International Journal of Computer Vision manuscript No.
(will be inserted by the editor)

Fast Human Pose Detection using Randomized Hierarchical Cascades of Rejectors

Grégory Rogez · Jonathan Rihan · Carlos Orrite-Uruñuela · Philip H. S. Torr

Received: date / Accepted: date

Abstract This paper addresses human detection and pose estimation from monocular images by formulating it as a classification problem. Our main contribution is a multi-class pose detector that uses the best components of state-of-the-art classifiers including hierarchical trees, cascades of rejectors as well as randomized forests. Given a database of images with corresponding human poses, we define a set of classes by discretizing camera viewpoint and pose space. A bottom-up approach is first followed to build a hierarchical tree by recursively clustering and merging the classes at each level. For each branch of this decision tree, we take advantage of the alignment of training images to build a list of potentially discriminative HOG (Histograms of Orientated Gradients) features. We then select the HOG blocks that show the best rejection performances. We finally grow an ensemble of cascades by randomly sampling one of these HOG-based rejectors at each branch of the tree. The resulting multi-class classifier is then used to scan images in a sliding window scheme. One of the properties of our algorithm is that the randomization can be applied on-line at no extra-cost, therefore classifying each window with a different ensemble of randomized cascades. Our approach, when compared to other pose classifiers, gives fast and efficient detection performances with both fixed and moving cameras. We present results using different publicly available training and testing data sets.

Keywords Human Detection · Pose Estimation · Cascade Classifiers

1 Introduction

Full-body human pose analysis from monocular images constitutes one of the fundamental problems in Computer Vision as shown by the recent special issue of the journal [47]. It has a wide range of potential applications such as Human-Computer interfaces, videogames, video annotation/indexing or surveillance. Given an input image, an ideal system would be able to localize any humans present in the scene and recover their poses. The two stages, known as *human detection* and *human pose estimation*, are usually considered separately. There is an extensive literature on both *detection* [14, 24, 43, 56, 57, 61] and *pose estimation* [1, 6, 18, 26, 29, 35, 41, 44, 52] but relatively few papers consider the two stages together [5, 9, 17, 37, 49]. Most algorithms for pose estimation assume that the human has been localized and the silhouette has been recovered, making the problem substantially easier.

Some techniques therefore separate the foreground from the background and classify a detected (and segmented) object as human or nonhuman. The pose of the human can then be estimated, for instance fitting a human model on the resulting blob or silhouette [29, 41], or applying pose regressors [1, 26]. These methods are very helpful when a relatively clean background image can be computed which is not always the case, depending on the settings and applications: for example if the goal is to detect humans in an isolated image (not from a video sequence) or in a moving camera sequence, the

Part of this work was conducted while the first author was a research fellow at Oxford Brookes University. This work was supported by Spanish grant TIN2010-20177 and FEDER, and by the EPSRC grant EP/C006631/1(P) and Sony.

Grégory Rogez Carlos Orrite-Uruñuela
Computer Vision Lab - University of Zaragoza
E-mail: grogez@unizar.es

Jonathan Rihan, Philip H. S. Torr
Department of Computing, Oxford Brookes University, Wheatley Campus, Oxford OX33 1HX, UK

computation of a background image and consequently the segmentation of the subject are not trivial.

We consider the problem of simultaneous human detection and pose estimation. We follow a sliding window approach to jointly localize and classify human pose using a fast multi-class classifier that combines the best components of state-of-the-art classifiers including hierarchical trees, cascades of rejectors and randomized forests. The novelty and properties of the algorithm are:

1. Cascade classifiers are designed to quickly reject a large majority of negative candidates to focus on more promising regions, we exploit this property by learning an ensemble of multi-class hierarchical cascades.
2. At each branch of the hierarchy, our training algorithm selects a small subset of informative features from a much larger feature space. This approach is computationally efficient and scalable.
3. Additionally, by randomly sampling from these candidate features, each cascade uses different sets of features to vote which adds some robustness to noise and helps to prevent over-fitting.
4. Each random cascade can vote for one or more class so the ensemble outputs a distribution over poses that can be useful for resolving pose ambiguities.
5. Both randomization and selection of the number of cascades can be performed on-line at no extra-cost, therefore classifying each window with a different ensemble of cascades. This adaptive classification scheme allows a considerable speed-up and an even more efficient pose detection than simply using the same fixed size ensemble over the image.

1.1 Related Previous Work

Exemplar based approaches have been very successful in pose estimation [35]. However, in a scenario involving a wide range of viewpoints and poses, a large number of exemplars would be required. As a result the computational time would be very high to recognize individual poses. One approach, based on efficient nearest neighbor search using histogram of gradient features, addressed the problem of quick retrieval in large set of exemplars by using Parameters Sensitive Hashing (PSH) [44], a variant of the original Locality Sensitive Hashing algorithm (LSH) [15]. The final pose estimate is produced by applying locally-weighted regression to the neighbors found by PSH.

The method of Agarwal and Triggs [1] is also exemplar based. They also use a kernel based regression but they do not perform a nearest neighbor search for exemplars, instead using a hopefully sparse subset of the exemplars learnt by the Relevance Vector Machines (RVM). Their method has the main disadvantage that it is silhouette based, and perhaps more serious it can not model ambiguity in pose as the regression is unimodal. In [53], an exemplar-based approach with dynamics is proposed for tracking pedestrians. Gavrila [24] presents a probabilistic approach to hierarchical, exemplar-based shape matching. This method achieves a very good detection rate and real time performance but does not regress to a pose estimation. Similar in spirit, Stenger [50] uses a hierarchical Bayesian filter for real-time articulated hand tracking. Recently, Lin and Davis [31] propose a hierarchical part-template matching for shape-based human detection and segmentation.

Much other work focuses on human detection specifically without considering pose [14, 24, 43, 56, 57, 61]. Dalal and Triggs [14] use a dense grid of Histograms of Orientated Gradients (HOG) and learn a Support Vector Machine (SVM) classifier to separate human from background examples. Later Zhu et al [61] extend this work by applying integral histograms to efficiently calculate HOG features and use a cascade of rejectors classifier to achieve near real time detection performance.

Several works attempt to combine localization and pose estimation. Dimitrijevic et al [17] present a template-based pose detector and solve the problem of huge datasets by detecting only human silhouettes in a characteristic postures (sideways opened-leg walking postures in this case). They extend this work in [22] by inferring 3D poses between consecutive detections using motion models. This method can be used to track walking people even with moving cameras, however, it seems somehow difficult to generalize to any actions that do not exhibit characteristic posture. Sminchisescu et al [49] jointly learn coupled generative-discriminative models in alternation and integrate detection and pose estimation in a common sliding window framework. Okada and Soatto [37] learn k kernel SVMs to discriminate between k predefined pose clusters, and then learn linear regressors from feature to pose space. They extend this method to localization by adding an additional cluster that contains only images of background. The poselet work of [9] presents a two-layer classification/regression model for detecting people and localizing body components. The first layer consists of poselet classifiers trained to detect local patterns in the image. The second layer combines the output of the classifiers in a max-margin framework. Ferrari et al. [21] use an upper-body detector to localize a human in an image, find a rough segmentation using a foreground and background model calculated using the detection window location, and then apply a pictorial structure model [19] in regions of interest. Such part-based methods are another

successful approach to simultaneous human detection and pose estimation [2, 36, 40] . Typically however they require multiple classifiers or appearance models to represent each of the the body parts.

We introduce a novel algorithm that jointly tackles human detection and pose estimation in a similar way to template tree approaches [24, 50], while exploiting some advantages of AdaBoost style cascade classifiers [55, 61] and Random Forests [11]. Random Forests (RF) have seen a great deal of success in many varied applications such as object recognition [8] or clustering [34, 45]. RF have shown to be fast and robust classification techniques that can handle multi-class problems [30], so makes them ideal for use in human pose estimation. Recently, Shotton et al [46] trained a decision forest to estimate body parts from depth images with excellent results on pose estimation.

1.2 Motivation and Overview of the Approach

Many different types of features have been considered for human detection and pose estimation: silhouette [1], shape [24], edges [17], HOG descriptors [14, 20, 61], Haar filters [56], motion and appearance patches [6], edgelet feature [57], shapelet features [43] or SIFT [32]. Driven by the recent success of HOG descriptors for both human detection [14, 61] and pose estimation [44], and that they can be implemented efficiently to achieve near real time speeds [61], we chose to use HOG descriptors as a feature in our algorithm. For pose estimation, an ideal dataset should contain variation in subject pose, camera viewpoint, appearance and physical attributes. Combining the dataset with a very dense image feature set such as HOG captures discriminative details between very similar poses [37] but also considerably increases the dimension of the training set.

Random Forests [11] allow for a better handling of large datasets as they can be faster to train and are less prone to over-fitting than selecting features from an exhaustive search over all features[1]. We performed an initial test of pose classification (see Fig. 1 and Sect. 4) and identified two main drawbacks with the algorithm and the existing implementation: as illustrated in Fig. 1 using denser HOG feature grids improves pose classification accuracy. Neighboring classes can be very close to one another in image space, and in practice are only separable by some sparse subset of features. This means that, having randomly picked an arbitrary feature to project on from a high dimensional feature space, it is highly unlikely that an informative split in this projection (i.e. one that improves the information measure) exists. While we do not need perfect trees, informative trees are still rare and finding them naively requires us to generate an infeasible number of trees.

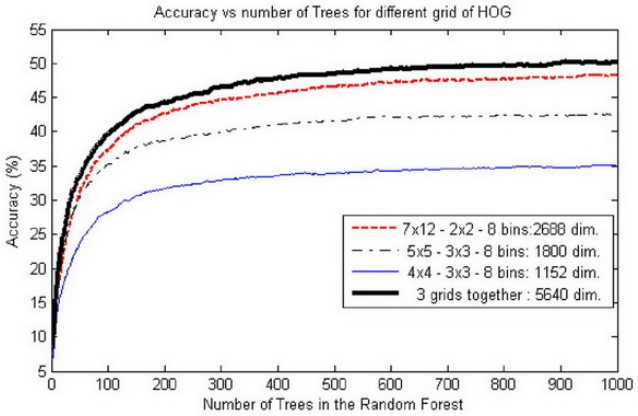


Fig. 1 Random Forest preliminary results. An initial test was performed on the MoBo walking dataset [25]: dense grid of HOG features are extracted for 15 different subjects from around 50,000 images which are grouped in 64 pose classes. We build the training subset by randomly sampling 10 subjects and keep the remaining 5 subjects for testing. We run the same test for 3 different grids of HOG and show the classification results varying the number of trees used in the forest. Using denser HOG grids improves pose classification accuracy but we are quickly facing memory issues that prevent us from working with denser grids.

Another drawback of the Random Forests algorithm is that it is not very well adapted for sliding window approaches. Even if on-demand feature extraction can be considered as in [16], for each scanned sub-image, the trees still have to be completely traversed to produce a vote/classification. This means that a non-negligible amount of features have to be extracted for each processed window, making the algorithm less efficient than existing approaches like cascades-of-rejectors that quickly reject most of the negative candidates using a very small subset of features. Works such as [55, 61] use AdaBoost to learn a cascade structure using very few features at the first level of the cascade, and increasing the number of features used for later stages. Other approaches such as those described in [33, 60] for multi-view face detection, organize the cascade in to a hierarchy structure consisting of two types of classifier; face/non-face, and face view detection. Zhang et al [59] present a probabilistic boosting network for joint real-time object

[1] RF are grown by randomly selecting a subset of features at each node of the tree to help avoid a single tree over fitting the training data. The best split is found for each dimension m_i by evaluating all possible splits along that dimension using a measure such as information gain [8]. The dimension m^* that best splits the data according to that score is used to partition the data at that node. This process continues recursively until all the data has been split and each node contains a single class of data.

detection and pose estimation. Their graph structured network also alternates binary foreground/background and multi-class pose classifiers.

Inspired by these ideas, we train multi-class hierarchical cascades for human pose detection. First a class hierarchy is built by recursively clustering and merging the predefined pose classes. Hierarchical template trees have been shown to be very effective for real time systems [24, 50], and we extend this approach to non-segmented images. For each branch of this tree-like structure, we use a novel algorithm to build a list of potentially discriminative HOG descriptor blocks. We then train a weak classifier on each one of these blocks and select the ones that show the best performances. We finally grow an ensemble of cascades by randomly sampling one of these HOG-based rejectors at each branch of the hierarchy. By randomly sampling the features, each cascade uses different sets of features to vote and adds some robustness to noise and prevents over fitting as with RF. Each cascade can vote for one or more class so the final classification is a distribution over classes.

