Extended Support Vector Machines for Object Detection and Localization

Jan Feyereisl, Bohyung Han (POSTECH, Korea)

Object detection is a fundamental task for many high-level computer vision applications such as image retrieval, scene understanding, activity recognition, visual surveillance and many others. Although object detection is one of the most popular problems in computer vision and various algorithms have been proposed thus far, it is also notoriously difficult, mainly due to lack of proper models for object representation, that handle large variations of object structure and appearance. In this article, we review a branch of object detection algorithms based on Support Vector Machines (SVMs), a well-known max-margin technique to minimize classification error. We introduce a few variations of SVMs--Structural SVMs and Latent SVMs--and discuss their applications to object detection and localization.

Ⅰ. Introduction

Object detection and localization is one of the most primitive operations in computer vision, and has potential to be applied to many applications. Image retrieval, scene understanding, visual surveillance, event detection and assistive driving are only a few examples. Although many researchers in the computer vision and machine learning communities have been making significant efforts for this problem, it is still a challenging problem, primarily due to the diversity of object appearance. Such diversity is typically originated from, not only photometric transformation by color variations and illumination changes, but also geometric or structural variations by non-rigid deformations and viewpoint changes. For example, the appearance of cars may vary due to their colors and viewpoints, and the structures of cars--sedan, sports car, wagon, and truck--may be inherently different.

Object detection is the process of learning the intra-class similarity and the inter-class discriminativeness, typically in a supervised manner. On the other hand, object localization typically means a method to find position and scale of an object to be detected. These two problems are sometimes independent of each other, but often closely related. The object detection and localization procedure comprises of feature detection, object representation, and classification. All these three steps are equally important to maximize detection and localization accuracy. This paper, however, mainly focuses on the classification algorithms among the three steps.

Many object detection and localization methods in computer vision are based on binary classification, in which discrimination between two classes is sought after by learning a function $f : \mathbb{X} \rightarrow \mathbb{Y}$, mapping the inputs $x \in \mathbb{X}$ to binary outputs $y \in \{-1, 1\}$. Alternatively, regression methods, where a mapping from one or more independent variables to a real valued variable is desired, can be employed to enrich classification power.

There are many kinds of classification techniques developed for various purposes. Among many options for classification, Support Vector Machines (SVMs) are probably the most popular due to the algorithm's

simplicity yet good performance even with a small number of training examples. The original SVM algorithm is a binary classifier, which requires learning a number of binary SVMs for the extension to multi-way classifier. Moreover, it cannot consider the more natural and non-trivial structure of the output $y \in \mathbb{Y}$. For example, when we attempt to detect complex objects such as trees, humans, and bicycles or obtain richer information --geometric structure, rigid/non-rigid deformation, etc.--about such objects, structural outputs based on multiple dependent variables may need to be provided. This problem is not reducible to simple classification or regression tasks in a straightforward manner; complex structures and associated inter-dependencies can be lost, resulting in limited prediction power of learned models.

Structural SVMs are a generalization of multi-class SVMs to provide structured information for unobserved data, given a trained model. Structured output prediction methods provide solutions to the incorporation of structures, within the learning stage itself. In other words, the mapping f is between inputs x and structured, complex, and multi-dimensional outputs y. As many such real-life prediction problems are NP-hard, methods have been developed, that allow for the exploitation of domain knowledge as part of the learning framework, resulting in efficient solutions. Structural SVMs are a family of methods generalizing the SVM paradigm, which allows for efficient solutions to such hard problems. On the other hand, latent SVMs can also be used to understand unobservable structures from data by using latent variables; in object detection problems, the variations in object appearance and structure can be handled by such latent variables. Note that latent SVMs is another variation of the original SVM method, closely related to structural SVMs. Both algorithms have been applied to many computer vision applications and have provided promising results.

Although our interest lies in learning structures via SVMs, we do not discuss the original SVM since it is already well-known and has sufficient references for various applications. Instead, our focus in this paper is on the structural SVM and the latent SVM algorithms, and their applications in object detection and localization problems.

This paper is organized as follows. We first present the basic concept of support vector machines for structured outputs in Section 2. A few important research outcomes involving structural SVMs and latent SVMs for the problem of object detection and localization are discussed in Section 3 and 4, respectively. We conclude our article with current challenges and future direction of this line of research.

