Multiple Instance Feature for Robust Part-based Object Detection

Zhe Lin
University of Maryland
College Park, MD 20742
zhelin@umiacs.umd.edu

Gang Hua
Microsoft Live Labs Research
Redmond, WA 98052
ganghua@microsoft.com

Larry S. Davis
University of Maryland
College Park, MD 20742
lsd@cs.umd.edu

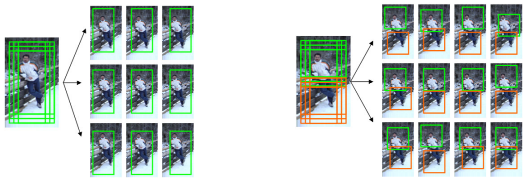
Abstract

Feature misalignment in object detection refers to the phenomenon that features which fire up in some positive detection windows do not fire up in other positive detection windows. Most often it is caused by pose variation and local part deformation. Previous work either totally ignores this issue, or naively performs a local exhaustive search to better position each feature. We propose a learning framework to mitigate this problem, where a boosting algorithm is performed to seed the position of the object part, and a multiple instance boosting algorithm further pursues an aggregated feature for this part, namely multiple instance feature. Unlike most previous boosting based object detectors, where each feature value produces a single classification result, the value of the proposed multiple instance feature is the Noisy-OR integration of a bag of classification results. Our approach is applied to the task of human detection and is tested on two popular benchmarks. The proposed approach brings significant improvement in performance, i.e., smaller number of features used in the cascade and better detection accuracy.[1] [2]

1. Introduction

Pose articulation and local part deformation are among the most difficult challenges which confront robust object detection. For scanning-window-based object detectors, they produce one common issue which we call *feature misalignment*. It refers to the phenomenon that features which fire up in some positive detection windows do not fire up in other positive detection windows.

Holistic methods [1,8,15,19,25] that concatenate local visual descriptors in a fixed order, such as HOGs [1], are inevitably cursed by this phenomenon. Part-based approaches such as [7,11,22] handle misalignment by training part classifiers and assembling their responses. Even though they can handle part misalignment somehow, training a classifier for every part might be computationally expensive. Intuitively, adopting a part-based representation in the holistic method, if appropriately modeled, may largely alleviate the feature misalignment issue.


(a) Global or Holistic

(b) Local, Part-based

Figure 1. Multiple instances. (a) An example of global (holistic) instances (9 instances from uniform neighborhood); (b) An example of local (part-based) instances for a two-part object (only 12 are shown among 9 × 9 instances). It is obvious that part-based multiple instances better capture articulation of a deformable object since each part can have independent placement.

Previous work [2,4,6,18] has approached this problem by allowing object parts to scale and/or shift spatially for better feature alignment, and leveraging machine learning algorithms to determine the optimal configuration of the visual parts. However, most of these approaches [4,6,18] manually picked the object parts to learn the configuration. There is no guarantee that the manual part selection would be optimal for detection.

On the other hand, some boosting-based approaches [5,16,19,23,25] implicitly select visual parts, but they do not take feature misalignment into consideration. Meanwhile, multiple instance learning [14,21] has been demonstrated to effectively handle object misalignment at the holistic level (i.e., the whole detection window). A natural question to ask is: can we extend the idea of multiple instance learning to handle the problem of feature misalignment? The answer is indeed quite straightforward by allowing multiple instances at the part levels. Figure 1 illustrates the difference between global [14,21] and local part-based multiple instance schemes.

[1] The majority of this work is carried out when Zhe Lin is an intern at Microsoft Live Labs Research.

[2] We illustrate and demonstrate our approach mainly in the context of human detection while the general approach we propose can be directly applied to many other object categories such as bikes, cars, animals, etc.

Hence, we introduce a framework for learning part-based object detectors which are robust to feature misalignment, and thus are robust to pose variation and local part deformation. The main idea is the introduction of a more powerful feature, or equivalently weak learner, into the Boosting framework for object detection. The new feature, called a *multiple instance feature*, is the Noisy-OR aggregation of the classification results of a bag of local visual descriptors. Essentially, each bag of local visual descriptors is considered as one local part of the targeted object.

To efficiently utilize multiple instance features, in contrast to Dollar *et al.*'s work [2], which runs multiple instance learning on a randomly selected set of parts, we propose a novel *seed-and-grow* scheme. In essence, at each feature selection step of boosting, we first select an optimal ordinary feature, e.g., a decision stump or a decision tree. Then based on the position of the ordinary feature, we grow out to define the local part (i.e., a bag of descriptors). A multiple instance boosting algorithm [21] is further performed to obtain the optimal multiple instance feature.

Our learning process is very efficient because of the proposed seed-and-grow approach. Additionally, we learn from multiple instances of both negative and positive examples and thus learning is done through very tight and fair competition between positive and negative bags. An overview of our training and testing algorithms is shown in Figure 3. Our main contributions are three-fold:

- A novel boosting-based learning framework to automatically align features for object detection.
- Multiple instance features for modeling part misalignment/deformations.
- A `seed-and-grow` scheme to efficiently construct the multiple instance feature.