In the next section, we present a new method for data driven discriminative feature selection that enables our proposal to deal with large datasets and high dimensional feature spaces. Next, we extend hierarchical template tree approaches [24, 50] to unsegmented images. Finally, we explain how we use random feature selection inspired by Random Forests to build an ensemble of multi-class cascade classifiers. The work presented in this paper is an extension of [42]. We take steps toward generalizing the algorithm and analyze two of its properties that allow a considerable speed-up of the pose detection. We have carried out an exhaustive experimentation to validate our approach with a numerical evaluation (using different publicly available training and testing datasets) and present a comparison with state-of-the-art methods for 3 different levels of analysis: human detection, human pose classification and pose estimation (with body joints localization).

Our approach gives promising fast pose detection performances with both fixed and moving cameras. In the presented work, we focus on specific motion sequences (walking), although the algorithm can be generalized for any action.

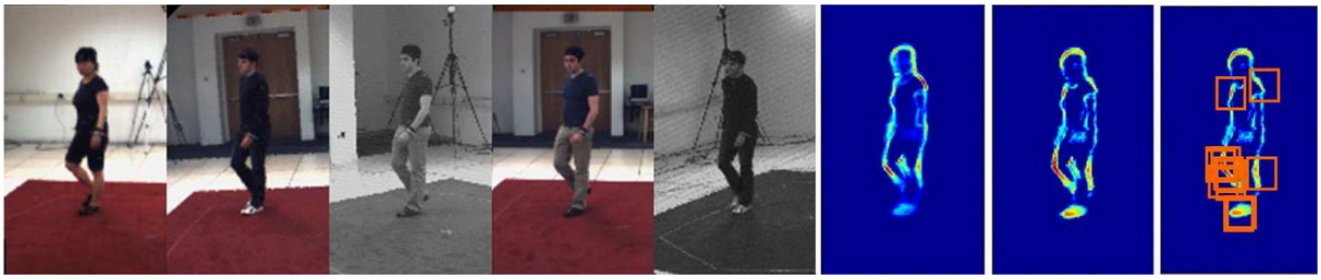
2 Sampling of Discriminative HOGs

Feature selection is probably the key point in most recognition problems. It is very important to select the relevant and most informative features in order to alleviate the effects of the *curse of dimensionality*. Many different types of features are used in general recognition problems. However, only a few of them are useful for exemplar-based pose estimation. For example, features like color and texture are very informative in general recognition problems, but because of their variation due to clothing and lighting conditions, they are seldom useful in exemplar-based pose estimation. On the other hand, gradients and edges are more robust cues with respect to clothing and lighting variations[2]. Guided by their success for both human detection [14, 61] and pose estimation [44] problems, we chose to use HOG descriptors as a feature in our algorithm.

Each HOG block represents the probability distribution of gradient orientation (quantized into a predefined number of histogram bins) over a specific rectangular neighborhood. The usage of HOGs over the entire training image, usually in a grid, leads to a very large feature vector where all the individual HOG blocks are concatenated. So an important question is how to select the most informative blocks in the feature vector. Some works have addressed this question for human detection and pose estimation problems using SVMs or RVMs [5, 14, 37, 61]. However, such learning methods are computationally inefficient for very large datasets.

AdaBoost is often used for discriminative feature selection such as in [54, 58]. Instead of an exhaustive search over all possible features, or uniformly sampling a random subset from these features, we introduce a guided sampling scheme based on log-likelihood gradient distribution. Collins and Liu [13] also use a log-likelihood ratio based approach in the context of adaptive on-line tracking, and select the most discriminative features from a set of 49 RGB features that separate a foreground object from the background. In our case, we have a much higher set of possible features (histogram bins) if we consider all the possible configurations of HOG blocks at all locations exhaustively. So to make the problem more tractable, rather than selecting individual features in a dense HOG vector, entire HOG blocks are randomly selected using a log-likelihood ratio derived from the edge gradients of the training data. Dimitrijevic et al [17] use statistical learning techniques during the training phase to estimate and store the relevance of the different silhouette parts to the recognition task. We use a similar idea to learn relevant gradient features, although slightly different because of the absence of silhouette information. In what follows, we present our method to select the most discriminative and informative HOG blocks for human pose classification. The basic idea is to take advantage of accurate image alignment and study gradient distribution over the entire training set to favor locations that we ex-

[2] Clothing could still be a problem if there are very few subjects in the training set: some edges due to clothing could be considered as discriminative edges when they should not.



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 2 Log-likelihood ratio for human pose. (**a** to **e**): examples of aligned images belonging to the same class (for different subjects and cameras) as defined in Sect. 4.1.1 using HumanEVA dataset [48]. Resulting gradient probability map $p(E|B)$ (**f**) and log-likelihood ratio $L(B,C)$ (**g**) for this same class vs all the other classes. Hot colors in (**g**) indicate the discriminative areas. The sampled HOG blocks are represented on top of the likelihood map in (**h**).

pect to be more discriminative between different classes. Intra-class and inter-class probability density maps of gradient/edge distribution are used to select the best location for the HOG blocks.

2.1 Formulation

Here we describe a simple Bayesian formulation to compute the log-likelihood ratios which can be used to determine the importance of different regions in the image when discriminating between different classes. Given a set of classes C, the probability that the classes represented by C could be explained by the observed edges E can be defined using a simple Bayes rule:

$$p(C|E) = \frac{p(E|C)p(C)}{p(E)}. \tag{1}$$

The likelihood term $p(E|C)$ of the edges being observed given classes C, can be estimated using the training data edges for the respective classes. Let $T = \{(I_i, c_i)\}$ be a set of aligned training images, each with a corresponding class label. Let $T_C = \{(I,c) \in T \mid c \in C\}$ be the set of training instances for the set of classes $C = \{c_i\}$. Then the likelihood of observing an edge given a set of classes C can be estimated as follows:

$$p(E|C) = \frac{1}{|T_C|} \sum_{(I,c)\in T_C} \nabla(I), \tag{2}$$

where $\nabla(\cdot)$ calculates a normalized oriented gradient edge map for a given image I, with the value at any point being in the range $[0,1]$. Note that an accurate alignment of the positive samples is required to compute $p(E|C)$. We refer the reader to the automatic alignment method proposed for human pose in Sect. 4.1.1 for a solution to this problem. Class specific information is represented by high values of $p(E|C)$ from locations where edge gradients occur most frequently across the training instances. Edge gradients at locations that occur in only a few training instances (e.g. due to background or appearance) will tend to average out to low values. To increase robustness toward background noise the likelihood can be thresholded by a lower bound:

$$p(E|C) = \begin{cases} p(E|C) & \text{if } p(E|C) > \tau \ , \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Suppose we have a subset of classes $B \subset C$. Discriminative edge gradients will be those that are strong across the instances within B but are not common across the instances within C. Using the log-likelihood ratio between the two likelihoods $p(E|B)$ and $p(E|C)$ gives:

$$L(B,C) = \log\left(\frac{p(E|B)}{p(E|C)}\right). \tag{4}$$

The log-likelihood distribution defines a gradient prior for the subset B. High values in this function give an indication of where informative gradient features may be located to discriminate between instances belonging to subset B and the rest of classes in C. For the example given in Fig. 2, we can see how the right knee is a very discriminative region for this particular class (see Fig. 2g). In Fig. 3, we present the log-likelihood distributions for 5 different facial expressions.

Gradient orientation can be included by decomposing the gradient map into n_θ separate orientation channels according to gradient orientation. The log-likelihood $L_\theta(B,C)$ is then computed separately for each channel, thereby increasing the discriminatory power of the likelihood function, especially in cases when there are many noisy edge points present in the images. Maximizing over the n_θ orientation channels, the log-likelihood gradient distribution for class B then becomes[3]:

$$\mathcal{L}(B,C) = \max_\theta \left(L_\theta(B,C)\right). \tag{5}$$

We also obtain the corresponding orientation map:

$$\Theta(B,C) = \operatorname*{argmax}_\theta \left(L_\theta(B,C)\right). \tag{6}$$

Uninformative edges from a varied dataset will generally be present for only a few instances and not be

[3] $\mathcal{L}(B,C) = L(B,C)$ if no separated orientation channels are considered.

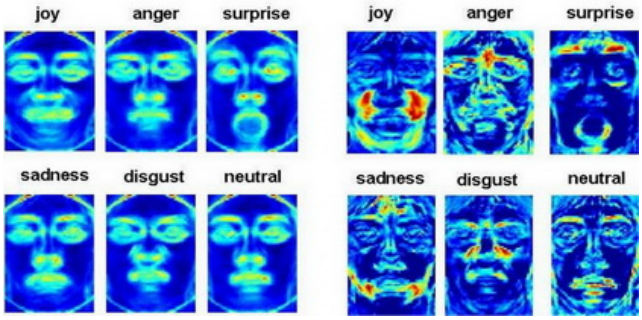


Fig. 3 Log-likelihood ratio for face expressions. We used a subset of [27] composed by 20 different individuals acting 5 basic emotions besides the neutral face: joy, anger, surprise, sadness and disgust. All images were normalized, i.e. cropped and manually rectified. Gradient probability map (*left*) and log-likelihood ratio (*right*) are represented for each one of the 6 classes. Hot colors indicate the discriminative areas for a given facial expression.

common across instances from the same class, whereas common informative edges for pose will be reinforced across instances belonging to a subset of classes B, and be easier to discriminate from edges that are common between B and all classes in parent set C. Even if background edges were shared by many images (e.g. if positive training samples consist of images of standing actions shot by stationary cameras such as the gesture and the box actions in the HumanEVA dataset [48]), then these edges become uninteresting as $p(E|B) \approx p(E|C)$ and a low log-likelihood value would be returned. However, when uninformative edges (from background or clothing) are not common across all training instances but occur in the images of the same class all the time by coincidence [4], edges can be falsely considered as potentially discriminative. To address this issue, we create a varied dataset of instances which is discussed more detail in Sec.4. Given this log-likelihood gradient distribution $\mathcal{L}(B,C)$, we can randomly sample HOG blocks from positions (x,y) where they are expected to be informative, thus reducing the dimension of the feature space. We then use $\mathcal{L}(B,C)$ as distribution proposal to drive blocks sampling $(x^{(i)},y^{(i)})$:

$$(x^{(i)},y^{(i)}) \sim \mathcal{L}(B,C). \quad (7)$$

Features are then extracted from areas of high gradient probability across our training set more than areas with low probability (see Fig. 2h). By using this information to sample features, the amount of useful information available to learn efficient classifiers is increased. Results using this feature selection scheme on facial expression recognition have been reported in [39].

[4] We observed that particular case when trying to work with the Buffy dataset from [21]: all the images for the same pose (standing with arms folded) correspond to one unique character with the same clothes and same background scene.

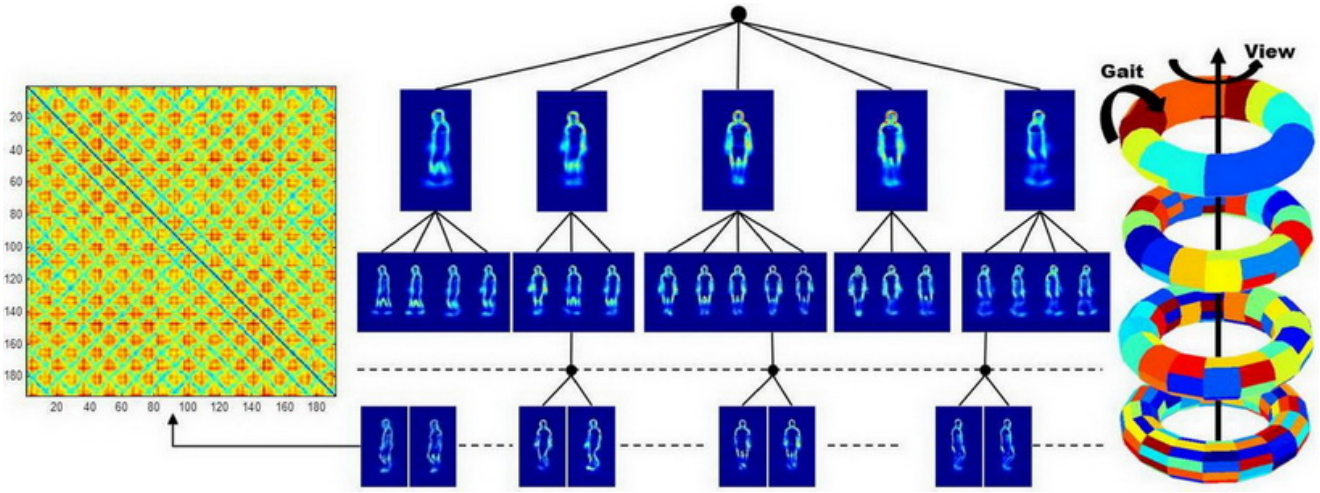
3 Randomized Cascades of Rejectors

The classifier is an ensemble of hierarchical cascade classifiers. The method takes inspiration from cascade approaches such as [55, 61], hierarchical template trees such as [24, 50] and Random Forests [8, 11, 30].

