PAPER

Fast Voting by Pyramid Match for Visual Object Detection

name[†*], **name**[†a)], ***and*** **name**[†*b)],

SUMMARY Visual object detection is still a difficult task. This paper proposes an detection method which is very efficient in both learning and detection phases. Pyramid match is employed by the method to accelerate both learning and detection. Compared with state-of-the-art detection methods on two challenging datasets, the proposed method gives competitive experimental results with much more efficient learning and detection.
key words: *Object detection, Fast voting*

1. Introduction

Bag-of-features [1], [2] schema can be considered as the watershed between traditional and modern detection methods. Instead of considering each target object as a collection of raw optical elements, i.e., pixels, the schema tries to consider each object as a set of semantic elements or so-called object parts which are usually some strong local image features. Then one visual object is said to be a target object if it possesses some certain local features of certain numbers, while does not contain other certain local features of certain numbers. While this is quite straightforward, the very precious information encoded in local features' relative positions is left over. Following some pioneering ideas [3], [4], this paper proposes a detection method which combines spatial and visual information of local image features in a way pursuing both efficiency and effectiveness.

Besides, inferring object status in a bottom-up manner fails to capture global information of each target object from the beginning. And this is also why recently the detection results of Hough transform based methods need refinement by discriminative methods in order to be competitive. Still the way how to use spatial information of local features is very illuminate.

Just as said in [5], Hough transform based methods and sliding-window methods are the two sides of the same coin. The method proposed in this paper calculates confidence of a target object class for each sub-window in an image. Instead of considering each object as a collection of visual patterns (appearance of local features), the method considers each object as a set of visual-spatial patterns. One object is considered as a set of points. Each point is a digital vector, with the last two dimensions the relative $x-$ and $y-$ coordinates to object center, and SIFT after principal component analysis as the remaining dimensions. The training procedure is about collecting all such visual-spatial points into a point set, which acts as a super template. During detection, each sub-image is considered as a point set, and it is matched against the super template. The confidence is then the match score.

The key to this method is how to define a match score for two point sets. Here pyramid matching procedure is employed, not only for efficiency, but also for combining visual and spatial information from local features in an effective manner. The visual-spatial space is divided from fine to coarse. Under a certain dividing parameter, points from the two matching point sets are considered as match if they fall into the same grid, and they are excluded from the respective point sets. The procedure continues till one point set is empty. Then the numbers of matched pairs under each dividing method is counted, and a weighted sum of all these numbers are considered as the match score for two point set, which will be referred to as Pyramid Match Score, or PMS for short. The weights under all dividing methods are learned during training, and how to divide the visual-spatial space is of great importance.

Obviously, each object is considered as a whole during detection in this method.

The proposed method also has several appealing properties, which include but not limited to:

- Feasibility of sequential/batch training, which will lead to easy deployment in distributed system.
- Time complexity of detection not related to the size of training examples.

This paper is organized as follows. Section 2 reviews most related work. Section 3 proposes the training and detecting procedure. Section 4 gives experimental results. Section 5 compares time complexities between the proposed method with other methods, discusses about the insights of the method, and explains why the proposed method is effective. Section 6 concludes.

2. Related Work

Detection methods still mainly follow the sliding-window schema or share similar structures with methods based on Hough transform. While the focus of the later is to infer about object status by use each online feature as a query against a well-trained codebook. These methods fail to consider target objects as a whole at the beginning. The problem of sliding-window schema is that it often ignores positional information when also following bag of features [2]. In

[†]The author is with the
a) E-mail: e-mail address
b) E-mail: e-mail address
DOI: 10.1587/trans.E0.??.1

the method of [3], positional information is considered in the kernel function. Here a kernel function is usually used in classifiers, which are usually support vector machines, as introduced in [6]. The assumption behind [3] is that two images are considered as similar if they possess similar object parts at similar relative positions. Despite of the good theory, its being embedded in support vector machines as kernel function limits the efficiency of this method.

The Bag-of-features [2] schema successfully improves detection performance, while still there is information which are not made use of in images. The positional information is not fully made use of, even of the method of [6]. While [7] provides a method to model the relationship between object parts, there are too many parameters to estimate in their model, which requires a large amount of training data for acceptable performance.

In the method proposed in [8], each object is modeled as a graph, when matching each object with another, constrains are made not only between the two objects, but also between different features of the same object. The relationship between elements of the same object is important. However, the inefficiency of this method prevents it from directly being used for object detection, while its performance on matching the same object under different views is promising.

The method in [9] instead of building some parametric or non-parametric model, directly maps the labels of similar images in the training images to the current image. In this manner, the descriptive capacity of model will not affect performance, and this in return makes the method robust. However, this kind of methods heavily rely on the manually marked labels in the training dataset, while such labels are very expensive in human power. The successes of HOG [10] on pedestrians benefit from its capability to encode relative spatial and visual information from each divided cells. Still the flexibility is not enough, and this leads to deformable part model [4], [11], and its enhanced versions [12], [13]. The model will be referred as DPM for short. It is currently employed by most state-of-the-art methods considering appearance information of object parts together with the relative positional information between object parts. In methods following DPM, a root template is used to detect each object as a whole, and HOG feature is usually used. When a potential object is detected, all possible object parts are detected accordingly. Finally, the confidence of the object is given by the confidence of the root object, the sum of confidence of the object parts, and the cost to deploy the object parts within the root object. Latent SVM is employed in these methods for optimization, and it is capable of representing information in a more complex form. Some methods motivated by DPM try to improve DPM by providing better solution searching strategies [14]. Two recent methods [15], [16] try to find sparse basis of object parts to reduce the number of parameters that need estimating. For efficiency, the method in [17] replace the dot operator of DPM with start-of-the-art hashing method [18]. There also exists hierarchical extensions of DPM [19], and multi-view extension of DPM [20].

Advantages of DPM include that it only need to encode the object parts in positive training examples, and that some of its invariants can give promising results in real time. Still DPM heavily rely of the latent SVM, which is trained in an expectation-maximization manner, and this stops it from adopting new training examples. While nowadays, training examples often come sequentially. A very flexible model, which will evolve with training examples is preferred. These evolutions include evolving of object part number, evolving of the appearance models of object parts, and evolving of the relative positions of object parts.

The pyramid match score method is most related to methods using [21] or [6] as kernel functions, methods employ Hough transforms, and the methods proposing efficient solution space searching techniques [22]. The method is also related to efforts trying to encode images [23] and methods combining sliding window with Hough transform [24].