The approach is applied to the task of human detection (see a survey [13]) and classification results are obtained on commonly used pedestrian benchmarks.

2. Multiple Instance Features

2.1. Aggregating Multiple Instances

A multiple instance feature basically refers to an aggregation function of multiple instances. More specifically, given a classifier C, it is the aggregated output, y, of a function, f, of classification scores, $\{y_j\}_{j=1...J}$, of multiple instances, $\{x_j\}_{j=1...J}$:

$$y = f(y_1, y_2, ...y_J) = f(C(x_1), C(x_2), ...C(x_J)). \quad (1)$$

Suppose we have a set of labeled bags $\{x_i, t_i\}_{i=1...n}$ where label $t_i \in \{0, 1\}$. A bag x_i consists of a set of instances: $x_{ij}, j = 1...N_i$. The number of bags is n, and the number of instances is $N = \sum_{i=1}^{n} N_i$. Given a real-valued (real-valued output) classifier C, the score y_{ij} of an instance x_{ij} can be computed as: $y_{ij} = C(x_{ij})$. The probability of an instance x_{ij} being positive is modeled by the standard logistic function [21] as: $p_{ij} = p_{ij}^{+} = \frac{1}{1+exp(-y_{ij})}$.

In [9], a multiple instance learning problem is formulated as the maximization of *diverse density* which is a measure of intersection of the positive bags minus the union of negative bags. The diverse density is probabilistically modeled using a Noisy-OR model for handling multiple instance learning problems. Under this model, the probability of a bag being positive is given by $p_i = 1 - \prod_{j=1}^{N_i}(1 - p_{ij})$. We modify this by taking the geometric mean to avoid numerical issues when N_i is large, so instead, we use $p_i = 1 - \left(\prod_{j=1}^{N_i}(1 - p_{ij})\right)^{1/N_i}$. Intuitively, the model requires that at least one instance in each positive bag has a high probability, while all instances in each negative bag have low probabilities. In terms of scores, the multiple instance aggregated score y_i is computed from instance scores y_{ij} as:

$$y_i = \log\left(\left(\prod_k (1 + e^{y_{ij}})\right)^{1/N_i} - 1\right), \quad (2)$$

which can be easily derived from the relation between p_i and y_i, *i.e.* $p_i = \frac{1}{1+exp(-y_{ij})}$. We refer to the above aggregated score computed from Equation 2 as multiple instance feature.

2.2. Learning Multiple Instance Features

Learning multiple instance features involves estimation of the function f and instance classifier C. Assuming that the aggregation function f is given as the Noisy-OR integration, we can learn the instance classifier C via multiple instance boosting (MILBoost), which was originally introduced in [21], as an application of the diverse density to rare event detection problems to handle misalignment of examples in training images. In MILBoost, each training example is represented as multiple instances residing in a bag, and the bag and instance weights are derived under the Anyboost framework [10] which views boosting as gradient descent.

Given an adaboost classifier $C = \{\lambda_t, h_t(\cdot)\}_{t=1...T}$, the score y_{ij} of an instance x_{ij} is computed as a weighted sum of scores from all weak classifiers $h_t(x)$:

$$y_{ij} = C(x_{ij}) = \sum_t \lambda_t h_t(x_{ij}), \quad (3)$$

where $h_t(x) \in \{-1, +1\}$ and the weight $\lambda_t > 0$.

The overall likelihood assigned to a set of labeled training bags is

$$\mathcal{L}(C) = \prod_i {p_i}^{t_i} (1 - p_i)^{1-t_i}. \tag{4}$$

Hence, the learning task is to design a boosted classifier C so that the likelihood $\mathcal{L}(C)$ is maximized. This is equivalent to maximize the log-likelihood

$$\log \mathcal{L}(C) = \sum_i t_i \log p_i + (1 - t_i) \log (1 - p_i). \tag{5}$$

According to the Anyboost framework [10], the weight of each example is given as the partial derivative of the likelihood function (which can be regarded as a negative cost function) w.r.t. a change in the instance score y_{ij}. Hence, the instance weights w_{ij} can be derived (see appendix for the detailed derivation) as:

$$\frac{\partial \log \mathcal{L}(C)}{\partial y_{ij}} = w_{ij} = \frac{1}{N_i} \frac{t_i - p_i}{p_i} p_{ij}, \tag{6}$$

where the weights for positive and negative examples are $w_{ij}^+ = \frac{1}{N_i} \frac{t_i - p_i}{p_i} p_{ij}$ and $w_{ij}^- = -\frac{1}{N_i} p_{ij}$, respectively. Interestingly, the formula for the weight values are exactly $1/N_i$-scaled versions of the ones derived in [21]. The boosting process is composed of iterative weight updates and additions of weak hypotheses. Based on [10,21], each round of boosting is conducted through the following two steps:

- search for the optimal weak classifier $h(x) \in \{-1, +1\}$ which maximizes the energy function $\psi(h)$:

  $$\psi(h) = \sum_{ij} w_{ij}^t h(x_{ij}) \tag{7}$$

  under the current weight distribution w_{ij}^t and set the optimal weak hypothesis as h^{t+1}.
- given the previous classifier $C^t(x) = \sum_\tau^t \lambda_t h^t(x)$ and the currently chosen weak classifier h^{t+1}, search for the optimal step size λ_{t+1}^* so that $\mathcal{L}(C^t(x) + \lambda_{t+1} h^{t+1}(x))$ is maximized, and set $C^{t+1}(x) = C^t(x) + \lambda_{t+1}^* h^{t+1}(x)$.

We learn multiple instance features at the part level, and adopt them as potential weak classifiers for learning boosted deformable object detectors, which is discussed in Sec. 3 in detail.

3. Part-based Object Learning

Modeling deformable objects as a composition of parts is a well-studied approach to object recognition. For example, the pictorial structure model [3] represents an object as a set of visual parts placed in a deformable configuration, where the appearance of each part is modeled separately and deformation is modeled as a set of spring-like connections between parts. In object detection, most existing part-based approaches [7, 11, 12, 22] manually decompose an object into semantic parts such as head-shoulder, torso, and legs. In contrast, we aim for automatically selecting discriminative parts through training on labeled examples.

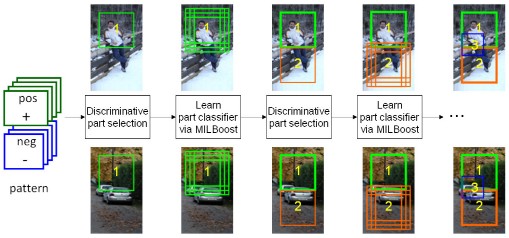

pos
+
neg
-
pattern
Discriminative part selection
Learn part classifier via MILBoost
Discriminative part selection
Learn part classifier via MILBoost
...
1
2
3

Figure 2. An example of part-based object model learning. The learning process consists of iterative modules of discriminative part selection and part classifier learning.

3.1. Learning Framework

Suppose the object of interest to be modeled consists of L parts (L is given). Each part is assumed to be square or rectangular blocks or regions[3]. The learning algorithm should automatically figure out the most discriminative combination of L parts (or regions) among the set of N_p candidate visual parts: $\mathcal{P} = \{P_1, ...P_{N_p}\}$. A part P_j is represented as a set of instances which correspond to a set of locally neighboring (possibly overlapping) local image regions.[4] From each instance (an image region) of a part, a predefined feature set F_j is computed and a subset of those features are chosen to contribute to the final strong classifier. From each selected part, a fixed number of weak classifiers are learned and added to the final strong classifier.

Our object model learning framework is visualized in Figure 2. It adopts an efficient `seed-and-grow` scheme to boost multiple instance features. The framework first selects the most discriminative part by locating a `seed` feature from the whole feature pool, and then training the part's classifier using multiple instance boosting by adding features to that part (*i.e.* `growing`). The two modules are repeated to progressively select parts and learn their classifiers. Note that the learning process focuses on a single part (region) at a time, *i.e.*, multiple instances are only considered for the currently active part. After a part is learned, all features belong to that part are removed from the current feature pool so that it will not be selected again. The first feature (or weak classifier) for the next part is chosen from the remaining (unexplored) feature pool as in the ordinary

[3] The regions can be arbitrary regions in a detection window and need not to be semantic object parts.

[4] Given a potential part and a training example, a bag includes (feature) instances extracted from a set of neighboring local image regions around the part's default location. The set of neighboring image regions can be obtained either by jittering the default part region or uniformly sampling regions in spatial-scale space around the default location.

boosting approach. Then the region (or block) corresponding to that part is localized. Next, subsequent (2nd, 3rd,...) weak hypotheses of that localized part are learned using MILBoost [21] (see Sec. 2.2). In this boosting framework, part selection and weak classifier training are simultaneously performed. Part selection provides a weak learner pool for the current boosting round. The final strong classifiers are constructed by boosting multiple instance features from selected parts. The multiple instance features are directly used as additive weak hypotheses to construct the final strong classifier (see Figure 3).

