

**PARTIALLY OCCLUDED OBJECT DETECTION**

by

**Di Sun**

B.S. in Electrical Engineering, Beijing Jiaotong University, Beijing, China, 2014

Submitted to the Graduate Faculty of  
Swanson School of Engineering in partial fulfillment  
of the requirements for the degree of  
Master of Science

University of Pittsburgh

2016

UNIVERSITY OF PITTSBURGH  
SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Di Sun

It was defended on

March 29, 2016

and approved by

Chen, Yiran, PhD, Associate Professor, Department of Electrical and Computer Engineering

Li, Ching-Chung, PhD, Professor, Department of Electrical and Computer Engineering

Li, Hai, PhD, Associate Professor, Department of Electrical and Computer Engineering

Thesis Advisor:

Chen, Yiran, PhD, Associate Professor, Department of Electrical and Computer Engineering

Copyright © by Di Sun

2016

# **PARTIALLY OCCLUDED OBJECT DETECTION**

Di Sun, M.S.

University of Pittsburgh, 2016

Object detection is a fairly important field in computer vision and image processing, and there are some domains that have been well researched, such like face detection and pedestrian detection, while these detections are based on situation that there are no shelters on objects in images or videos. But most objects are partially occluded in real world. And this becomes one major obstacle in object detection. In this paper, I utilize a state-of-art model, Deformable Part Model (DPM), which present a mixture model contains multi-scale deformable parts. And attach visibility flags to both cells and part of it. A visibility flag is used to indicate whether a specific cell or part is visible or occluded. By removing the contribution of occluded cells and parts to the whole detection score, the performance of partially occluded object detection can be significantly improved. To get the value of visibility flags, I use a mathematical method called Alternating Direction Method of Multipliers (ADMM). The experiment shows that by using this model, the detection accuracy of partially occluded object is improved.

## TABLE OF CONTENTS

<b>TABLE OF CONTENTS</b> .....	<b>V</b>
<b>LIST OF TABLES</b> .....	<b>VII</b>
<b>LIST OF FIGURES</b> .....	<b>VIII</b>
<b>1.0 INTRODUCTION</b> .....	<b>1</b>
<b>2.0 RELATED WORK</b> .....	<b>6</b>
<b>3.0 PROBLEM BACKGROUND</b> .....	<b>8</b>
<b>3.1 HISTOGRAMS OF ORIENTED GRADIENTS</b> .....	<b>8</b>
<b>3.1.1 Implementation of Algorithm</b> .....	<b>9</b>
<b>3.2 ALTERNATING DIRECTION METHOD OF MULTIPLIER</b> .....	<b>10</b>
<b>3.2.1 Implementation of Algorithm</b> .....	<b>10</b>
<b>4.0 PART BASED MODEL</b> .....	<b>12</b>
<b>4.1 MODEL</b> .....	<b>12</b>
<b>4.1.1 Deformable Part Models</b> .....	<b>14</b>
<b>4.1.2 Matching</b> .....	<b>15</b>
<b>4.1.3 Mixture Model</b> .....	<b>16</b>
<b>4.2 LSVM</b> .....	<b>18</b>
<b>4.2.1 Semi-convexity</b> .....	<b>18</b>
<b>4.2.2 Optimization</b> .....	<b>19</b>

<b>4.3</b>	<b>TRAINING</b> .....	<b>20</b>
<b>4.3.1</b>	<b>Learning Parameter</b> .....	<b>20</b>
<b>4.3.2</b>	<b>Initialization</b> .....	<b>23</b>
<b>5.0</b>	<b>OCCLUSION HANDLING</b> .....	<b>25</b>
<b>5.1</b>	<b>MODEL</b> .....	<b>25</b>
<b>5.2</b>	<b>IMPLEMENTATION</b> .....	<b>28</b>
<b>6.0</b>	<b>EXPERIMENTS</b> .....	<b>29</b>
<b>6.1</b>	<b>DETECTION OF FULLY VISIBLE OBJECTS</b> .....	<b>29</b>
<b>6.2</b>	<b>DETECTION OF PARTIALLY OCCLUDED OBJECTS</b> .....	<b>32</b>
<b>7.0</b>	<b>CONCLUSION</b> .....	<b>38</b>
	<b>BIBLIOGRAPHY</b> .....	<b>39</b>

## LIST OF TABLES

Table 1. Comparison of occlusion handling model and DPM.....	36
--	----

## LIST OF FIGURES

Figure 1. Person detection model.....	2
Figure 2. Two components car detection model.....	5
Figure 3. Image pyramid (left) and its corresponding feature pyramid (right).....	13
Figure 4. Star model for bottle.....	14
Figure 5. Procedure of matching.....	17
Figure 6. Pseudo code of training process .....	22
Figure 7. Model for bicycle .....	24
Figure 8. Some model examples trained on PASCAL 2007 dataset .....	30
Figure 9. Some detection result on PASCAL test dataset .....	31
Figure 10. Alpha from 1 to 1.2 .....	33
Figure 11. Alpha from 1.3 to 1.5 .....	34
Figure 12. Alpha from 1.6 to 1.8 .....	35
Figure 13. Some detection results:.....	37

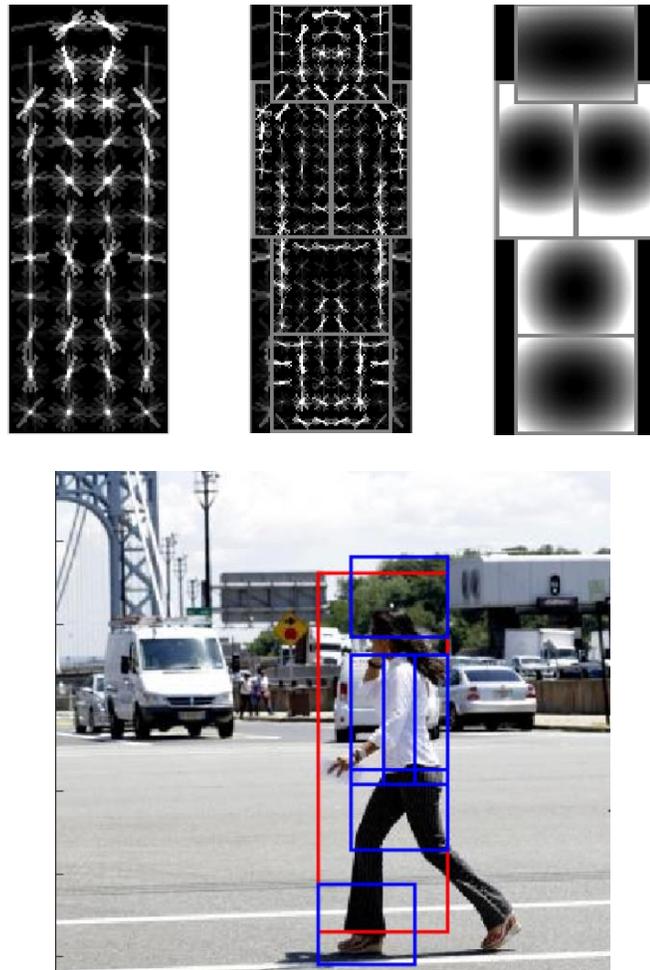
## 1.0 INTRODUCTION

This paper introduces a method for object detection with partially occluded based on a state-of-art model, Deformable part models (DPM) [1]. The problem of detecting and localizing is a fundamental and important challenge in computer vision area. Detecting objects such as pedestrians and cars in static images is a tough problem because these objects, especially pedestrians, can vary significantly from image to image. Besides their poses and shapes, image illumination, complexity of background and viewpoint can also arise the variation. Despite significant effort has been put and encouraging development has been made recent years, most of them are only focus on detecting fully visible objects. When comes with objects with partial occlusion, the detection results are not good enough.

Most popular detection models use method of bounding box, which supposes that all pixels in the bounding box belong to the object. But in real world, not all objects are entirely visible, which means not all pixels support the object. For example, in the Galtech Pedestrian Dataset [22], at least 70% of pedestrian are partially or fully occluded in at least one frame of a video sequence. Dollar et al. [22] show that detection performance for standard methods drops significantly even under partial occlusion, and drastically under heavy occlusion. [6] So a detection model that can learn the visibility of pixels of an object is very important.

Method in this paper is based on a mixture Deformable Part Model, which provides an outstanding framework for the problem of object detection. Deformable part model can

significantly capture the object variations, but using only one is not sufficient. Discriminatively Trained Deformable Part Model [1] uses a mixture of star-structured deformable part models, which contains a “root” filter and a set of part filters to better represent the variations on both pixel level and part level. Figure 1 shows a star-structured model for pedestrian.



