Recursive Coarse-to-Fine Localization for fast Object Detection

Marco Pedersoli, Jordi Gonzàlez, Andrew D. Bagdanov, and Juan J. Villanueva

Dept. Ciències de la Computació & Centre de Visió per Computador,
Edifici O, Campus UAB 08193 Bellaterra (Cerdanyola) Barcelona, Spain
{marcopede,poal,bagdanov,juanjo}@cvc.uab.es

Abstract. Cascading techniques are commonly used to speed-up the scan of an image for object detection. However, cascades of detectors are slow to train due to the high number of detectors and corresponding thresholds to learn. Furthermore, they do not use any prior knowledge about the scene structure to decide where to focus the search. To handle these problems, we propose a new way to scan an image, where we couple a recursive coarse-to-fine refinement together with spatial constraints of the object location. For doing that we split an image into a set of uniformly distributed neighborhood regions, and for each of these we apply a local greedy search over feature resolutions. The neighborhood is defined as a scanning region that only one object can occupy. Therefore the best hypothesis is obtained as the location with maximum score and no thresholds are needed. We present an implementation of our method using a pyramid of HOG features and we evaluate it on two standard databases, VOC2007 and INRIA dataset. Results show that the Recursive Coarse-to-Fine Localization (RCFL) achieves a 12x speed-up compared to standard sliding windows. Compared with a cascade of multiple resolutions approach our method has slightly better performance in speed and Average-Precision. Furthermore, in contrast to cascading approach, the speed-up is independent of image conditions, the number of detected objects and clutter.

Keywords: Object Detection, Machine Learning, SVM

1 Introduction

Many improvements and enhancements have been developed on object detection. However, the state of the art for detection is still far from the level necessary for real applications in terms of both speed and accuracy [1]. These two aspects are highly correlated: the newest and best performing methods for object detection, where multiple features [2–4], multiple and non-linear kernels [5, 3] or deformable models [6] are employed, rely on high computational power. All these approaches are based on the concept of moving a classifier around over all possible scales and positions, scanning the image and searching for maximal detection responses, which is commonly called Sliding Windows (SW). However, standard SW is based on a brute-force approach.

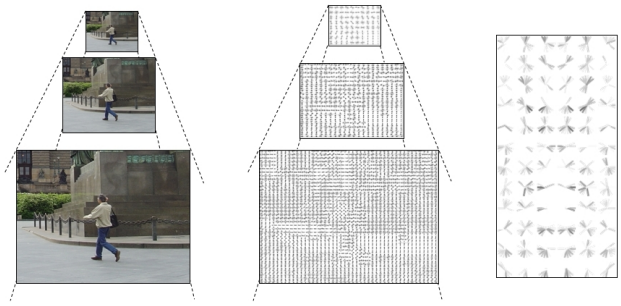
Techniques to avoid the complete image scans have been proposed in the literature. In [7, 8] the authors avoid a dense scan of the image by localizing the object as maximal response in a transformed space of local feature voting. Unfortunately this approach considers every feature independently from the rest, thus rendering it too sensitive to background clutter. Lampert et al. [9] proposed an interesting solution based on a branch-and-bound search over intervals representing bounding box positions and dimensions. However, the method depends on the existence and quality of the bound.

Other methods are able to speed-up SW scanning. A common approach is to decompose the base classifier into a cascade of rejecting classifiers, where the first one is fast but not very effective and the last is very accurate but computationally expensive. The first real-time classifier based on this strategy was proposed in [10], where a pedestrian is localized searching the best match in a hierarchy of human silhouette models. Cascades of classifiers are often based on Adaboost [11, 12], where at each level of the cascade a new *strong* classifier is created by adding more and more *weak* classifiers. The main drawbacks of Adaboost cascades are the complexity of training, which can last for days on a standard PC due to the high number of detectors to be built, the selection of features and the learning of rejection thresholds for each cascade level.

For this reason, especially when dealing with large databases of images such as VOC2007 [13] or the INRIA person dataset [14], fast training methods are essential. Following this trend, the use of cascades based on SVMs with a hierarchy of features of increasing discriminative power [15] or with a hierarchy of kernels from linear to quasi-linear and non linear [3, 5] has been proposed. These approaches, however, require computing all possible windows in the image, not use neither prior knowledge nor spatial constraints to guide the search, and require complex hierarchies of detectors.

In order to overcome the limitations mentioned above we present a method based on a single detector built on a multiresolution pyramid of dense features that benefits from spatial constraints to reduce the search space. In the search process, the image is divided into possible locations or neighborhoods, each one containing at most one object instance. Subsequently, for each location, a recursive coarse-to-fine localization refinement is applied based on the response of the detector at each resolution. The cost of a local search is thus reduced from linear (i.e. proportional to the number of possible locations) to logarithmic time.