3.1 Bottom-up Hierarchical Tree Construction

Tree structures are a very effective way to deal with large exemplar sets. Gavrila [24] constructs hierarchical template trees using human shape exemplars and the chamfer distance between them. He recursively clusters together similar shape templates selecting at each node a single cluster prototype along with a chamfer similarity threshold calculated from all the templates that the cluster contains. Multiple branches can be explored if edges from a query image are considered to be similar to cluster exemplars for more than one branch in the tree. Stenger [50] follows a similar approach for hierarchical template tree construction applied to articulated hand tracking, the main difference being that the tree is constructed by partitioning the state space. This state space includes pose parameters and viewpoint. Inspired by these two papers, Okada and Stenger [38] present a method for human motion capture based on tree-based filtering using a hierarchy of body poses found by clustering the silhouette shapes. Although these existing template tree techniques are shown to have interesting qualities in terms of speed, they present some important drawbacks for the task we want to achieve. First, templates need to be stored for each node of the tree leading to memory issues when dealing with large sets of templates. The second limitation is that they require that a clean silhouette or template data is available from manual segmentation [24] or generated synthetically from a 3D model [38, 50]. Their methodology can not be directly applied to unsegmented image frames because of the presence of too many noisy edges from background and clothing of the individuals which dominate informative pose-related edges. By using only the silhouette outlines as image features, the approaches in [24, 38] ignore the non-negligible amount of information contained in the *internal edges* which are very informative for pose estimation applications. We thus propose a solution to adapt the construction of such a hierarchical tree structure for images.

Instead of successively partitioning the state space at each level of the tree [50] or clustering together similar shape templates from bottom-up [24] or top-down [38], we propose a hybrid algorithm. Given that a parametric model of the human pose is available (3D or 2D joint locations), we first partition the state space into



(a) (b) (c)

Fig. 4 Bottom-up hierarchical tree construction: the structure $\mathcal{S}$ is built using a bottom-up approach by recursively clustering and merging the classes at each level. We present an example of tree construction from 192 classes (see class definition in Sect. 4.1.2) on a torus manifold where the dimensions represent gait cycle and camera viewpoint. The matrix presented here (**a**) is built from the initial 192 classes and used to merge the classes at the very lowest level of the tree. The similarity matrix is then recomputed at each level of the tree with the resulting new classes. The resulting hierarchical tree-like structure $\mathcal{S}$ is shown in (**b**) while the merging process on the torus manifold is depicted in (**c**). We can observe (**b**) how the first initial node acts as a viewpoint classifier.

a series of classes[5]. Then we construct the hierarchical tree structure by merging similar classes in a bottom-up manner as in [24] but using for each class its gradient map. This process only requires that the cropped training images have been aligned without the need for clean silhouettes or templates.

We thus recursively cluster and merge similar classes based on a similarity matrix that is recomputed at each level of the tree (Fig. 4a). The similarity matrix $M = \{M_{i,j}\}$ with $i, j \in \{1, \cdots, n_l\}$ (being n_l the number of classes at each level) is computed using the L2 distance between the log-likelihood ratios of the set of classes C_n that represent the classes that fall below each node n of the current level and the global edge map C constructed from all the classes together:

$$M_{i,j} = ||\mathcal{L}(C_i, C) - \mathcal{L}(C_j, C)||. \quad (8)$$

Using the log-likelihood ratio to merge classes reduces the effect of uninformative edges on the hierarchy construction while at the same time increasing the influence of discriminative edges. At each level, classes are clustered by taking the values from the similarity matrix in ascending order and successively merge corresponding classes until they all get merged, then go to next level. The class hierarchy construction is depicted in Algorithm 1. This algorithm is fully automatic as it does not require any threshold to be tuned, and works well with continuous and symmetrical pose spaces like the ones considered in this paper. However, we have observed that it fails with non-homogeneous training data or in presence of outliers. In that case, the use of a threshold on the similarity value in the second while loop should be considered to stop the clustering process before merging together classes which are too different. This process leads to a *hierarchical* structure $\mathcal{S}$ (Fig. 4b). The leaves of this tree-like structure $\mathcal{S}$ define the partition of the state space while $\mathcal{S}$ is constructed in the feature space: the similarity in term of image features between compared classes increases and the classification gets more difficult when going down the tree and reaching lower levels as in [24]. But in our case each leaf represents a cluster in the state space as in [50] while in [24], templates corresponding to completely different poses can end-up being merged in the same class making the regression to a pose difficult or even impossible. In [24] the number of branches are selected before growing the tree, potentially forcing dissimilar templates to merge too early. while in our case, each node in the final hierarchy can have 2 or more branches as in [38, 50].

Instead of storing and matching an entire template prototype at each node as in [24, 38, 50], we now propose a method to build a reduced list of discriminative HOG features, thus making the approach more scalable to the challenging size and complexity of human pose datasets.

[5] Two methods have been implemented to obtain the discrete set of classes: the torus manifold discretization (see Sect. 4.1.2) and 2D pose space clustering (see Sect. 4.2.1).

Algorithm 1: Class Hierarchy Construction.

input : Labeled training images.
output: Hierarchical structure $\mathcal{S}$.

while *num. of classes of the level* $n_l > 1$ **do**
  for *each class* n **do**
    Compute new $\mathcal{L}(C_n, C)$ (cf. § 2);
  Compute the $n_l \times n_l$ similarity matrix M (cf. Eq. 8);
  Set the number of merged classes $n_m = 0$;
  Set the number of clusters $n_{cl} = 0$;
  while $n_m < n_l$ **do**
    Take the next 2 closest classes (C_1, C_2) in M;
    – **case 1:** C_1 and C_2 have not been merged yet. Create a new cluster $\mathcal{C}_{n_{cl}+1}$ with C_1 and C_2: $\mathcal{C}_{n_{cl}+1} = C_1 \cup C_2$;
      Update $n_{cl} = n_{cl} + 1$ and $n_m = n_m + 2$;
    – **case 2:** C_1 and C_2 already merged together. Do nothing;
    – **case 3:** C_1 has already been merged in $\mathcal{C}_r$. Merge C_2 in $\mathcal{C}_r$: $\mathcal{C}'_r = \mathcal{C}_r \cup C_2$;
      $n_m = n_m + 1$;
    – **case 4:** C_2 has already been merged in $\mathcal{C}_s$. Merge C_1 in $\mathcal{C}_s$: $\mathcal{C}'_s = \mathcal{C}_s \cup C_1$;
      $n_m = n_m + 1$;
    – **case 5:** $C_1 \in \mathcal{C}_r$ and $C_2 \in \mathcal{C}_s$. Merge the 2 clusters $\mathcal{C}_r$ and $\mathcal{C}_s$: $\mathcal{C}'_r = \mathcal{C}_r \cup \mathcal{C}_s$;
      $n_{cl} = n_{cl} - 1$;
  Create a new level l with new hyper-classes $\{C'_n\}_{n=1}^{n_{cl}}$;
  Update the number of classes for that level $n_l = n_{cl}$;
  Update the structure $\mathcal{S}' = \mathcal{S} \cup \{C'_n\}_{n=1}^{n_l}$;

3.2 Discriminative HOG Blocks Selection

While other algorithms (PSH, RVMs, SVMs, etc) must extract the entire feature space for all training instances during learning, making them less practical when dealing with very large datasets, our method for learning hierarchical cascades only selects a small set of discriminative features extracted from a small subset of the training instances at a time. This makes it much more scalable for very large training sets.

For each branch of our structure $\mathcal{S}$, we use our algorithm for feature selection to build a list of potentially discriminative features, in our case a vector of HOG descriptors. HOG blocks need only to be placed near areas of high edge probability for a particular class. Feature sampling will then be concentrated in locations that are considered discriminative following the discriminative log-likelihood maps (discussed in Sect. 2). For each node, let the set of classes that fall under this node be C_n and the subset of classes belonging to one of its branches b be C_b. Then for each branch b, n_H locations are sampled from the distribution $\mathcal{L}(C_b, C_n)$ (as described in Sect. 2.1) to give a set of potentially discriminative locations for HOG descriptors:

$$H_p = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n_H} \sim \mathcal{L}(C_b, C_n). \quad (9)$$

For each of these positions we sample a corresponding HOG descriptor parameter:

$$H_\Psi = \{\psi^{(i)}\}_{i=1}^{n_H} \in \Psi \quad (10)$$

from a parameter space $\Psi = (W \times B \times A)$ where $W = \{16, 24, 32\}$ is the width of the block in pixels, $B = \{(2,2),(3,3)\}$ are the cell configurations considered and $A = \{(1,1),(1,2),(2,1)\}$ are aspect ratios of the block. An example is proposed in Fig. 5a for a 16×16 HOG block with 2×2 cells.

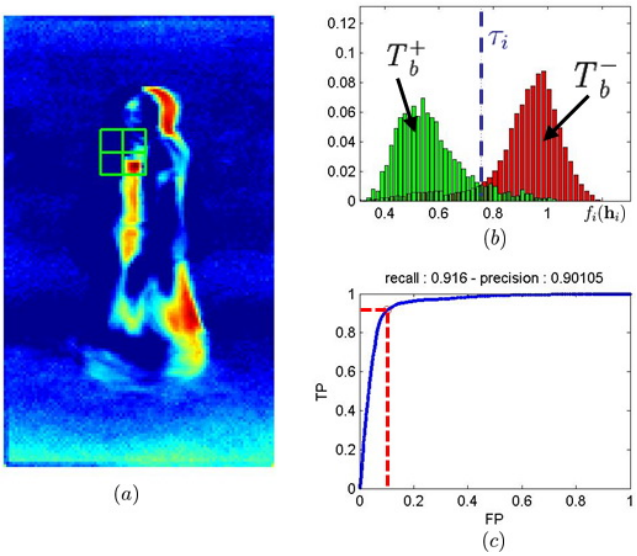


Fig. 5 Example of a selected HOG block from $\mathcal{B}$: we show the location of the 16×16 block (2×2 cells) on top of the log-likelihood map for the corresponding branch of the tree (**a**). We represent in (**b**) the 2 distributions obtained after training a binary classifier: the green distribution corresponds to the set of training images T_b^+ that should pass through that branch while the red one corresponds to the set T_b^- that should not pass. In this example, $f_i(\mathbf{h}_i)$ is the projection of the block $\mathbf{h}_i$ on the hyperplane found by Support Vector Machines (SVM). Finally, we represent in (**c**) the ROC curve with True Positive (TP) vs False Positive (FP) rates varying the decision threshold. We give the precision and recall values for the selected threshold (red dotted line).