**Figure 1.** Person detection model

At a specified position and scale, score of the star-structured model is the sum of responses of root filter and every part filter. The response of each filter is calculated by the dot product of filter weights and image features inside the detection window. In this model, part filters are applied to feature maps twice the resolution of root filters. By doing this, feature descriptors can better represent parts of objects and experiments show that by doing this, the detection performance is improved. However, datasets available now are all weakly labeled, which means there are labels only for root bounding box of an object. So we need a Support Vector Machine for Multiple-Instance Learning (MI-SVM) [17], in the following paper we will call it latent SVM (LSVM), to train the star-structured model with partially labeled dataset. We introduce a latent variable, which is used to indicate the positions of root and every part according to the object. The detection score of an input example  $x$  is calculated by the following form of function,

$$f_{\beta} = \max_{z \in Z(x)} \beta \cdot \Phi(x, z)$$

where  $\beta$  is a vector of model parameters,  $z$  is the vector of latent values, and  $\Phi(x, z)$  is feature vector. In the star-structured model,  $\beta$  is the concatenation of the root filter weights, part filters weights and deformation cost weights,  $\Phi(x, z)$  is the concatenation of feature pyramid image inside root filter and part filters detection windows according to the object. A model for a specified class consists of several star-structured models, each one of which represents a component. Figure 2 shows a model for cars with two components.

I attach latent visibility flags to part filters and every cell of root filter of deformable part model to deal with the detection of partially occluded objects. A visibility flag is a binary variable indicates whether the cell or part of a particular object is occluded or visible, 0 for occluded and 1 for visible. In order to get the value of visibility flags, I use an optimization

method called Alternating Direction Method of Multiplier (ADMM). Since we know that generally occluded parts of an object, especially pedestrian, are on the bottom and tend to cluster together, so the concave function we use to be maximized totally contains four parts: sum of the detection score of root filter and every part filter, a penalty term to encourage occluded parts appear at the bottom of the object, a consistency term to encourage cells concatenating have the same visibility flag, a consistency term to make sure that overlapped cells of root filter's detection window and parts have the same visibility flag. Experiments show that by removing the contribution of occluded cells and parts to final detection score, the performance of partially occluded object detection is significantly improved.

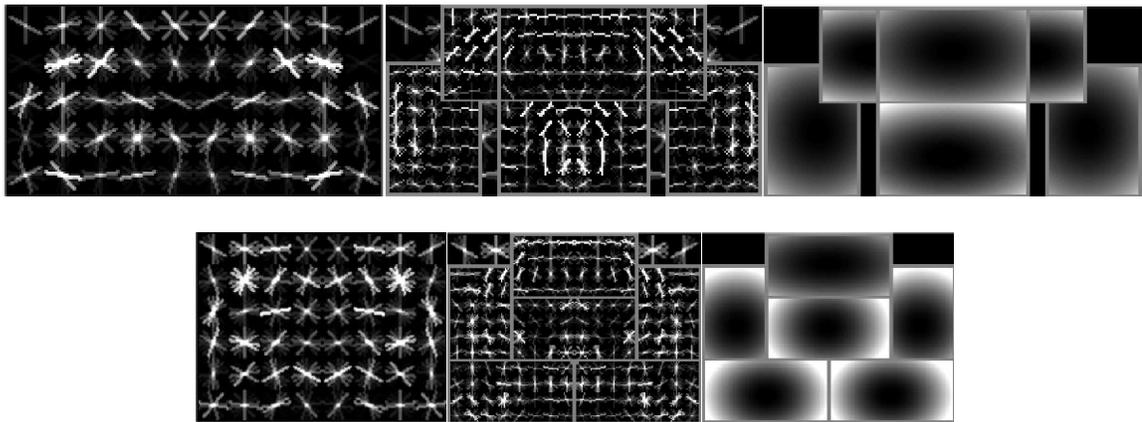
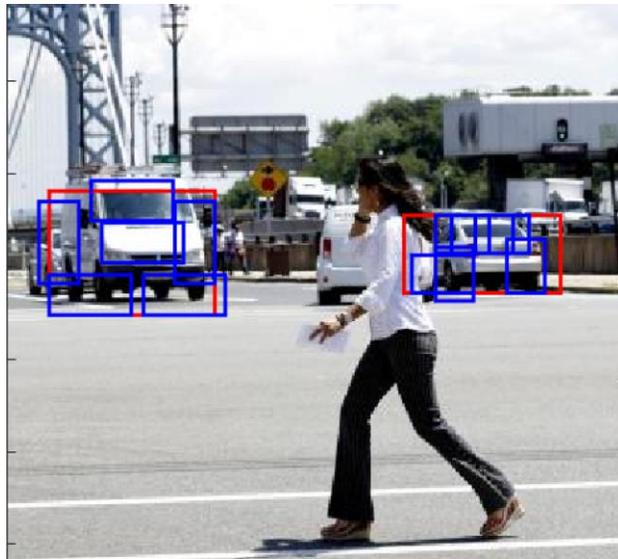


Figure 2. Two components car detection model

## 2.0 RELATED WORK

In recent years, object detection without occlusion has significantly developed, and most of the methods can achieve a considerable accuracy rate. However, occlusion handling is still an obstacle in this area, and a lot of researchers address the problem of partially occluded object detection. Paper [4] combines the Histogram of Oriented Gradients (HOG) with Local Binary Pattern (LBP) to get the feature set. They use the response of each block in HOG feature vector to construct occlusion likelihood maps and classify them with Mean Shift approach. Paper [5] use latent variables to indicate the visibility of every cell, but the model they construct can only be used in the situation that the object is truncated. In real world most objects are occluded by other objects, so this method cannot solve common problems. Paper [6] provides a segmentation-aware object detection model. A latent variable is also attached to every cell in their model to indicate whether the cell belongs to the object. They construct a structural SVM to learn the values of latent variables, which means this model need to be trained with features of both objects being occluded and occluder. This will be time-consuming and the training dataset needed is also relatively large.

Models above are using holistic detectors, while Paper [7], [8], [8], [9] use part-based detectors instead of the whole object detector and attach visibility flags to every part. Like method in this paper, Paper [7] also based on deformable part model and use latent variable to indicate the visibility of every part. They construct a discriminate deep model to learn the

visibility relation shape of overlapping parts. Paper [8] use the contextual and spatial configuration information from deformable part model to train conditional random field (CRF) parameters and use stochastic gradient descent with belief propagation to optimize CRF to get visibility flag value. But all these methods only based on part detectors and ignore the occlusion information on finer cell level.

There are some other approaches like training occlusion specified dataset, which is quite time consuming. Paper [14] provides an approach to train a double person detector to handle person-to-person occlusion, which cannot be handling with in this paper.

### **3.0 PROBLEM BACKGROUND**

#### **3.1 HISTOGRAMS OF ORIENTED GRADIENTS**

A good object detection method must base on a robust feature descriptor, which can greatly represent the object and make it discriminate from complex background. Edge and gradient based descriptors are most widely applied to object detection methods, and among these approaches, grids of Histograms of Oriented Gradient (HOG) descriptors plays a very important role because its outstanding performance.

The basic idea of HOG is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradient or edge directions, even without precise knowledge of the corresponding gradient or edge position [2]. To implement this descriptor, we can divide the image inside detection window into small connected regions, which we called cells, and in each cell, we calculate centered horizontal and vertical gradient of every pixel in the cell and accumulate them together. The combination of these histograms represents the descriptor. Then by creating larger spatial regions (blocks) and accumulating “energy” inside the block, the local histograms can be contrast normalized to get better performance. This normalization will improve the descriptor’s robustness to illumination and shadow changes.

### 3.1.1 Implementation of Algorithm

The Algorithm implementation is consisted of four parts: gradient computation, orientation binning, creating blocks and block normalization.

The first step is to compute centered horizontal and vertical gradient and gradient's orientation and magnitude. The most common method is to apply a centered point discrete derivative mask in horizontal and vertical directions. The two filter kernels are:

$$D_x = [-1 \ 0 \ 1] \quad D_y = [1 \ 0 \ -1]^T$$

The second step is to create cell histogram by casting weighted vote from every pixel, where vote can be the magnitude of gradient or square or square root of it. Histogram bins are evenly spread over 0 to 180 degrees or 0 to 360 degrees, which is depending on whether the gradient is “unsigned” or “signed”. After experiments, Dalal and Triggs [2] found that unsigned gradients use in conjunction with nine histogram bins performs best.

In order to make the descriptor robust to the change of illumination and image contrast, a local normalization should be applied to the gradients inside a block. Since generally these blocks are overlapped, cells in one block may contribute twice or more to the final descriptor. Many normalization factors like  $l_1$  norm and  $l_2$  norm can be used here to do the block normalization. A created feature descriptor can be put into a classifier like linear SVM to train and get the final detector.

## 3.2 ALTERNATING DIRECTION METHOD OF MULTIPLIER

In computer vision and machine learning area, many problems can be finally posed in a framework of convex function optimization. Due to the large size and high complexity of most dataset we needed in this area, a good optimization method becomes critical. Both computing time and storage of large data size should be taken into consideration when choosing an optimization approach. In this paper, I use a method called Alternation Direction Method of Multiplier (ADMM), which is well suited to distributed convex optimization, especially to large-scale problem. This method was developed in the 1970s, with roots in the 1950s, and is closely related to many other algorithms, such as dual decomposition, Douglas-Rachford splitting, Spingran's method of partial inverses, the method of multiplier and others.