Ⅱ. Structural Support Vector Machines

SVMs were originally developed for the binary classification problem [Cortes95]. Their theoretical foundation provides them with a set of attractive properties, such as a flexible choice of a loss function, convexity of the training problem, and non-linearity through kernelization. Formally, (hard-margin) SVMs can be defined as follows:

$$\min_w \frac{1}{2}\|w\|^2, \text{ s.t. } y_i(w \cdot \Phi(x_i) - b) \geq 1 , \qquad (1)$$

where $\| \cdot \|$ denotes L_2 norm, b is the bias and $\Phi(x_i)$ represents a feature map for input data x_i $(i = 1, \ldots, n)$. In short, SVMs seek a decision hyperplane that maximizes the geometric margin between the two target classes $y_i \in \{-1, 1\}$ as to minimize the generalization error in practice.

Structural SVMs [Tsochantaridis2004], on the other hand, are an extension of SVMs, that allow for the consideration of structure of the output y within the model learning stage, while maintaining the above desirable properties.

At a high level, structured learning can be thought of as multi-class learning, where each structure in $y \in \mathbb{Y}$ denotes a separate class and the prediction is a task of correctly assigning the input x to the correct structure according to $f(x, y) = w_y \cdot \Phi(x)$ with class-specific weight vector w_y. Thus, in this initial scenario, we seek to find the highest scoring structure amongst all the possible structures, with the help of $h(x) = \arg\max_{y \in \mathbb{Y}} f(x, y)$, leading to the following optimization problem,

$$\min_w \frac{1}{2}\|w\|^2$$
$$\text{s.t. } f(x_i, y_i) - f(x_i, \hat{y}) \geq 1 \quad (\forall i, y_i \neq \hat{y}), \qquad (2)$$

where $\hat{y}$ denotes a predicted structure. This quadratic convex problem generalizes across x, but not the outputs y and can result in a very large number of parameters, as there exist w_y for each structure y.

This analogy thus leads to a number of issues that need to be overcome by the structural SVMs. As the number of possible structures, i.e., classes, $|\mathbb{Y}|$ can be very large, the following four issues arise that need to be solved:

- Compact representation of $\mathbb{Y}$
- Efficient prediction using $h(x)$,
- Error calculation using loss $\Delta(y, \hat{y})$,
- Training algorithm sub-linear in $|\mathbb{Y}|$.

• Representation

First, to provide a compact representation of $\mathbb{Y}$, a so-called joint feature map $\Psi(x, y)$ was proposed instead of the standard feature map $\Phi(x)$ [Tsochantaridis2004]. This map extracts features from input-output pairs with contributions from the combinations of x and y, rather than only x. This leads to generalizations across both input and output, as well as the reduction in the number of parameters needed, as now $f(x, y) = w \cdot \Psi(x,y)$ instead of $f(x, y) = w_y \cdot \Phi(x)$. The problem thus becomes

$$\min_w \frac{1}{2}\|w\|^2$$
$$\text{s.t. } w \cdot \Psi(x_i, y_i) - w \cdot \Psi(x_i, \hat{y}) \geq 1 \quad (\forall i, y_i \neq \hat{y}). \qquad (3)$$

It is important to note that the choice of Ψ is problem specific and therefore needs to be chosen accordingly by the user. Examples of different Ψ for various domains can be found in [Tsochantaridis2004].

• Prediction

Even with a joint feature map Ψ, computing predictions using $h(x)$ may be very expensive as the number of outputs grows exponentially. Thus an exhaustive search might not always be possible. For this reason, some exploitation of the structures is required to allow for the decomposition of $\mathbb{Y}$ in a way that will allow for efficient maximization in $h(x)$. Again, this process is problem specific, and domain knowledge is required in order to select an appropriate formulation and optimization technique. Numerous useful examples of different solutions, to this step of structural SVMs, for various problems can be found in [Tsochantaridis2005].