Next, at each location $(x^{(i)}, y^{(i)}) \in H_p$ HOG features are extracted from all positive and negative training examples using corresponding parameters $\psi^{(i)} \in H_\Psi$. For each location a positive set T_b^+ is created by sampling from instances belonging to C_b and a negative set T_b^- is created by sampling from $C_n - C_b$. An out-of-bag (OOB) testing set is created by removing 1/3 of the instances from the positive and negative sets. A discriminative binary classifier g_i (e.g. SVM) is trained

Algorithm 2: HOG Blocks Selection

input : Hierarchical structure $\mathcal{S}$, and training images. Discriminative classifier $g(\cdot)$.
output: List of discriminative HOG blocks $\mathcal{B}$.

for *each level* l **do**
  for *each node* n **do**
    Let C_n = set of classes under n;
    for *each branch* b **do**
      Let C_b = set of classes under branch b;
      Compute $\mathcal{L}(C_b, C_n)$ (cf. § 2);
      $H_p = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n_H} \sim \mathcal{L}(C_b, C_n)$;
      $H_\Psi = \{\psi^{(i)}\}_{i=1}^{n_H} \in \Psi$;
      for $i = 1$ *to* n_H **do**
        $(x^{(i)}, y^{(i)}) \in H_p$;
        $\psi_i \in H_\Psi$;
        Let $T_b^+ \in C_b$;
        Let $T_b^- \in C_n - C_b$;
        for *all images under* n **do**
          Extract HOG at $(x^{(i)}, y^{(i)})$ using $\psi^{(i)}$;
        Let h_i^+ = HOG from T_b^+;
        Let h_i^- = HOG from T_b^-;
        Train classifier g_i on $\frac{2}{3}$ of: h_i^+ and h_i^-;
        Test g_i on OOB set $\frac{1}{3}$ of: h_i^+ and h_i^-;
        Rank block $(x^{(i)}, y^{(i)}, \psi^{(i)}, g_i)$
      Select n_h best blocks, $B_b = \{(x_j, y_j, \psi_j, g_j)\}_{j=1}^{n_h}$;
      Update the list $\mathcal{B}' = \mathcal{B} \cup B_b$;

using these examples. We then test this weak classifier on the OOB test instances to select a threshold τ_i and rank the block according to the actual True Positive (TP) and False Positive (FP) rates achieved: the overall performance of the weak classifier g_i is determined by selecting the point on its ROC curve lying closest to the upper-left hand corner that represents the best possible performance (i.e. 100% TP and 0% FP). We then rank the rejector using the Euclidean distance to that point. Each weak classifier $g_i(h_i)$ thus consists of a function f_i, a threshold τ_i and a parity term p_i indicating the direction of the inequality sign:

$$g_i(\mathbf{h_i}) = \begin{cases} 1 & \text{if } p_i f_i(\mathbf{h}_i) < p_i \tau_i \ , \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Here $\mathbf{h}_i$ is the HOG extracted at location $(x^{(i)}, y^{(i)})$ using parameters $\psi^{(i)}$. In the example proposed in Fig. 5b, $f_i(\mathbf{h}_i)$ is the projection of the block $\mathbf{h}_i$ on the hyperplane found by SVM. The corresponding ROC curve is shown in Fig. 5c. For each branch, the n_h best blocks are kept in the list $\mathcal{B}$ which is a bag/pool of HOG blocks and associated weak classifiers. If n_B is the total number of branches in the tree over all levels, the final list $\mathcal{B}$ has $n_B \times n_h$ block elements. The selection of discriminative HOG blocks is depicted in Alg. 2.

By this process, features are extracted from areas of high edge probability across our training set more than areas with low probability. By using this information to sample features, the proportion of useful information available for random selection is increased.

3.3 Randomization

Random Forests, as described in [11] are constructed as follows. Given a set of training examples $T = \{(y_i, \mathbf{x_i})\}$, where y_i are class labels and $\mathbf{x_i}$ the corresponding D-dimensional feature vectors, a set of random trees F is created such that for the k-th tree in the forest, a random vector Φ_k is used to grow the tree resulting in a classifier $t_k(\mathbf{x}, \Phi_k)$. Each element in Φ_k is a randomly selected feature index for a node. The resulting forest classifier F is then used to classify a given feature vector $\mathbf{x}$ by taking the mode of all the classifications made by the tree classifiers $t \in F$ in the forest.

Each vector Φ_k is generated independently of the past vectors $\Phi_1..\Phi_{k-1}$, but with the same distribution:

$$\phi \sim \mathcal{U}(1, D), \forall \phi \in \Phi_k, \tag{12}$$

where $\mathcal{U}(1, D)$ is a discrete uniform distribution on the index space $\mathcal{I}_D = \{1, \cdots, D\}$. The dimensionality of Φ_k depends on its use in the tree construction (i.e. the number of branches in the tree). For each tree, a decision function $f(\cdot)$ splits the training data that reaches a node at a given level in the tree by selecting the best feature m* from a subset of $m << D$ randomly sampled features (typically $m = \sqrt{D}$ dimensions). An advantage of this method over other tree based methods (e.g. single decision trees) is that since each tree is trained on a randomly sampled $\frac{2}{3}$ of the training examples [10] and that only a small random subset of the available dimensions are used to split the data, each tree makes a decision using a different view of the data. Each tree in the forest F learns quite different decision boundaries, but when averaged together the boundaries end up reasonably fitting the training data.

In Random Forests (RF), the use of randomization over feature dimensions makes the forest less prone to over-fitting, more robust to noisy data and better at handling outliers than single decision trees [11]. We exploit this random selection of features in our algorithm so that our classifier will also be less susceptible to overfitting and more robust to noise compared to a single hierarchical decision tree that would use all the selected features together: a hierarchical tree-structured classifier $r_k(I_N, \Phi_k, \mathcal{S}, \mathcal{B})$, with I_N being a normalized input image, is thus built by randomly sampling one of the HOG blocks in the list $B_b \in \mathcal{B}$ at each branch b of

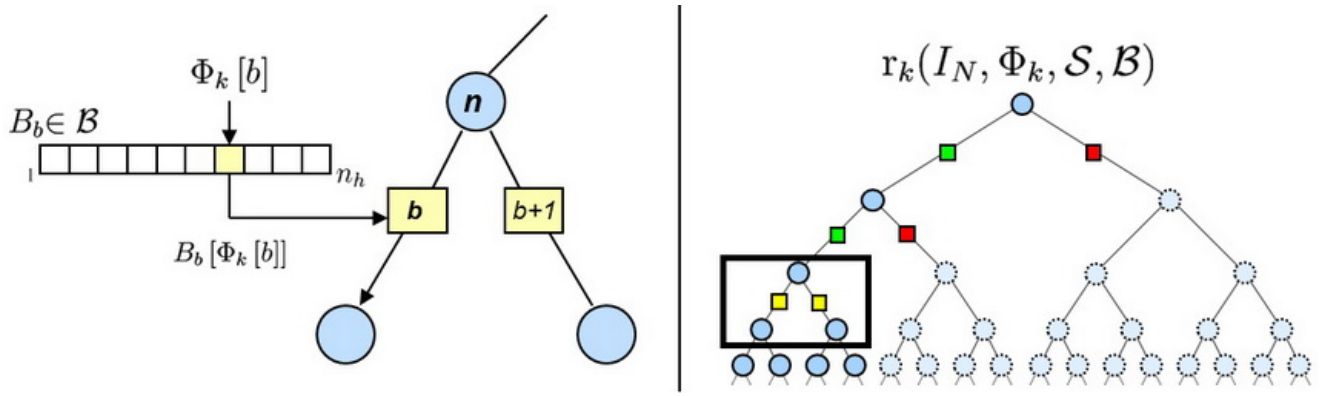


Fig. 6 Rejector branch decision: shown here is a diagram of a rejector classifier. Note that the structure allows for any number of branches at each node and is not fixed at 2. The tree-like structure of the classifier is defined by $\mathcal{S}$ and the blocks (and associated weak classifiers) stored during training for each branch for this structure are held in $\mathcal{B}$. The random vector Φ_k defines a cascade r_k by selecting one of the rejector blocks in B_b at each branch b, being $B_b \in \mathcal{B}$ the bag of HOG blocks for branch b.

the hierarchical structure $\mathcal{S}$. This gives a random n_B-dimensional vector Φ_k where each element corresponds to a branch in the structure $\mathcal{S}$:

$$\Phi_k \in \mathcal{I}^{n_B} \text{ where } \phi \sim \mathcal{U}(1, n_h), \forall \phi \in \Phi_k. \tag{13}$$

Here $\mathcal{U}(1, n_h)$ is a discrete uniform distribution on the index space $\mathcal{I} = \{1, \cdots, n_h\}$. The value of each element in Φ_k is the index of the randomly selected HOG block from B_b for its corresponding branch. An ensemble R of n_c hierarchical tree-structured classifiers is grown by repeating this process n_c times:

$$R(I_N) = \{r_k(I_N, \Phi_k)\}_{k=1}^{n_c}, \tag{14}$$

where $\mathcal{S}$ and $\mathcal{B}$ are left out for brevity. R thus has 2 design parameters n_c and n_h whose main effects on performance will be evaluated in Sect. 4.2. Let $\mathcal{P}_\omega$ be the path through the tree-structure $\mathcal{S}$ from the top node to the class leaf ω. $\mathcal{P}_\omega$ is in fact an ordered sequence of ${n_b}^\omega$ branches[6]: $\mathcal{P}_\omega = \left(b_1^\omega, b_2^\omega, \cdots, b_{{n_b}^\omega}^\omega\right)$. For each classifier $r_k \in R$ and for each class $\omega \in \Omega$, where $\Omega = \{1, \cdots, n_\omega\}$, we have the corresponding ordered sequence of HOG blocks and associated weak classifiers:

$$\mathcal{H}_k^\omega = \{(x_j, y_j, \psi_j, g_j)\}_{j=1}^{{n_b}^\omega}, \tag{15}$$

$$\text{with } (x_j, y_j, \psi_j, g_j) = B_b[\Phi_k(b)], \tag{16}$$

where the branch index b is the j^{th} element in $\mathcal{P}_\omega$ (i.e $b = \mathcal{P}_\omega[j]$) and $B_b \in \mathcal{B}$. See Fig. 6a.

For each tree-structured classifier r_k, the decision to explore any branch in the hierarchy is based on a accept/reject decision of a simple binary classifier g_j that works in a similar way to a cascade decision (see Fig. 6b). The ensemble classifier R is then, in fact, a series of *Randomized Hierarchical Cascades of Rejectors*.

[6] Note that ${n_b}^\omega$ could be different for each class $\omega \in \Omega$

The decision at each node of a hierarchical cascade classifier r_k is made in a one-vs-all manner, between the branch in question and its sibling branches. In this way multiple paths can be explored in the cascade that can potentially vote for more than one class. This is a useful attribute for classifying potentially ambiguous classes and allows the randomized cascades classifier to produce a distribution over multiple likely poses. Each cascade classifier r_k therefore returns a vector of binary outputs (yes/no) $\mathbf{o}_k = \left(o_k^1, o_k^2, \cdots, o_k^{n_\omega}\right)$ where a given o_k^ω from $\mathbf{o}_k$ for class index ω, takes a binary value:

$$o_k^\omega = \begin{cases} 1 & \text{if } \forall (x_j, y_j, \psi_j, g_j) \in \mathcal{H}_k^\omega,\ g_j(\mathbf{h}_j) = 1\ , \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Here $\mathbf{h}_j$ is the HOG extracted at location (x_j, y_j) using parameters ψ_j. Each output o_k^ω is initialized to zero, which means that if no leaf is reached during classification, r_k will return a vector of zeros and will not contribute to the final classification. The uncertainty associated with each rejector during learning could be considered to provide a crisp output (i.e. each cascade voting for only one class) or to compute a soft confidence values of the cascade classifiers. Although other classifier combination techniques could be considered, here we choose to use a simple sum rule to combine the binary votes from the different cascade classifiers in the ensemble R that outputs the vector $\mathbf{O}$:

$$\mathbf{O} = \left(O^1, O^2, \cdots, O^{n_\omega}\right) = R(I), \tag{18}$$

where each $O^\omega \in [0, n_c]$ represents a number of *votes*:

$$O^\omega = \sum_{k=1}^{n_c} o_k^\omega, \tag{19}$$

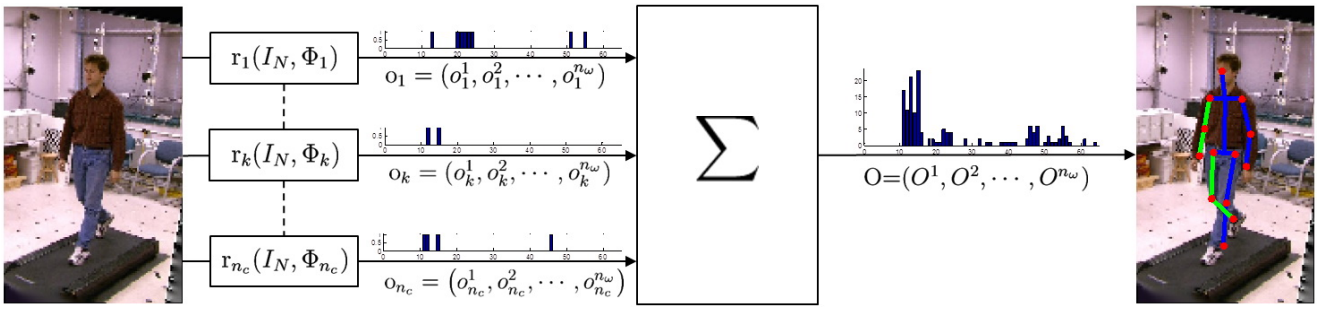


Fig. 7 Diagram of Image classification using an ensemble of n_c randomized cascades (*from left to right*): When applied to a normalized input image I_N, each cascade classifier $r_k(I_N, \Phi_k, \mathcal{S}, \mathcal{B})$ returns a vector of binary outputs $\mathbf{o}_k = \left(o_k^1, o_k^2, \cdots, o_k^{n_\omega}\right)$. A sum-rule is used to combine the binary votes from the different cascade classifiers in the ensemble R that outputs the distribution over classes $\mathbf{O} = (O^1, O^2, \cdots, O^{n_\omega})$, with $O^\omega = \sum_{k=1}^{n_c} o_k^\omega$ and $O^\omega \in [0, n_c]$, being n_ω the number of classes and n_c the number of cascades. The image I_N can be classified by taking, for instance, the class ω^* that received more *votes*, i.e. the peak in the final distribution over classes. The average 2D pose corresponding to class ω^* is shown on the *right*.

and can be used to assign a confidence value[7] (or score) to class ω. $\mathbf{O}$ can ultimately be an input in to another algorithm, e.g. tracking, or a class can be estimated by majority voting, i.e. taking for instance the mode of all the classifications as in the example presented in Fig. 7.

