*, 2009.

[3] B. Schiele P. Dollar, C. Wojek and P. Perona. Pedestrian detection: A benchmark. In *CVPR*, 2009.

[4] Shuai Tang, Xiaoyu Wang, Xutao Lv, Tony X. Han, James Keller, Zhihai He, Marjorie Skubic, and Shihong Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. ACCV'12.

[5] Omar Oreifej and Zicheng Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[6] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *International Conference on Robotics and Automation*, 2011.

[7] P. Viola and M.J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, May 2004.

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[9] Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[11] Long Zhu, Yuanhao Chen, Alan L. Yuille, and William T. Freeman. Latent hierarchical structural learning for object detection. In *CVPR*, 2010.

[12] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient . In *ICCV*, 2009.

[13] Chaitanya Desai, Deva Ramanan, and Charless Fowlkes. Discriminative models for multi-class object layout. In *ICCV*, 2009.

[14] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.

[15] Qiang Zhu, Shai Avidan, Mei chen Yeh, and Kwang ting Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, 2006.

[16] Yuan Li, Haizhou Ai, Takayoshi Yamashita, Shihong Lao, and Masato Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2008.

[17] Junge Zhang, Kaiqi Huang, Yinan Yu, and Tieniu Tan. Boosted local structured hog-lbp for object localization. In *CVPR*, 2010.

[18] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[19] Shai Avidan. Ensemble tracking. In *CVPR*, pages 494–501, 2005.

[20] Robert T. Collins, Yanxi Liu, and Marius Leordeanu. On-line selection of discriminative tracking features. In *ICCV*, 2003.

[21] Helmut Grabner and Horst Bischof. On-line boosting and vision. In *CVPR*, 2006.

[22] Yuan Li and Haizhou Ai. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans. In *CVPR*, 2007.

[23] Xiaoming Liu and Ting Yu. Gradient feature selection for online boosting. In *ICCV*, 2007.

[24] Minh-Tri Pham and Tat-Jen Cham. Online learning asymmetric boosted classifiers for object detection. In *CVPR*, 2007.

[25] Gert Cauwenberghs and Tomaso Poggio. Incremental and Decremental Support Vector Machine Learning. In *NIPS*, pages 409–415, 2000.

[26] Cha Zhang, Raffay Hamid, and Zhengyou Zhang. Taylor expansion based classifier adaptation: Application to person detection. In *CVPR*, 2008.

[27] Guang Chen, Tony X. Han, and Shihong Lao. Adapting an object detector by considering the worst case: a conservative approach. In *CVPR*, 2011.

[28] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[29] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[30] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. 2006.

[31] Eon Bottou and Vladimir Vapnik. Local learning algorithms. *Neural Computation*, 1992.

[32] Guang Chen, Yuanyuan Ding, Jing Xiao, and Tony X. Han. Detection evolution with multi-order contextual co-occurrence. In *CVPR*, 2013.

[33] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/.

[34] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 2011.

[35] Yuanyuan Ding and Jing Xiao. Contextual boost for pedestrian detection. In *CVPR*, 2012.

[36] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, 2012.

[37] Guang Chen, Jianchao Yang, Hailin Jin, Jon Brandt, Eli Shechtman, Aseem Agarwala, and Tony X. Han. Large-scale visual font recognition. In *CVPR*, 2014.

[38] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee S. Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[39] E. Robert. A Brief Introduction to Boosting. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[40] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28, 2000.

[41] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*. MIT Press, Cambridge, MA, USA, 1988.

[42] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science++ Bussiness Media, LLC, 2006. 3.

[43] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. *CVPR*, 2009.

[44] *The PASCAL Visual Object Classes Challenge 2007*. http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/.

[45] *The ImageNet dataset*. http://www.image-net.org/index.

[46] *The WordNet dataset*. http://wordnet.princeton.edu/.

[47] Zhuowen Tu. Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. *ICCV*, 2005.

[48] Boris Babenko, P. Dollar, Zhuowen Tu, and Serge Belongie. Simultaneous learning and alignment: Multi-instance and multi-pose learning. In *ECCV Faces in Real-Life Images*, 2008.

[49] Hedi Harzallah, Frédéric Jurie, and Cordelia Schmid. Combining efficient object localization and image classification. In *ICCV*, sep 2009.

[50] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on PAMI*, 2001.

[51] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vision*, 2001.

[52] L.Levina. *StatisticalIssues in Texture Analysis*. PhD thesis, Department of Statistic, University of California, Berkeley, 2002.

[53] Marzia Polito and Pietro Perona. Grouping and dimensionality reduction by locally linear embedding. In *NIPS*, 2001.

[54] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[55] Lihi Zelnik-manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, 2004.

[56] Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *SIGKDD '09*, 2009.

[57] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral Channel Features. In *British Machine Vision Conference*, 2009.

[58] Dennis Park, Deva Ramanan, and Charless Fowlkes. Multiresolution models for object detection. In *ECCV*, 2010.

[59] Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010.

[60] Zheng Song, Qiang Chen, ZhongYang Huang, Yang Hua, and Shuicheng Yan. Contextualizing object detection and classification. In *CVPR*, 2011.