• Loss

In order to be able to assess the quality of a prediction and therefore allow for a model to be learned, a loss function $\Delta : \mathbb{Y} \times \mathbb{Y} \rightarrow \mathbb{R}$ is necessary. Thus $\Delta(y, \hat{y})$ denotes the loss associated with a prediction $\hat{y}$, when the true value is y. In structural SVMs, similar to SVMs, this loss is minimized and consequently a structure classifier is learned during the training procedure. Again, the choice of a loss function is problem specific. In computer vision, examples of common loss functions are zero-one loss, Hamming loss, hierarchical multi-class loss and area overlap [Nowozin2011]. Thus far we have presented structural SVMs only for the separable case (hard-margin). With the introduction of a loss function and the allowance for mistakes, the optimization problem needs to be changed to allow for violations of the constraint in Eq. (3). This can be achieved with the introduction of slack variables and a penalty term for such violations. There are two formulations of the optimization problem, incorporating the slack-rescaling formulation,

$$\min_{w, \xi_i \geq 0} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$$
$$\text{s.t. } w \cdot \Psi(x_i, y_i) - w \cdot \Psi(x_i, \hat{y}) \geq 1 - \frac{\xi_i}{\Delta(y, \hat{y})} \quad (\forall i, y_i \neq \hat{y}), \qquad (4)$$

and the margin-rescaling approach with slightly different constraint formulation,

$$w \cdot \Psi(x_i, y_i) - w \cdot \Psi(x_i, \hat{y}) \geq \Delta(y, \hat{y}) - \xi_i. \qquad (5)$$

The main difference between the two formulations is that the slack-rescaling approach is scaling invariant, whereas the margin-rescaling approach is not. On the other hand, the latter method is simpler and thus easier to apply in practice.

• Training

Given the formulations of the structural SVM problem in Eq. (4) and (5), the remaining task is to present an efficient algorithm for finding optimal w, that is independent of the cardinality of $\mathbb{Y}$. Again, the possibly very large number of incorrect predictions $\hat{y}$ means that all constraints in the optimization cannot be enumerated. Therefore, a custom algorithm has been proposed in [Tsochantaridis2004]. This, so-called, cutting-plane algorithm can find an ϵ-accurate solution, whilst considering only $O(1/\epsilon)$ constraints, rather than the entire constraint space. This algorithm first finds the most violated constraint in Eq. (5) if the error is smaller than ϵ, and then the constraint is added to a working set of constraints, over which the quadratic program is solved in the next iteration. This is repeated, until convergence, when the working set remains unchanged. This algorithm has also been shown to be independent of the number of training examples. During training, a similar maximization step is required as in prediction. In many scenarios, however, the optimization technique for prediction can also be exploited within this loss-augmented step.

1. Kernelization

One of the main benefits of the SVM method is the ability to use kernel functions and thus allow for training of non-linear classifiers. Due to the fact that structural SVMs are based on SVM theory, by using a dual formulation, the optimization problem only depends on inner products in the joint feature space Ψ and thus allows for the use of joint kernel functions,

$$k((x,y),(x',y')) = <\Psi(x,y),\Psi(x',y')> , \quad (6)$$

where $< \cdot >$ denotes an inner product.

In the case of structural SVMs, the kernels become joint kernels as they operate between two input-output pairs. When using kernels, an explicit expression of Ψ is not necessary, however one has to ensure that both the prediction and loss-augmented training remain feasible.

2. Application

The structural SVM method along with the cutting-plane algorithm allows for learning of a wide variety of structured output problems. The generic nature of the method allows for such wide applicability, however the following set of implementations are required for each new problem.

- Joint feature map $\Psi(x,y)$,
- Loss $\Delta(y, \hat{y})$,
- Optimization of $h(x)$ in prediction and training.

Each of these require domain understanding and thus a problem tailored solution.

Ⅲ. Object Detection by Structural SVM

Structural SVMs for object detection and localization were first introduced by Blaschko and Lampert [Blaschko2008]. Their aim was to approach object localization with the sliding window method in a principled way. Rather than classifying individual image regions using positive and negative examples, their aim is a direct prediction of the bounding box of objects in images. In other words, rather than a binary classification problem, they posed the problem of object localization as a structured output task. Such formulation is much more natural, as the localization task is one of structured regression, whereby rather than a binary $\{-1,1\}$ space, the output space comprises the set of all possible bounding boxes, parameterizable by some coordinate measures, that, in addition, are dependent on each other. Computationally, their motivation was two-fold. First, the sliding window approach is inefficient, and secondly, no apparent method for training discriminant functions for object localization optimally is known.

1. Problem Formulation

In their setting, input images $\{x_1,...,x_n\} \subset \mathbb{X}$ have an associated annotation $\{y_1,...,y_n\} \subset \mathbb{Y}$. The task is to learn a mapping $h: \mathbb{X} \rightarrow \mathbb{Y}$, that predicts annotations on unseen images. The structured output space $\mathbb{Y} \equiv \{(\omega,t,l,b,r)\} \mid \omega \in \{-1,1\}, (t,l,b,r) \in \mathbb{R}^4\}$ consists of a vector of four indicators of bounding box location (top, left, bottom, right) and an object presence/absence label ω.