3.2. Weak Classifiers for Multiple Instance Features

As mentioned before, for constructing a cascade classifier, the training algorithm needs to repeatedly add weak learners $h_t(x)$ to the classifier C. Each weak learner $h_t(x)$ is chosen based on the criterion of maximizing $\psi(h) = \sum_{ij} w^t_{ij} h_t(x_{ij})$. Instead of binary stumps [20], we use more informative domain-partitioning weak classifiers as in [17], where each weak hypothesis makes its prediction based on a partitioning of a feature's domain $\mathcal{X}$ into disjoint regions $X_1, ...X_K$. Suppose we partition the domain of $h(x)$ into K bins and assign binary values $c_k = \{+1, -1\}$ to each bin $k = 1, 2..K$, *i.e.*, $c_k = h(x)$ for $x \in X_k$. The goal is to find the optimal weak hypothesis $h^*(x)$ (consisting of the $K-1$ partition thresholds and the binary values $\{c_k\}_{k=1,2...K}$) such that $\psi(h)$ is maximized. Finding partition thresholds is equivalent to learning a binary decision tree with $\lfloor log_2 K \rfloor$ levels, *e.g.* for $K = 8$, we learn a three-level decision tree classifier. For each feature, a greedy optimization process is applied to adjust the partition thresholds (by an iterative binary splitting algorithm) and optimal score values c_k for the partitioned intervals are determined so that $\psi(h)$ can be minimized for that feature.

For each k and for $b \in \{-1, +1\}$, let

$$W^k_b = \sum_{ij:x_{ij} \in X_k \wedge t_{ij} = b} w_{ij}. \qquad (8)$$

Then, $\psi(h)$ in Equation 7 can be rewritten as:

$$\psi(h) = \sum_k \sum_{ij:x_{ij} \in X_k} w_{ij} h(x_{ij}) = \sum_k c_k (W^k_+ + W^k_-). \qquad (9)$$

This means that we only need to choose c_k based on the sign of $W^k_+ + W^k_-$, *i.e.*, $c_k = sgn(W^k_+ + W^k_-)$.

All that remains is to estimate the step size α_t for the chosen weak hypothesis $h_t(x) = h^*(x)$ so that the objective function $\mathcal{L}(C + \lambda_t h_t)$ is maximized. This is done by line search in some interval $[0, \gamma]$ ($\gamma = 5$ and search step size as 0.01 are shown to be adequate in our experiment).

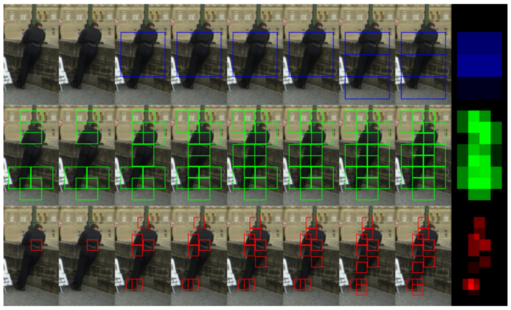


Figure 4. Visualization of an example feature selection process (better viewed in color). Red, green and blue square blocks denote 1st, 2nd and 3rd scale/resolution level features respectively. Columns from left to right indicate features selected in different stages of the learning algorithm, and the final column shows accumulated evidence.

The resulting step sizes λ_t from the above process are uniformly assigned to all intervals of a feature's domain. This poses the question of how we can generalize it to confidence-rated step sizes (nonuniform step sizes λ^k_t for different interval k). We introduce a greedy optimization to improve the step sizes as follows:

```
• initialize step sizes $\lambda_{t,k} = \lambda_t, k = 1, 2...K$
• for k=1:K
    – search over an interval $(\lambda_t - \delta, \lambda_t + \delta)$, find maximum likelihood estimation $\lambda_{t,k} = \lambda^*_{t,k}$ so that $\mathcal{L}(C, \lambda^*_{t,1}, ..., \lambda^*_{t,k-1}, \lambda_{t,k}, ...\lambda^K_t)$ is maximized.
• endfor
• return $\lambda^*_{t,1}, ..., \lambda^*_{t,K}$.
```

The overall training and testing algorithms are shown in Figure 3. We assume that both positive and negative examples consist of multiple instances and reside in bags so that a discriminative classifier can be learned via a tight competition between positive and negative examples. An example visualization of part selection is shown in Figure 4. As we can see, most discriminative parts are selected in legs, head, head-shoulder areas and more features are chosen from salient regions around human silhouette boundaries.