3. Pyramid Match Score

In this section, firstly the typical procedure of pyramid matching is reviewed, and how a match score between two point sets by using pyramid matching is defined. Then based on the defined matric, how from the training examples, a super template can be learnt and how the super template can be used for object detection in a test image is proposed. In the definition of the matching score, there are parameters very important, finally in this section, how these parameters are estimated is introduced in three sub-sections.

In the remaining content of this paper, all is, js and ks are local symbols.

3.1 Pyramid Matching

The Pyramid Matching method is designed to find the best one-one match, as shown in Fig. 2, between two point sets in a heuristic manner.

Given two point sets, $S_1 = \{u_1, u_2, ..., u_m\}, u_i \in R^d$ and $S_2 = \{v_1, v_2, ..., v_n\}, v_i \in R^d$, there exists a best one-one matching π^* that minimizes the sum of $L1$ -distances between matched pairs,

$$\pi^* = \arg\min_{\pi} \sum_{u_i \in S_1} \|u_i - v_{\pi(i)}\|_1 \ .$$

Here $m \leq n$, and π maps each feature u_i in S_1 to a unique feature $v_{\pi(i)}$ in S_2. There is a 2D example in Fig. 1

The best matching exists, and can be found by simple brute-force enumeration. In special cases, the Hungarian algorithm [25] is also applicable.

Sub-optimal solution can be found by heuristic methods. A very intuitionistic method is to find matched pairs of nearest distance, exclude corresponding points from both point sets, and repeat until no matched pair can be found.

The Pyramid Matching method is straightforward. Divide the point space from fine to coarse, find pairs of points from different point sets in the same grid under the current dividing parameter, exclude the matched pairs, and continue

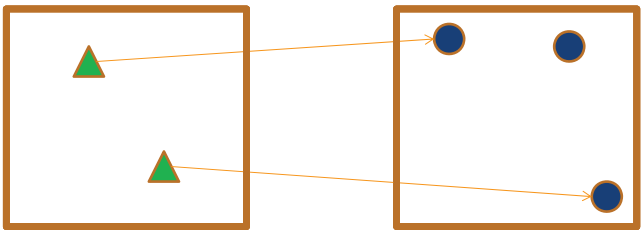

Fig. 1 A best one-one match problem in 2D space. There are two points in the first point set, three in the second point set. The arrows show correspondence between the two point set.

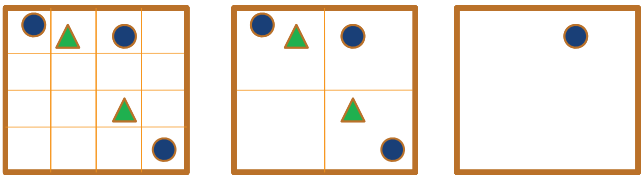

Fig. 2 Pyramid matching procedure which takes the 2D one-one match problem in Fig. 1 as an example. The pyramid matching method divides the 2D space from fine to coarse in the 2D space. Notice, the triangle points belong to point set one, and the circle points belong to point set two. In the left, each dimension is divided into 4, results in totally 16 grids, and 0 matched point pairs are found. In the middle, each dimension is divided into 2, results in totally 4 grids, and two pairs of points belonging to different point sets are found and excluded. In the right, since the matched points belonging to matched point pairs are excluded, then only one point from the second point set is left. So the number of pairs found under all dividing methods are, 0, 2, and 0. The pyramid match score is calculated as a weighted sum of these 0, 2, and 0.

this procedure until the smaller point set is empty. The Pyramid Matching method is very efficient, and its time complexity is bounded by $O(dmL)$ [21]. Here d is the number of dimensions in each point set, m is size of the smaller point set, and L is number of dividing methods. In the example of Fig. 2, L is 3.

In [21], pyramid matching helps to define kernel functions for SVMs. The meaning of pyramid matching is that, it changes how the way to define similarity between two objects. Originally two objects are considered as similar if they both contain certain number of certain object parts, while the idea of one-one match will only favor the objects parts which have corresponding counterparts.

Let $\gamma = \{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_L\}$ be an ordered set, which contains all the dividing methods from fine to coarse. Let $N(S_1, S_2; \mathbf{g}_i)$ be the numbers of matched pairs of points under dividing method $\mathbf{g}_i$. Then the pyramid match score between S_1 and S_2 on γ is defined by,

$$\mathrm{P}(S_1, S_2; \gamma) = \frac{\omega_1 \times N(S_1, S_2; \mathbf{g}_1) + \sum_{i=2}^{L} \omega_i \times (N(S_1, S_2; \mathbf{g}_i) - N(S_1, S_2; \mathbf{g}_{i-1}))}{m}. \tag{1}$$

To exactly follow the procedure as shown in Fig. 2, the definition in (1) is rewritten as,

$$\mathrm{P}(S_1, S_2; \gamma) = \frac{\sum_{i=1}^{L} \omega_i \times N(S_1^{(i-1)}, S_2^{(i-1)}; \mathbf{g}_i)}{m}. \tag{2}$$

Here, S_1^i and S_2^i represent the point set after excluding the points which are found match after the ith round matching respectively from $S_1^{(i-1)}$ and $S_2^{(i-1)}$. Actually, $S_1^0 = S_1$, and $S_2^0 = S_2$.

The procedure is as follows, 1) given the original S_1 and S_2, find the point pairs which fall into the same grid in the space defined by $\mathbf{g}_1$, 2) exclude the matched points respectively from S_1 and S_2 to give S_1^1 and S_2^1, and 3) continue until $i = L$ or one point set is empty.

Then how to construct the dividing methods in γ and how to define the corresponding weight, ω_i, for each $\mathbf{g}_i \in \gamma$ are left to be defined. And these also belong to the factors which distinguish the proposed method from [21].

3.2 Training and Detection

The Pyramid Match Score is a matric between two point sets. In [21], image features are considered as points, while in the proposed method, each point encodes both appearance and location information of each local feature. Each visual-spatial point is $d-$dimensional, and, the first $(d-2)$ dimensions are SIFT after PCA, while the last 2 dimensions are relative $x-$ and $y-$ coordinates after considering scale and width-height ratio changes.

