Reconfigurable Templates for Robust Vehicle Detection and Classification

Yang Lv[1,3], Benjamin Yao[2,3], Yongtian Wang[1] and Song-Chun Zhu[2,3]
[1]School of Computer Science, BIT
[2]Department of Statistics, UCLA
[3]Lotus Hill Research Institute

Abstract

In this paper, we learn a reconfigurable template for detecting vehicles and classifying their types. We adopt a popular design for the part based model that has one coarse template covering entire object window and several small high-resolution templates representing parts. The reconfigurable template can learn part configurations that capture the spatial correlation of features for a deformable part based model. The features of templates are Histograms of Gradients (HoG). In order to better describe the actual dimensions and locations of "parts" (i.e. features with strong spatial correlations), we design a dictionary of rectangular primitives of various sizes, aspect-ratios and positions. A configuration is defined as a subset of non-overlapping primitives from this dictionary. To learn the optimal configuration using SVM amounts, we need to find the subset of parts that minimize the regularized hinge loss, which leads to a non-convex optimization problem. We solve this problem by replacing the hinge loss with a negative sigmoid loss that can be approximately decomposed into losses (or negative sigmoid scores) of individual parts. In the experiment, we compare our method empirically with group lasso and a state of the art method [7] and demonstrate that models learned with our method outperform others on two computer vision applications: vehicle localization and vehicle model recognition.

1. Introduction

Compositional hierarchy [1, 14, 9, 11] and deformable parts [7, 2, 3, 8, 16] are widely used for modelling and detecting objects in computer vision. For example, the star model [10, 8] and the constellation model [9] can formulate the parts and their deformations by Gaussian models, and recently, new features such as HoG [4] have contributed to the steady improvement of detection performance. In these models, "parts" are generally used to group features with strong spatial correlations that should move together. However, most of previous methods use either fixed or hand-designed windows for parts, while in real world applica-

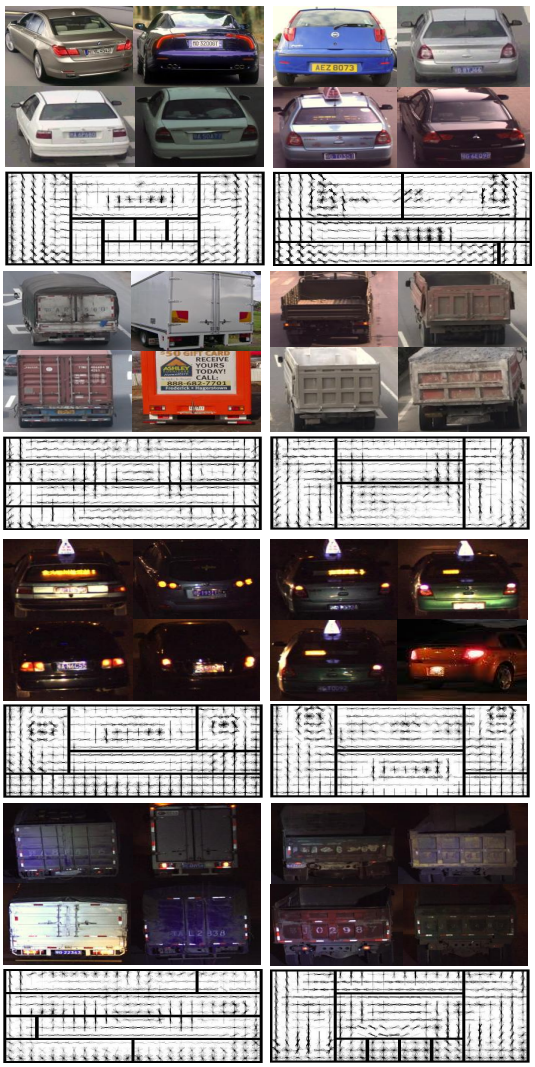


Figure 1. By using various shapes of parts, our method can learn global optimal tiling of parts that correspond to different types of vehicles under various viewing angle, lighting condition and configurations (left and right columns).

tions such as the vehicle type classification task illustrated in Fig.1, we face the problem of large variations of categories, viewing angles, lighting conditions that require different configurations of parts.