### 3.2.1 Implementation of Algorithm

Here I will give a simple overview of ADMM. More details please refer to the [16]. Giving an example of optimizing a convex function  $f(x)$ .

$$\begin{aligned} & \text{minimize } f(x) \\ & \text{subject to } x \in C \end{aligned} \tag{3-1}$$

here  $x$  is input variable that subjects to condition set  $C$ , and  $f(x)$  is a convex function. We can first rewrite the function to the following form,

$$\begin{aligned} & \text{minimize } f(x) + I_C(z) \\ & \text{subject to } x = z \end{aligned} \tag{3-2}$$

where  $I_C(z)$  is the indicator function on  $C$ , which means  $I_C(z) = 0$  for  $z \in C$  and  $I_C(z) = +\infty$  for  $z \notin C$ . Then we can get the augment Lagrangian function for this problem as,

$$L_\rho(x, z, u) = f(x) + I_C(z) + \frac{\rho}{2} \|x - z + u\|_2^2 \quad (3-3)$$

Where  $u$  is a scaled dual variable associated with the constraint  $x = z$  and  $\rho > 0$  is penalty parameter. In each iteration, we perform alternation minimization of the augment Lagrangian function over  $x$  and  $z$  and update value following steps as below,

$$\begin{aligned} x^{k+1} &:= \operatorname{argmin} \left\{ f(x) + \frac{\rho}{2} \|x - z^k + u^k\|_2^2 \right\} \\ z^{k+1} &:= \Pi_C(x^{k+1} + u^k) \\ u^{k+1} &:= u^k + (x^{k+1} - z^{k+1}) \end{aligned} \quad (3-4)$$

When updating  $x$ , we fix  $z$  and  $u$ . Similarly, when updating  $z$ ,  $x$  and  $u$  are fixed and updating  $u$  while  $x$  and  $z$  are fixed.  $\Pi_C$  denotes the Euclidean projection onto constrain set  $C$ .

A typical criterion to stop the iteration is,

$$\|e_p^k\|_2 \leq \epsilon^{pri}, \|e_d^k\|_2 \leq \epsilon^{dual} \quad (3-5)$$

Where  $e_p^k$  and  $e_d^k$  are primal and dual residuals  $\epsilon^{pri}$  and  $\epsilon^{dual}$  are tolerances can be set via an absolute plus relative criterion.

$$\begin{aligned} e_p^k &= (x^k - z^k), e_d^k = \rho(z^k - z^{k-1}) \\ \epsilon^{pri} &= \sqrt{n}\epsilon^{abs} + \epsilon^{rel} \max\{\|x^k\|_2, \|z^k\|_2\} \\ \epsilon^{dual} &= \sqrt{n}\epsilon^{abs} + \epsilon^{rel}\rho\|u^k\|_2 \end{aligned} \quad (3-6)$$

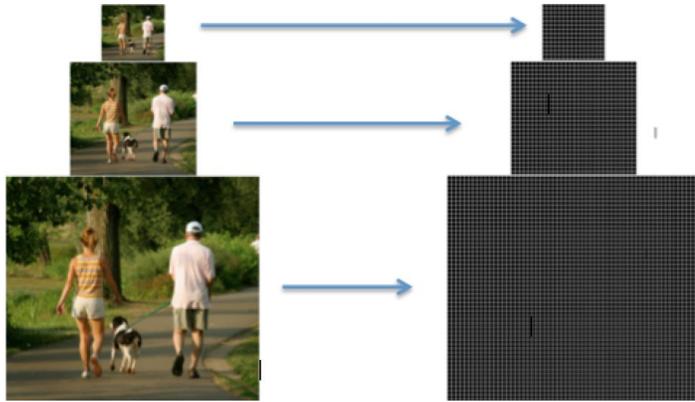
## 4.0 PART BASED MODEL

In this section, I will mainly introduce the mixture deformable part based model that the occlusion handling model based on. This section includes four parts: introduction to models, features descriptor, the utilization of latent SVM and training process.

### 4.1 MODEL

The basic idea of models is applying linear filters to dense feature map of the input image. A linear filter is an array of  $d$ -dimensional weight vectors and a feature maps present an array of feature vectors computed inside a sub-window. Here the model use feature maps based on HOG features introduced in [2], but following discussion is independent of the form of feature descriptor.

The response (score) of a filter  $F$  on a feature map  $G$  is calculated by summing the dot product of the filter and a sub-window feature weights. Since there may be not only one object in the whole image and the size may various greatly and we would like to detect all of them, changing size of detection window is needed here. But instead of changing the size of detection window, we prefer to calculate a feature pyramid, which is more suitable for this model. Feature pyramid is implemented by the iteration of smoothing and subsampling on the original image. Then compute feature maps on every level of the pyramid. Figure 3 explains the construction.



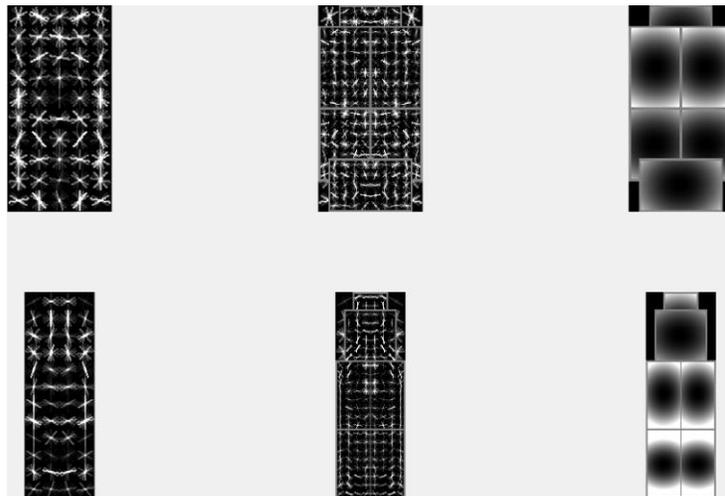
**Figure 3.** Image pyramid (left) and its corresponding feature pyramid (right)

As we discussed before, in this model, part filters will be applied to the feature maps twice resolution as the root filter. So here they introduce a parameter  $\lambda$  to indicate the number of levels in an octave of feature pyramid. That means image resolution will increase twice as much when pyramid level go down  $\lambda$  levels. Experiments show that by doing this, the detection rate can be improved.

The advantage of deformable part model is that it can better present the variation of object but single filter is still not good enough. In this model, mostly two components are included in one object model. Suppose that there is a filter  $F$  with the size  $w \times h$  and is applied to a feature pyramid  $G$ . Let  $\Phi(G, p)$  denote the feature vector obtained from concatenation specific  $w \times h$  sub-window of feature pyramid. Here  $p = (x, y, l)$  is used to indicate the position  $(x, y)$  in the  $l$ -th level of pyramid. Let  $F'$  be the vector obtained by concatenating the weight vectors in  $F$ . All the concatenation is done in row-major order. Then the score of the filter  $F$  applied to  $G$  is  $F' \cdot \Phi(G, p)$ .

### 4.1.1 Deformable Part Models

This model analyzes features both on cell-based level and part-based level. A star model is consist of a root filter coarsely cover the entire object and a set of part filters cover small parts of the object. Figure 4 shows a star model for a bottle. A root filter is first applied to the feature pyramid as a detector and then part filters will be applied to the feature map  $\lambda$  level down in the pyramid according to root filter. By doing this, feature map detected by part filter is twice the resolution as that detected by root filter. This step is very important. Higher resolution the image has, more detailed feature can be captured. For example, in the process of detecting a face, root filter can capture edges like boundary of face while part filters can capture features like eyes and mouth. Experiments also show that applying part filters to higher resolution image can get much higher accuracy.



**Figure 4.** Star model for bottle

Every model for an object with  $n$  parts is defined by a  $(n + 2)$  tuple  $(F_0, P_1, \dots, P_n, b)$  where  $F_0$  is the root filter,  $P_1$  to  $P_n$  are models for part filters,  $b$  is the bias term. Every part model is defined by a three tuple  $(F_i, v_i, d_i)$  where  $F_i$  is the filter for the  $i$ -th part,  $v_i$  is a two value vector indicate the “anchor” position according to the position of root filter and  $d_i$  defines the deformation cost for every placement of the part relative to the anchor position.

Here they introduce a latent vector  $z = (P_0, \dots, P_n)$  to indicate the position of every filter where  $P_i = (x_i, y_i, l_i)$  specifies the position and level in feature pyramid. Since we apply part filters to feature map twice the resolution of root filter, we can get the relation that  $l_i = l_0 - \lambda$ .

Give a object hypothesis  $z = (P_0, \dots, P_n)$ , the score of it can be computed by the sum of each filter response minus the deformation cost depends on the positions of parts relative to the root filter plus the bias term,

$$score(P_0, \dots, P_n) = \sum_{i=0}^n F'_i \cdot \phi(G, p_i) - \sum_{i=0}^n d_i \cdot \phi_d(dx_i, dy_i) + b \quad (4-1)$$

Deformation cost term is a four values vector specifies a quadratic function of the displacement. Bias term is introduced to make multiple models able to mixture together. As we have discussed, generally every object model is a mixture of two models.