[61] Congcong Li, Devi Parikh, and Tsuhan Chen. Extracting adaptive contextual cues from unlabeled regions. In *ICCV*, 2011.

[62] Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. In *CVPR*, 2000.

[63] Michael Jones, Paul Viola, Paul Viola, Michael J. Jones, Daniel Snow, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *ICCV*, 2003.

[64] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV*, 2004.

[65] Geremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. In *ECCV*, 2008.

[66] Antonio Torralba. Contextual priming for object detection. *IJCV*, 2003.

[67] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Contextual models for object detection using boosted random fields. In *NIPS*, 2004.

[68] Peter Carbonetto, Nando de Freitas, and Kobus Barnard. A statistical model for general contextual object recognition. In *ECCV*, 2004.

[69] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. *IJCV*, 2008.

[70] Deva Ramanan. Using segmentation to verify object hypotheses. In *CVPR*, 2007.

[71] Shai Avidan. Spatialboost: adding spatial reasoning to adaboost. In *ECCV*, 2006.

[72] Zhuowen Tu and Xiang Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *PAMI*, 2010.

[73] Takumi Kobayashi and Nobuyuki Otsu. Bag of hierarchical co-occurrence features for image classification. In *ICPR*, 2010.

[74] Haibin Ling and Stefano Soatto. Proximity distribution kernels for geometric context in category recognition. In *ICCV*, 2007.

[75] Yi Yang and Shawn Newsam. Spatial pyramid co-occurrence for image classification. In *ICCV*, 2011.

[76] Mustafa Özuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *PAMI*, 2010.

[77] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *ECCV*, 2010.

[78] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011.

[79] Takumi Kobayashi. Higher-order co-occurrence features based on discriminative co-clusters for image classification. In *BMVC*, 2012.

[80] Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 2001.

[81] Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *ICML*, 2009.

[82] Pedro F. Felzenszwalb, Ross B. Girshick, and David Mcallester. Cascade object detection with deformable part models. In *CVPR*, 2010.

[83] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[84] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. Discovering important people and objects for egocentric video summarization. In *CVPR*, pages 1346–1353, 2012.

[85] Yin Li, Alireza Fathi, and James M. Rehg. Learning to predict gaze in egocentric video. In *ICCV*, 2013.

[86] *Microsoft Corp.* http://www.xbox.com/en-US/kinect.

[87] *PrimeSense Corp.* http://www.primesense.com/.

[88] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[89] Sho Ikemura and Hironobu Fujiyoshi. Real-time human detection using relational depth similarity features. In *The Tenth Asian Conference on Computer Vision*, 2010.

[90] L. Xia, C.-C. Chen, and J. K. Aggarwal. Human detection using depth information by kinect. In *Workshop on Human Activity Understanding from 3D Data in conjunction with IEEE Conference on Computer Vision and Pattern Recognition (HAU3D)*, 2011.

[91] L. Bo, X. Ren, and D. Fox. Depth kernel descriptors for object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[92] L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[93] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Sparse distance learning for object recognition combining rgb and depth information. In *International Conference on Robotics and Automation*, 2011.

[94] Hui Ding, Fabien Moutarde, and Ayet Shaiek. 3d object recognition and person facial identification using time-averaged single-views from time-of-flight 3d depth-camera. In *Eurographics Workshop on 3D Object Retrieval*, 2010.

[95] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, 2012.

[96] Jiang Wang, Zicheng Liu, Jan Chorowski, Zhuoyuan Chen, and Ying Wu. Robust 3d action recognition with random occupancy patterns. In *ECCV*, 2012.

[97] David G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, 1999.

[98] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[99] Timo Ahonen, Abdenour Hadid, and Matti Pietikinen. Face recognition with local binary patterns. In *ECCV*, 2004.

[100] B C Vemuri, A Mitiche, and J K Aggarwal. Curvature-based representation of objects from range data. *Image Vision Comput.*, 4:107–114, May 1986.

[101] Bikash Sabata, Farshid Arman, and J. K. Aggarwal. Segmentation of 3d range images using pyramidal data structures. *CVGIP: Image Underst.*, 57:373–387, May 1993.

[102] Youding Zhu and Kikuo Fujimura. 3d head pose estimation with optical flow and depth constraints. *3D Digital Imaging and Modeling*, 2003.

[103] Andreas Ess, Bastian Leibe, and Luc Van Gool. Depth and appearance for mobile scene analysis. In *IEEE International Conference on Computer Vision*, 2007.

[104] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *TPAMI*, 1999.

[105] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–109, 1943.

[106] Qin Cai, David Gallup, Cha Zhang, and Zhengyou Zhang. 3d deformable face tracking with a commodity depth camera. In *European Conference on Computer Vision*, 2010.

[107] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgbd mapping: Using depth cameras for dense 3d modeling of indoor environments. In *RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 2010.

[108] Hao Du, Peter Henry, Xiaofeng Ren, Marvin Cheng, Dan B Goldman, Steven M. Seitz, and Dieter Fox. Interactive 3d modeling of indoor environments with a consumer depth camera. In *ACM International Conference on Ubiquitous Computing*, 2011.