It was noted in the literature [20] that the difficulty of learning structured model lies in the ambiguity of its parts. Many techniques have been developed to address this problem. For example, L. Zhu et al. proposed a bottom-up clustering method by discovering suspicious coincident structures [20], and Dollar et al. proposed a multiple component learning framework [5] that learns parts based on the adaboost [12]. Similarly, the work of Huang [13] also used adaboost to select a local optimal configuration of templates. P. Felzenswalb et al. presented a line of work [7, 6] that treated part locations as latent variables and learned the appearance and deformation parameters jointly using a latent-SVM algorithm. While their method enjoys good popularity, it has one drawback that the algorithm is sensitive to the initialization of part. The heuristic method used to initialize parts does not always find good placement and it also relies on manual input for shape and size of parts. In contrast, we propose to use parts with different shapes and sizes by learning an optimal configuration to construct part-level templates.

In this paper, we present a novel reconfigurable template to learn part configurations for vehicle detection and classification. We adopt a popular design, the Part Based Templates model [7] (which will be later referred to in short as "PBT"). The difference between our method and PBT is that our templates learn an optimal configuration of parts to represent vehicles. Given a dictionary with rectangular primitives of various positions, sizes and aspect-ratios, a configuration is defined as a set of primitives that tile a detection window (as shown in Fig.1). The objective of learning configuration is to find a non-overlapping subset of primitives that minimize certain loss function (e.g. the empirical classification error). The absence of previous methods addressing this issue, however, may largely due to the fact that, in general, this objective involves a non-convex optimization problem. We replace the non-convex loss function with a negative sigmoid score as our loss function, which can be approximately decomposed into negative sigmoid scores (margins) of individual parts. Then our training algorithm is carried out in three steps: 1) Learn appearance model for all primitives via the Latent SVM algorithm [7] and compute their sigmoid scores; 2) Build the And-Or search tree. Starting from the object window, an And-Or search tree is built recursively by enumerating all possible ways to bisect a rectangle window into two smaller windows so that every leaf node on the tree corresponds to a primitive from the dictionary. Therefore, each path on this tree corresponds to a valid configuration; 3) Search the optimal tiling configuration with Dynamic Programming (DP).

The contributions of this paper are as follows,

- **Optimized Configuration**. We propose a reconfigurable template and an algorithm for learning part configurations that captures the actual spatial correlations of features for robust vehicle detection and classification. This method can be easily applied on general deformable part-based model. By comparing with the original PBT method [7] and group lasso, we demonstrate empirically that models using configurations learned with our method work best. Besides, our learning results are insensitive to initialization because unlike heuristic search strategy [13] and pursuit method [15, 16], we seeks the global optimal solution.
- **High efficiency and performance**. Compared with other multi-layer hierarchical model and constellation model, our final model is similar to the two layer PBT model, which can be computed very efficiently by cascading [6]. We apply our method on a traffic surveillance system. The camera locates in the crowded environment where most of the tracking methods fail to work. Our optimized detection algorithm can compute 8 templates at a speed of 200ms/frame in a dual core CPU (3.9GHz) for vehicle detection. The model learned through our method can accurately count vehicle flow, classify vehicles, and provide context for locating license plate. The detection rate approximates 95% with average of 0.3 false alarm per frame in traffic (See Fig.6 and supplementary video).

2. Representation

2.1. Reconfigurable templates

We adopt a model that is similar to the popular PBT model [7], which has one coarse template covering entire object window and several small high-resolution templates representing parts. The Histogram of Gradients (HoG) feature is used for the templates [4]. There are two differences between our model and the PBT: 1) parts in our model have various sizes, aspect ratios, whereas in the PBT, parts are square windows of fixed sizes; 2) During detection, we allow a part not only to displace horizontally and vertically, but also rotate locally. Formally, a template consisting of a coarse-level template T_O, and m part-level templates is the following,

$$T = (T_O, m, \{T_P^j\}_{j=1}^m) \tag{1}$$

where $\{T_P^j\}_{j=1}^m$ denotes a set of part-level templates. Each part has a set of independent variables, which can be specified by a 3-tuple: $T_P^j = (h_j, \omega_j, S_j)$, where

i) $h_j = (d_j, r_j)$ denotes the latent variables of the j^{th} part including its displacement $d_j = (dx, dx^2, dy, dy^2)$

w.r.t. an anchor point and its rotation. We allow each part to have small rotations $r_j = \{-20°, 0, 20°\}$ by circularly left or right shifting bins of HoG histograms. Similar to PBT, we use a 2-D quadratic function to penalize the displacements d_j. Rotation is not penalized.