Let $\mathbf{p}$ be a visual-spatial point in the point set of an

image, I, and $F_{\mathbf{p}}$ be the image feature of $\mathbf{p}$, which is $(d-2)-$dimentional. Let $x_{\mathbf{p}}$ and $y_{\mathbf{p}}$ be the $x-$ and $y-$ coordinates of $\mathbf{p}$. Let w_I and h_I be the width and height of I. Then

$$\mathbf{p} = [F_{\mathbf{p}}^1, F_{\mathbf{p}}^2, ..., F_{\mathbf{p}}^{d-2}, \frac{x_{\mathbf{p}}}{w_I}, \frac{y_{\mathbf{p}}}{h_I}] .$$

Instead of following [21], PMS does not server as kernel functions for SVMs. And, a procedure similar to Hough transform is employed. Each training image is considered as a point set. From all the training images, the method generates a point set as a super template, $S_{\mathbf{T}}$, following Algorithm 1. This is just a procedure to collect all points from point sets generated from training images into one point set.

Algorithm 1 Template Generation

```
1: S_T ← ∅
2: for S_Itr ∈ {S_Itr} do
3:     S_T ← S_T + S_Itr
4: end for
5: return S_T
```

In Algorithm 1, each $S_{I_{tr}}$ in $\{S_{I_{tr}}\}$ is the point set generated from the corresponding training image I_{tr}, and the + operator is defined on two sets.

Actually, $S_{\mathbf{T}}$ plays a role similar to a codebook as in methods based on Hough transform.

For detection, a most popular pipeline is employed, as in Algorithm 2. All possible hypotheses are generated, given by $\{\eta\}$. Each hypothesis, η is a rectangle in the image where target objects will be detected, and

$$\eta = [x_\eta, y_\eta, w_\eta, h_\eta].$$

So each η is defined by its starting (x, y) coordinate, its width, and its height. To generate $\{\eta\}$, the sliding window schema is followed, and it works by enumerating all possible rectangles by considering sub-windows' positions and sizes. In Algorithm 2, Ω is the set of final detection results, P_{th} is a threshold to accept hypotheses as detections, I_{te} is a test image, and S_η is the point set generated by local features contained in η.

Algorithm 2 Detection Procedure

```
1: Ω ← ∅, generate {η} from I_te
2: for η ∈ {η} do
3:     Calculate P(S_η, S_T; γ)
4: end for
5: Sort {η} by P(S_η, S_T; γ) in descending order
6: while P(S_η1, S_T; γ) >= P_th do
7:     Ω ← Ω + η_1
8:     {η} ← {η} − η_1
9:     for η ∈ {η} do
10:        for η' ∈ Ω do
11:            for p ∈ S_η do
12:                if (p^(d−1), p^d) is inside η' then
13:                    S_η ← S_η − p
14:                end if
15:            end for
16:        end for
17:        Calculate P(S_η, S_T; γ)
18:    end for
19:    Sort {η} by P(S_η, S_T; γ) in descending order
20: end while
21: return Ω
```

3.3 Dividing Visual-spatial Space

What is very important in Algorithm 2 is how to define the set of dividing methods, γ. In the method of [21], $\mathbf{g}_i$ means dividing each dimension of the point space into 2^i intervals. However, the space in [21] is a pure feature space, while the space here is a visual-spatial space. And also, in [21], the two point sets both belong to objects, while here one point set belongs to the super template.

The space-dividing method proposed here divides the dimensions of visual features and spatial coordinates at different grid sizes. Let

$$\mathbf{g} = g(i, j), i, j \in N.$$

Here $g(i, j)$ is a function which defines how to divide the

visual-spatial space. And i means each dimension belonging to visual channel is divided in to 2^i intervals, and j means each dimension belonging to spatial channel is divided into 2^j intervals. Note, that for a point, $\mathbf{p}$, the first $(d-2)$ dimensions belong to visual channel, while the remaining 2 dimensions belong to spatial channel. For example, if $d = 3$, then $g(2,3)$ will divide the whole space into $(2^i)^{(d-2)} \times (2^j)^2 = 256$ grids.

In Fig. 3, an example is given by considering visual information as one dimension, and spatial information as the other dimension. Note, the total dimension of a point is actually d, while in the example it is 2.

About γ, not only its members, but also the order of its members is important. For (1) to work, a requirement must be fulfilled, that if $i < j$, $\mathbf{g}_i$ is finer than $\mathbf{g}_j$, which means if two points are decided as match under $\mathbf{g}_i$, they must be decided as match under $\mathbf{g}_j$. This is,

$$i < j, G(\mathbf{p}_{S_1}; \mathbf{g}_i) = G(\mathbf{p}_{S_2}; \mathbf{g}_i) \;\Rightarrow\; G(\mathbf{p}_{S_1}; \mathbf{g}_j) = G(\mathbf{p}_{S_2}; \mathbf{g}_j)\,. \tag{3}$$

If $\mathbf{g}_i$ is finer than $\mathbf{g}_j$, it is also written as $\mathbf{g}_i > \mathbf{g}_j$.

In (3), $G(\mathbf{p}; \mathbf{g})$ is a function to map $\mathbf{p}$ into a particular grid, given dividing method $\mathbf{g}$. And $G(\mathbf{p}; \mathbf{g})$ on the kth dimension is defined by,

$$G^k(\mathbf{p}; g(i,j)) = \begin{cases} \lfloor \frac{2^i \times (\mathbf{p}^k_{max} - \mathbf{p}^k)}{\mathbf{p}^k_{max} - \mathbf{p}^k_{min}} \rfloor & \text{if } k \le (d-2) \\ \lfloor \frac{2^j \times (\mathbf{p}^k_{max} - \mathbf{p}^k)}{\mathbf{p}^k_{max} - \mathbf{p}^k_{min}} \rfloor & \text{otherwise} \end{cases}.$$

Here, $\mathbf{p}^k_{max}$ and $\mathbf{p}^k_{min}$ are the maximum and minimum values on the kth dimension, which are determined by training dataset.

There is no such constrain which requires γ is descending ordered by the fineness level in (2). Thus, in the following paper, (2) will be used. In fact, when (3) is satisfied, (1) and (2) are the same.

Though (2) can be used to calculate a pyramid match score for two point sets, given any set, γ, still the dividing methods and the order of the dividing methods will affect performance. For a largest fineness level, $l_{max}, l_{max} \in N$, γ is defined in Algorithm 3.