Finally the computed score should be put into a latent SVM framework to train detector parameters, so we rewrite the model parameters as  $\beta = (F'_0, \dots, F'_n, d_1, \dots, d_n, b)$  and feature vector as  $\psi(G, z) = (\phi(G, p_0), \dots, \phi(G, p_n), -\phi_d(dx_1, dy_1), \dots, \phi_d(dx_n, dy_n), 1)$ , which can illustrate the relation between this model and linear classifier.

### 4.1.2 Matching

The overall score of a detection window is calculated according to the best placements  $(P_0, \dots, P_n)$  for parts,

$$score(p_0) = \max_{P_1, \dots, P_n} score(P_0, \dots, P_n) \quad (4-2)$$

Then we can get high-scoring root locations define the possible location of the whole object and best placements for parts according to it. Root filter is applied as a sliding window detector to every level and position of the feature pyramid, so we can detect multiple instances of the object with different location can scale. And the score of every root location defines the score of every detection window.

Dynamic programming and generalized distance transform [25] are used to calculate the best placements for parts according to the location of root. The overall score can be calculated by summing the root filter response at the specific level and position and shifted versions of transformed and subsampled part responses,

$$score(x_0, y_0, l_0) = R_{0, l_0}(x_0, y_0) + \sum_{i=0}^n D_{i, l_0 - \lambda}(2(x_0, y_0) + v_i) + b \quad (4-3)$$

After getting a high-scoring root location  $(x_0, y_0, l_0)$ , we can find the relative part locations by optimal displacement  $P_{i, l_0 - \lambda}(2(x_0, y_0) + v_i)$ . Figure 5 shows the procedure of matching.

### 4.1.3 Mixture Model

As we discussed before, use only one model to represent an object is not enough. Here we introduce the mixture model, which is consisted of several components. A mixture model with  $m$  components is defined by a  $m$  tuple,  $M = (M_1, \dots, M_m)$ , where  $M_i$  is the model for the  $i$ -th component. Every model includes a root filter and a set of part filters. For example, when detecting people, we may need a model to represent people standing up and another one for people sitting or crouching down. For cars detecting, we may need two models to represent cars coming from right hand and left hand separately.

For a single component model, the score of a hypothesis is the dot product between the vector of model weight and the feature vector inside the detection window. For an m components model, the model weight vector is concatenation of weight vectors for every model. To detect an object with mixture model, we can use the process discussed above to find high-scoring root location and relative parts locations independently for each component

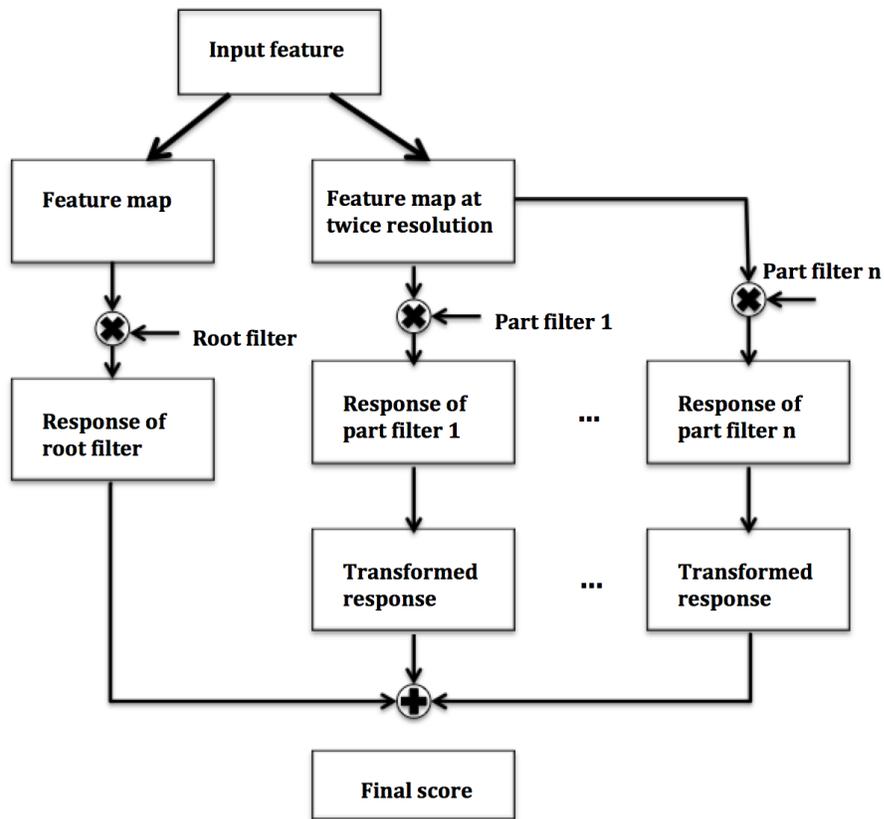


Figure 5. Procedure of matching

## 4.2 LSVM

As we have discussed before, filter response is calculated by dot product of filter weight vector and feature vector as the following form,

$$f_{\beta}(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z) \quad (4-4)$$

where  $\beta$  is the vector of filter weight and  $z$  are latent values. We have a set of possible latent values  $Z(x)$  for an input  $x$ . Then we can set a threshold the to the score to label the input  $x$ . Like training classical SVM, we can train  $\beta$  from labeled data set  $D = ((x_1, y_1), \dots, (x_n, y_n))$ , where  $y_i \in \{-1, 1\}$  is the label of  $x_i$ , by minimizing objective function,

$$L_D(\beta) = \frac{1}{2} \|\beta\|^2 + c \sum_{i=1}^n \max(0, 1 - y_i f_{\beta}(x_i)) \quad (4-5)$$

where  $\max(0, 1 - y_i f_{\beta}(x_i))$  is the hinge loss and  $C$  is a constant value used to control the weight of regularization term. If there is only one possible latent value for an input  $x$ , then the problem is simplified to a linear SVM training problem. So linear SVM is a special case of latent SVM.

### 4.2.1 Semi-convexity

Training a linear SVM is convex optimization problem because  $f_{\beta}(x)$  is linear in  $\beta$  and hinge loss is convex as maximum of two convex functions. However training a latent SVM is a non-convex optimization problem. But in some specified cases, it can be treated as convex. If there is single latent value possible for positive examples, then the problem becomes linear SVM training problem.

As we know, the maximum of a set of convex functions is also a convex function. From function above we can see that  $f_{\beta}(x)$  is a maximum of a function, which is linear in  $\beta$ . So  $f_{\beta}(x)$

is convex in  $\beta$  and hinge loss,  $\max(0, 1 - y_i f_\beta(x_i))$ , is also convex when  $y_i = -1$ . That means the loss function is convex in  $\beta$  for negative examples, which is called semi-convexity.

Generally speaking, hinge loss in latent SVM for positive examples is not convex because it is a maximum of a convex function and a concave function.

## 4.2.2 Optimization

Here we introduce a value  $Z_p$  to specify the latent value for each positive example in training set  $D$ . Let  $D(Z_p)$  is a dataset derived from  $D$  by restricting latent values  $Z_p$  for positive examples. That is we set  $Z(x_i) = \{z_i\}$  for a positive example according to  $Z_p$ . Then we can rewrite the loss function of latent SVM as,

$$L_D(\beta) = \min_{Z_p} L_D(\beta, Z_p) \quad (4-6)$$

where  $L_D(\beta) \leq L_D(\beta, Z_p)$ . So we can train latent SVM by minimizing  $L_D(\beta, Z_p)$ .

We implement the minimization of  $L_D(\beta, Z_p)$  by using a “coordinate descent” approach, which iteratively update variables in a function while set others constant. The procedure is shown below,

- (1) Optimize  $Z_p$ : Set  $\beta$  as constant and minimize  $L_D$  over  $Z_p$  to get highest scoring latent value for each positive example.
- (2) Optimize  $\beta$ : Set  $Z_p$  as constant and minimize  $L_D$  over  $\beta$ .

Follow the steps above repeatedly until converge. After convergence we will get a relatively strong optimum because step 1 search all possible latent values to each positive example and find the highest scoring one while step 2 search a large set of values to get best model weights. Here

the initialization of  $\beta$  is important because bad initialization of  $\beta$  may lead to unreasonable  $Z_p$  in step1 and then lead to bad model finally.

The semi-convex property is very important here. In step 1, we specify the latent value to all positive examples and then in step 2, latent SVM becomes linear for positive examples and convex for negative examples.

## 4.3 TRAINING

Most training data available are images labeled with a bounding box around an object of interest. Here we use the PASCAL datasets, which contains thousands of labeled images. But this dataset is weakly labeled because it only has labeled bounding box for the whole object but no components or parts label. The parameter of latent SVM and best latent values are learned by the procedure of coordinate descent approach introduced in 4.2.2.