ii) w_j is the parameter vector of the j^{th} part which includes the weights for the HoG histograms, a bias term and negative weights for displacements d_j.

iii) $S_j = (s_j, a_j, l_j)$ represents the size s_j, aspect ration a_j, and anchor point location l_j of the j^{th} part. In this paper, we use all possible rectangles composed of no less than one 3×3 HoG cell (,which equivalents to 24×24 pixels), while is no greater than half of the average size of the object window (determined by annotations). The parts are placed according to a pre-determined grid $l_j \in G^2$ spaced apart by 3×3 HoG cells. For example, an object window of 96×192 pixels (12×24 HoG cells) allows $s_j \in [9, 72](cells^2)$, $a_j \in [1/4, 8]$ and a 4×8 grid, which accounts for a total of 267 different choices of S_j. We define the set of all possible primitives S as a dictionary, which will be discussed in further details in the next section.

2.2. Part dictionary and configurations

Given a dictionary S with M candidate primitives, we use a vector $\beta \in \{0, 1\}^M$ to indicate which primitives appear in the final template (i.e. j^t primitives is "on" if $\beta_j = 1$, or "off" if $\beta_j = 0$). A template T with m parts implies $\sum_j \beta_j = m$. It is worth mentioning that since each primitive in the dictionary has a unique S (size, aspect-ratio and location), if β is given, $\{S_j\}_{j=1}^m$ is also known. A configuration is defined as a subset of primitives, which can also be represented by β. We require a valid configuration β^c to have no overlapping "on" primitives. Intuitively a configuration is a tiling of detection window with rectangular primitives (see Fig.1). Formally, let $R(j)$ denotes the window of primitive $j, \beta^c \in \{0, 1\}^M$, s.t. $\forall j, k$, if $\beta_j^c = 1$ and $\beta_k^c = 1$, $R(j) \cap R(k) = \emptyset$. By adding this non-overlapping constraint, we are implicitly enforcing sparsity of features i.e. $m \ll M$. For example, using the example of $M = 267$ from previous section, the maximum number of m is 32. Most of configurations in our experiments have m around 8.

3. Learning

Given the pair of training data as $\mathcal{D} = \{(x_i, y_i), \cdots, (x_n, y_n)\}$, where x_i is the feature extracted from the i^{th} image, $y_i \in \{-1, +1\}$ denotes the label of sample i, according to our model, we need to learn a linear classifier that scores an example x with a function of the form,

$$f(x) = \max_{h \in H} \sum_{j=1}^{m} \beta_j \omega_j^T \phi_j(x, h_j) + \omega_0 \phi_0(x, h_0) \qquad (2)$$

where function $\phi_j(x, h_j, S_j)$ is a feature vector of part j on a sample extracted at window S_j, plus h_j displacement/rotation (since S_j is dictated by j, we will omit it in future references). ω_j is the parameter vector.

3.1. Latent-SVM

If the part configuration β is given, the problem (2) is reduced to the original PBT, whose parameters ω can be learned by minimizing the the following objective function over $\mathcal{D}$ using the latent-SVM algorithm [7],

$$\min_{\omega} \frac{1}{2}\|\omega\|_2 + \frac{C}{n} \sum_{i=1}^{n} \max(0, 1 - y_i f(x_i)) \qquad (3)$$

where $\omega = [\omega_0, \cdots, \omega_m]$ is the joint parameter vector. The learning algorithm has two main steps: i) imputing hidden variable h_j; ii) updating the value of parameter ω.