Each hierarchical cascade in the ensemble can make a decision and efficiently reject negative candidates by only sampling a few features of the available feature space. This makes our classifier more suitable for sliding window detectors than RF where the trees have to be completely traversed to produce a vote.

3.4 Application to Human Pose Detection

Since we learn a classifier that is able to discriminate between very similar classes, we can also tackle localization. Given an image I, a sliding-window mechanism then localizes the individual within that image and, at each visited location (x, y) and scale s, a window $I_{\mathbf{p}}$ can be extracted and classified by our multi-class pose classifier R obtaining $\mathbf{O}_{\mathbf{p}} = \left(O_{\mathbf{p}}^\omega\right)_{\omega=1}^{n_\omega}$ where $\mathbf{p} = (x, y, s)$, is a location vector that defines the classified window.

A multi-scale saliency map $\mathcal{M}$ of the classifier response can be generated by taking the maximum value of the distribution for each classified window $I_{\mathbf{p}}$:

$$\mathcal{M}(x, y, s) = \max_{\omega \in \Omega}(\mathbf{O}_{\mathbf{p}}) = \max_{\omega \in \Omega}\left((O_{\mathbf{p}}^\omega)_{\omega=1}^{n_\omega}\right). \tag{20}$$

A dense scan (trained on pose only) is shown in Fig. 8a where many isolated false positives appear where the classifier responds incorrectly.

Zhang et al [59] tackle joint object detection and pose estimation by using a graph-structured network that alternates the two tasks of foreground/background discrimination and pose estimation for rejecting negatives as quickly as possible. Instead of combining binary foreground/background and multi-class pose classifiers, we propose to perform the two tasks simultaneously by including hard background examples in the negative set T_b^- (see Fig. 5 and Sect. 3.2) during the training of our rejectors. To create the hard negatives dataset, the classifiers trained on pose images only can be run on negative examples (e.g. from the INRIA dataset [14]) and strong positive classifications are incorporated as hard negative examples (see details in Sect. 4.2.3). Repeating the process of hard negative retraining several times helps to refine the classifiers as can be appreciated in Fig. 8 b and c. Numerical evaluation is given in the results Sect. 4.2.

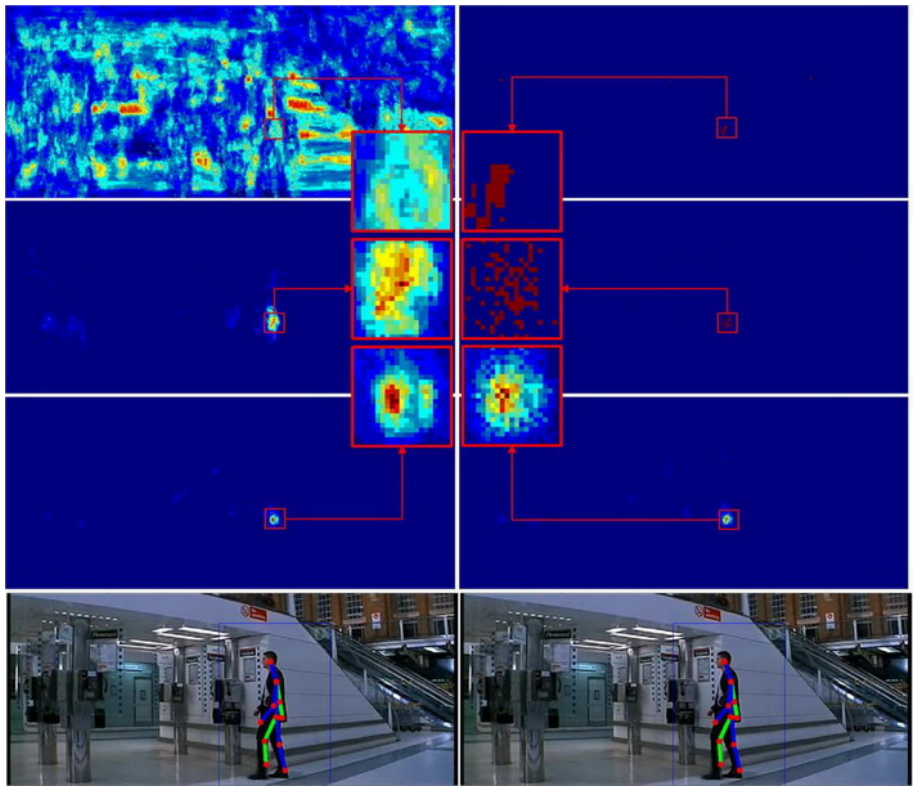
By generating dense scan saliency maps for many images, we have found that humans tend to have large "cores" of high confidence value as in Fig. 8c. This means that a coarse localization can be obtained with a sparse scan and a local search (e.g. gradient ascend method) can then be used to find the individual accurately. In the example proposed in Fig. 8, taking the maximum classification value over the image, (after exploring all the possible positions and scales) results in reasonably good localization of a walking pedestrian. In that case, we have:

$$\mathbf{p}^* = (x^*, y^*, s^*) = \underset{(x,y,s)}{\operatorname{argmax}}(\mathcal{M}(x, y, s)), \tag{21}$$

and the corresponding distribution over poses $\mathbf{O}^* = \mathbf{O}_{\mathbf{p}^*} = \left(O_{\mathbf{p}^*}^\omega\right)_{\omega=1}^{n_\omega}$.

Once a human has been detected and classified, 3D joints for this image can be estimated by weighting the mean poses of the classes resulting from the distribution using the distribution values as weights, or regressors can be learnt as in [37]. The normalized 2D pose is computed in the same way and re-transformed from the

[7] Although it can be expressed as a *percentage of votes* by dividing by the number of cascades n_c, $\widetilde{O}^\omega = \frac{1}{n_c}\sum_{k=1}^{n_c} o_k^\omega$ and $\widetilde{O}^\omega \in [0, 1]$, it does not express a probability ($\sum_{k=1}^{n_\omega} \widetilde{O}^\omega \neq 1$).



(a) 100-cascade classifier trained on pose images only

(b) 100-cascade classifier after 1 Pass of hard negatives retraining

(c) 100-cascade classifier after 3 Passes of hard negatives retraining

(d) Input image with the detected pose from the "peak" in (c)

(e) 1-cascade classifier after 3 Passes of hard negatives retraining

(f) On-line randomization of a 1-cascade classifier (3 Passes of hard neg)

(g) On-line randomization of a 100-cascade classifier (3 Passes of hard neg)

(h) Input image with the detected pose from the "peak" in (g)

Fig. 8 Cascade classifier response for a dense scan. *Left side, from top to down*: saliency map $\mathcal{M}$ after 0 (**a**), 1 (**b**) and 3 (**c**) passes of hard negatives retraining (i.e. adding hard negative examples in the cascade learning process). For visualization purpose we represent $\mathcal{M}$ for ground truth scale (i.e. $s = 0.6$) and show a zoom around the ground truth location. (**d**) input image with the pose corresponding to the peak in (**c**). *Right side, from top to down*: saliency map $\mathcal{M}$ obtained with a 1-cascade (**e**), 1 cascade randomized on-line (**f**), a 100-cascade ensemble randomized on-line (**g**). In (**h**) we show the pose corresponding to the peak in (**g**).

normalized bounding box to the input image coordinate system obtaining the 2D joints location (see Fig. 8d).

3.5 Properties of the Random Cascades Classifiers

In addition to the advantages stated so far, our classifier has additional interesting properties. Since the tree structure $\mathcal{S}$ and HOG block rejectors list $\mathcal{B}$ remain fixed after training, a new cascade classifier $\widetilde{r_k}(I_N, \widetilde{\Phi}_k)$ can be constructed *on-line* by simply creating a new random vector $\widetilde{\Phi}_k$. This means that for each classified window $I_{\mathbf{p}}$ in an image, a different ensemble $R_{\mathbf{p}}$ can be regenerated instantly at no extra-cost:

$$R_{\mathbf{p}}(I_N) = \{\widetilde{r_k}(I_N, \widetilde{\Phi_k})\}_{k=1}^{n_c}, \tag{22}$$

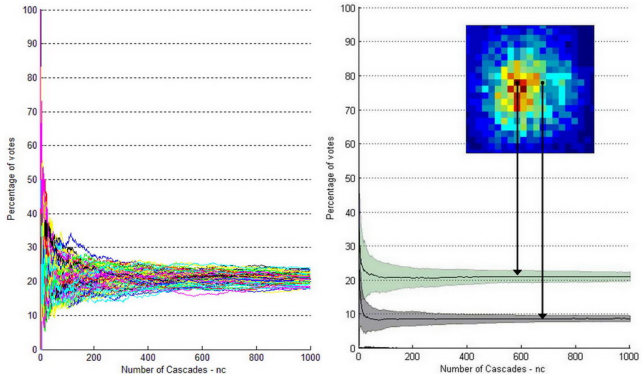
where $\mathbf{p} = (x, y, s)$, is the location vector that defines $I_{\mathbf{p}}$ and $\widetilde{\Phi_k}$ is sampled using Eq. 13.

The qualitative effects of this *on-line randomization* can be appreciated in Fig. 8: a dense scan (1-pixel stride) with a 1-cascade classifier (Fig. 8e) produces responses which are grouped while randomizing Φ on-line (using a different random cascade at each location) produces a cloud of responses around the ground truth (see Fig. 8f). This property will favor an efficient localization at a higher search stride, as verified later by the numerical evaluation presented in Sect. 4.2 [8]. When randomizing a 100-cascade classifier (Fig. 8g), the saliency map becomes similar to the one obtained with a regular 100-cascade ensemble without on-line randomization (Fig. 8d).

Performing construction on-line, each new random vector $\widetilde{\Phi_k}$ contributes to the final distribution. The probability that any given randomly generated cascade classifier $\widetilde{r_k}$ will vote close to an object location increases as the position gets closer to the true location of the object. Even if the classification by the initial cascade for the pose class is wrong, it is generally close to an area where the object is. Subsequent classifications from other randomized rejectors push the distribution toward a stable result. As more classifications are made, the distribution converges and becomes stable as shown in Fig 9a and Fig 9b where we can observe the higher

[8] This property may also be exploited to spread the computation of image classification and localization over time. They may also be readily parallelized due to the structure $\mathcal{S}$ and $\mathcal{B}$ being fixed once the classifier has been trained.

variability in classification with few cascades compared to the one obtained using bigger ensembles. This explains the similarity of the saliency map obtained with and without on-line randomization (Fig. 8) when considering a large number of cascades in the ensemble R.



(a) (b)

Fig. 9 On-line Randomization: (**a**) Detection score (in percentage of votes) $\max(\mathbf{O}) = \max(\frac{100}{n_c}\sum_{k=1}^{n_c} o_k^\omega)$, i.e. the peak of the distribution, obtained when classifying the same location (x, y, s) with 100 different randomized ensembles. When increasing n_c, the number of cascades in the ensemble R, all the curves converge to 20%. In (**b**), we present the average and std across all these curves for 2 different locations in the image: ground truth location and 5 pixels away from ground truth.