### 4.3.1 Learning Parameter

Assume for an object class  $c$ , the training dataset  $D$  is consisted of positive examples  $P$  with labeled bounding boxes around objects of interest in every image and negative examples  $N$ .  $P$  is a set of pairs  $(I, B)$  where  $I$  is a image and  $B$  are the bounding boxes for objects of interest in image  $I$ .

$M$  is a mixture model with weight parameters vector  $\beta$ . We use coordinate descent approach to learn the values of  $\beta$  with positive examples from  $P$  and backgrounds from  $N$ . We compute feature pyramid  $G(x)$  for every image  $(x, y)$  in dataset  $D$  where  $x$  and  $y$  is the label for

it. Latent value  $z \in Z(x)$  specify the root location and relative part locations. We define  $\phi(x, z)$  as the feature vector. Then  $\beta \cdot \phi(x, z)$  is the score of hypothesis  $z$  for  $M$  on  $G(x)$ .

We want the detection score high in area of bounding box  $B$  for image  $I$ . We define a positive example  $x$  for every  $(I, B) \in P$  and the detection window of root filter specified by  $Z(x)$  should overlap  $B$  at least 50%. Here we treat the root location as a part of latent value, which is helpful for compensating for noisy bounding boxes in  $P$ . On the other hand, we want the detection score low when applied on negative examples.

The pseudo code of training process is shown in Figure 6. The first loop from line 2 to 15 implements fixed iteration of coordinate descent. Lines 3 to 5 do the positive example relabeling to optimize  $z$  while lines 7 to 14 optimize  $\beta$ . Data-mining algorithm is used to remove hard negative examples for reducing the data size. In every iteration, highest scoring hypothesis is stored in  $F_p$  while hard negative examples are all collected into  $F_n$ . Then training  $\beta$  and remove easy feature vectors from  $F_n$ . The process break when we reach the memory limitation.

Positive examples  $P = \{(I_1, B_1), \dots, (I_n, B_n)\}$   
 Negative examples  $N = \{J_1, \dots, J_m\}$   
 Initial model  $\beta$

```

1  $F_n = \phi$ 
2 for  $relabel = 1$  to  $num-relabel$  do
3    $F_p = \phi$ 
4   for  $i = 1$  to  $n$  do
5     Add  $detect-best(\beta, I_i, B_i)$  to  $F_p$ 
6   end
7   for  $datamine = 1$  to  $num-datamine$  do
8     for  $j = 1$  to  $m$  do
9       if  $|F_n| \geq memory-limit$  then break
10      Add  $detect-all(\beta, J_j, -(1+\delta))$  to  $F_n$ 
11     end
12      $\beta = \underline{gradient-descent}(F_n \cup F_p)$ 
13     Remove  $(i, v)$  with  $\beta \cdot v < -(1+\delta)$  from  $F_n$ 
14   end
15 end

```

**Figure 6.** Pseudo code of training process

The function  $detect - best(\beta, I, B)$  is used to find the highest scoring hypothesis that specifies a root filter overlap  $B$  on a large area. The  $detect - all(\beta, I, th)$  is used to find the best hypothesis for a specified root and select ones whose score above  $th$ . The function  $gradient - descent(F)$  is used to train  $\beta$  with hypothesis values.

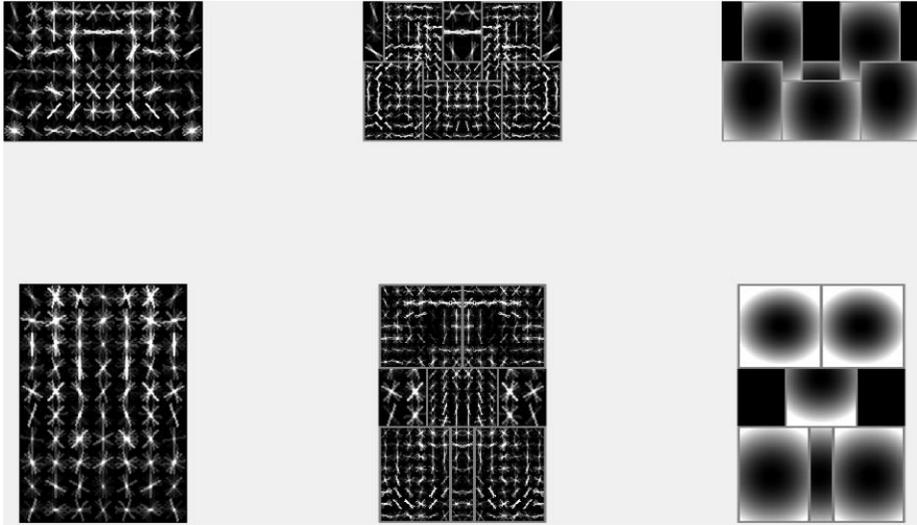
### 4.3.2 Initialization

The coordinate descent algorithm for latent SVM is sensitive to the initialization of model parameters. So here we discuss the procedure of initialization and training.

For training a mixture model with  $m$  components, we first sort the bounding boxes by their aspect ratio and send them into  $m$  groups. Every group has equal size bounding boxes and we train  $m$  different root filters, one for each group. We use standard SVM to train these filters on positive dataset  $P$  with no latent variables. Figure 7 shows two trained models for bicycle.

After we got initialed root filters, we need to combine them together to form a mixture model. This mixture model has no parts information now. Then we train this mixture model following the pseudo code shown above on the whole dataset  $D$ . In this case, the latent information only contains the information of component label and root location.

Next step is to initialize and train part filters. We fix the number of parts for each component as six and we assume parts should be placed on high-energy regions of root. We use a pool of part-shape rectangular to cover these regions and once a region is covered on the root, it is set to be zero. Then we look for a next high-energy region until six parts are chosen. The deformation parameters are initialed as  $d = (0,0,.1,.1)$ .



**Figure 7.** Model for bicycle

## 5.0 OCCLUSION HANDLING

In this section, I will mainly introduce an occlusion handling approach based on the mixture deformable part models discussed in section 4.0 and the idea is mainly based on paper [26]. This section includes two parts: introduction of model and implementation of model.

### 5.1 MODEL

As we discussed before, mixture deformable part model introduced in section 4.0 is consisted of a root filter and a set of part filters. The model can be define as  $M = (F_0, P_1, \dots, P_b, b)$  where  $F_0$  is the root filter,  $P_i$  is the model for the  $i$ -th part filter and  $b$  is the bias term. Part model is defined as  $P_i = (F_i, v_i, d_i)$  where  $F_i$  is the filter for the  $i$ -th part and  $d_i$  is the deformation coefficients, which we will use here to calculate part filter responses.

Given a detection window at position  $p$  and level  $l$ , the response of root filter is calculated as,

$$R(p, l) = F_0^T \cdot G(p, l) + b \quad (5-1)$$

Where  $G(p, l)$  is the concatenation of features computed on cell level inside the detection box and  $b$  is the bias term. Since this is a linear operation, we can rewrite the filter response as a summation of responses on cell level. Then the root detector score can be written as,

$$R_0(p, l) = \sum_{c=1}^C [(F_0^c)^T \cdot G^c(p, l) + b^c] = \sum_{c=1}^C R_0^c(p, l) \quad (5-2)$$

Where  $C$  is the number of cells and  $b = \sum b^c$ . Here we suppose  $R_p(p, l)$  is the filter response of  $p$ -th part according to the root position and as we know, part filter response is linear on part level. Then the total score of a detection window, which is the summation of root filter response and every part filter response can be calculated as,

$$R(p, l) = \sum_{c=1}^C R_0^c(l, p) + \sum_{p=1}^P R_p(p, l) \quad (5-3)$$

When an object is partially occluded, the occlusion parts may have negative effect to the filter response, which will decrease the detection score. So what we need to do here is to remove the contribution of occluded part to both root filter response and part filter response. So I attach a visibility flag to every cell and part to indicate that whether the cell or part is occluded or not. The visibility flag is a binary variable that 0 indicates occluded and 1 indicates visible. By doing this, we will only aggregate the filter responses of visible cells and parts.

The problem of finding put the values of visibility flags can be treated as a optimization problem on the following score function,

$$\widehat{v}_0, \{\widehat{v}_p\}_{p=1}^P = \arg \max_{v_0, \{v_p\}} (\sum_{c=1}^C R_0^c v_0^c + \sum_{p=1}^P R_p v_p) \quad (5-4)$$

Where  $\widehat{v}_0 = [v_0^0, \dots, v_0^C]$  and  $\{\widehat{v}_p\}_{p=1}^P$  are the visibility flags for cells of root and parts. Here we omit the parameter  $p$  and  $l$  for simplicity and the function only focus on one detection window. The maximization result of the score function above will simply set visibility flag 0 for cells and parts whose filter response below zero and 1 for others. But occluded regions are not likely to be distributed evenly on the object. There some occlusion “rules” we need to follow.