• Joint feature map

In their work, the margin-rescaling formulation (Eq. (5)) is explored, along with a joint kernel map (Eq. (6)) for localization. To allow for the use of such a kernel, the dual formulation of Eq. (5) is required and can be found by replacing,

$$w = \sum_{i=1}^{n} \sum_{\hat{y} \neq y_i} \alpha_{iy} (\Psi(x_i, y_i) - \Psi(x_i, y)) \tag{7}$$

in Eq. (5), with α_{iy} denoting the Lagrange multiplier. In their scenario, the joint kernel map exploits the fact that kernels are capable of size invariant image comparison. Thus cropping an image and subsequently applying an image kernel, allows for image region comparison. Their joint kernel map is defined as follows,

$$k((x,y),(x',y')) = k_x(x|_y, x'|_{y'}) \ , \tag{8}$$

where $x|_y$ denotes an image region within a bounding box y and $\Psi(x|_y)$ its representation in Hilbert space. An important property of Eq. (8) is that overlapping regions will have common features and statistics. Many other image kernels, such as spatial pyramids and pyramid match kernel, can all be used within this framework.

• Loss

To assess the quality of the performance of the learned model, the choice of $\Delta(y,\hat{y})$ is essential. Blaschko and Lampert have created a loss, based on the notion of area overlap, as used in VOC challenges in the past. The loss function has been defined as follows,

$$\Delta(y,\hat{y}) = \begin{cases} 1 - \dfrac{Area(y \cap \hat{y})}{Area(y \cup \hat{y})} & \text{if } y_{iw} = \hat{y}_{\omega} = 1 \\ 1 - \left(\dfrac{1}{2}(y_{i\omega}\hat{y}_{\omega} + 1)\right) & \text{otherwise} \end{cases} \tag{9}$$

where $y_{i\omega} \in \{-1,1\}$ denotes the absence or presence of an object, respectively. An interpretation of the above loss is that when the bounding boxes y_i and $\hat{y}_i$ are identical, the loss is 0, otherwise it is 1. This loss is invariant to both translation as well as scale, while providing a smooth measure to the degree of overlap, exploited in optimization.

• Optimization

Given both, the joint kernel map and the loss function, a suitable method for the maximization step in $h(x)$, for prediction and loss-augmented training, is necessary. The sliding window approach can be used in this case, however this is computationally inefficient. Only a subset of possible bounding boxes can be evaluated and thus this results in only an approximate solution. The authors suggest another method, that uses a branch-and-bound optimization strategy over the whole space $\mathbb{Y}$, that guarantees a globally optimal solution. This method employs a priority queuing procedure, ordered by the upper bound of the objective function. Iteratively, this method splits the space $\mathbb{Y}$ and operates on those individually, until only one bounding box remains in the priority queue [Blaschko2008].

• Results

Blaschko and Lampert tested their method on two separate datasets, namely the PASCAL VOC 2006 dataset and TU Darmstadt dataset. SURF descriptors [Bay2006] were used to create visual codewords that were used as the underlying feature space. They have found that structured learning, with linear kernel, consistently achieves tighter contours at the correct locations in images in the TU Darmstadt dataset, resulting in superior results compared to binary classification and to previously developed methods. Similarly, in the more challenging PASCAL VOC 2006 dataset, both precision and recall of the detection has been improved in comparison to binary classification when the chi-square

statistic is used. The employed method beat the VOC challenge winners in 5 out of 10 tasks, and in all except one, when compared to binary classification.

The authors showed that despite the use of a single feature set and a simple image kernel, superior results are achieved when compared to existing methods. They argue that this is due to the structural SVMs being able to exploit the full structure of the problem, rather than only a subset. This is in contrast to binary classification approaches, that include negative examples, which are possibly non-informative. Furthermore they argue, that unlike binary classification, structured learning can handle partial detections due to the flexibility of loss scaling.

In summary, the proposed method avoids prediction errors by considering the entire space of possible object locations, optimizes for the task of location directly, and appropriately handles partial detections, resulting in the observed superior results. All this is achieved efficiently by constraint generation and a branch-and-bound strategy within the structural SVM framework.