This convergence property can also be exploited for fast localization using *cascade thresholding*: a cascade r_k is drawn at random (without replacement) from R until the aggregated score reaches a sufficient confidence level. The confidence level can be selected based on a desired trade-off between speed and accuracy. Another possibility is to consider an *adaptive cascade filtering* by classifying with the entire ensemble R ($R_{\mathbf{p}}$ if on-line randomization is considered) only the locations $\mathbf{p} = (x, y, s)$ that yield a detection using a single or few cascades from the ensemble, i.e. filtering with the first n_f cascades of the ensemble:

$$\mathcal{M}(x, y, s) = \begin{cases} \max_{\omega\in\Omega}\left(O_{\mathbf{p}}^{\omega} \in \mathbf{O}_{\mathbf{p}}\right) & \text{if } S_{n_f} > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (23)$$

where the detection score S_{n_f} is basically:

$$S_{n_f} = \max_{\omega\in\Omega}\left(\sum_{k=1}^{n_f} o_k^\omega\right). \quad (24)$$

In other words, if a strong vote for background has been made with the first n_f cascades (i.e. no vote for any human pose classes and $S_{n_f} = 0$), then the classifier stops classifying with the rest of the cascades in the ensemble. Localizing using this approximate approach means that a classifier with a few cascades can be used as a region of interest detector for a more dense classification. The *adaptive cascade filtering* combined with *on-line randomization* will produce an efficient and fast detection, as verified later in Sect. 4.2.

4 Experiments

4.1 Evaluation using HumanEva dataset

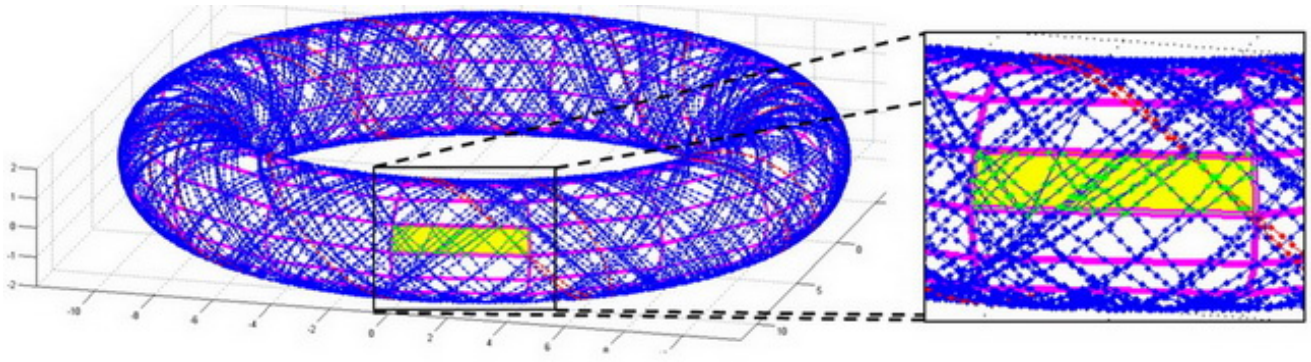
The first dataset we consider is the HumanEva dataset [48]: HumanEVA I for training and HumanEVA II for testing. This dataset consists of 4 subjects performing a number of actions (e.g. walking, running, gesture) all recorded in a motion capture environment so that accurate ground truth data is available.

4.1.1 Alignment of Training Data

Before training the algorithm, strong correspondences are established between all the training images.

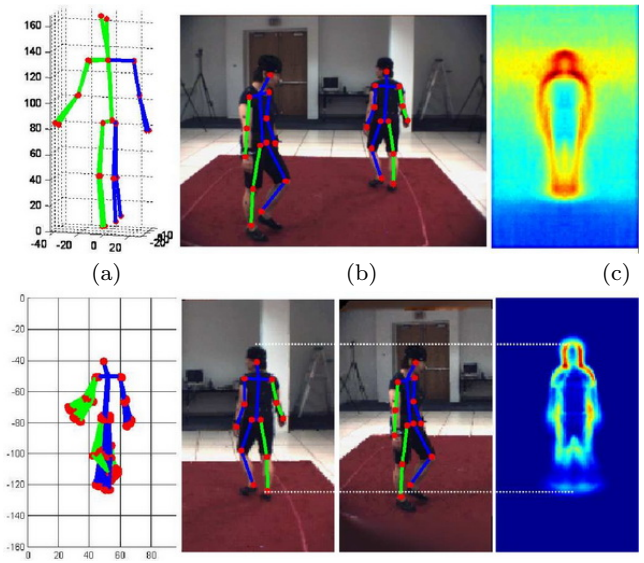
While other approaches require a manual process [14] or a clean silhouette shape (either synthetic [1, 17, 44] or from manual labeling [24, 41]) for feature alignment, we propose a fully automatic process for accurately aligning real training images. It is assumed that the 3D mocap data (3D poses) or other pose labeling corresponding to the training images are available. Then by applying a simple but effective way of using their 2D pose projections in the image plane, all of the training images are aligned.

The complete process is depicted in Fig. 10: the 3D joints of every training pose are first projected onto the corresponding image plane (Fig. 10b). The resulting 2D joints are then aligned using rigid-body Procrustes alignment [7], uniformly scaled and centered in a reference bounding-box (Fig. 10d). For each training pose, we then estimate the four parameters corresponding to a similarity transformation (one for rotation, two degrees of freedom for translation and one scale parameter) between the original 2D joints locations in the original input image and the corresponding aligned and resized 2D joints. This transformation is finally applied to the original image leading to the cropped and aligned image (Fig. 10e and f). In this work, we normalize all the training images to 96×160 as in [14]. The dataset (images and poses) is then flipped along the vertical axis to double the training data size. Applying the process described above to this data (3 Subjects and 7 camera views) for training, we generate a very large dataset of more than 40,000 aligned and normalized 96×160 im-



(a) (b)

Fig. 11 Class definition - HumanEVA: (**a**) Torus manifold with discrete set of classes (12 for gait and 16 for camera viewpoint) and training sequences (each blue dot represents a training image). (**b**) Zoom on a particular class (yellow) with corresponding training images (green dots). Please refer to Fig. 2 where we present 6 examples of aligned images belonging to the class highlighted in (**b**).



(a) (b) (c) (d) (e) (f) (g)

Fig. 10 Alignment of training images. (**a**) Training 3D poses from Mocap data. (**b**) Training images with projected 2D Poses. (**c**) Average gradient image over INRIA training examples (96×160). (d) Aligned and re-sized 2D poses (96×160). (**e**) and (**f**) cropped images (96×160) with normalized 2D poses. (**g**) Average gradient image over aligned HumanEva [48] training examples.

ages of walking people[9], with corresponding 2D and 3D Poses. After that is done, classes need to be defined to train our pose classifier.

[9] The average gradient image obtained with our training dataset (Fig. 10g) shows more variability in the lower region compared to the one obtained from INRIA dataset (Fig. 10c). This is due to the fact that most of the INRIA images present standing people.

4.1.2 Class Definition - Torus Manifold Discretization

Since similar 3D poses can have very different appearance depending on the camera viewpoint (see Fig. 10a and Fig. 10b), the viewpoint information has to be included into the class definition. To define the set of classes, we thus propose to utilize a 2D manifold representation where the action (consecutive 3D poses) is represented by a 1D manifold and the viewpoint by another 1D manifold. Because of the cyclic nature of the viewpoint parameter, if it is modeled with a circle the resulting manifold is in fact a cylindrical one. When the action is cyclic too, as with gait, jog etc., the resulting 2D manifold lies on a "closed cylinder" topologically equivalent to a torus [18, 41]. The walking sequences are then mapped on the surface of this torus manifold and classes are defined by applying a regular grid on the surface of the torus thus discretizing gait and camera viewpoint[10]. By this process, we create a set of 192 homogeneous classes with about the same number of instances (see Fig. 11).

We choose to define non-overlapping class quantization, due to the property of our hierarchical cascade classifier that each cascade compares feature similarity rather than make a greedy decision, so can traverse more than one branch in the hierarchical cascade. When an image reaches a node with a decision between very similar classes (and subsequently close in pose space), then it is possible that a query image that lies close to a quantization border between those two classes can arrive at both class leaves.

[10] Since we only learnt our pose detector from walking human sequences, we do not attempt to detect people performing other actions than walking.

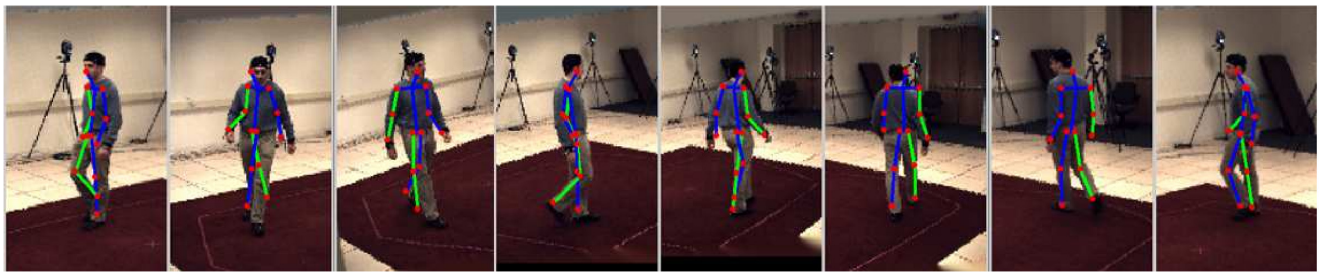


Fig. 12 Pose detection results on HumanEva II dataset: normalized 96×160 images corresponding to the peak obtained when applying the pose detection in one of the HumanEva II sequences (subject S2 and Camera 1). For each presented frame (1, 50, 100, 150, 200, 250, 300 and 350) the resulting pose is represented on top of the cropped image.

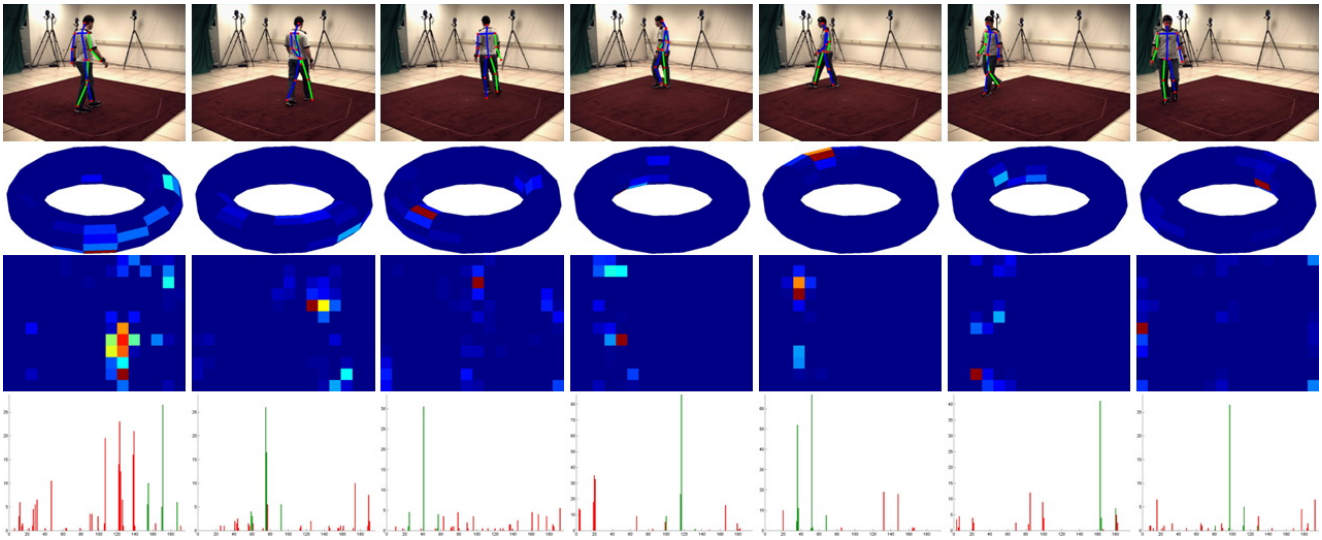


Fig. 13 Pose detection results on HumanEva II Subject S2-Camera C2. *From top to down*: for each presented frame (2, 50, 100, 150, 200, 250 and 300) the reprojected 2D pose is represented on top of the input image (*top row*). The resulting distributions over the 192 classes after classification using our random cascades classifier is represented on the 3D and 2D representations of the torus manifold. In the bottom row, we show the distribution after selection (in green) of the most probable classes based on spatiotemporal constraints (i.e. transitions between neighboring classes on the torus).