4. Implementation

We use multi-scale, multi-resolution shared HOG feature descriptors as the feature pool of our object learning and detection algorithms. The descriptor is a multi-resolution generalized version of histograms of oriented gradients (HOGs) [1] and is formed by extracting and concatenating dense descriptor elements from multiple resolutions. A descriptor element is a 36-dimensional vector computed from a 2×2-cell spatial block. Descriptor elements computed at higher pyramid levels correspond to larger spatial blocks at lower resolutions. We concatenate all descriptor elements (from three different resolution levels in which

```
TRAINING ALGORITHM
Input:
  ⋆ Labeled training examples $\{x_i, t_i\}_{i=1...N}$, where $t_i \in \{0, 1\}$.
  ⋆ The number of parts to learn $L$ and a pool of candidate parts $P_j, j = 1, 2..N_p$. Each part $j$ corresponds to a set of bags $\{x_{ij}, t_i\}_{i=1...N}$, where each bag $x_{ij}$ consists of instances $\{x_{ijk}, t_i\}_{i=1...N}$.
Initialization:
  ⋆ Initialize weights $w_i, i = 1...N$ such that $w_{pos} = \frac{N_{neg}}{N_{pos}+N_{neg}}$ and $w_{neg} = -\frac{N_{pos}}{N_{pos}+N_{neg}}$.
  ⋆ Set $F^1 \leftarrow F$, classifier $C \leftarrow \phi$ (empty set), and example scores $y_i \leftarrow 0, i = 1...N$, where $F = F_1 \cup F_2 ... \cup F_{N_p}$ is the set of all features from all candidate parts.
Training:
For $l = 1...L$:
  ⋆ From $F^l$, form a weak learner $h$ (corresponding to a feature $f_{mn}$) to maximize $\sum_{i=1}^{N} w_i h(x_i)$ under the ordinary adaboost setting (no multiple instances are allowed), and set $h_1^l = h$.
  ⋆ Localize a part based on feature $f_{mn}$ and set the part index as $IDX(l) = m$.
  ⋆ Train a MILBoost classifier $C^l$, $y = C^l(x) = \sum_{t=1}^{T_l} \lambda_t^l h_t^l(x) = \sum_{t=1}^{T_l} c_t^l(x)$.
  ⋆ Update the overall strong classifier as: $C \leftarrow C + C^l$.
  ⋆ Use Equation 2 to compute the multiple instance features $y_{il}$ for all $i = 1...N$.
  ⋆ Update example scores by boosting multiple instance features $y_{il}, i = 1...N$, $y_i \leftarrow y_i + y_{il}$.
  ⋆ $F^{l+1} = F^l - F_m$. (remove all features belonging to part $m$ from the current feature pool.)
  ⋆ Set the rejection threshold $rth(l)$ of stage $l$ as the minimum positive score: $\min_{t_i=1}\{y_i\}$ .
Output: Overall classifier $C = \{C^l\}_{l=1...L}$ and the set of rejection thresholds $\{rth(l)\}_{l=1...L}$.
```

```
TESTING ALGORITHM
For a test example $x$, set the initial score $y = 0$
For $l = 1...L$:
  ⋆ Use Equation 2 to compute the multiple instance feature $y_l$, and update the score $y \leftarrow y + y_l$.
  ⋆ if $y < rth(l)$, reject; else continue.
Decide positive if $l = L$ and $y \geq rth(L)$; else negative. Detection confidence: $prob = \frac{1}{1+exp(-y)}$.
```

Figure 3. Part-based Object Learning and Testing Algorithms.

consecutive levels differ by a factor of 2) which are relevant to a detection window to form the descriptor for that window. Hence, for a typical 64×128 pedestrian detection window, there are total of $105 + 21 + 3 = 129$ spatially overlapping blocks. The dimensionality of the descriptor is $36 \times 129 = 4644$.[5]

In the object learning algorithm implementation (see Figure 3), each training and testing example x_i represents a 64×128 image patch, and each candidate part P_j is defined as a HOG block (one of 129 multi-resolution blocks) and its feature pool F_j consists of 36 features (*i.e.* HOG features of a block). The overall set of features is $F = F_1 \cup F_2 ... \cup F_{N_p}$ and the feature set of each part P_j is represented as the feature pool $F_j = \{f_{j1} ... f_{jd}\}$ for part j. Each part j of an example x_i is modeled as a bag denoted as x_{ij} and instances of the bag is denoted as x_{ijk}. We fix the number of weak learners for a part to be a constant ($T_l = T, l = 1, 2...L$) in our current implementation.

In practice, instead of adding a fixed number of weak classifiers for each part, we use an adaptive method by adding weak learners until the contribution of the next weak hypothesis falls below some threshold $\delta > 0$, *i.e.* terminate adding weak hypothesis when $\sum_{ik} w_{ijk} h_t(x_{ijk}) < \delta$ (where i, j, and k denote example, part, part instance, respectively) or the step size is small $\lambda_t < \delta$.