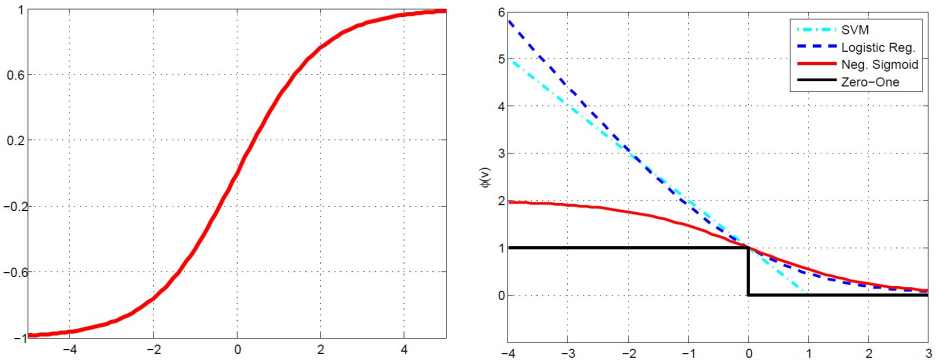
3.2. Learning optimal configuration

In addition to learn appearance parameter ω in the previous section, we also have to learn an optimal configuration $\beta = [\beta_1, \cdots, \beta_m], \beta_j \in \{0, 1\}$. To learn it, we should solve the following optimization problem:

$$\begin{aligned} &\min_{\omega, \beta} \frac{1}{2}\|[\omega_0, \beta_1\omega_1, \cdots, \beta_M\omega_M]\|_2 \\ &+\frac{C}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(f_0(x_i) + \sum_{j=1}^{M} \beta_j f_j(x_i))) \\ &s.t. \forall j, k \in [1, M], if \beta_j = 1 \& \beta_k = 1, R(j) \cap R(k) = \emptyset \end{aligned} \qquad (4)$$

where $f_j(x) = \omega_j^T \phi_j(x, h_j, S_j)$ is the prediction function for each part with the hidden variable h_j, and ω_0 is the root parameter. This is a highly non-convex problem that cannot be easily solved with numerical methods. The enormous number of valid configurations β also makes brutal-force search method impractical. We will discuss two approximation steps in the following to make this problem solvable. For future references, we rewrite the objective function of problem (4) into $l_D(\omega, \beta)$.

i) **Learn appearance model for all primitives.** One reason that $l_D(\omega, \beta)$ is not solvable is because there are two parameters ω and β in this function that could affects each other. However, given the fact that the appearance parameters of each part is relatively independent, we can first learn ω by turning "on" all primitives, i.e. let $\beta_j = 1, \forall j \in [1, M]$. Therefore, $\omega^o = argmin_\omega l_D(\omega|\beta = 1)$ is a semi-convex optimization problem that can be solved with the latent-SVM algorithm [7].



(a) Sigmoid function. (b) Loss function.

Figure 2. (a) Sigmoid function is almost linear in its middle section. (b) Comparison of negative sigmoid loss and loss functions of SVM (hinge loss) and logistic regression.