Table 1 2D Pose Estimation Error on HumanEva II dataset: mean (and standard deviation) of the 2D joints location error (in pixels) obtained by other state-of-the-art approaches and our Randomized Cascades.

Subject	Camera	Frames	Rogez et al [41]	Gall et al [23]	Bergtholdt et al [4]	Andriluka et al [3]	our approach
S2	C1	1-350	16.96 ± 4.83	4.10 ± 1.11	25.48 ± 13.18	10.49 ± 2.70	12.98 ± 3.5
S2	C2	1-350	18.53 ± 5.97	4.38 ± 1.36	25.48 ± 13.18	10.72 ± 2.44	14.18 ± 4.38
S4	C1	1-290	16.36 ± 4.99	3.58 ± 0.74	38.82 ± 16.32	-	16.67 ± 5.66
S4	C2	1-290	14.88 ± 3.44	3.35 ± 0.51	38.82 ± 16.32	-	13.03 ± 3.49

4.1.3 Pose Estimation

The resulting cascades classifier is first validated in similar conditions (i.e. indoor with a fixed camera) using HumanEVA II dataset (Fig. 12 and Fig. 13). We apply a simple Kalman filter on the position, scale and rotation parameters along the sequence and locally look for the maxima, selecting only the probable classes based on spatiotemporal constraints (i.e. transitions between neighboring classes on the torus, see [41] for details). By this process, we do not guarantee to reach the best result but a reasonably good one in relatively few iterations. Quantitative evaluation is provided in Tab.1 together with the results reported by [3, 4, 23, 41]. Gall et al [23] present the best numerical results using a multi-layer framework based on background subtraction with local optimization while Andriluka et al [3] use extra training data and optimize over the entire sequence. Our results are obtained training on HumanEVA I only and estimating the pose frame by frame without background subtraction.

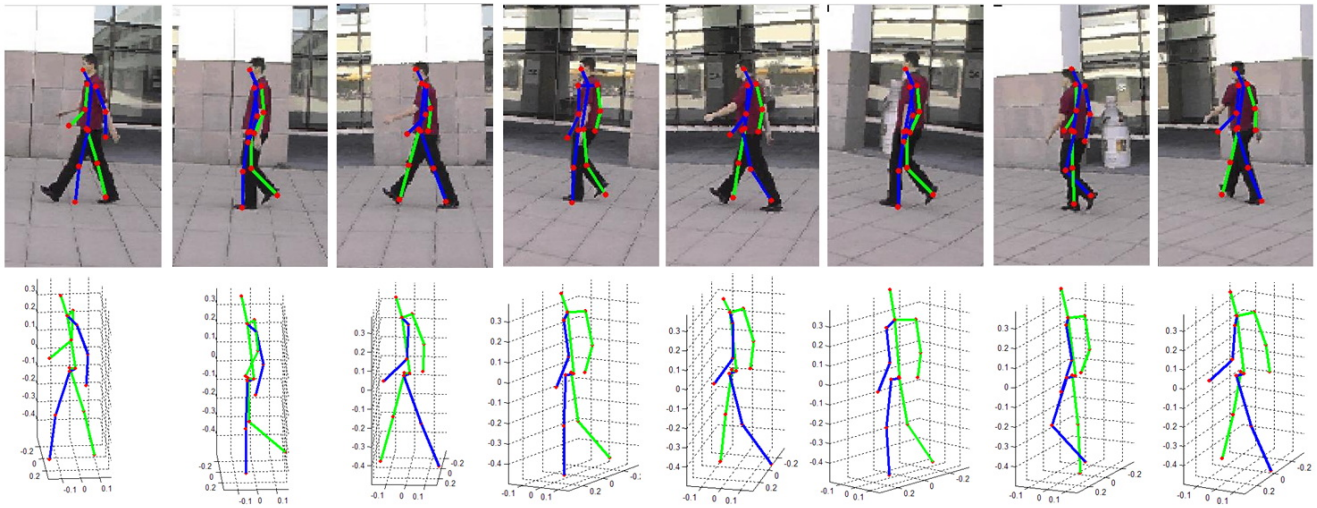


Fig. 14 Pose detection result with a moving camera. Normalized 96×160 images corresponding to the peak obtained applying the pose detection to a moving camera sequence (from [22]). For each presented frame, the mean 2D pose corresponding to the "winning" class is represented on top of the cropped image while the corresponding 3D pose is presented just below.

The cascades classifier is also applied to a moving camera sequence (see Fig. 14): even if the cascade classifier shows good performances, it can be observed in Fig. 14 that the classifier is unable to make a good pose estimate if the gait stride is too wide (e.g. when the subject was walking at speed). This is due to the low variability in pose present in HumanEVA walking database: even if 40,000 images are available, they are not representative of the variability in terms of gait style since only 3 subjects walking at the same speed have been considered. Additionally, HumanEva has very little background (one unique capture room) and clothing (capture suit) variation, which make the classifier unrobust against cluttered background.

4.2 Experimentation using the MoBo Dataset

Our second set of experiments is performed using the CMU Mobo dataset [25]. Again, we consider the walking action but this time add more variability in the dataset by including 15 subjects, two walking speeds (high and low) and a discretized set of 8 camera viewpoints, uniformly distributed between 0 and 2π. Between 30 and 50 frames of each sequence were selected in order to capture exactly one gait cycle per subject. By this process we generate a training database encompassing around 8000 images and the corresponding 2D and 3D pose parameters. The same alignment procedure is applied to this dataset as with the HumanEva dataset, but in addition we use hand labeled silhouette information to superimpose each of the instances on random backgrounds to increase the variability of the nonhuman area of the image (see Fig. 15). For more details about manual labeling of silhouettes and pose, please refer to [41].

Fig. 15 MoBo dataset (*from top to bottom*): we show an original image for each one of the 6 available camera views (*1st row*). We then present a normalized 96×160 binary segmentation per training view (*2nd row*), the corresponding segmented images (*3rd row*) and images with a random background (*4th row*).

We observed that the classification in section 4.1 was very strict so that very fine alignment was necessary during localization. Looser alignment in the training should allow for a smoother detection confidence landscape. Following [28] and [21], the training set is thus augmented by perturbing the original examples with small rotations and shears, and by mirroring them

horizontally. This improves the generalization ability of the classifier. The augmented training set is 6 times larger and contains more than 48,000 examples with different background. The same dataset is generated for the 3 following configuration: original background, no background, and random background (see Fig. 15). This dataset will enable to measure the effect of cluttered background on classification. The richer background variation compared to HumanEva dataset allows a more robust cascade to be learnt, as demonstrated later by our experiments.

4.2.1 Class Definition - 2D Pose Space Clustering

Class definition is not a trivial problem because changes in the human motion are continuous and not discrete. In other words, it is not trivial to decide where a class ends and where the next one starts. Previously no one has attempted to efficiently define classes for human pose classification and detection. In [37] they clustered 3D pose space and also considered a discrete set of cameras. Gavrila [24] clustered the silhouette space using binary edge maps. Ferrari et al [21] defined classes as 3D pose but only considered frontal views and a limited set of poses. In PSH [44], they automatically build the neighborhood in 3D pose space but only consider frontal views. This definition produced good results in the absence of viewpoint changes. However, two poses which are exactly the same in the pose space could still have completely different appearances in the images due to changes in viewpoint (see example in Fig. 10a and b). Thus it is critical to consider viewpoint information in the class definition.

Ideally, classes should be defined in the feature space or, at least, in the 2D pose space (2D projection of the pose in the image) that takes into account the information of the camera viewpoint. When we considered the HumanEVA dataset (section 4.1), classes were defined applying a regular grid on the surface of the 2D manifold of pose plus viewpoint (torus manifold). Mapping training poses on a torus worked nicely with HumanEva because only 3 subjects with similar aspect and walking style were considered. When the dataset is richer (more subjects, more walking speeds and styles, more variability in morphology), it is more difficult to map the poses on a torus and the discretization of the manifold produces non-homogeneous classes. Additionally, the discretization process assigns the same number of classes for quite different viewpoints. This means that, for instance, frontal and lateral viewpoints of a walking cycle were quantized with equal number of classes, despite the difference in appearance variability. However, since there is much less visual change over the gait cycle when viewed from front or back views than for lateral views, differences between classes do not reflect the same amount of change in visual information over each of these views. This over-quantization of visually quite similar views can make the class data unrealistically difficult to separate for certain viewpoints, and can introduce some confusion when a cascade must classify those examples. Therefore it is important to define homogeneous classes. Too many class clusters become too specific to a particular subject or pose, and do not generalize well enough. Too few clusters can merge poses that are too different and no feature can be found to represent all the images of the class.

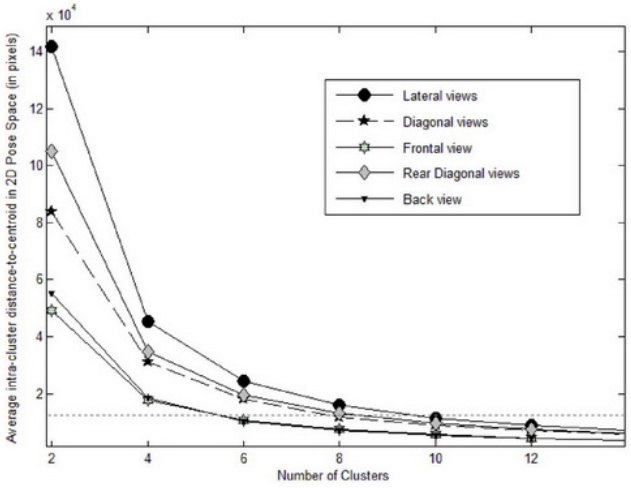


Fig. 16 Class Definition for MoBo dataset: For each training view, the 2D pose space is clustered and for each number of clusters, the intra-cluster distances to centroid are computed. By selecting a thresholding value of this average intra-cluster distance, the views are quantized into a number of classes that reflect the variability in appearance for that view. For the selected threshold (dashed line), frontal views of walking have a coarser quantization (6 classes) compared to the diagonal (8 classes) and lateral views (10 classes).

For each training view, the 2D pose space is clustered several times with K-means and for each number of clusters K, the intra-cluster distances to centroid are computed in the entire space. This distance indicates the tightness of the clusters. By selecting a thresholding value of the average intra-cluster distance (See Fig.16), the views are quantized into a number of classes that reflect the variability in appearance for that view. For the selected threshold (dashed line) frontal views of walking have a coarser quantization (6 classes) compared to the diagonal (8 classes) and lateral views (10 classes)[11]. We can see on Fig. 16 that if we choose the same number of clusters for all the views as we did in section 4.1, the resulting average intra-cluster distances are very different

[11] We select this threshold in order to obtain a minimum of 6 clusters in the frontal and back views. See [41] for details.

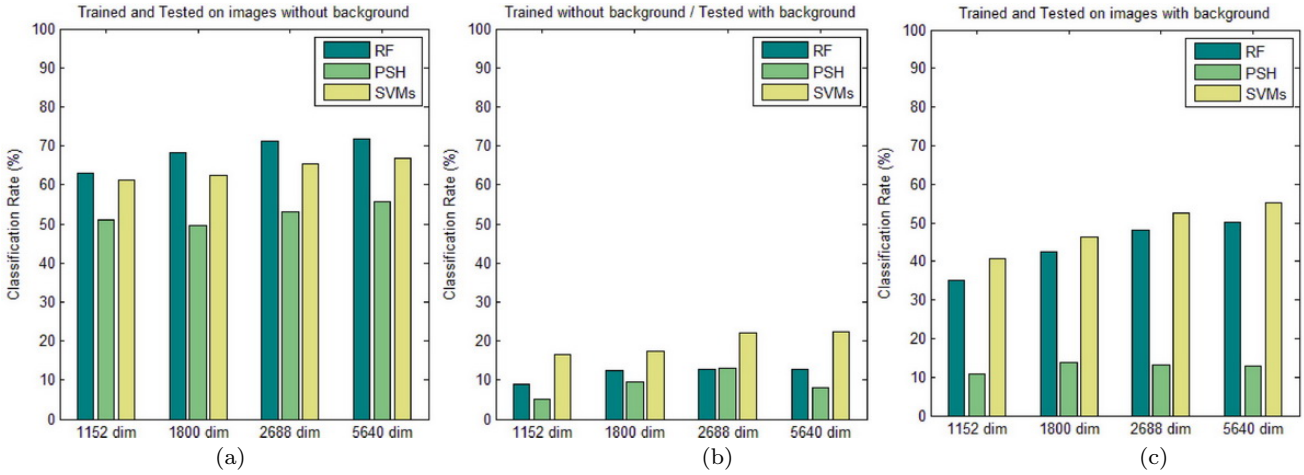


Fig. 18 Pose classification for PSH, Random Forest (1000 trees) and SVMs classifiers trained on a subset of the MoBo database containing 10 subjects and tested on the remaining 5 subjects. We compare the results using segmented images without background and images with a random background for 4 different grids of HOG descriptors.