5. Experiments

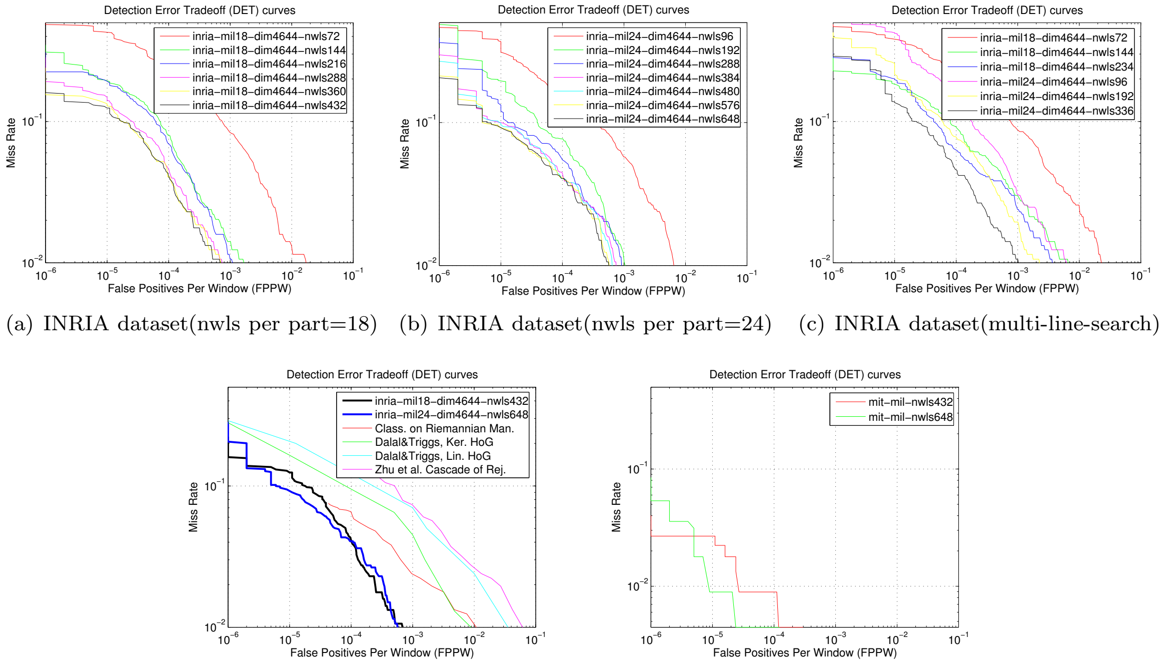
5.1. Datasets

We use the INRIA person dataset[6] [1] and the MIT-CBCL pedestrian dataset[7] [12, 15] for performance evaluation. In these datasets, training and testing samples all consist of 64×128 image patches. Negative samples are randomly selected from raw (person-free) images; positive samples are cropped (from annotated images) such that persons are roughly aligned in location and scale. The MIT-CBCL dataset contains

[5] In this section and the algorithm in Figure 3, the meanings of subscripts i, j, k are example, part and part instance, respectively, and are different from previous sections.

[6] http://lear.inrialpes.fr/data

[7] http://cbcl.mit.edu/software-datasets/PedestrianData.html



(a) INRIA dataset(nwls per part=18) (b) INRIA dataset(nwls per part=24) (c) INRIA dataset(multi-line-search)

(d) INRIA dataset(comparison) (e) MIT-CBCL dataset(evaluation)

Figure 5. Performance evaluation results on the INRIA dataset (nwls means number of weak classifiers). (a) Evaluation of our part-based detectors (18 weak classifiers for a part) with increasing numbers (72-432) of weak classifiers. (b) Evaluation of our part-based detectors (24 weak classifiers for a part) with increasing numbers (96-576) of weak classifiers. (c) Evaluation of our (multi-line-search) part-based detector. (d) Two detectors trained using our part-based approach are compared to HOG-Ker.SVM [1], HOG-Lin.SVM [1], HOG Cascade [25], and Classification on Riemannian Manifolds [19]. The results of [1] are approximate and are obtained from the original paper, and the results of [19, 25] are obtained by running their original detectors on the test data. (e) Evaluation of the two detectors (trained on the INRIA dataset) on the MIT dataset.

Table 1. Other comparison results on the INRIA dataset. **Miss rates(%)** (with respect to FPPW values) are compared. Results of [2, 6, 8, 14] are approximate and are obtained from their original papers for comparison purposes.

FPPW	[2]	[14]	[6]	[8]	nwls432	nwls648
1e-6	N/A	N/A	20.9	16.0	16.0	20.6
1e-5	15.0	7.2	12.7	6.0	12.5	9.0
1e-4	4.0	4.2	5.8	2.5	4.0	4.0
1e-3	1.5	<1.5	1.6	<1.0	0.7	0.4

924 front/back-view positive images, and the INRIA dataset contains 2416 positive training samples and 1218 negative training images plus 1132 positive testing samples and 453 negative testing images. Compared to the MIT dataset, the INRIA dataset is much more challenging due to significant pose articulations, occlusion, clutter, viewpoint and illumination changes.

5.2. Performance Evaluation

In training and testing, we quantize the scale space by a factor of $\sigma = 2^{1/4} = 1.1892$ and scan windows by a stride of 8 pixels. Negative examples are generated by exhaustively scanning the original negative images. We use Detection-Error-Tradeoff (DET) curves [1], plots of miss rates versus false positives per window (FPPW) for detection performance evaluation. Note that all of our result curves are generated using 1281 different thresholds $(-64 : 0.1 : 64)$ applied to the final strong classifier.