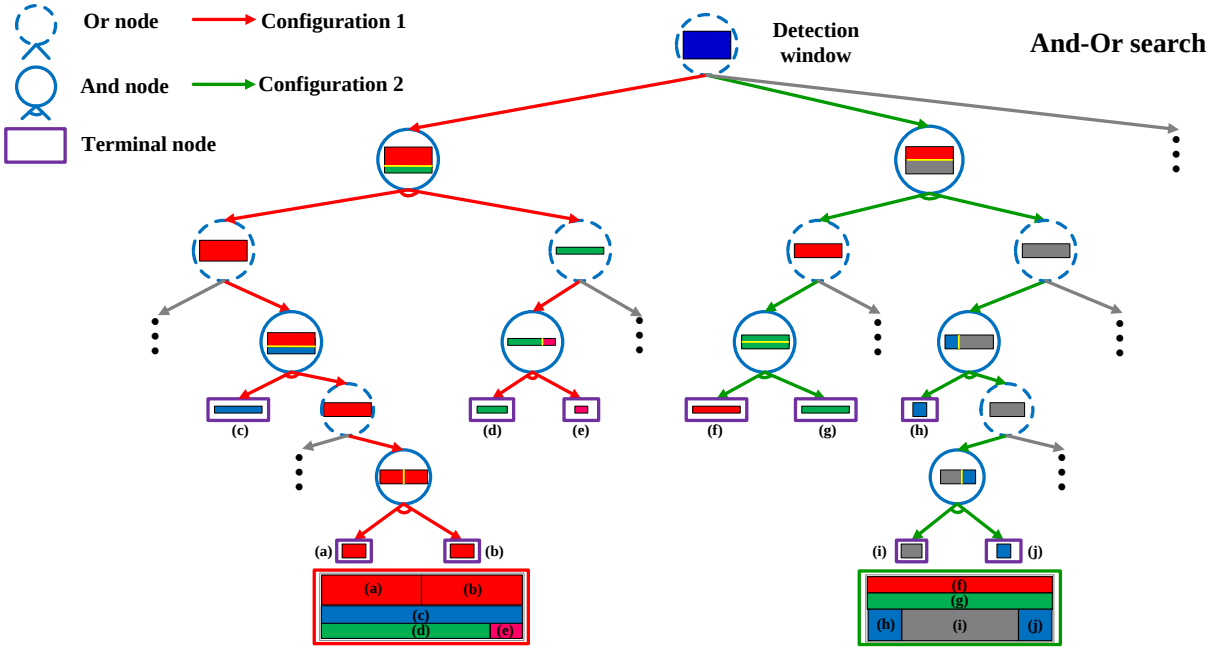


Figure 3. Composition of configurations by using dictionary of primitives. Or node represents orswitch for different bisecting methods; And node bisects a part into two. (Better in color).

ii) **Separate part scores.** With ω^0 learned from the previous step, we can now compute the regularized hinge loss $l_D(\beta|\omega^0)$ for any given β. The search space is still too large to compute. However, if we can manage to separate loss (or equivalently negative scores) for each part, we can apply searching tricks such as dynamic programming. Let the score of the j^{th} part be $v_j = \sum_{i=1}^{n} y_i f_j(x_i)$, which is the sum of margins on n samples. The hinge loss of SVM involves a maximum operation of v, $max(0, 1 - \sum_j v_j)$ and therefore is not separable. Instead, we can use a negative Sigmoid function $NSig(v) = \frac{2}{1+\exp(v)}$ that is also an upper bound of the zero-one loss (as shown in Fig.2(b))(The sigmoid function $Sig(v) = \frac{2\exp(v)}{1+\exp(v)}$ is illustrated in Fig.2(a)).

The reason to choose the negative Sigmoid loss is two-fold:

1) As shown in Fig.2(a), when v is close to 0, $Sig(v)$ approximates a linear function, therefore,

$$\begin{aligned} &NSig(v_1 + \cdots + v_M) \\ \approx &1 - \frac{1}{M}[Sig(M * v_1) + \cdots + Sig(M * v_M)] \end{aligned} \tag{5}$$

We empirically observe that most of v_j in our experiment satisfy this condition.

2) The RHS of Eq.5 is always bounded by its LHS, which is in turn bounded by $[0, 2]$. Therefore, the RHS of Eq.5 is always a reasonable approximation to the zero-one loss function. Intuitively, choose sigmoid function is reasonable

because it penalize parts with extremely scores beyond normal range.

With the above approximation, we can now rewrite the loss function $l_D(\beta|\omega^0)$ as

$$\begin{aligned}\beta* &= \arg\min_\beta \sum_{j=1}^{M} \beta_j \cdot [\frac{1}{2}\|\omega_j\|_2 - \frac{C}{n} Sig(\sum_{i=1}^{n} y_i f_j(x_i))] \\ &= \arg\min_\beta \sum_{j=1}^{M} \beta_j \cdot r_j(\omega_j) \\ s.t. &\forall j,k \in [1,M], if \beta_j = 1 \& \beta_k = 1, R(j) \cap R(k) = \emptyset \end{aligned} \quad (6)$$

where $R(j)$ is the window of the j^{th} part, $r_j(\omega_j)$ is the regularized empirical loss of part j. The above optimizing problem (6) can be efficiently solved by an And-Or search algorithm described in the next section.

iii) **Adjust the weights.** Given the configuration β^* derived from the previous step, we adjust the weights $\omega^1 = \arg\min_\omega l_D(\omega|\beta^*)$ with the latent-SVM algorithm. This step generally converges within a few steps of gradient descent, which confirms that ω^0 is a fairly good approximation.