Our method extends other contributions to object detection which also apply some kind of multiresolution strategy. The authors in [15] propose a cascade of detectors at different resolutions to speed up the sliding windows scan. Due to the use of multiple and separate detectors, they do not employ multiple resolution features in the same classifier, which reduces their discriminative power. In [16] the authors introduced a human detector based on the joint use of multiresolution gradient features. In this method the multi-resolution pyramid is used for better discriminative power but not for improving the search speed and object localization as in ours. The authors in [6] propose a 2-level dyadic pyramid for the object model: the first for the whole object and the second for parts. As



(a) (b) (c)

Fig. 1. Sliding window components: (a) Pyramid of images I_s: computed by repeated smoothing and sub-sampling of the original image. (b) Pyramid of features H_s: from every scale of the pyramid of images, the corresponding matrix of features is extracted. (c) Object model M: a $h \times w$ matrix of f-dimensional weight vectors.

in our method, no further feature computation is necessary because the same features are used for both multi-scale and multiresolution. But, in contrast to our method, they do not exploit local search to speed-up the scan process. Recently, the same authors propose in [17] a cascade algorithm for their detector. The method is similar to ours in the sense that it decomposes a single classifier into partial scores and uses these to prune hypotheses. However, their pruning method is based on thresholding object parts, while ours is based on recursive object localization refinements and so no threshold is necessary.

Our work uses features at different resolutions in the same classifier and exploits a greedy localization refinement to speed-up the image scan. Our implementation of the Recursive Coarse-To-Fine Localization (RCFL) based on HOGs does require no thresholds and runs twelve times faster than standard SW. In contrast to cascade approaches, the speed up is constant and independent of (i) the quality of the detector, (ii) the complexity of the image and (iii) the number of objects in the image.

2 The Image Scanning Approach

In this section we first describe the standard SW as a vectorial convolution between an object model and image features. Next, this formulation is extended to describe RCFL.

2.1 Sliding Windows

In SW, as described in [14], an object model is scanned over a pyramid of features representing an image. The pyramid of features is a set of matrices $H_s(x, y)$ (see Fig. 1 (b)), where each element is an f-dimensional feature vector. Each matrix H_s is built from a smoothed and sub-sampled version $I_s(x, y)$ of the original

image at a certain scale s, as shown in Fig. 1 (a). The object model for a linear classifier is an $h \times w$ matrix $M(x, y)$, where each element is an f-dimensional weight vector, as shown in Fig. 1 (c). The scale sampling of the pyramid of features is established by a parameter λ defining the number of levels in an octave, that is the number of levels we need to go down in the pyramid to get twice the feature resolution of the previous one.

The response D_s, or score, of the object model centered at position (x, y) and scale s is defined as:

$$D_s(x, y) = \sum_{\hat{x},\hat{y}} M(\hat{x}, \hat{y}) \cdot H_s(\hat{x} + x - w/2, \hat{y} + y - h/2), \tag{1}$$

where $\hat{x} \in \{0, 1, \ldots, w-1\}$, $\hat{y} \in \{0, 1, \ldots, h-1\}$. Note that the symbol $(- \cdot -)$ represents the scalar product because each element M_s and H_s are f-dimensional vectors. In this way, D_s is a pyramid of matrices of the same size as H_s, but where each element is a scalar that represents the response of the object model in the corresponding position and scale. Each element of $D_s(x, y)$ is converted to the corresponding image bounding box center $B_s(x, y)$:

$$B_s(x, y) = (2^{\frac{s}{\lambda}} kx, 2^{\frac{s}{\lambda}} ky) \tag{2}$$

$$\equiv k2^{\frac{s}{\lambda}}(x, y), \tag{3}$$

where k is the size of the feature at level $s = 0$ in pixels. For the sake of simplicity, in the following we will use the notation of Eq. (3), i.e. coordinate-wise scalar multiplications, as equivalent to notation in Eq. (2). Therefore, Eq. (3) describes SW in terms of image coordinates, which is more natural.

The same conversion of Eq. (2) is also applied for the bounding box size (w, h). In this way, we obtain all the necessary information to associate each score $D_s(x, y)$ with the corresponding image bounding box. Applying a Non-Maximum-Suppression (NMS) like in [18], we obtain the bounding box of the final detection.