Algorithm 3 Generation of Dividing Method Set

```
γ ← ∅, r ← 2 × (l_max − 1)
while r ≥ 0 do
    if r ≥ l_max − 1 then
        i ← l_max − 1
    else
        i ← r
    end if
    j ← r − i
    while i ≤ (l_max − 1) and i ≥ 0 and j ≤ (l_max − 1) and j ≥ 0 do
        γ ← γ + g(i, j), i ← i − 1, j ← r − i
    end while
    r ← r − 1
end while
return γ
```

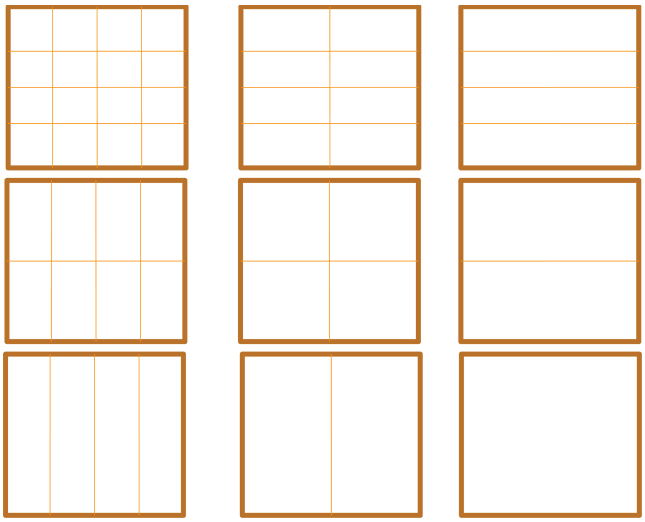

Fig. 3 An example set of methods to divide the visual-spatial space. $x-$ and $y-$ coordinates represent visual and spatial information respectively. From left to right, the first line is $g(2,2)$, $g(1,2)$, and $g(0,2)$. The second line is $g(2,1)$, $g(1,1)$, and $g(0,1)$. And the third line is $g(2,0)$, $g(1,0)$, and $g(0,0)$. And γ is defined as an ordered set of all the dividing methods with different parameters, i.e., $\gamma = \{g(2,2), g(1,2), g(2,1), g(0,2), g(1,1), g(2,0), g(0,1), g(1,0), g(0,0)\}$.

The size of γ, $L = l_{max} \times l_{max}$. For two dividing methods $\mathbf{g}_i, i \in 1, 2, ..., L$ and $\mathbf{g}_j, j \in 1, 2, ..., L$, if $\mathbf{g}_i > \mathbf{g}_j$, then $i < j$, which means if one dividing method is finer than the other, it will appear earlier in the set of dividing methods. There are also dividing methods, of which the fineness level cannot be compared, i.e., $g(1,2)$ and $g(2,1)$ as shown in Fig. 3.

3.4 Deciding Weights for Dividing Methods

After how to divide the visual-spatial space is decided, the remaining task is, for each dividing method $\mathbf{g}$, defining a corresponding weight. When talking about two points which are found in the same grid under $\mathbf{g} = g(i,j)$, there is an upper bound to their $L1$-distance, which is given by

$$D_{ub} = (d-2) \times \frac{1}{2^i} + 2 \times \frac{1}{2^j},$$

if unit length is assumed for all $(\mathbf{p}^k_{max} - \mathbf{p}^k_{min}), k \in \{1, 2, ..., d\}$. Since the first $(d-2)$ dimensions of each grid under $g(i,j)$ possess length of $\frac{1}{2^i}$, while the last 2 dimensions possess length of $\frac{1}{2^j}$.

Following [21], for two point from different point sets, if they are in the same grid under $g(i,j)$, which means $G(\mathbf{p}_{S_1}; g(i,j)) = G(\mathbf{p}_{S_2}; g(i,j))$, the visual difference between $\mathbf{p}_{S_1}$ and $\mathbf{p}_{S_2}$ is defined as $\frac{(d-2)}{2^i}$, and the spatial difference is defined as, $\frac{2}{2^j}$. A weight, ω, defined for a dividing method $g(i,j)$ shows the importance of two matched points, and measures how difficult it is to match under such dividing method.

$$\omega_{g(i,j)} = \sqrt{((d-2) \times 2^i) \times (2 \times 2^j)}. \tag{4}$$

As is seen in (4), the finer one grid is in $g(i, j)$, the larger a weight will be assigned for it. The weight is the confidence that the point set belong to a target object based on a point has corresponding evidence from the super template under the current $\mathbf{g}$.

3.5 Learning Weights for Dividing Methods

Besides directly assigning weights to all the dividing methods in a deterministic way, as in (4), a framework for learning weights is here proposed.

Often Gaussian kernels are used to measure differences between two features or two positions in Hough transform based methods following [26]. For two visual-spatial points found match in the same grid under dividing method, $g(i, j)$, the visual difference is $\frac{(d-2)}{2^i}$, and the spatial difference is $\frac{2}{2^j}$. The total difference between two points found match in the same grid is modeled using a 2D Gaussian kernel, as

$$\omega_{g(i,j)} = \frac{\exp(-\frac{1}{1-\rho^2}(\frac{(\frac{(d-2)}{2^i})^2}{{\sigma_1}^2} + \frac{(\frac{2}{2^j})^2}{{\sigma_2}^2} - \frac{2\rho\frac{(d-2)}{2^i}\frac{2}{2^j}}{\sigma_1\sigma_2}))}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}. \quad (5)$$

In (5), ρ is the correlation between visual and spatial channel, while σ_1 and σ_2 are standard deviations for visual and spatial channel.

To make (5) clear, it is rewritten as,

$$\begin{aligned}\omega_{g(i,j)} =& t\exp(-a(\frac{1}{2^i})^2 - b(\frac{1}{2^j})^2 + c(\frac{1}{2^i})(\frac{1}{2^j})) \\ =& t\exp(-a(\frac{1}{2^i})^2)\exp(-b(\frac{1}{2^j})^2)\exp(c(\frac{1}{2^i})(\frac{1}{2^j})).\end{aligned} \quad (6)$$

Where,

$$\begin{aligned} t &= \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}, \\ a &= \frac{(d-2)^2}{(1-\rho^2){\sigma_1}^2}, \\ b &= \frac{2^2}{(1-\rho^2){\sigma_2}^2}, \text{ and} \\ c &= \frac{2(d-2)(2)\rho}{(1-\rho^2)\sigma_1\sigma_2}. \end{aligned}$$

In (6), the larger, the visual difference, or the larger, the spatial difference, the smaller the corresponding weight. This is decided by the 2D Gaussian kernel, and also this is consistent and very similar with Hough transform based methods.