5.2.1. Evaluation on the INRIA Dataset

Figure 5(a) and 5(b) show performance of our detectors learned with 18 and 24 weak classifiers, respectively. We can see that the deeper into the cascade (*i.e.* the more parts learned for the cascade), the better the performance achieved, while the difference gets smaller and performance generally converges at some point (after adding certain number of parts). This indicates that parts learned in earlier stages of the cascade take much more important role than ones learned in later stages. Figure 5(c) shows performance of our detectors extended using multi-line-search-based learning algorithm. The same trends of performance improvement and convergence are observed as in the cases of Figure 5(a) and 5(b). On the other hand, even though richer weak classifiers are learned and fewer parts are required to complete training, the performance does not improve compared to the results of detectors using binary decision tree (Figure 5(a) and 5(b)). This implies that heavily exploring information in small portions of all available parts can lead to over-fitting; instead it might be better to choose an appropriate number of weak learners for each part so that broader information can be utilized in training by exploring more visual parts.

Figure 5(d) shows performance comparison of our approach with previous techniques [1, 19, 25]. Both of our listed detectors perform better than the other approaches. The results are also compared to [2, 6, 8, 14] on individual FPPW values (see Table 1). The table indicates that our approaches are comparable to the most recent pedestrian detectors.

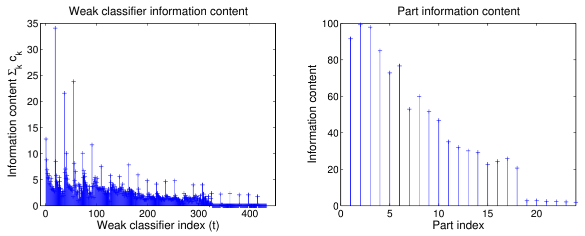
5.2.2. Evaluation on the MIT Dataset

For cross validating on different datasets, instead of training on the MIT dataset, we use two of our trained detectors on the INRIA dataset and tested them on the MIT dataset; the result is shown in Figure 5(e). Even though we have not directly trained on this dataset, the performance of our detectors achieve near perfect detection performance, comparable to the detector [1] which is directly trained on the MIT dataset. This indicates that our detectors have good generalization performance.

5.3. Analysis on Training and Testing

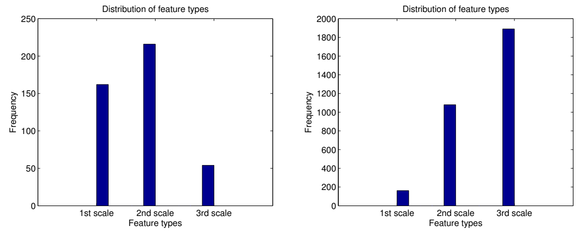
We also provide detailed analysis of our soft cascade learning and testing results using the cascade classifier `inria-dim4644-nwls432`. Figure 6(a) shows feature information contents w.r.t. the number of weak learners in the cascade. We can see that the information content[8] of weak classifiers monotonically decreases in each part and increases to a high value at every part-to-part switching point. In contrast, in the part-level, as shown in Figure 6(b), a part's information content tends to decrease to 0. This is reasonable because we constrain the feature pool to a selected part when learning part classifiers (corresponding to the inner loop of the learning algorithm), while a part's outer loop is a greedy boosting algorithm, where each part is a weak classifier.

The distribution and occurrence frequency of feature types in the cascade are shown in Figure 7. More than half of the features are chosen from the 2nd and 3rd scales/resolutions as shown in Figure 7(a). The importance of features extracted at lower resolutions (higher scales) is more clearly illustrated in Figure 7(b) by showing the selection frequencies of each feature type. Higher scale features are much more likely to be chosen than the 1st scale features. This further validates the usefulness of multi-scale, multi-resolution descriptors. Figure 8(a) shows the score traces of the first 5000 test windows based on the learned soft cascade classifier. We observe that most of the positive and negative test windows are well separated in earlier stages of the cascade.



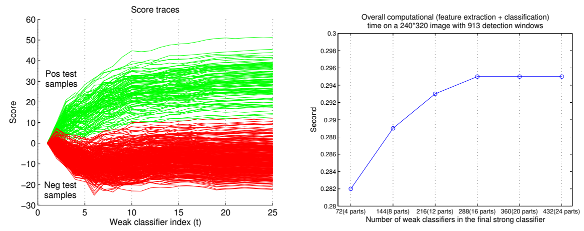
(a) Weak learner's info. content (b) Part's info. content

Figure 6. Analysis on the learning result.



(a) Total number of times (b) Frequency

Figure 7. Distribution of feature types being selected.



(a) Score trace (b) Detection time

Figure 8. Analysis on the testing results. (a) An example of score traces for 5000 (1132 positive and 3868 negative) test examples based on our learned soft cascade; (b) Detection time w.r.t. the number of weak learners.

5.4. Computational Requirements

Our system is implemented in C++. Typically, the training process takes less than half a day and testing a 320 × 240 image (scanning 913 windows) less than 0.3 seconds. Figure 8(b) shows a plot of computational time with respect to the number of weak classifiers. The propagation through the cascade classifier takes a negligible amount of time compared to feature computation. Our testing approach can be further optimized for speed using the multiple instance pruning techniques [24].