3.3. Finding an optimal configuration via And-Or search

The And-Or search algorithm has two steps, the first step is to enumerate all configuration in a top-down approach, and the second step is to find an optimal configuration in a bottom-up approach. Starting from the object window, the And-Or search tree is built recursively by enumerating all possible ways to bisect a rectangle window into two smaller windows so that every leaf node on the tree corresponding to a primitive from the dictionary. Therefore, each path on this tree corresponds to a valid configuration. As illustrated in Fig.3, the search procedure starts from a root node. At each step, we expand a node using And-Or operators. The Or operators transform a primitive into all possible bisecting. The And operators bisect a primitive into two smaller ones. By expanding the starting node recursively until all nodes reach a terminal node, we can enumerate the solution space of configurations. The red route and the green route in Fig.3 show two solutions generated by And-Or search tree. The problem of learning the optimal configurations is to search the minimum regularized loss $\sum_{j=1}^{M} \beta_j r_j$ in all solution space of β. And the optimal solution can be solved by dynamic programming on the And-Or search. Given learned parameter ω and enumerated configurations, we summarize the procedures to seek optimal configuration in the following steps.

i) Initialization. Given training data $\widehat{\mathcal{D}} \in \mathcal{D}$ for a type of object, we can compute regularized empirical loss r_j on $\widehat{\mathcal{D}}$ (Eq.6) and assign scores to corresponding leaf nodes. For other leaf nodes, the value is assigned zero. At the same time, $\forall \beta_j, \beta_j = 0$.

ii) DP in bottom-up approach. After initialization, we can compute maximum score of each node in a bottom-up way. For each And node, score is the sum of its two child nodes $r_{and} = r_1 + r_2$. For each Or node, score is the max value of its child nodes $r_{or} = \max(r_1, \cdots, r_{last})$. In this way, we can compute the max score of root node.

iii) Retrieving the optimal solution. By tracing back, we can get a solution sub-tree T_{slu} of the And-Or search T_{search}. For each β_j, if its part appears in the leaf node of T_{slu}, $\beta_j = 1$, and an optimal configuration is learned. The learning algorithm is outlined in Algorithm 1.

Algorithm 1 The dynamic programming algorithm for learning configuration

Require: And-Or search T_{search}, and appearance parameter ω;
Ensure: The solution vector β;
1: Initialize node score of T_{search}, and β.
2: **for** each node **do**
3: $r_{and} = r_1 + r_2$.
4: $r_{or} = \max(r_1, \cdots, r_{last})$.
5: **end for**
6: Retrieve T_{search} and get solution tree T_{slu}.
7: **for** each terminal node in T_{slu} **do**
8: $\beta_j = 1$.
9: **end for**
10: **return** β.

3.4. Learning with Group Lasso

Recent asymptotic analysis of the $l1$ norm optimization proved that under certain conditions the estimated sparse coefficient of feature enjoys a property called sparsistency [17], suggesting their applicability for meaningful variable selection in high-dimensional feature space. The mixed-norm $l1/l2$ regularization has been used for recovery of joint sparsity across input dimension [18]. A latent-SVM under $l1/l2$ regularization for problem (4) has the form,

$$\min_\omega \lambda \sum_{j=1}^{M} \frac{\sqrt{\omega_j^T \omega_j}}{\sqrt{df_j}} + \sum_{i=1}^{n} \max(0, 1 - y_i f(x_i)) \quad (7)$$

where tuning parameter λ controls the amount of penalization, and df_j is the dimension of the j^{th} group of ω_j. This shrinkage tends to encourage features in the same group to be jointly zero. Therefore, the sparsity is now imposed on the configuration β. Similar to latent-SVM, the learning algorithm has two steps, i) imputing hidden variable h; ii) updating the value of parameter ω by block coordinate gradient descent [19].