In Algorithm 1, the weights are not needed, and the super template, $S_{\mathbf{T}}$, can be generated firstly. Then the performance of Algorithm 2 will rely on the set of dividing method, γ, and each corresponding weight, ω.

In (6), to define the weight for $g(i, j)$, besides i and j, still there are four parameters, t, a, b, and c, need to be given. Here, t is just a factor, it won't affect the results of Algorithm 2, which are the final detection results.

Here, a, b, and c have their meanings. When a is larger, visual information will play a more important role, and when b is larger, spatial information will play a more important role. What is more interesting is that, there is c, which will be responsible for modeling correlating visual-spatial information.

To estimate a, b, and c for a particular γ, all positive training images, $\{I_p\}$, and negative training images, $\{I_n\}$, are used. For brevity, the pyramid match score against the super template, with defined γ, is rewritten according to (2),

$$\begin{aligned}\mathrm{P}(S_I, S_{\mathbf{T}}; \gamma) &= \frac{\sum\limits_{i,j}\omega_{g(i,j)} \times N(S_I^{(i-1)}, S_{\mathbf{T}}^{(i-1)}; g(i,j))}{m} \\ &= t\sum_{i,j}\exp(-\frac{a}{(2^i)^2} - \frac{b}{(2^j)^2} + \frac{c}{2^i \times 2^j} \times \frac{N(S_I^{(i-1)}, S_{\mathbf{T}}^{(i-1)}; g(i,j))}{m}\end{aligned} \quad (7)$$

In (7), after summing up along i, and j at given γ and $S_{\mathbf{T}}$, it will be a function which changes according to I, a, b, and c. It is rewritten as $\mathrm{PMS}(I; a, b, c)$ for brevity.

The objective function is written as the gap between pyramid match scores of positive training images and negative training images under normalizing condition as,

$$\begin{aligned}&\arg\max_{a,b,c} \frac{\frac{\sum \mathrm{PMS}(I_p; a, b, c)}{|\{I_p\}|} - \frac{\sum \mathrm{PMS}(I_n; a, b, c)}{|\{I_n\}|}}{\frac{\sum \mathrm{PMS}(I_p; a, b, c) + \sum \mathrm{PMS}(I_n; a, b, c)}{|\{I_p\}| + |\{I_n\}|}} \\ &s.t.: \ a > 0; \\ &\qquad b > 0\ .\end{aligned}$$

Note, about $\mathrm{PMS}(I_p; a, b, c)$, it is not $\mathrm{P}(S_{I_p}, S_{\mathbf{T}}; \gamma)$, but $\mathrm{P}(S_{I_p}, S_{\mathbf{T}} - S_{I_p}; \gamma)$.

After all positive and negative training examples are given, the objective function will only contain a, b, and c. Thus for such a parameter estimating problem, brute-force solutions will be feasible, especially that $\exp(\cdot)$ can be easily expanded.

Till now, Algorithm 1 gives how $S_{\mathbf{T}}$ can be generated, Algorithm 3 gives how γ is defined, how ω is estimated is given in (4) and (7) respectively, so Algorithm 2 can be used for detection.

4. Experimental Results

The key value of the method is the proposed matric between an object hypothesis and the super template trained from training examples. Accordingly, two experiments are carried on and the results are reported in this paper.

4.1 UIUC Cars

UIUC cars [27] can be considered as one of the most famous datasets, and it has been used as benchmarks in the area of detection. There are 1,050 training images, of which 550 are

positive, and 500 are negative. There are 200 target objects in 167 test images without significant scale changes. In the evaluation following, together with the 200 target objects, 669 negative rectangles from the test images are extracted from testing images for the evaluation.

Performance of the method upon different parameters will be evaluated on the UIUC cars, and compared with DPM [12]. In the experiments, (4) is used. So there are only two parameters left, which will lead to difference in performance, which are, the largest fineness level, l_{max} and the dimension of SIFT after PCA, $(d-2)$.

In Fig. **??**, are some detection results on UIUC cars.

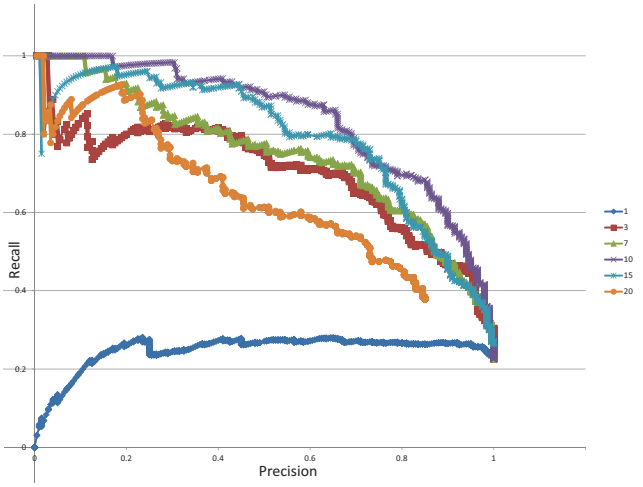

Fig. 4 Result evaluation on UIUC cars with different $(d-2)$s.

In Fig. 4, are the results given by using different $(d-2)$s. Precision-recall curves are generated directly by calculating pyramid match score. Obviously, when appearance information takes 10 dimensions, the performance is the best. The result is consistent with [21] and [28].

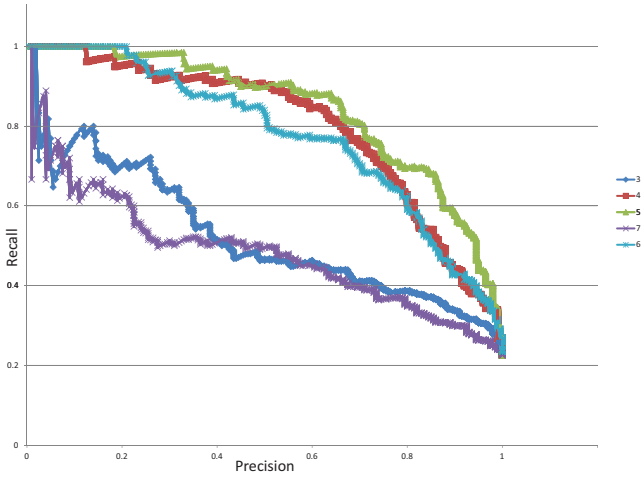

Fig. 5 Result evaluation on UIUC cars with different l_{max}s.