First of all, the object is usually occluded by another object, which means occlusion likely to occur on a continuous region. What’s more, the root cells and parts that are overlapped

should have the same visibility flag value. To account these spatial relations, we can add two penalty terms, cell-to-cell consistency and part-to-part consistency, to the score function. Then the function takes the following form,

$$R_{occ} = \max_{v_0, \{v_p\}_{p=1}^P} \sum_{c=1}^C R_0^c v_0^c + \sum_{p=1}^P R_p v_p - \gamma \sum_{c_i \sim c_j} |v_0^{c_i} - v_0^{c_j}| - \gamma \sum_{c_i \sim p_j} |v_0^{c_i} - v_0^{p_j}| \quad (5-5)$$

Where  $c_i$  and  $c_j$  are adjacent cells,  $c_i$  and  $p_j$  are overlapped cell and part,  $\gamma$  is the regularization parameter.

Since the dataset I will use to test the model is a pedestrian detection dataset, in street scene, lower part of the pedestrian are more likely to be occluded than the upper part. So here I also introduce two penalty terms to control the position of the occlusion region. Then the score function can be written as,

$$R_{occ} = \max_{v_0, \{v_p\}_{p=1}^P} \sum_{c=1}^C [R_0^c v_0^c - \alpha \lambda_0^c (1 - v_0^c)] + \sum_{p=1}^P [R_p v_p - \beta \lambda_p (1 - v_p)] - \gamma \sum_{c_i \sim c_j} |v_0^{c_i} - v_0^{c_j}| - \gamma \sum_{c_i \sim p_j} |v_0^{c_i} - v_0^{p_j}| \quad (5-6)$$

Where  $\lambda_0^c$  and  $\lambda_p$  are penalties when visibility flags of cells and parts are set occluded,  $\alpha$  and  $\beta$  are weights of the penalty term. The value of  $\lambda_0^c$  and  $\lambda_p$  should be proportional to the height from the bottom of image to the cell or part. When the cell or part is in upper area of the image, the penalty of setting their visibility flags to 0 should be large. It can be calculated follow the sigmoid function as,

$$\lambda = \frac{1}{2} \left[ \frac{1}{1 + e^{-\tau(h-h_0/2)}} + \frac{1}{2} \right] \quad (5-7)$$

where  $h$  is the height of cell or part from the bottom of image and  $\tau$  controls the step size.

## 5.2 IMPLEMENTATION

There are a large number of detection windows in an image, so computing visibility flags for all these windows will be very time-consuming. So here we run the DPM detector on an image first and pick top 10% from all detection windows as our candidates. Then we attach visibility flags to these candidates and here we introduce Alternation Direction Method of Multipliers (ADMM) [16] to speed up the optimization process.

The function (5-6) can be rewritten as the following ADMM form,

$$\begin{aligned}
 & \text{minimize } -\omega^T q + \lambda \|z_1\|_1 + J_{[0,1]}(z_2) \\
 & \text{subject to } Dq = z_1, q = z_2, q \in [0,1]^{C+P}
 \end{aligned} \tag{5-8}$$

where  $q = [v_0^0, \dots, v_0^C, v_1, \dots, v_p]^T$  is the concatenation of visibility flags of cells and parts,  $\omega = [(R_0^0 + \alpha\lambda_0^0), \dots, (R_0^C + \alpha\lambda_0^C), (R_1 + \beta\lambda_1), \dots, (R_p + \beta\lambda_p)]^T$  is the stack of filter responses and a penalty term parameter.  $D$  here is a  $(C + P) \times (C + P)$  differentiation matrix for cell-to-cell consistency and cell-to-part consistency penalty terms.  $J_{[0,1]}(z_2)$  is an indicator function.  $J_{[0,1]}(z_2) = 0$  when  $z_2$  between 0 to 1, otherwise  $J_{[0,1]}(z_2) = +\infty$ . Then we can get the augment Lagrangian function as following,

$$\begin{aligned}
 L(\omega, z_1, z_2, u_1, u_2) = & -\omega^T q + \lambda \|z_1\|_1 + J_{[0,1]}(z_2) \\
 & + \frac{\rho_1}{2} \|Dq - z_1 + u_1\|_2^2 \\
 & + \frac{\rho_2}{2} \|q - z_2 + u_2\|_2^2
 \end{aligned} \tag{5-9}$$

where  $\rho_1 > 0, \rho_2 > 0$  are penalty terms. Here tried different values for parameters  $(\lambda, \rho_1, \rho_2)$  and find best value set. This will be discussed in detail in next section.

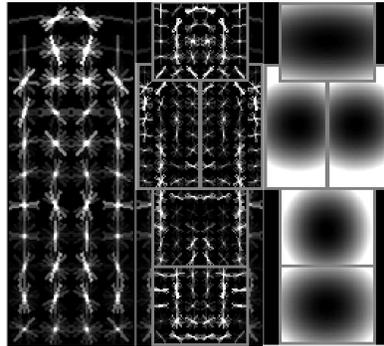
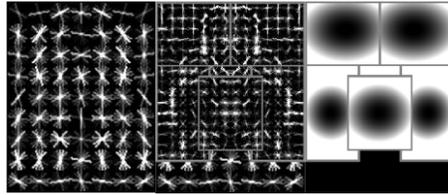
## **6.0 EXPERIMENTS**

### **6.1 DETECTION OF FULLY VISIBLE OBJECTS**

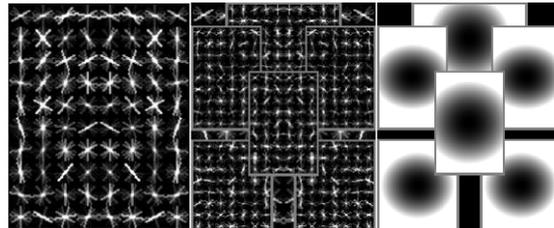
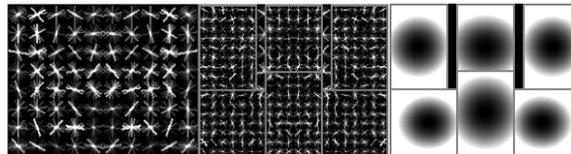
The Deformation Part Model is tested on PASCAL VOC 2006 [20], 2007 [18] and 2008 [19]. These are very difficult dataset for object detection challenge. Each dataset has thousands of images with one or more different objects on them. Every object is has a ground-truth bounding box. In the test process, our goal is to predict the bounding box for a given class object. Since the system returns several high-scoring bounding box candidates, so we need to set a threshold to pick out better results from these candidates. A prediction of bounding box can be treated as correct when it overlap the ground-truth bounding box over 50%. Sometimes the system with return several bounding boxes that overlap together, in this case, we will only consider one of them is correct, and the others are false.

Every model consists of two components. Figure 8 shows some mixture models trained on PASCAL 2007 set. Figure 9 shows some detection results we get by using these models.

person



dog



aeroplane

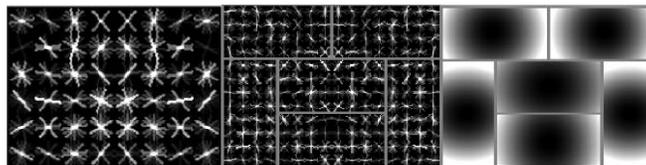
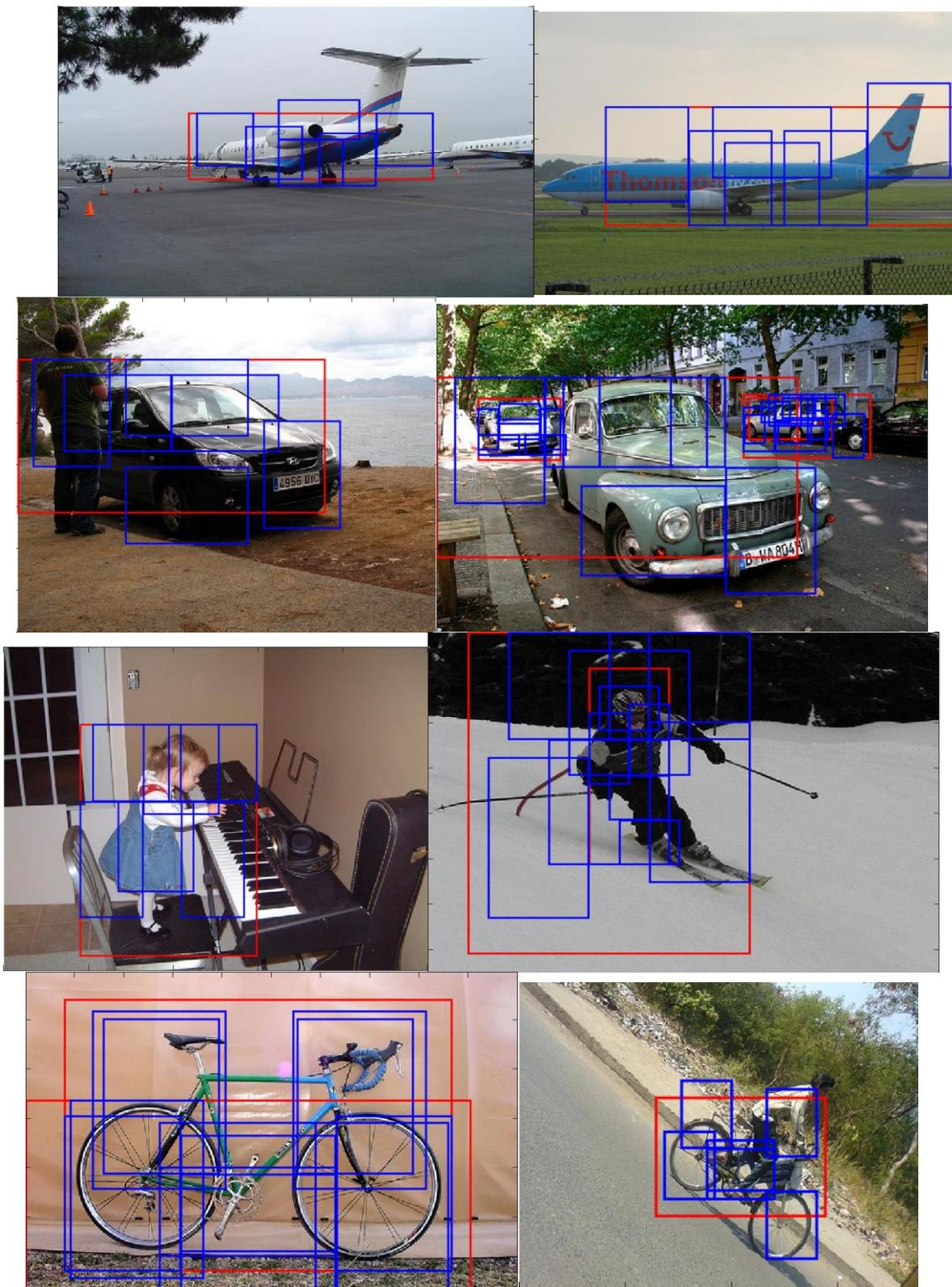