2.2 Recursive Coarse-to-Fine Localization

In RCFL the object is searched in space but at different resolutions, from coarse to fine. The final score of the detector is now the sum of partial scores, one for each resolution. For this reason, the object model is a dyadic pyramid composed of l levels, where each level d is a matrix M_d of weight vectors. An example of a 3-level pyramid model for the class person in shown in Fig. 2, while an example of recursive localization refinement is shown in Fig. 3.

The computation of the partial score R_s^d for a resolution level d of the object model pyramid at a position (x, y) and scale s of the pyramid of features is then:

$$R_s^d(x, y) = \sum_{\hat{x}_d,\hat{y}_d} M_d(\hat{x}_d, \hat{y}_d) \cdot H_{s+\lambda d}(\hat{x}_d + (x - \frac{w}{2})2^d, \hat{y}_d + (y - \frac{h}{2})2^d), \tag{4}$$

where $\hat{x}_d \in \{0, 1, \ldots, w2^d - 1\}$, $\hat{y}_d \in \{0, 1, \ldots, h2^d - 1\}$. When $d = 0$ this is exactly Eq. (1). When the resolution level d is greater than 0, it is necessary to

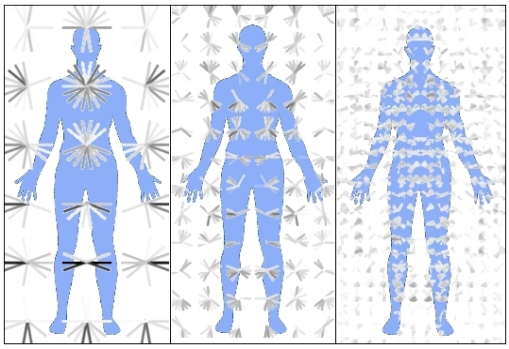


Fig. 2. HOG pyramid model M for the class person with $w = 3$, $h = 6$ and $l = 3$. The low resolution features ($d = 0$) give a general coarse representation of the human silhouette, while the high resolution ($d = 2$) focuses more on details.

move-down λd levels in the feature pyramid to reach the corresponding resolution level. For each H_{s+d}, the search space is split into adjacent neighborhoods Δ_δ:

$$\Delta_\delta(x, y) = \{(\hat{x}, \hat{y}) | \hat{x} = x + d_x, \hat{y} = y + d_y\}, \tag{5}$$

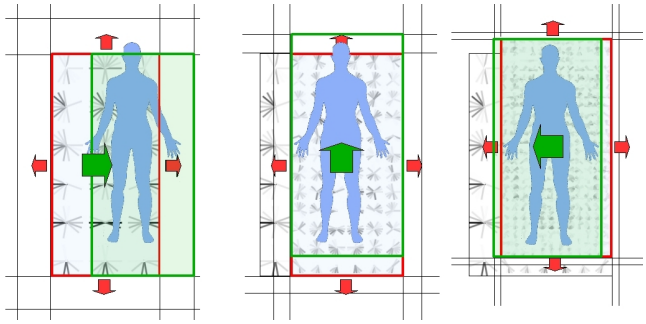
where $d_x, d_y \in \{-\delta, -\delta+1, \ldots, \delta-1, \delta\}$ and δ is the radius of the neighborhood. The neighborhood represents all the locations where an object can be found. While in SW the number of hypotheses corresponds to the number of possible locations of the object, in RCFL the number of hypotheses corresponds to the number of neighborhoods. We define Π_s^0 for each (x, y) and scale s as the location that maximizes the partial score R_s^0 over the neighborhood Δ_δ :

$$\Pi_s^0(x, y) = \underset{(\hat{x},\hat{y})\in\Delta_\delta(x,y)}{\arg\max} \; R_s^0(\hat{x}, \hat{y}). \tag{6}$$

Notice that (x, y) is the location of the center of the neighborhood at the coarse resolution at scale s, while Π_s^0 is the location of the object estimated by M_0. Since we optimize the score of $R_{0,s}$ over the neighborhood Δ_δ, it is not necessary to compute each (x, y). To select the correct sub-sampling of (x, y) is necessary that all locations be scanned at least once, which implies a sampling of $(\hat{\delta}x, \hat{\delta}y)$ with $\hat{\delta} \leq \delta$. The optimal position at levels $d > 0$ is recursively defined as a refinement of the position at $d - 1$:

$$\Pi_s^d(x, y) = \underset{(\hat{x},\hat{y})\in\Delta_1(2\Pi_s^{d-1}(x,y))}{\arg\max} \; R_s^d(\hat{x}, \hat{y}). \tag{7}$$