In Fig. 5, detection performance by using different l_{max}s is given. Note, the best performance is given at $l_{max} = 5$. The reasons are two-fold: 1) the weights are given by (4), and favor larger fineness levels, and 2) both training examples and testing examples are of size 100×40, and $2^5 = 32$ is the best dividing parameter for positional information.

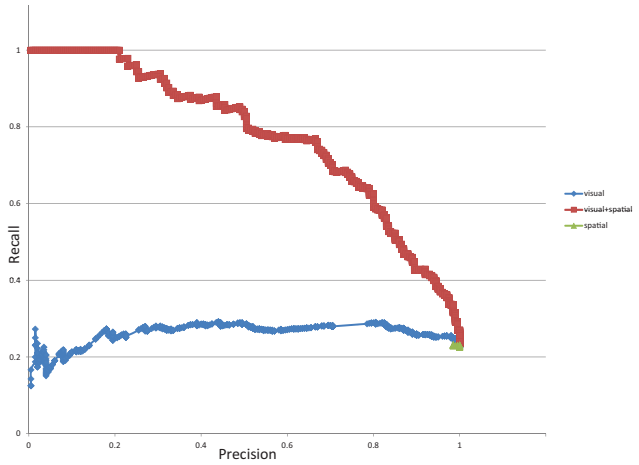

Fig. 6 Result evaluation on UIUC cars by using visual, spatial, and visual + spatial information.

In Fig. 6, are the evaluation of visual and spatial information. Actually spatial information alone is not able to distinguish positive and negative target objects at all. The performance of just using visual information is much worse than using visual-spatial information.

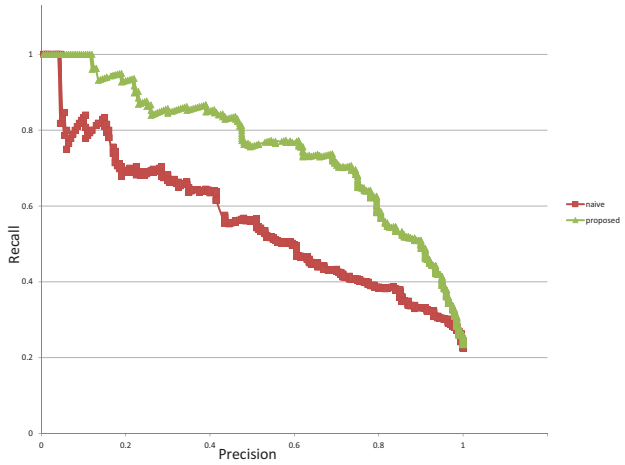

Fig. 7 Result evaluation on UIUC cars by using γ which is generated using Algorithm 3 with $l_{max} = 5$ and $\gamma = \{g(4,4), g(3,3), g(2,2), g(1,1), g(0,0)\}$.

How the visual-spatial space is divided is also critical to the performance of the method, and in Fig. 7, results are evaluated by using the proposed space-dividing methods and [21].

In Fig. 8, performance are evaluated between the proposed method, and a state-of-the-art method, DPM [12].

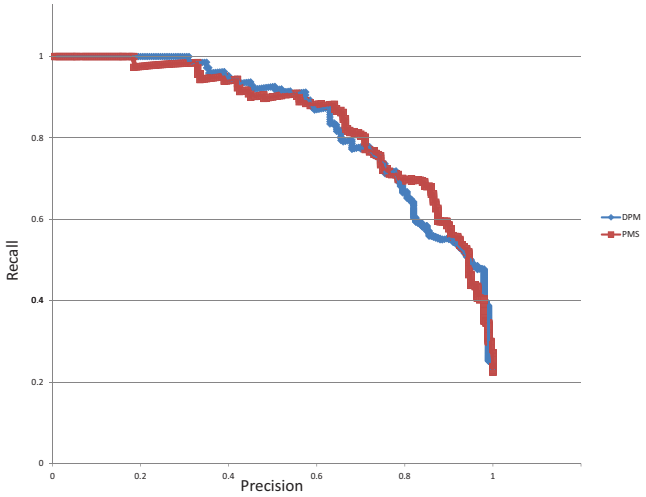

Fig. 8 Result evaluation on UIUC cars between PMS and DPM.

The proposed method performs no worse than DPM, while the training time between the two models are 1 minute vs 16 hours. In this experiment, the training speed of the proposed method is 1,000 times faster. Also during the training of the proposed method, since (4) is used, only the positive training examples are used, while DPM uses both positive and negative training examples.

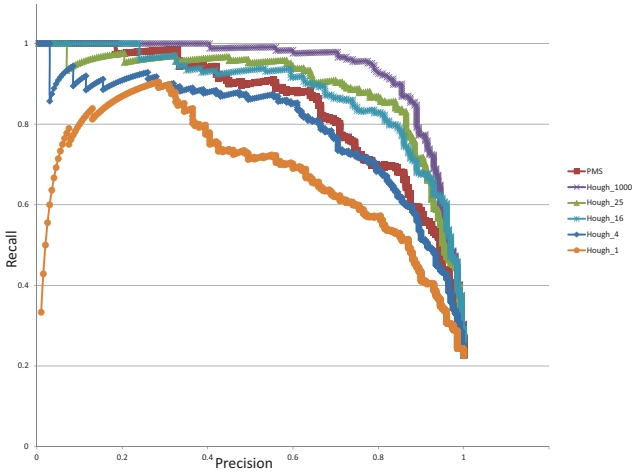

Fig. 9 Compare the proposed method with different-speed Hough transforms.

In Fig. 9, performance are evaluated among the proposed method and variants of Hough transform methods. By taking into consideration of the size of the codebook, the original Hough transform method is 1,000 (the indexes in Fig. 9 mean the times of time consumptions of the variants) time-consuming than the proposed method in giving the confidence of one sub-window's being a target object. Naive k-means method is employed to accelerate the original Hough transform method during training by condensing the size of the codebook. The original Hough transform method performs better than the proposed method. When the time consumptions are the same, the proposed method performs much better. And the proposed method performs better than the variant which is 4 times time-consuming.

In Fig. 10, the average numbers of matched pairs of positive and negative test examples are shown. It is clear that under finer dividing methods, positive test examples possess larger average numbers of matched pairs, while under coarser dividing methods, negative test examples possess larger matched numbers of matched pairs.

4.2 P-campus

The method is also tested on a dataset of pedestrians [29]. For a fair comparison, both training images and test images are exactly same for method comparisons. In Fig. 11, there are some detection results on the dataset.