2. An Extension

Another approach for object detection by structural SVMs is proposed to handle more complex structure such as alignment, multiple aspects, occlusion, and variable number of objects in [Vedaldi2009]. The model in this work is formulated as structural SVMs with latent variables, where occlusion and multiple aspects can be handled by the latent variables, and the variable number of objects is encoded by the loss function.

In this framework, the joint feature map and the loss function involve latent variable as

$$(\hat{y}_x(w), \hat{z}_x(w)) = \arg\max_{(y,z)\in\mathbb{Y}\times\mathbb{Z}} w \cdot \Psi(x,y,z),$$

where $z \in \mathbb{Z}$ is a latent variable, and $\Psi(x,y,z)$ is a joint feature map. Given training data, the model parameter w is learned by minimizing the following objective function:

$$\min_w \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \Delta(y_i, \hat{y}_{x_i}(w), \hat{h}_{x_i}(w)),$$

where $\Delta(y_i, y, h)$ is a loss function. The objective function is not convex and requires iterative optimization step between the model parameter w and latent variable z. An extension of the Concave-Convex Procedure algorithm [Yu2009], a technique for margin scaling, is required for the slack rescaling.

Ⅳ. Part-Based Deformable Models by Latent SVM

Many object detection algorithms describe objects using holistic models, therefore it is not straightforward to handle variation in object appearances, typically caused by pose and viewpoint changes. Part-based modeling is a potential solution to overcome the limitation, but it is not practical to maintain models for all possible poses and viewpoints in both training and testing. Recently, Felzenszwalb et al. [Felzenszwalb2009] proposed an algorithm to address this problem using latent variables, which represent the configuration of an object. The algorithm only needs loosely labeled training data with arbitrary pose variations, and learns a single object model, regardless of potential variations of poses. Multiple models are however required to handle significant viewpoint changes. In other words, the training data do not include annotations about the parts of objects, but the appearance and layout of each part is learned during the training procedure automatically.

The latent SVM is employed to construct the classifier between object and non-object and learn the spatial layout of individual parts. In the testing phase, the detection score as well as deformation cost for each part are incorporated to compute overall response for an object. In contrast to the object detection algorithm using structural SVMs, the technique in [Felzenszwalb2009] needs to search for the optimal location and scale for target object searching by a sliding window method. However, this method provides structured output of the spatial configuration of multiple parts comprising the entire object based on latent SVM. In this section, we discuss how latent SVM is utilized for object detection in this framework and review the training and testing procedure for more practical understanding of the algorithm.

1. Object Detection Modeling by Latent SVM

One of the most important contributions of [Felzenszwalb2009] is that it combines a holistic model and a star-structured part-based model. Both kinds of models are implemented by linear filters; the model for an entire object is called the "root" filter. The score of the star-structured models, at a particular position and scale within an image, is composed of three components; the score of the root filter, the scores of the part filters, and the deformation cost for the deviation of the parts from their ideal locations, with respect to the root position. Clearly, to find objects in a scene, one wants to maximize the scores of the root filter and part filters, while minimizing the deformation cost.

The computation of the score for each filter is performed by a simple procedure. Denote a $w \times h$ filter by F and a feature pyramid by H. Let $\phi(H, p)$ be the vector obtained by concatenating the feature vectors from the $w \times h$ subwindow of H at $p = (x, y, l)$, where (x, y) and l represent location and scale in the feature pyramid, respectively. The response of the filter F for H at p is given by a simple dot product as

$$F' \cdot \phi(H, p), \tag{10}$$

where F' is the concatenation of the vectorized weight vectors F. Since there are n parts, we should add $(n+1)$ scores, including the one corresponding to the root filter. The score of a hypothesis is also given by a deformation cost that depends on the relative position of each part with respect to the root, given by

$$d \cdot \phi_d(dx, dy), \tag{11}$$

where $\phi_d(dx, dy) = (dx, dy, dx^2, dy^2)$ is deformation feature. The displacement of the i-th part relative to its anchor position is defined as $(dx, dy) = (x, y) - (2(x_0, y_0) + v)$, where v is the desired position vector of a part. Finally, after adding the bias term, the overall score for object detection with N parts is formally defined by

$$\sum_{i=0}^{N} F_i' \cdot \phi(H, p_i) - \sum_{i=1}^{N} d_i \cdot \phi_d(dx_i, dy_i) + b. \tag{12}$$