For $d > 0$ the neighborhood is fixed to Δ_1 because between the level d and $d+1$ the feature resolution doubles and setting the maximum displacement to 1 allows refinement of the object model location at the new resolution. Recall our notational convention for coordinate-wise scalar multiplication, so that $2\Pi_s^{d-1}(x, y)$ represents a doubling of the coordinates for the object estimate at resolution



(a) (b) (c)

Fig. 3. Example of RCFL for detection. In (a), at a certain position (x, y) and scale s (red box) of the pyramid of features H, the best location $\Pi_s^0(x, y)$ (green box) for the low resolution model of the object M_0 is searched in the local neighborhood $\Delta_\delta(x, y)$. In (b), the same procedure is repeated for the next resolution level $s + \lambda d$, using as center of the neighborhood the best location computed at low resolution $\Pi_s^0(x, y)$. The process is recursively repeated for all feature resolution levels. In (c), the location obtained at the finest resolution $\Pi_s^2(x, y)$ is the location of the final detection and can be converted to pixels using Eq.(9).

$d - 1$. Knowing the optimal position of the object model at each level d, we calculate the total score $D_s(x, y)$ as:

$$D_s(x, y) = \sum_{\hat{d}} R_s^{\hat{d}}(\Pi_s^{\hat{d}}(x, y)), \tag{8}$$

where $\hat{d} = \{0, 1, \ldots, l - 1\}$. The computation of the bounding box of each score $D_s(x, y)$ is similar to the standard sliding windows. However, now (x, y) represents the location of the detection at the coarsest level. To obtain the location at the finest level it is necessary to convert the coordinates at Π_s^{l-1}. The center of the bounding box B, for the position (x, y) and scale s is thus:

$$B_s(x, y) = k2^{\frac{s+\lambda(l-1)}{\lambda}} \Pi_s^{l-1}(x, y). \tag{9}$$

The final detection is computed like in normal SW by applying NMS.

3 Learning

Given a set of input data $\{x_1, , x_n\}$ and the associated labels $\{y_1, , y_n\}$, we find a parameter vector w of a function y that minimizes the regularized empirical risk:

$$\frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i y). \tag{10}$$

In our problem the input data x_i is a set of multiple resolution features (extracted from the pyramid H_s defined in previous section) associated to an image region,

while the output data y_i is a binary label indicating whether the object is present in the region. The estimated output y depends on the relative position of each feature level with respect to the previous level. We introduce a structured latent variable h that is a vector of tuples (x_d, y_d) representing the relative position of a certain level d with respect to the previous $d-1$. Using latent variables allows us to obtain a better alignment of the object model with training data, which is useful to improve detector performance as shown in [6]. The estimated output is:

$$y = \max_h \langle w, f(x,h) \rangle, \tag{11}$$

where $f(x,h)$ is a function that maps the input features x to the corresponding latent variable h. In our case:

$$\langle w, f(x,h) \rangle = \sum_d R_s^d(x + x_d, y + y_d). \tag{12}$$

From Eq. (4) we see that w corresponds to the flattened version of M, our object model. Instead of computing the current maximum of f we compute the coarse-to-fine refinement approximation of this, which is Eq.(8):

$$\max_h \langle w, f(x,h) \rangle \approx D_s(\hat{x}, \hat{y}), \tag{13}$$

where $\hat{x}, \hat{y}, s$ corresponds to object location and scale at the lowest resolution. In contrast to normal SVM optimization, y is no longer linear in w, therefore the empirical risk is no longer convex, and standard optimization techniques can not be used. However$f$ is still convex in w since it is a maximum of linear functions. Thus, the empirical risk is is convex for $y_i = -1$ but concave for $y_i = 1$. In order to optimize this function we use a stochastic gradient descent, where learning is divided into two iterative steps: the optimization of w with h fixed for the positive examples and the estimation of the best h using the computed w [6].

Another problem of the learning is the number of negatives examples. While positive examples are costly to obtain and thus their number is always quite limited, the number of negative examples can be very high and can be obtained from images not containing the object to detect. This very high number of negative examples can help to boost performance, but it can make the learning process prohibitive in terms of time and memory. To solve this we use a cutting plane technique consisting of an iterative algorithm that first estimates w using a subset of the full training set and then selects the most violated constraints that will be added to the training set of the next estimation of w. This allows much faster learning and assures that the algorithm converges to the solution obtained with the full set of examples.

4 Discussion

RCFL scans the image in two ways at the same time. On one hand, it scans the image spatially, searching as a standard SW for the object. On the other hand,

it scans the image in the resolution space, from coarse to fine. The number of hypotheses to scan is established at the coarsest level of the pyramid model as a set of neighborhood regions uniformly distributed over the image. Subsequent levels of the pyramid object model refine the hypotheses to the location with highest score inside each neighborhood.