[109] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d object discovery via multi-scene analysis. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[110] Kai Yu and Tong Zhang. Improved local coordinate coding using local tangents. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010.

[111] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In *Advances in Neural Information Processing Systems 22*, pages 2223–2231. 2009.

[112] Dong Xu and Wenli Xu. Description and recognition of object contours using arc length and tangent orientation. *Pattern Recognition Letters*, pages 855–864, 2005.

[113] Xiaoyu Wang, Tony X. Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *IEEE International Conference on Computer Vision*, 2009.

[114] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3-d object dataset: Putting the kinect to work. In *ICCV Workshops*, 2011.

[115] Tao Wang, Xuming He, and Nick Barnes. Learning structured hough voting for joint object detection and occlusion reasoning. In *CVPR*, 2013.

[116] B. Kim, S. Xu, and S. Savarese. Accurate localization of 3d objects from rgb-d data using segmentation hypotheses. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2013.

[117] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[118] M.-C. Jung, Y.-C. Shin, and S. N. Srihari. Multifont classification using typographical attributes. In *Proc. of Intl. Conf. on Document Analysis and Recognition*, 1999.

[119] Y. Zhu, T. Tan, and Y. Wang. Font recognition based on global texture analysis. *PAMI*, 2001.

[120] H. Ma and D. Doermann. Gabor filter based multi-class classifier for scanned document images. In *Proc Intl Conf on Document Analysis and Recognition*, 2003.

[121] H.-M. Sun. Multi-linguistic optical font recognition using stroke templates. In *ICPR*, 2006.

[122] R. Ramanathan, L. Thaneshwaran, and V. Viknesh. A novel technique for english font recognition using support vector machines. In *Proc Intl Conf on Advances in Recent Technologies in Communications and Computing*, 2009.

[123] Martin Solli and Reiner Lenz. Fyfont: find-your-font in large font databases. In *SCIA*, 2007.

[124] C. Aviles-Cruz, R.Rangel-Kuoppa, M. Reyes-Ayala, A. Andrade-Gonzalez, and Rafael Escarela-Perez. High-order statistical texture analysisfont recognition applied. *Pattern Recognition Letter*, 2005.

[125] A. Zramdini and R. Ingold. Optical font recognition using typographical features. *PAMI*, 1998.

[126] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *ICCV*, 2013.

[127] *Google Tesseract-OCR Eigine*. https://code.google.com/p/tesseract-ocr/.

[128] X. Zhou, K. Yu, T. Zhang, and T. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010.

[129] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[130] T. Ahonen, A. Hadid, and M. Pietikinen. Face description with local binary patterns: Application to face recognition. *PAMI*, 2006.

[131] B. Yao, G. Bradski, and L. Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *CVPR*, 2012.

[132] S. Yang, L. Bo, J. Wang, and L. G. Shapiro. Unsupervised template learning for fine-grained object recognition. In *NIPS*, 2012.

[133] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large-scale image classification: generalizing to new classes at near-zero cost. In *ECCV*, 2012.

[134] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[135] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[136] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[137] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[138] R. Farrell, O. Oza, N. Zhang, V. Morariu, T. Darrell, and L. Davis. Birdlets: subordinate categorization using volumetric primitives and pose-normalized apparances. In *ICCV*, 2011.

[139] S. Yan, X. Zhou, M. Liu, M. Hasegawa-Johnson, and T. S. Huang. Regression from patch-kernel. In *CVPR*, 2008.

[140] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classificatio. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[141] Fatih Porikli and Tekin Kocak. Robust license plate detection using covariance descriptor in a neural network framework. In *AVSS*, 2006.

[142] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 2002.

[143] Samy Bengio, Jason Weston, and David Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010.

[144] Jia Deng, Sanjeev Satheesh, Alexander C. Berg, and Fei-Fei Li. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, 2011.

[145] Baoyuan Liu, Fereshteh Sadeghi, Marshall Tappen, Ohad Shamir, and Ce Liu. Probabilistic label trees for efficient large scale image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[146] A. O. Hatch, S. Kajarekar, and A. Stolcke. Within-class covariance normalization for svm-based speaker recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2006.

VITA

I was born in Zhumadian, a city of Henan province, which locates right in the middle of China. After the joyful time as a teenager, I joined Beijing University of Posts and Telecommunications (BUPT) as an undergraduate student. I had a very happy time there and met some very good friends, sharing the same habits and goals. In year 2009, I obtained a B.S. in Electronic and Information Engineering from BUPT and then I was accepted by University of Missouri as a PhD student. I got a M.S. in Electrical and Computer Engineering in 2011 and continued my PhD study at Mizzou till now. Under supervision by Dr. Tony X. Han, I have been doing research in computer vision, with special interest in object detection, fine-grained recognition and large-scale image classification. We also did some research related to machine learning, including model adaptation, local learning and sparse representation.

After my graduation, I will go to a research institute to continue my research. It is so exciting that I could continue to put my passion into computer vision research and try my best to help the machine understand the world. I hope I could develop some algorithms to benefit everyone's life in the future.