In Fig. 12, there are the results of comparing the proposed method with ordinary Hough transform and common

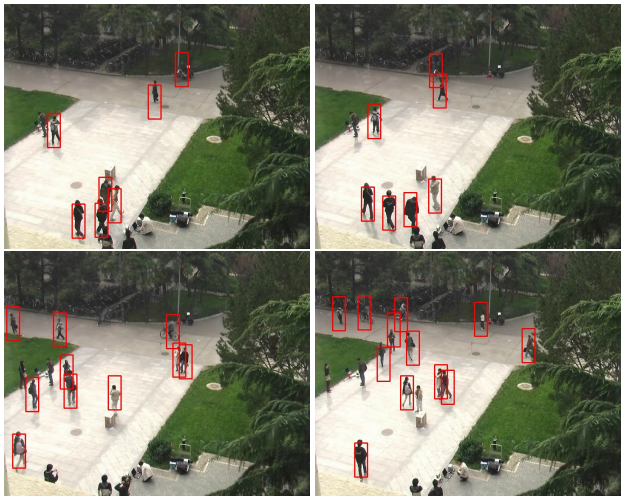

Fig. 11 Detection results on P-campus dataset.

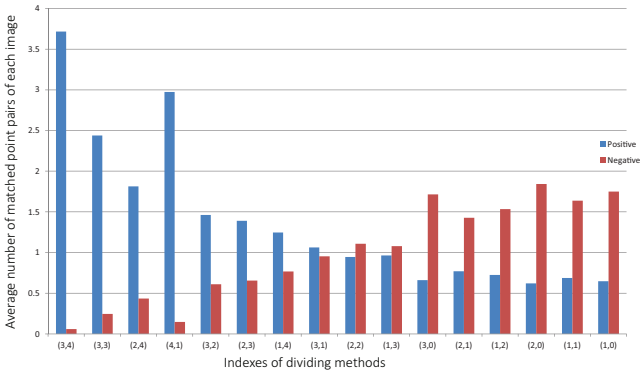

Fig. 10 Comparison of average matched numbers between positive and negative test examples under different visual-spatial space dividing parameters.

fate Hough transform [29]. It can be seen that, common fate Hough transform performs the best since the method employs motion information. The proposed method almost perform as well as the Hough transform on this dataset.

The method detects at a frame rate of 8 frames per second with multi-thread accelerations, while common fate Hough transform needs two minutes to deal with one frame in its old implementation. For a very fair comparison, multi-thread accelerations are disabled for this method, and both [30] and [29] are re-implemented and re-compiled under the same settings. Then all three methods run on the same computer, and the time consumptions are reported in Table 1. Obviously, the proposed method consumes the shortest time. There are more than 5,800 codes in the codebook, and the proposed method is more than 200 times faster than a method based on Hough transform to deal with one candidate sub-image in theory. When dealing with all candidate sub-images in one frame, it is only about 5 times faster. This is because that calculations for one sub-image can be cached and can be used for the decisions on its neighboring sub-images. For example, if two sub-images share the same image feature on the frame, and if best matched codes are found for this image feature, these codes can be used for decisions of both the sub-images.

	Total time (ms)	Average time (ms)
Proposed method	363,678	4,603
Hough transform	1,986,583	25,146
Common fate Hough transform	8,296,161	105,014

Table 1 Time consumptions of the proposed method, [30], and [29]. Total time means the running time on all the 79 testing frames, and average time means average running time per frame.

The training time of the three methods are also compared. Both Hough transform and common fate Hough transform spend 483 ms for training, while pyramid match score spend 978 ms for training. All methods spend less than one second for training.

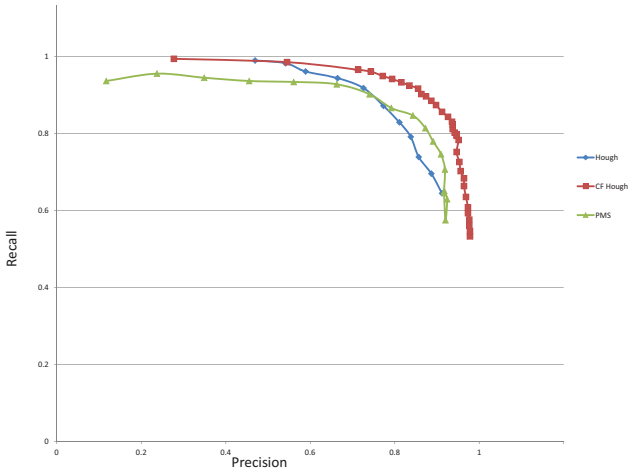

Fig. 12 Precision-Recall curves for the method, Hough transform, and common fate Hough transform.

5. Discussion

5.1 Time Complexity

The time complexity of find a sub-optimal best matching between two point sets is $O(dml^2_{max})$ when pyramid matching is employed. Here, d is the dimension of each visual-spatial point, m is the size of the point set of each object hypothesis, and l_{max} is the largest fineness level, which is usually 5. The match complexity of [21] is $O(dmL), L = l_{max}$. However, this is just the time complexity of matching between two point sets. To give the confidence of a hypothesis, the time complexity of the proposed method is still $O(dml^2_{max})$, while for [21], it is $O(dml_{max}n_{sup})$, where n_{sup} is the number of support vectors in the used SVM, which is usually larger than 5. So, considering about the time complexity to give confidence for a hypothesis, the proposed method is no worse than [21]. Also n_{sup} is related to the training dataset, while l_{max} is less related. The case of [3] is almost the same with [21].

In theory, the time complexity of methods based on Hough transform based methods should be $O(dmn_{tr}^{\frac{1}{1+\varepsilon}})$, if the kNN framework is employed. Here n_{tr} is the size of the codebook, which in the case of UIUC cars is about $50 \times 500 = 25,000$. And, $\varepsilon > 0$ is a factor which affects the quality of the kNN. And compared to PMS, this is 1,000 times time-consuming. However, methods based on Hough transform usually does not consider about object detection in the sliding window manner, which makes such methods unable to well deal with scale changes.

The time complexity of DPM should be an exponential function to the number of parts used. Tricks can be used to avoid such heavy calculation in exchange of performance.

Methods employing bag-of-features schema, and using linear SVM have the lowest time complexity, which is usually $O(dmn_{sup})$, while such methods are currently no longer the dominant.