In contrast to previous methods based on cascades [19, 20, 15, 11], there is only one classifier to train. The only assumption required for the validity of the model is that the object has an appearance that can be modeled in a top-down manner. That is, global views of an object contain most of the relevant information needed to support reliable recognition [21], although specific details may be helpful for further refinement.

In cascade approaches, for each sliding window location the score of the classifier is used to evaluate whether to discard the location or continue to the next classifier of the cascade. This means that the score provided by a small number of features has to be precise enough to take a difficult choice that will greatly affect overall detector performance. Therefore, to obtain reliable decisions, the detector has to be conservative, discarding only a fraction of hypotheses. In contrast, our method does not require a binary decision about continuing in the search, but a localization decision about where to focus the search. This is much easier to take, because it is just about finding a local maxima in a small neighborhood of hypotheses. This is what allows RCFL to perform as fast as cascades without sacrificing accuracy and without any rejection threshold, as shown hereafter.

5 Implementation Details

Our aim is to evaluate the performance of the RCFL framework in terms of speed and performance. For this reason we implemented a RCFL based on HOG features, which are widely used in SW-based object detection [14, 5]. However, the proposed framework is not limited to only HOG so it can be applied to any (and multiple) features.

Features. We used the HOG feature implementation proposed in [18]. The features for each square region are 31-dimensional: 9 contrast insensitive features, 18 contrast sensitive features and 4 representing the overall gradient of four neighbor regions. In contrast to the standard HOG proposed in [14], these features are not composed of 50% overlapping blocks, which saves memory space and simplifies the representation.

Object model definition. The object model has few parameters to tune. We set only the parameters of the lowest resolution object representation. All the rest follow from the use of a dyadic pyramid representation. The number of HOG features for the object representation is a trade-off between better discrimination (high number of HOGs) and the capability to detect small objects (low number of HOGs). The aspect ratio of the object model is chosen based on the mean aspect ratio of the training bounding boxes.

Positive examples. We convert an image containing positive examples into a pyramid of features (as described in section 2) and then search over space and

scale for the best fit between the object model bounding box and the training example bounding box using the overlap ratio as defined in [13]. If the overlap o is greater than 0.5, the example is taken as a positive sample and added to T_p, otherwise it is discarded.

Negative examples. Negatives examples T_n are extracted from images not containing the object class. As for the positives, the image is converted to a pyramid of features. The examples are initially drawn using a uniform distribution over both space and scale. Subsequently, they are selected based on the cutting plane technique explained before.

SVM training. Positive T_p and negative T_n pyramids of features are flattened to vectors and used to train a linear SVM using libSVM [22]. The result of this is a weighted sum of support vectors. Since the kernel is lineal, these are summed up into a single vector of weights w. This is then converted back to the dyadic pyramid representation, resulting in our object model M.

6 Experiments

We evaluate our RCFL detector on two different and complementary databases: VOC2007 [1] and the INRIA person dataset [2]. We use VOC2007 as reference to evaluate our method on 20 different object classes and to obtain the most general detector configuration. Then, we test our method on the INRIA dataset, where many state-of-the-art methods are evaluated in terms of accuracy and speed.

6.1 Neighborhood radius, Resolution levels and Speed-up Factor

The neighborhood radius δ and resolution levels l are the two most important parameters that influence the final performance of the detector. While for resolution levels greater than zero δ is forced to be 1 to ensure coherence of representation of the model over resolutions, for the zero level δ is free and greatly affects the speed-accuracy trade-off. Using a neighborhood of radius δ for level zero corresponds to scanning $q = (2\delta + 1)^2$ locations at the first level and subsequently 9 locations for the next levels. So, a model of l levels requires $q + 9l$ evaluations instead of $q4^l$ as in standard SW working at the finest level. However, the cost of scanning a location is proportional to the object model resolution which doubles at each level of the pyramid model. So, the ratio between the cost of brute-force search and the recursive localization approach is:

$$g(l, q) = \frac{q4^{l-1}}{\sum_d \frac{9}{4^d} + \frac{q}{4^{l-1}}} \tag{14}$$

where $d = \{0, 1, \ldots, l-2\}$. The computational cost of RCFL, compared to standard SW, is reduced proportionally to the number of levels of the object model l and the neighborhood locations q. In experiments l is bounded by the resolutions available in images of the object and the memory space needed for the training.