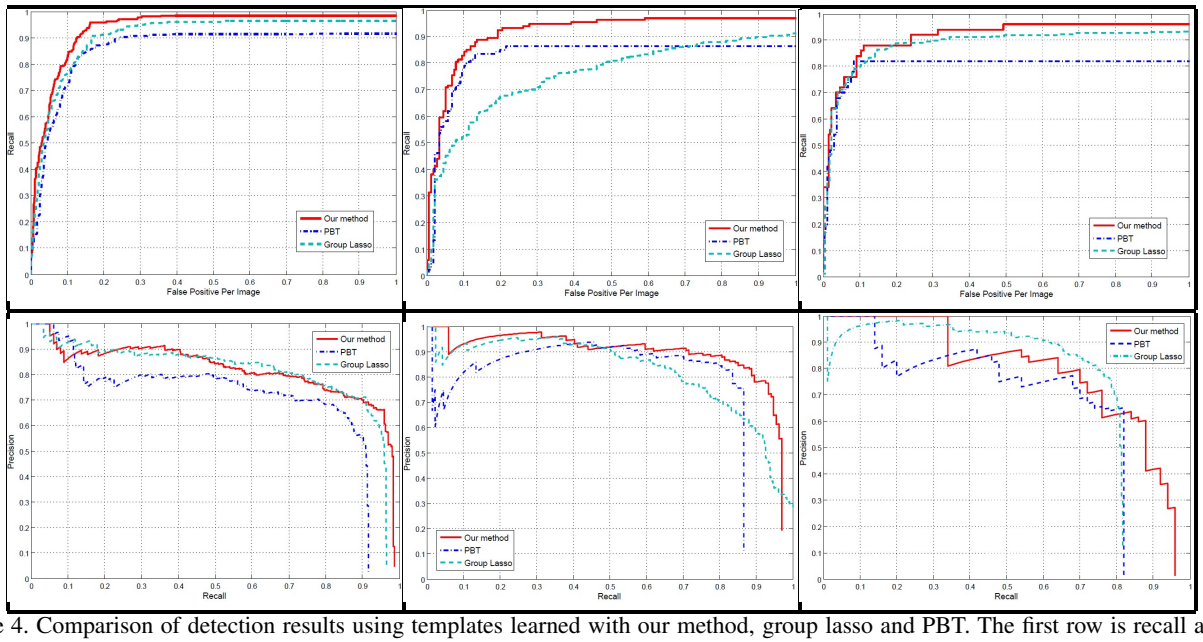
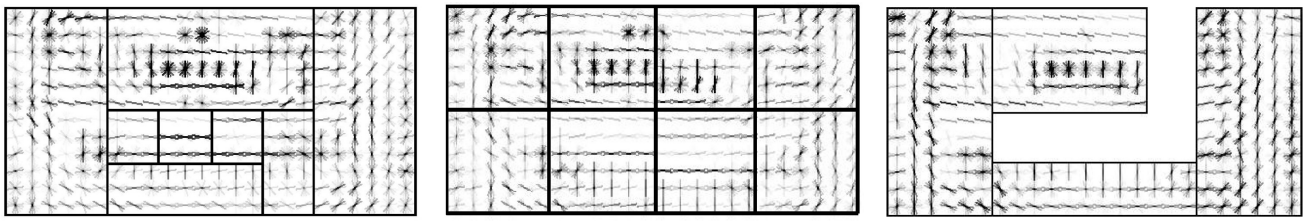


Figure 4. Comparison of detection results using templates learned with our method, group lasso and PBT. The first row is recall against false positive per image, and the second row is precision against recall. From left to right are comparisons of detection results using our method, PBT and group lasso on the dataset of sedan in day time, sedan in night time, and rear truck at all time.



(a) Learned with our method (b) Learned with the PBT (c) Learned with Group lasso

Figure 5. Template/configuration learned with (a) our method, (b) the PBT and (c) the group lasso. The highlighted boxes represent the part configurations. It is noticeable that the PBT template covers the license plate area with two parts.

4. Experiments

4.1. Dataset

As the first attempt to tackle the problem of detecting and classifying types of vehicles at both daytime and night, we have no existing dataset to use and compare with (The popular Pascal dataset has only two types of vehicles at day time, 2501 cars and near 500 buses). Instead we have compiled a new dataset sorted into several subtypes according to the illumination conditions (day and night), types of vehicles (truck and sedan), views (frontal and rear), and configurations. Images are collected from surveillance videos and Internet, as shown in Fig.1. The number of image in each subtypes varies from 500 to 3000. Here we highlight the properties of our dataset:

- **Time variances**. The images are captured by surveillance system 24 hours consecutively,
- **Views**. Vehicles are captured in both frontal view and rear view,
- **Inner-class variance**. The same type of vehicle in the same view can have different configurations,
- **Complexity of background**. Our dataset are captured on real surveillance environment locating in crowded roads, which are full of pedestrians, vehicles, and trees.

We have also obtained a ground truth annotation of each image. The annotation information provides the exact location of each vehicle in the image. For training, we use around 100 different positive examples for each category as well as a shared set of 500 negative images without any vehicles. For testing, we collected 500 positive examples for each category of sedan and 300 examples for all categories of truck (due to lack of truck data in general) and 500 negative images.