In summary, the proposed method is competitive in the aspect of time complexity, when considering about giving a reliable confidence to an object hypothesis.

5.2 Usage of Visual and Spatial Information

Since there are two main information channels of the local features, the methods which make full use of such information should be robust and effective.

Let's take Fig. 3 as reference. Bag-of-features schema consider similarity between objects in $g(2,0)$. Actually, [21] considers matched image features in $\{g(2,0), g(1,0), g(0,0)\}$, while [3] considers in a single fineness level of visual information, i.e., $\{g(1,2), g(1,1), g(1,0)\}$. Generally speaking, method based on Hough transform only consider about the features in larger fineness level of both visual and spatial information, i.e., $\{g(2,2), g(2,1), g(1,2)\}$. And methods based on Hough transform, when in a bottom-up manner, can not deploy the object parts in a way similar

to DPM. For example, an object part will contribute to an object it does not belong to.

Together with DPM, PMS considers about visual and spatial information at all fineness levels, and is able to model the correlation between visual and spatial information. However, the way is different. DPM picks several object part at a certain fineness level of visual information, and makes assumptions of their corresponding fineness level of spatial information. For example, DPM will use the most top-left grid of $g(2,1)$, and use the most top-right grid of $g(1,2)$. This grid-picking procedure is achieved by structure SVMs. In the case of PMS, the philosophy is different. Instead of finding some very representative features at certain locations, all features are used, the noisy features are averaged out, since a larger number, i.e., 50, of features are used, compared to DPM, which usually performs good with 6 local features.

Another advantage of PMS is that the number of parameters which need estimating is very small. While, $d = 10$ is an easy conclusion. Other parameters, l_{max}, a, b, and c are all super parameters, which actually contain abundant information. What is attractive is that even using (4) to directly assign weights, the performance on UIUC cars is still promising. These results are consistent with [21].

To summarize, the way how the proposed method combines visual and spatial information of local features is theoretically promising.

6. Conclusion

This paper proposes a method to efficiently and effectively combine visual and spatial information of the local image features. Experiments show the method is comparable with state-of-the-art detection methods on benchmark datasets. What make the method different are: 1) pyramid matching between a codebook and an object hypothesis, 2) the set of methods to dividing the visual-spatial space, and 3) the way to define or learn weights for corresponding dividing method.

The underlying principle of PMS, which is defined by the order of dividing methods in Algorithm 3, is that use the template to explain every visual-spatial point of an object hypothesis at the best visual-spatial level.

To the best of the author's knowledge, this is the first attempt of pyramid matching between a codebook and an object hypothesis.

References

[1] F.f. Li, "A bayesian hierarchical model for learning natural scene categories," CVPR, pp.524–531, 2005.

[2] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," IJCV, vol.87, no.3, pp.316–336, February 2010.

[3] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," CVPR, pp.2169–2178, 2006.

[4] V. Ferrari, F. Jurie, and C. Schmid, "Accurate object detection with deformable shape models learnt from images," CVPR, pp.1–8, 2007.

[5] A. Lehmann, B. Leibe, and L. Van Gool, "Fast prism: Branch and bound hough transform for object class detection," IJCV, vol.94, no.2, pp.175–197, September 2011.

[6] J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, New York, NY, USA, 2004.

[7] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," CVPR, pp.II: 264–271, 2003.

[8] H. Jiang and S. Yu, "Linear solution to scale and rotation invariant object matching," CVPR, pp.2474–2481, 2009.

[9] C. Liu, J. Yuen, and A. Torralba, "Nonparametric scene parsing via label transfer," PAMI, vol.33, no.12, pp.2368–2382, Dec. 2011.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," CVPR, pp.886–893, 2005.

[11] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," PAMI, vol.32, no.9, pp.1627–1645, 2010.

[12] P.F. Felzenszwalb, D.A. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," CVPR, pp.1–8, 2008.

[13] P.F. Felzenszwalb, R.B. Girshick, and D.A. McAllester, "Cascade object detection with deformable part models," CVPR, pp.2241–2248, 2010.

[14] M. Pedersoli, A. Vedaldi, and J. Gonzàlez, "A coarse-to-fine approach for fast deformable object detection," CVPR, pp.1353–1360, 2011.

[15] H. Pirsiavash and D. Ramanan, "Steerable part models," CVPR, pp.3226–3233, 2012.

[16] H.O. Song, S. Zickler, T. Althoff, R. Girshick, M. Fritz, C. Geyer, P. Felzenszwalb, and T. Darrell, "Sparselet models for efficient multiclass object detection," ECCV, pp.716–723, 2012.

[17] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, "Fast, accurate detection of 100,000 object classes on a single machine," CVPR, 2013.

[18] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," VLDB, pp.518–529, 1999.

[19] L.L. Zhu, Y. Chen, A. Yuille, and W. Freeman, "Latent hierarchical structural learning for object detection," CVPR, pp.1062–1069, 2010.

[20] R.J. López-Sastre, T. Tuytelaars, and S. Savarese, "Deformable part models revisited: A performance evaluation for object category pose estimation," ICCV Workshops, pp.1052–1059, 2011.

[21] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," ICCV, pp.1458–1465, 2005.

[22] C.H. Lampert, M.B. Blaschko, and T. Hofmann, "Efficient subwindow search: A branch and bound framework for object localization," PAMI, vol.31, no.12, pp.2129–2142, 2009.

[23] B. Olshausen and D. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," Nature, vol.381, pp.607–609, June 1996.

[24] K. Ohba and K. Ikeuchi, "Detectability, uniqueness, and reliability of eigen windows for stable verification of partially occluded objects," PAMI, vol.19, no.9, pp.1043–1047, September 1997.

[25] H.W. Kuhn, "The hungarian method for the assignment problem," Naval Research Logistics Quarterly, pp.83–97, 1955.

[26] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," IJCV, vol.77, no.1-3, pp.259–289, May 2008.

[27] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," PAMI, vol.26, no.11, pp.1475–1490, Nov. 2004.

[28] Y. Liu, X. Wang, and H. Zha, "Dimension amnesic pyramid match kernel.," AAAI, pp.652–658, 2008.

[29] Z. Wang, J. Cui, H. Zha, M. Kagesawa, and K. Ikeuchi, "Object detection by common fate hough transform," ACPR, pp.613 – 617, 2010.

[30] O. Barinova, V. Lempitsky, and P. Kohli, "On detection of multiple object instances using hough transforms," CVPR, pp.2233–2240, 2010.