[1] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/
[2] http://pascal.inrialpes.fr/data/human/

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
l=1	22.4	39.2	10.5	3.6	17.4	37.5	36.8	23.4	15.5	20.8	33.6
l=2	28.3	43.3	11.5	4.5	29.0	45.7	39.3	28.8	16.0	27.4	36.3
l=3	28.0	37.3	9.6	3.6	22.1	45.8	36.7	26.6	14.8	35.2	31.6

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean	speed
l=1	19.2	45.4	36.5	23.6	16.2	19.3	33.3	26.5	44.7	26.3	1.0
l=2	24.7	42.9	38.0	22.1	16.3	27.7	34.1	31.3	47.7	29.7	3.2
l=3	26.7	43.9	37.7	21.5	15.1	27.2	30.6	28.2	46.0	28.1	12.2

Table 1. Average-Precision computed on positive examples of train+validation of the VOC2007 database for different number of levels of resolution.

For the choice of δ we have to consider that a neighborhood must contain a unique hypothesis for an object. Therefore, to correctly localize partially overlapping bounding boxes it is necessary that, within a neighborhood, all possible detections overlap each other enough to be grouped together by NMS.

We computed the distribution of overlapping instances in the VOC2007 database for all classes and evaluated that a maximum overlapping of 0.2 assures fusing 99% of the object instances correctly. Limiting the minimum resolution for the lower resolution model to 15 HOG cells assures a minimum overlapping of 0.2 by setting $\delta \leq 1$.

6.2 Levels of Resolutions

We also must establish how many levels of feature resolution are best for our problem. For this, we evaluate the RCFL method against all classes of the PASCAL VOC2007 database. Because we are interested only on the relative performance of different configurations, we used only positive examples. In order to make the comparison fair, we choose the number of features for the maximum resolution level to be the same for all configurations.

Detection results are reported in Table 1. Mean values show that the best performance in terms of average-precision is the configuration with 2 resolution levels. However, the speed-up of this configuration is only 3.2 times. Moving to 3 resolution levels the performance is still good, but the speed-up is increased to 12.2 times. This makes this configuration an optimal trade-off between performance and speed and it will be the configuration used also in all the following experiments.

6.3 Comparison with Cascades

Although interesting methods implementing cascades based on HOG have been developed in recent years [11, 15, 17], all the methods use different HOG implementation, different parameter configurations and different learning strategies. We implemented our own cascade detector and to allow a best comparison we keep the same configuration based on three levels of feature resolution. The cascade is similar to [15], but we improve the learning strategy by joining all

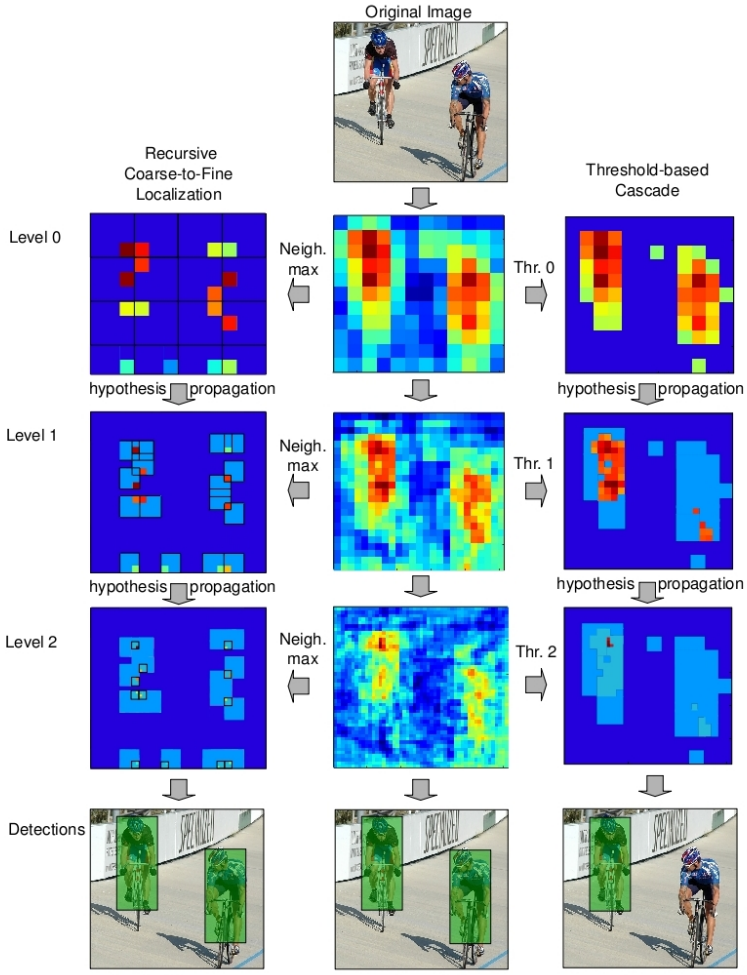


Fig. 4. Example of scan over resolutions with three methods: left column *RCFL*, central column *Exact*, right column *Cascade*. In the cascade the threshold is too high and prunes good hypothesis loosing the second cyclist. In RCFL both detections are made because the algorithm requires that hypotheses over all space are kept.

features from different levels into a single SVM optimization, exactly the same used for RCFL and explained in section 3. Using same learning and features assures that changes in accuracy or speed are totally due to the method, not to implementation details nor different learning strategies. To select the optimal thresholds we used the method proposed in [17].