Table 1. Average precision of vehicle type classification. Car1 and car2 have different configurations as illustrated in Fig.1

	car1	car2	truck1	truck2
Our method	70.36	73.32	78.12	80.92
PBT	61.70	39.58	51.40	78.95

4.2. Vehicle detection by configuration learned with our method v.s. other methods

We learn the eight classes of templates (as shown in Fig.1) separately against the common negative training examples, and compare the vehicle detection performance of our method, PBT [7] and group lasso on our testing dataset. Note that the same type of model setting and feature (HoG) are used across all experiments, the only difference between these models are how the configurations are learned. The PBT code is from [7] and we modify it by allowing HoG to have small rotation, which can bring 2% improvement in terms of both detection and classification. The training code is run on a computer cluster with 20 cores and 48Gb memory. We measure the detection performance in three groups: 1) sedan detection in day time (i.e. combine the detection results of two categories together), 2) sedan detection at night, and truck rear detection all day (all combined together, due to the lack of testing data). In all experiments, only the shared negative training samples are used. Fig.4 shows experiment results of three categories. It is clear to see, our method is generally better than both group lasso and the PBT model [7].

Fig.3.4 illustrates the learned type1 car template using three different methods. It is clear that the PBT model has two parts covering the same license plate area due to fixed configuration. Both our method and group lasso, captures certain important areas such as the two sides and the license plate. The the group lasso method is not robust because it turns on either too many parts that are overlapping or too few parts that cannot cover the detection window.

(See supplementary file for vehicle detection video)

4.3. Vehicle type classification.

For vehicle classification, we train four different "1 v.s. all" classifiers for each model with training examples of other categories as negative examples. The experiment is carried out in only day time data, because separation between day/night data is trivial in real-world. The average precisions for the classification results of our model and the PBT are shown in Table1 respectively. In this task, our model significantly outperforms the PBT model. This result reinforces our believe that when classifying object that are similar but with subtle differences, it is more important to have learned configurations.

5. Discussion

In this paper, we present a reconfigurable template and learning algorithm of learning part configurations for vehicle detection and classification that is shown to outperforms the other methods. The major difficulty of accurate detection and classification types of vehicles, in our opinion, is the lack of effective models that can represent the large variations in configurations and appearance. By tiling the templates with rectangular primitives, we limit the solution space of configurations so that we can find a global optimal solution by dynamic programming quickly. Based on this algorithm, we also build a practical real-time traffic surveillance systems that demonstrates high performance in cluttered traffic scenes.

Acknowledgement

We thank Wenze Hu and Jiangen Zhang for very helpful discussions and suggestions. The data used in this paper is from the Lotus Hill Institute vehicle dataset. The authors supported by NSF China grant 60827003 and 61072096.

References

[1] H. Chen, Z. Xu, Z. Liu, and S.-C. Zhu. Composite templates for cloth modeling and sketching. In *CVPR*, 2006.

[2] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *TPAMI*, 23(6):681–685, Jan. 2001.

[3] J. Coughlan, A. Yuille, C. English, and D. Snow. Efficient deformable template detection and localization without user initialization. *CVIU*, 78(2):303–319, 2000.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[5] P. Dollar, B. Babenko, S. Belongie, P. Perona, , and Z. Tu. Multiple component learning for object detection. In *ECCV*, 2008.

[6] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010.

[7] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 32(9), Sept. 2010.

[8] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 1 2005.

[9] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[10] R. Fergus, P. Perona, and A. Zisserman. *A Sparse Object Category Model for Efficient Learning and Complete Recognition*, volume 4170 of *LNCS*. Springer, 2006.

[11] S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *CVPR*, 2007.

[12] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boost-

Figure 6. Detection results.

ing. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[13] C. Huang, H. Ai, Y. Li, and S. Lao. High performance rotation invariant multiview face detection. *TPAMI*, 29(4), 4 2007.

[14] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. In *CVPR*, Jan. 2006.

[15] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal processing*, 41:3397–3415, 1993.

[16] Y. N.Wu, Z. Si, H. Gong, and S.-C. Zhu. Learning active basis model for object detection and recognition. *IJCV*, 2009.

[17] P.Ravikumar, M.Wainwright, and J.Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *Annals of Statistics*, 2009.

[18] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *Journal of Machine Learning Research*, (9):2491–2521, 2008.

[19] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Math. Programmng*, 117(1):387–423, 2007.

[20] L. Zhu, C. Lin, H. Huang, Y. Chen, and A. Yuille. Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. In *ECCV*, 2008.