The score of a hypothesis z can be given by the dot product of the model parameter β and a vector of appearance-deformation feature vector $\psi(H, z)$, $\beta \cdot \psi(H, z)$, where

$$\beta = (F_0', ..., F_N', d_1, ..., d_N, b) \tag{13}$$

and

$$\psi(H, z) = (\phi(H, p_0), ..., \phi(H, p_N), - \phi_d(dx_1, dy_1), ..., -\phi_d(dx_N, dy_N), 1). \tag{14}$$

Note, that this dot product should be performed for every possible combination of the base location (the location and scale for a root filter) p_0 and the spatial variations of n parts of an object. Therefore, finding the best configuration of a deformable object in an image involves a huge amount of iterations, thus the reduction of the computational cost is an extremely important problem in this framework.

2. Efficient Search

The overall score of each root location, with the best possible configuration of the parts, needs to be computed for object detection in an image. The detection score can be expressed as

$$\begin{aligned} score(p_0) &= \max_{p_1, ..., p_N} \text{score}(p_0, ..., p_N) \\ &= \max_z \beta \cdot \Psi(H, z) \\ &= \max \sum_{i=0}^{N} F_i' \cdot \phi(H, p_i) - \sum_{i=1}^{N} d_i \cdot \phi_d(dx_i, dy_i) + b \end{aligned} \tag{15}$$

where H is a feature vector extracted from root location p_0. Since a naive algorithm to search every location and scale, with consideration of potential deformations of parts, is too costly, it is necessary to improve search speed by reducing redundant operations. Fortunately, the solution can be found by dynamic programming and generalized distance transforms efficiently in $O(NK)$ time, once filter responses are computed, where N is the

number of parts in the model and K is the total number of locations in the feature pyramid. The implementation of the search process is surprisingly simple, involving convolutions of linear filters to compute the responses for each object part including root and generalized distance transform to handle the deformation cost of parts; it is illustrated in Figure 4 in [Felzenszwalb2009].

3. Learning Latent SVMs

Recall a classifier that scores an example x with a function of the form

$$f_\beta = \max_{z \in Z(x)} \beta \cdot \Phi(x,z). \quad (16)$$

A binary label for x can be obtained by thresholding its score. In the classical SVM, we need to train β given training dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $y_i \in \{-1,1\}$ by minimizing the following objective function:

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i f_\beta(x_i)) \ , \quad (17)$$

where $\max(0, 1 - y_i f_\beta(x_i))$ is the standard hinge loss and C is the constant for regularization term. The issue in training a latent SVM is that it is a non-convex optimization problem, but it becomes convex, once latent values for positive examples are fixed. The optimization for training latent SVM is done by minimizing an auxiliary objective function, $L_D(\beta, Z_p) = L_{D(Z_p)}(\beta)$, where $D(Z_p)$ is derived from a training set D by restricting the latent values Z_p for the positive examples. The original problem involves latent variables, the solution is given by iterative procedure between choosing the best latent variable z for each positive example and optimizing β via gradient descent algorithm. Specifically, it is composed of the following two iterative steps:

1) Step 1: minimize $L_D(\beta, Z_p)$ over Z_p by selecting the highest scoring latent value for each positive example, $z_i = \arg\max_{z \in Z(x_i)} \beta \cdot \Phi(x_i, z)$.

2) Step 2: minimize $L_D(\beta, Z_p)$ over β by solving the convex optimization problem defined by $L_{D(Z_p)}(\beta)$.

The construction of initial (latent) part models is tricky, and the initial parts are selected heuristically from the high response subwindows within the region defined by the root filter. Note that the number of body parts are given as a parameter.

4. Object Detection Result by Latent SVM

For person detection, [Felzenszwalb2009] utilized a variation of the histogram of oriented gradients (HOG) feature [Dalal2005], and the sliding window method with multiple scales is used to localize the object. For each location and scale, the trained latent SVM is applied to PASCAL VOC datasets to detect objects. The performance of this algorithm is better than other techniques in most object classes, as illustrated in Table 3 of [Felzenszwalb2009].

V. Discussion

We discussed a recent trend related to support vector machines for the task of object detection, including two different but related techniques—structural SVMs and latent SVMs. Both techniques are employed for the object detection problem, particularly to benefit from and obtain structured outputs; the "structure" can be any dependent information, that is useful to overcome the limitations originating from the independence assumptions among related variables. The proposed techniques are successfully applied to object localization and configuration problems and show superior experimental results in challenging standard datasets.