We compare the two methods also with a brute force approach in all classes of VOC2007. In this experiment we train the detectors only with the training set, while the validation set is used for threshold learning. The threshold value

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
Exact	24.1	41.3	11.3	3.9	20.8	36.8	35.4	25.5	16.0	19.4	21.2
Cascade	24.1	38.7	12.9	3.9	19.9	37.3	35.7	25.9	16.0	19.3	21.2
Speed	9.3	9.8	9.3	9.9	3.9	18.1	13.8	17.3	9.5	12.1	6.4
RCFL	23.6	39.4	12.9	2.7	19.7	39.2	34.5	25.9	17.0	21.6	23.1

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean	speed
Exact	23.0	42.9	39.8	24.9	14.6	14.3	33.0	22.8	37.4	25.4	1.0
Cascade	23.0	40.2	41.5	24.9	14.6	15.1	33.2	23.0	42.2	25.6	10.9
Speed	3.3	17.6	20.1	3.6	6.4	19.0	15.0	9.8	2.8		
RCFL	24.1	42.0	41.1	25.3	14.2	15.8	29.6	22.5	41.0	25.8	12.2

Table 2. Average-Precision computed on positive examples of train of the VOC2007 database. *Exact* shows the results of a brute force method; *Cascade* represents the result of a cascade method with thresholds chosen for obtaining the same performance as exact up to precision equals recall; thresholds are computed using the validation set of VOC2007; *Speed* is the average speed-up per class achieved for Cascade; *RCFL* is our method using three resolution levels.

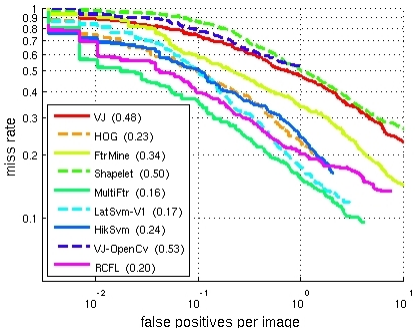
is set so the resulting precision-recall curve of the cascade detector should reach the precision-equals-recall point without any pruning. Results are reported in Table 2. For the cascade we also report per-class speed-up, while the final speed-up is the average of all classes. It is interesting to notice that both speed-up methods, not only improve speed, but also in some case average-precision. This is due to the pruning of false positives. Notice that even without any threshold expressly tuned for it, RCFL obtain an average performance slightly better than the cascade and more important, the speed-up is constant, while for cascades it can vary a lot and it is unpredictable. This demonstrates that recursive localization is a suitable strategy for pruning hypotheses because (i) it obtains same or better performance than cascades in most of the classes (ii) it assures that speed does not depend on object class or image which is very important for real-time applications (iii) no threshold computation is necessary.

Fig. 4 show the pipeline of the pruning strategy of RCFL and Cascade of the class person for a certain scale. Although both strategies use exactly the same observations (central column), in this example Cascade is not able to detect an obect due to a too low partial score in the second level. RCFL is not affected by this problem because no thresholds are used for the pruning which is done in a spatial manner.

6.4 INRIA pedestrian dataset

The INRIA database is the standard benchmark for human detection [14]. Evaluation is done on a per-image basis. This is equivalent to a precision recall curve, but for certain tasks it is preferred because it gives an absolute measure of the average number of false positives to expect per-image (FPPI).