Figure 8. Some model examples trained on PASCAL 2007 dataset



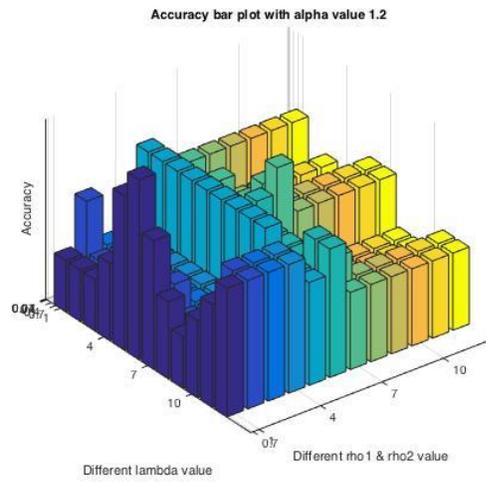
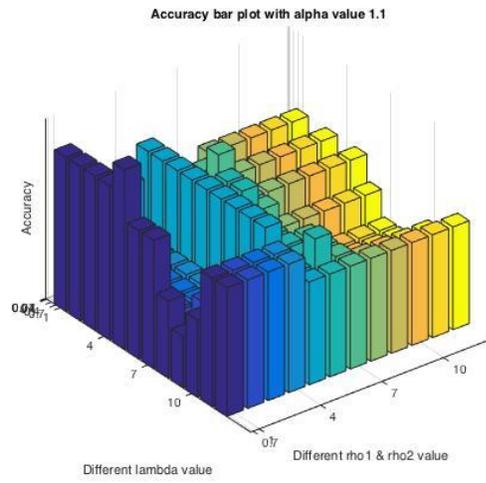
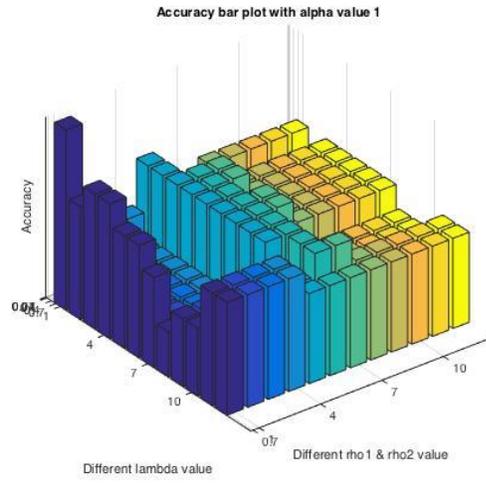
**Figure 9.** Some detection result on PASCAL test dataset

## 6.2 DETECTION OF PARTIALLY OCCLUDED OBJECTS

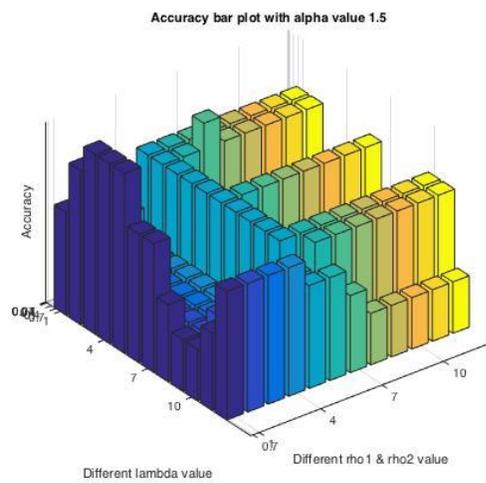
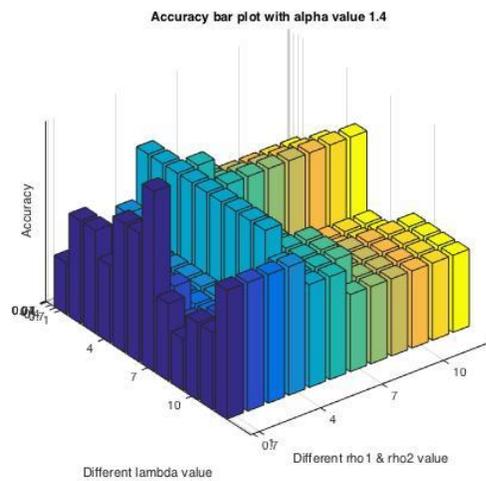
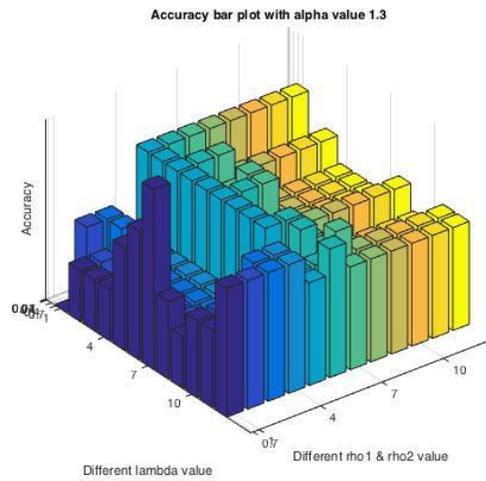
The test dataset for partially occluded objects we choose here is Daimler Multi-Cue Occluded Pedestrian Classification benchmark [13], which contains a large variety set of partially occluded pedestrians with cars or other non-pedestrian objects as occluders. Note that the occlusion handling model here in this paper is not able to deal with pedestrian-to-pedestrian occlusion. That's why the most other relative datasets like ETH [23] and Caltech Pedestrian Detection Benchmark [24] cannot be used here.

The training and testing part of the dataset contain labeled pedestrians and non-pedestrians bounding boxes. All the images of the dataset are taken from a vehicle-mounted camera in an urban environment. Positive samples are manually labeled while the negative samples are based on pre-processed with relaxed threshold setting. Besides formal intensity image, the dataset also contains dense stereo samples and dense optical flow samples. There are totally 80,000 training samples as well as 50,000 testing samples. Each sample has a 96\*48 resolution with a border of 12 pixels around the pedestrian. Intensity samples are the only samples used in my test.