We believe discriminative structure learning for object detection and localization is a meaningful step for more comprehensive image and scene understanding. However, the use of structural SVMs and latent SVMs is not straightforward since some critical information such as joint feature map and a loss function should be given

by the users. Also, it is questionable how well such discriminative learning works with large-scale data, thus efficient training and testing procedures should be developed to allow for practical use of structural and latent SVMs.

Acknowledgement

This research was supported in part by the Brain Korea 21 Project in 2012 and in part by the MKE (The Ministry of Knowledge Economy), Korea, under the "IT Consilience Creative Program" support program supervised by the NIPA (National IT Industry Promotion Agency) (C1515-1121-0003).

References

[Blaschko2008] M. B. Blaschko and C. H. Lampert, "Learning to Localize Objects with Structured Output Regression," In Proc. of European Conference on Computer Vision (ECCV), pp.2-15, Marseille, France, 2008

[Bay2006] H. Bay, T. Tuytelaars and L.J. Van Gool, "SURF: Speeded Up Robust Features," In Proc. of European Conference on Computer Vision (ECCV), pp.404-417, Graz, Austria, 2006.

[Cortes95] C. Cortes and V. Vapnik, "Support-Vector Networks. Machine Learning," 20(3), pp.273-297, Sep., 1995.

[Tsochantaridis2004] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support Vector Machine Learning for Interdependent and Structured Output Spaces," In Proc. of International Conference on Machine learning (ICML), New York, NY, USA, 2004.

[Nowozin2011] S. Nowozin, C. H. Lampert, "Structured Learning and Prediction in Computer Vision," Foundations and Trends in Computer Graphics and Vision 6(3-4). pp.185-365, 2011.

[Tsochantaridis2005] I. Tsochantaridis, Thorsten Joachims, T. Hofmann, and Y. Altun, "Large Margin Methods for Structured and Interdependent Output Variables," The Journal of Machine Learning Research, 6, pp.1453-1484, Dec., 2005.

[Vedaldi2009] A. Vedaldi and A. Zisserman, "Structured Output Regression for Detection with Partial Truncation," In Advances in Neural Information Processing Systems, Vol.21, MIT Press, pp.1928-1936, 2009.

[Yu2009] C.-N Yu and T. Joachims, "Learning Structured SVMs with Latent Variables," In Proc. of International Conference on Machine learning (ICML), Montreal, Canada, 2009.

[Felzenszwalb2009] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Model," IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9), pp.1627-1645, Sep., 2009.

[Dalal2005] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," In Proc. of IEEE Conference on Computer Vision Pattern Recognition, pp.886-893, San Diego, CA, 2005.

Jan Feyereisl

2010년 7월 PhD in Computer Science, The University of Nottingham, UK

2005년 7월 BSc in Computer Science, The University of Nottingham, UK

2011년 09월~Present Research Felllow, Dept. of Computer Science and Engineering, POSTECH, Korea

2010년 01월~2011년 08월 Research Fellow, Intelligent Modelling and Analysis Research Group, The University of Nottingham, U.K.

2010년 01월~2010년 08월 Research Assistant, Horizon: Digital Economy Hub, The University of Nottingham, U.K.

2009년 03월~2009/09월 Short-Term Research Fellowship, Marie-Curie Early Stage Training in Bioinformatics Optimisation (BIOPTRAIN), Poznan University of Technology, Poland

〈Research Interests〉 Empirical inference science, statistical learning theory, machine learning, computer vision, information security and biological computation

Bohyung Han

2005년 12월 Ph.D. in Computer Science, University of Maryland at College Park, MD, USA
2000년 8월 M.S. in Computer Engineering, Seoul National University, Seoul, Korea
1997년 2월 B.S. in Computer Engineering, Seoul National University, Seoul, Korea
2010년 7월~Present Assistant Professor, Dept. of Computer Science and Engineering, POSTECH, Korea
2010년 2월~2010년 7월 Assistant Professor, School of Electrical and Computer Engineering, UNIST
2008년~2010년 Researcher, Mobileye Vision Technologies, Princeton, NJ, USA
2006년~2007년 Researcher, Samsung R&D Center, Irvine, CA, USA
〈Research Interests〉 Computer vision, machine learning, pattern recognition, computer graphics