A comparison of RCFL with other methods is shown in Fig. 5 (a). The configuration of the detector is the same as in previous experiments with 3



(a)

Method	Features	Classifier	M-R	Time
HikSvm[16]	HOG-like	HIK SVN	0.24	140.0
Shapelet[23]	Gradients	AdaBoost	0.50	60.0
FtrMine[12]	Haar	AdaBoost	0.34	45.0
MultiFtr[4]	HOG+Haar	AdaBoost	**0.16**	18.9
HOG[14]	HOG	lin. SVN	0.23	13.3
LatSvm[6]	HOG	lat. SVM	0.17	6.3
Haar[19]	Haar	AdaBoost	0.48	7.0
RCFL	HOG	lin. SVM	0.20	**1.2**

(b)

Fig. 5. (a) False Positive Per-Image in the INRIA database. All curves but RCFL HOG are drawn using the data provided by [1]. (b) Comparison of different pedestrian detectors [1]. *M-R* represents the miss-rate at 10^0 false positive per-image. *Time* represents the seconds to compute an image of 640×480 pixels. RCFL reduces the miss-rate of the HOG detector performing much faster than any other method.

resolution levels. RCFL reduces the standard HOG miss-rate by 3 points at 10^0 FPPI, by 10 points at 10^{-1} FPPI and by 14 points at 10^{-2} FPPI. Globally, two methods perform better than RCFL. However, *MultiFtr* uses multiple and complementary features to improve the HOG results while *LatSvm* learns the object deformations using latent SVM.

Table 5 (b) summarizes the main characteristics of each method. HOG RCFL performs better than all comparable methods, but with a higher speed. Our method takes 1.2 seconds to process an image: around 1 second is used for feature computation and only 0.2 for the scan. In contrast to most methods, where the most significant part of the time is used for scanning, with RCFL this scanning time is reduced to a small fraction.

7 Conclusions

In this paper we introduced a new method to speed-up object detection. The method join prior information about the search of object hypotheses with a coarse-to-fine localization to optimally distribute the computation necessary to detect objects. Compared to cascades approaches, our method obtains similar detection performance, assures a constant speed-up independent of object class and image conditions and do not need any threshold to prune hypotheses. Finally, the great generality of the idea behind RCFL allows it to be applied to most of current object detector methods: from deformable models [6] to bag of words pyramids [5], but also multiple features [3].

Acknowledgements This work has been supported by the Spanish Research Programs Consolider-Ingenio 2010:MIPRCV (CSD200700018) and Avanza I+D ViCoMo (TSI-020400-2009-133); and by the Spanish projects TIN2009-14501-C02-01 and TIN2009-14501-C02-02. Thanks to Ivan Huerta for his precious help.

References

1. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: A benchmark. In: CVPR. (2009)
2. Schwartz, W.R., Kembhavi, A., Harwood, D., Davis, L.S.: Human detection using partial least squares analysis. In: ICCV. (2009)
3. Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple kernels for object detection. In: ICCV. (2009)
4. Wojek, C., Schiele, B.: A performance evaluation of single and multi-feature people detection. In: DAGM, Berlin, Heidelberg (2008) 82–91
5. Harzallah, H., Jurie, F., Schmid, C.: Combining efficient object localization and image classification. In: ICCV. (2009)
6. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: CVPR. (2008)
7. Mikolajczyk, K., Leibe, B., Schiele, B.: Multiple object class detection with a generative model. In: CVPR. (2006) 26–36
8. Leibe, B., Leonardis, A., Schiele, B.: Robust object detection with interleaved categorization and segmentation. IJCV **77** (2008) 259–289
9. Lampert, C.H., Blaschko, M.B., Hofmann, T.: Beyond sliding windows: Object localization by efficient subwindow search. In: CVPR. (2008)
10. Gavrila, D., Philomin, V.: Real-time object detection for smart vehicles. In: ICCV. (1999) 87–93
11. Zhu, Q., Yeh, M.C., Cheng, K.T., Avidan, S.: Fast human detection using a cascade of histograms of oriented gradients. In: CVPR. (2006) 1491–1498
12. Dollar, P., Tu, Z., Tao, H., Belongie, S.: Feature mining for image classification. In: CVPR. (2007)
13. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results (2007)
14. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005) 886–893
15. Zhang, W., Zelinsky, G., Samaras, D.: Real-time accurate object detection using multiple resolutions. In: ICCV. (2007)
16. Maji, S., Berg, A., Malik, J.: Classification using intersection kernel support vector machines is efficient. In: CVPR. (2008)
17. Felzenszwalb, P., Girshick, R., McAllester, D.: Cascade object detection with deformable parts models. In: CVPR. (2010)
18. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. PAMI **31** (2009)
19. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR. (2001)
20. Dollar, P., Babenko, B., Belongie, S., Perona, P., Tu, Z.: Multiple component learning for object detection. In: ECCV. (2008)
21. Torralba, A.: How many pixels make an image? Visual Neuroscience **26** (2009) 123–131
22. Chang, C., Lin, C.: LIBSVM: a library for support vector machines (2005)
23. Sabzmeydani, P., Mori, G.: Detecting pedestrians by learning shapelet features. In: CVPR. (2007)