5.5. Qualitative Results

Some qualitative results of our detectors on the INRIA test images are shown in Figure 9. Given raw detection results, windows are simply merged based on their overlapping ratios.

[8] defined as $\nu = \sum_k |c_k|$, the larger ν is the more evidence the weak classifier contribute to final scores on average.

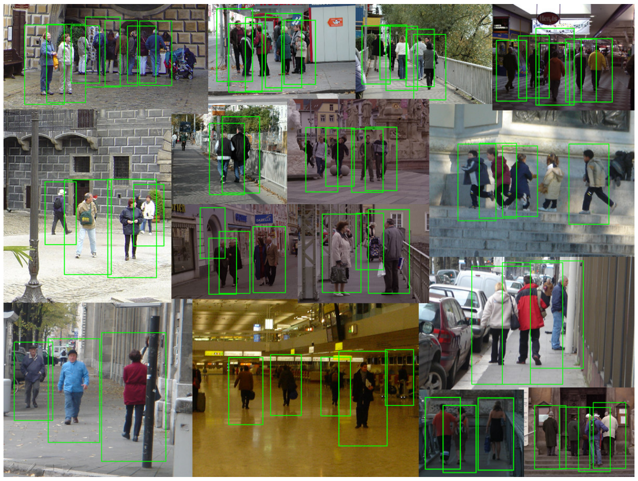


Figure 9. Some qualitative detection results.

6. Conclusion

We proposed a part-based, deformable object model learning approach. The approach models object shape articulations by considering multiple instances for each part. Multiple part instances are generated by jittering default local image patches spatially around its neighborhood. Specifically, it selects the most discriminative visual parts and learns deformable part detectors in a single, unified boosting framework. The framework is constructed as soft cascade learning with part-level multiple-instance boosting. In contrast to previous holistic detection approaches which need a large set of training examples for handling variations in pose articulation and obtaining reasonable detectors, our approach is more flexible in dealing with large pose variation or labeling misalignment in the training set. Our approach models an object as a set of parts, but is still sensitive to severe occlusion, hence the combination of our approach with occlusion analysis schemes would make detection of occluded objects more accurate. Additionally, it is possible to extend current part transformation (spatial translation) to more general cases (*e.g.* similarity or affine transformation) in order to better handle shape articulations.

Appendix

Derivation of Equation 6:

$$\frac{\partial \log \mathcal{L}(C)}{\partial y_{ij}} = \frac{\partial \left(\sum_i t_i \log p_i + (1-t_i)\log(1-p_i)\right)}{\partial y_{ij}}$$

$$= \frac{t_i}{p_i}\frac{\partial p_i}{\partial y_{ij}} - \frac{1-t_i}{1-p_i}\frac{\partial p_i}{\partial y_{ij}} = \frac{t_i - p_i}{p_i(1-p_i)}\frac{\partial p_i}{\partial p_{ij}}\frac{\partial p_{ij}}{\partial y_{ij}}$$

$$= \frac{t_i - p_i}{p_i(1-p_i)}\frac{1}{N_i}\frac{1-p_i}{1-p_{ij}}p_{ij}(1-p_{ij}) = \frac{1}{N_i}\frac{t_i-p_i}{p_i}p_{ij}.$$

References

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

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

[3] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial Structures for Object Recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[4] P. F. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. CVPR, 2008.

[5] I. Laptev. Improvements of object detection using boosted histograms. BMVC, 2006.

[6] Z. Lin and L. S. Davis. A pose-invariant descriptor for human detection and segmentation. ECCV, 2008.

[7] Z. Lin, L. S. Davis, D. Doermann, and D. DeMenthon. Hierarchical part-template matching for human detection and segmentation. ICCV, 2007.

[8] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. CVPR, 2008.

[9] O. Maron and T. L. Perez. A framework for multiple instance learning. NIPS, 1998.

[10] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. NIPS, 1999.

[11] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. ECCV, 2004.

[12] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Trans. PAMI*, 23(4):349–361, 2001.

[13] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE Trans. PAMI*, 28(11):1863–1868, 2006.

[14] J. Pang, Q. Huang, and S. Jiang. Multiple instance boost using graph embedding based decision stump for pedestrian detection. ECCV, 2008.

[15] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In Proc. of Intelligent Vehicles, 1998.

[16] P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. CVPR, 2007.

[17] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[18] D. Tran and D. A. Forsyth. Configuration estimates improve pedestrian finding. NIPS, 2007.

[19] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifold. CVPR, 2007.

[20] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. CVPR, 2001.

[21] P. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. NIPS, 2005.

[22] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. ICCV, 2005.

[23] B. Wu and R. Nevatia. Optimizing discrimination-efficientcy tradeoff in integrating heterogeneous local features for object detection. CVPR, 2008.

[24] C. Zhang and P. Viola. Multiple-instance pruning for learning efficient cascade detectors. NIPS, 2007.

[25] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng. Fast human detection using a cascade of histograms of oriented gradients. CVPR, 2006.