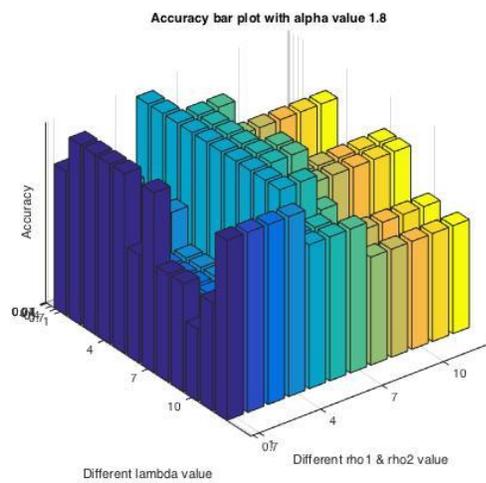
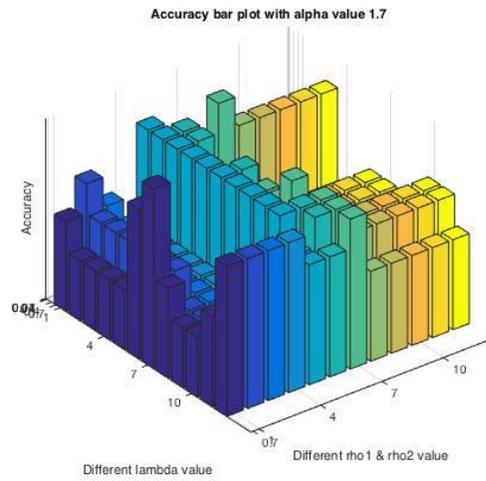
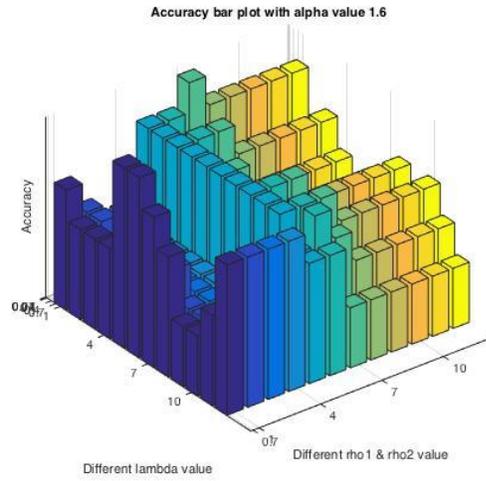
In practice, the minimization problem of function (5-9) has parameters such as  $\lambda$ ,  $\rho_1$ ,  $\rho_2$ ,  $\alpha$ , to be set. But the Daimler dataset only provides test set for occluded pedestrian detection. There is not enough data used to train these parameters. So here I just try a range of values of every parameter and find the parameter set that produces highest detection score.



**Figure 10.** Alpha from 1 to 1.2



**Figure 11.** Alpha from 1.3 to 1.5



**Figure 12.** Alpha from 1.6 to 1.8

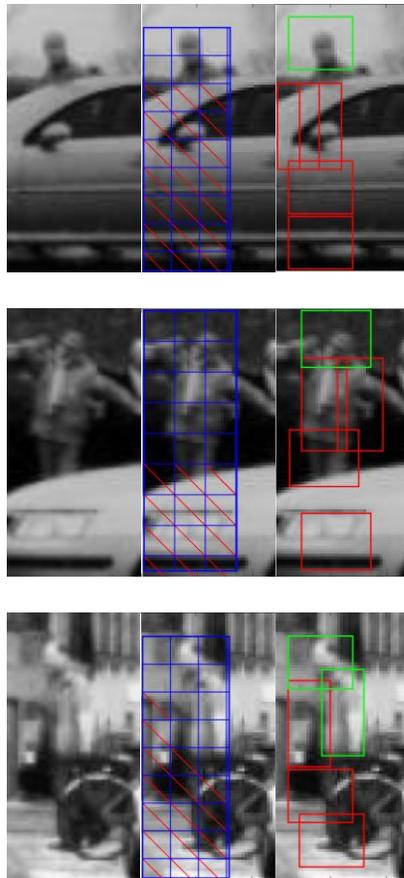
Figure 10, Figure 11 and Figure 12 shows the detection accuracy value according to different parameter set. To reduce the search space, I assume that  $\rho_1 = \rho_1$ . When the bounding box returned by the system overlaps the ground-truth bounding box over 50%, then we treat the detection result as accurate. Otherwise, we treat it as false detection. The highest scoring parameter set we get is ( $\lambda = 1.4, \rho_1 = 1, \rho_2 = 1, \alpha = 1.4$ ).

Even though only the cell information and part information alone cannot determine whether a region is occluded or not, this information combined with penalty term will provide a relatively robust model handling occlusion problems. Figure 13 shows some detection results on the Daimler dataset.

I compare the occlusion handling model with mixture deformation part model by testing both of them on the partially occluded and non-occluded pedestrian datasets. And the test result is shown in Table 1. From the table we can see clearly that the accuracy rate of partially occluded object detection is significantly improved compared with DPM.

**Table 1.** Comparison of occlusion handling model and DPM

	Partially occluded	Non-occluded
Occlusion Handling	0.9717	0.9884
DPM	0.350	0.6047



**Figure 13.** Some detection results:

Original input image (left). Cell occlusion detection (middle). Regions covered with red lines are occluded. Part occlusion detection (right). Parts with red bounding box lines are occluded.

## 7.0 CONCLUSION

In this paper, we introduce a state-of-art object detection model, mixture Deformable Part Model, and based on this model, we introduce an occlusion handling approach. We attach visibility flags to every root cells and parts and by doing convex function optimization, then we get the flag values. We do test on the Daimler Multi-Cue Occluded Pedestrian Classification benchmark and the experiments show that the accuracy rate of partially occluded object detection improved significantly. But there is still storage of this model: it cannot deal with pedestrian-to-pedestrian occlusion. I will do more future research on it.

## BIBLIOGRAPHY

- [1] Felzenszwalb, Pedro F., et al. "Object detection with discriminatively trained part-based models." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.9 (2010): 1627-1645.
- [2] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.
- [3] Dollar, Piotr, et al. "Pedestrian detection: An evaluation of the state of the art." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.4 (2012): 743-761.
- [4] Wang, Xiaoyu, Tony X. Han, and Shuicheng Yan. "An HOG-LBP human detector with partial occlusion handling." *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009.
- [5] Vedaldi, Andrea, and Andrew Zisserman. "Structured output regression for detection with partial truncation." *Advances in neural information processing systems*. 2009.
- [6] Gao, Tianshi, Benjamin Packer, and Daphne Koller. "A segmentation-aware object detection model with occlusion handling." *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011.
- [7] Ouyang, Wanli, and Xiaogang Wang. "A discriminative deep model for pedestrian detection with occlusion handling." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012.
- [8] Niknejad, Hossein Tehrani, et al. "Occlusion handling using discriminative model of trained part templates and conditional random field." *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013.
- [9] Azizpour, Hossein, and Ivan Laptev. "Object detection using strongly-supervised deformable part models." *Computer Vision–ECCV 2012*. Springer Berlin Heidelberg, 2012. 836-849.

- [10] Girshick, Ross B., Pedro F. Felzenszwalb, and David A. Mcallester. "Object detection with grammar models." *Advances in Neural Information Processing Systems*. 2011.
- [11] Yu, Xiang, et al. "Consensus of regression for occlusion-robust facial feature localization." *Computer Vision–ECCV 2014*. Springer International Publishing, 2014. 105-118.
- [12] Ghiasi, Golnaz, and Charless Fowlkes. "Occlusion coherence: Localizing occluded faces with a hierarchical deformable part model." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.
- [13] Baumgartner, Tobias, Dennis Mitzel, and Bastian Leibe. "Tracking people and their objects." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013.
- [14] Enzweiler, Markus, et al. "Multi-cue pedestrian classification with partial occlusion handling." *Computer vision and pattern recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010.
- [15] Mathias, Markus, et al. "Handling occlusions with franken-classifiers." *Proceedings of the IEEE International Conference on Computer Vision*. 2013.
- [16] Tang, Siyu, Mykhaylo Andriluka, and Bernt Schiele. "Detection and tracking of occluded people." *International Journal of Computer Vision* 110.1 (2014): 58-69.
- [17] Wahlberg, Bo, et al. "An ADMM algorithm for a class of total variation regularized estimation problems." *arXiv preprint arXiv:1203.1828* (2012).
- [18] Andrews, Stuart, Ioannis Tsochantaridis, and Thomas Hofmann. "Support vector machines for multiple-instance learning." *Advances in neural information processing systems*. 2002.
- [19] Everingham, Mark, et al. "The pascal visual object classes challenge (voc2007) results." [Online] Available: <http://www.pascal-network.org/challenges/VOC/voc2007/>
- [20] Everingham, Mark, et al. "The pascal visual object classes challenge (voc2008) results." [Online] Available: <http://www.pascal-network.org/challenges/VOC/voc2008/>
- [21] Everingham, Mark, et al. "The pascal visual object classes challenge (voc2006) results." [Online] Available: <http://www.pascal-network.org/challenges/VOC/voc2006/>
- [22] Boyd, Stephen, et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers." *Foundations and Trends® in Machine Learning* 3.1 (2011): 1-122.

- [23] Dollár, Piotr, et al. "Pedestrian detection: A benchmark." *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009.
- [24] Ess, Andreas, Bastian Leibe, and Luc Van Gool. "Depth and appearance for mobile scene analysis." *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007.
- [25] Tang, Siyu, Mykhaylo Andriluka, and Bernt Schiele. "Detection and tracking of occluded people." *International Journal of Computer Vision* 110.1 (2014): 58-69.
- [26] Felzenszwalb, Pedro, and Daniel Huttenlocher. *Distance transforms of sampled functions*. Cornell University, 2004.
- [27] Chan, Kai Chi, Alper Ayvaci, and Bernd Heisele. "Partially occluded object detection by finding the visible features and parts." *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015.