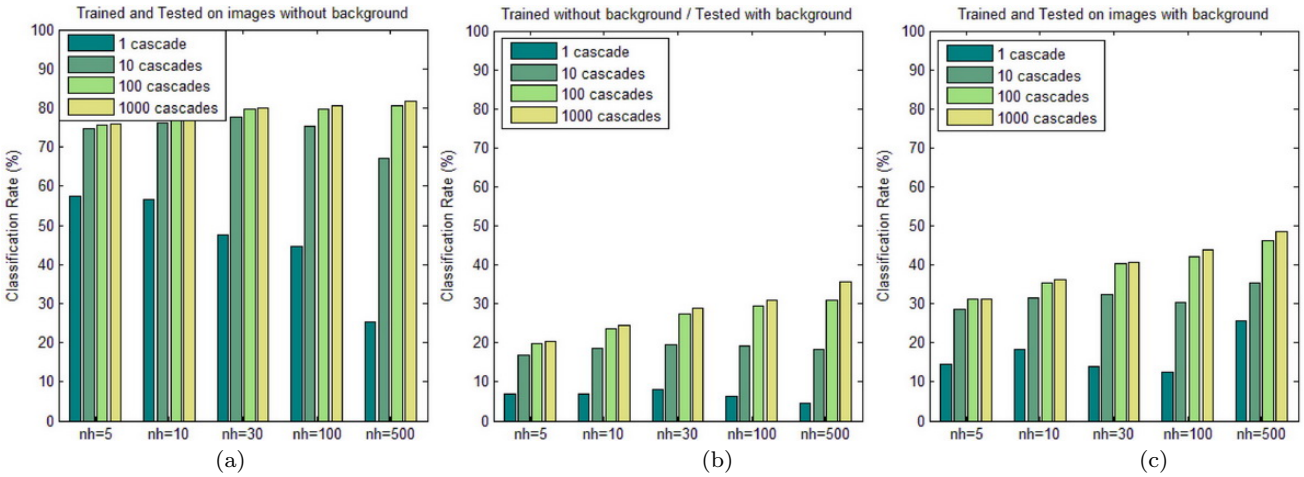


Fig. 19 Pose classification rates for 4 different ensembles (1, 10, 100 and 1000 cascades) trained on the subset of the MoBo database containing 10 subjects and tested on the remaining 5 subjects. We compare the results using segmented images without background and images with a random background for different number n_h of sampled HOG blocks.

from a view to another: for example, if we select 6 clusters per view, the resulting clusters from the frontal view are about two times tighter than the ones from lateral views.

In the proposed automatic class definition described here, views are quantized into a number of classes that reflect the variability in appearance for that view, and frontal views of walking would have a coarser quantization (i.e. less classes) compared to the lateral views. Using this method it is possible to create class definitions that better reflect the differences in variation over the gait cycle between different views. The resulting 64 classes are presented in Fig. 17.

4.2.2 Classification Results

A training subset is first built by randomly selecting 10 of the 15 available subjects from the database and a testing subset with the remaining 5 subjects, thus considering 30,000 images for training and 18,000 for testing. Benchmark experiments were performed on this dataset to get initial baseline results with three state-of-the-art multi-class (pose) classifiers:

- Random Forests [11], inherently good for multi-class problems, that share similarities with our approach.
- PSH [44] which is a fast and effective way of finding the neighboring poses of a query image. We will take the class of the nearest pose as a classification.
- A multi-SVMs classifier, similar in spirit to that of [37], which learns k one-vs-all linear SVMs to

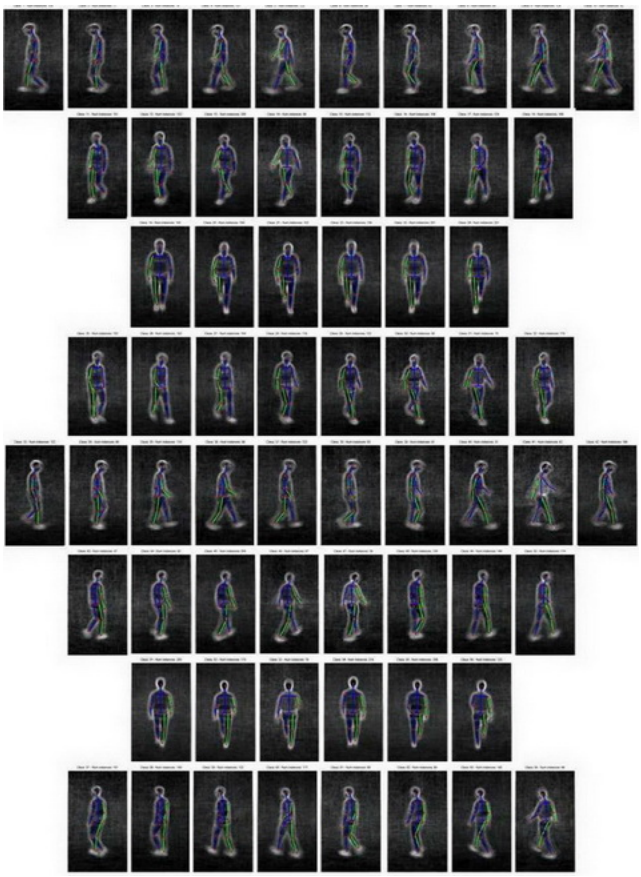


Fig. 17 Classes (64) obtained with the class definition method presented in Fig.16. The 8 rows correspond to the 8 training views available in the dataset. For each class, we show the average gradient map and average pose.

discriminate between k predefined pose clusters[12]. During classification the class is determined by choosing the cluster that has the highest probability $p(C_k|\mathbf{x})$ from the SVMs. We will refer to this work as multi-SVMs or SVMs in the rest of the paper.

We tuned the parameters of PSH and RF to get optimal performances on our 64 class MoBo dataset: we tailored PSH parameters to the number of images and used 200 18-bit hash functions and empirically validated that the optimum number m of randomly selected features for RF was near $\sqrt{D}$ (as indicated in [11]). We trained 64 one-vs-all linear SVMs. Two groups of classifiers are trained using segmented images without background and images with a random background respectively. We run the same test for 3 different grids of HOG descriptors. Table 2 gives the corresponding training time and Fig. 18 shows the performances[13] when classifying images with or without background. RF results are given for a 1000-tree forest since convergence is reached for that number as observed in Fig. 1.

Table 2 Classifiers training time on an Intel Core Quad Processor at 2.00GHz with 8Gb of RAM (see experiments in Fig. 18).

HOG Grid		4x4	5x5	7x12	3 together
Dimension		1152	1800	2688	5640
Backgrd	RF	5h30	6h45	13h00	21h45
	SVMs	2h00	4h00	8h00	17h30
	PSH	1h50	2h30	3h00	5h30
No Backgrd	RF	4h40	5h30	11h00	18h30
	SVMs	45min	52min	1h00	2h20
	PSH	1h20	2h20	3h08	5h30

The first observation we made is that all of the classifiers trained on segmented images perform poorly when classifying images with background (see Fig. 18b). This demonstrates the importance of considering a training dataset that includes a wider background appearance. The second observation is that using denser HOG feature grids improves pose classification accuracy but we are quickly facing memory issues that prevent us from working with denser grids. The third observation we can make is that PSH does not perform well in presence of cluttered background (Fig. 18c) while performing decently on segmented images (Fig. 18a). PSH tries to take a decision based on 1-bin splits. That could well be a reason as the histogram will be altered by the presence of background and affect that bin. In presence of cluttered background (Fig. 18c), SVM is the best classifier in terms of accuracy while Random Forest achieves the highest classification rate when working with segmented images (Fig. 18a).

After constructing the tree structure $\mathcal{S}$ (112 branches using $n_\theta = 1$), two lists of HOG block rejectors $\mathcal{B}$ are built using the two same training subsets of the MoBo database (with and without random background). The training took about 2 hours (testing 2500 HOG blocks at each branch) which is much faster than both SVMs and RF classifier training. Then, different ensembles are created varying the number n_c of cascades and the number n_h of HOG blocks that are considered for sampling. Corresponding classification rates on MoBo dataset are given in Fig. 19.

[12] As we do not perform pose-dependent feature selection and use linear SVMs instead of the ARD-Gaussian kernel SVMs, it is expected to perform a little lower than that of [37] but the training will be much faster. Since we use linear SVMs to train our cascade rejectors, we find it is a reasonable comparison.

[13] If neighboring classes are not considered as misclassification, some issues with images on the "boundaries between classes" are solved and the classification rates increase by about $15-20\%$

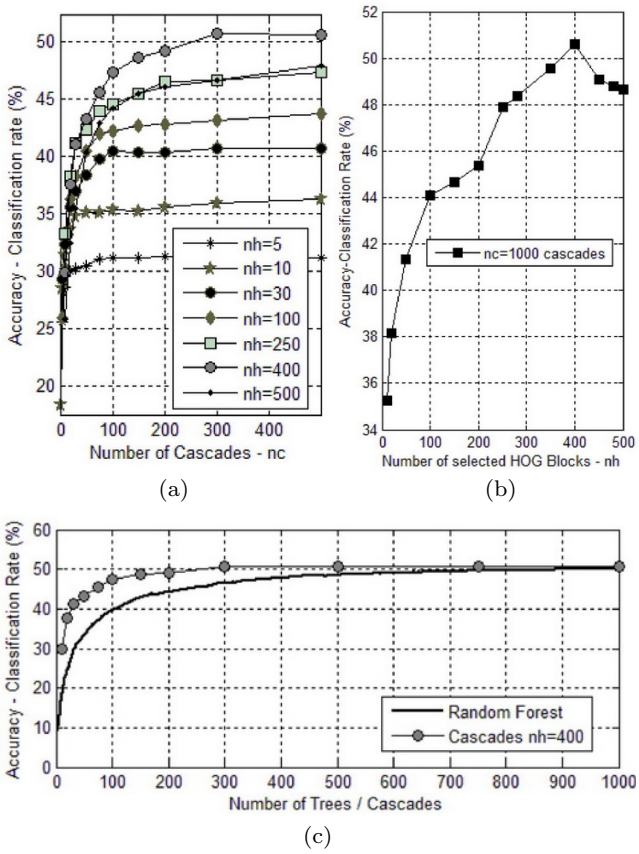


Fig. 20 Pose classification experiments on MoBo using randomized cascades: (**a**) we run the same experiment with our randomized cascades classifier (trained on the subset containing 10 subjects and tested the remaining 5 subjects from MoBo dataset) and compare the results obtained sampling from different numbers n_h of selected HOG blocks at each branch of the tree. (**b**) We show the results varying n_h with $nc = 1000$. (**c**) We compare the performance of the best Random Forest vs our randomized cascades classifier (with $n_h = 400$).

We can observe that the accuracy increases with both n_c and n_h for the 3 different tests and the cascades outperform the other classifiers when trained on images without background (Fig. 19 a and b). In particular, the cascades show better generalization performances when the testing data is significantly different from the training data (Fig. 19b). Performances seems to be lower than SVMs and RF when the classifiers are trained and tested on images with random background (see Fig. 19c vs Fig. 18c). Detailed results are reported in Fig. 20 for that concrete case. Fig. 20a shows that convergence is reached sooner when n_h is low and the accuracy improves when increasing n_h until $n_h = 400$ for $n_c = 1000$ (See Fig. 20b). The figure 20c compares the performances of a cascades classifier grown using the 400 best HOG blocks with the best RF classifier from Fig 1. The classification rate of the cascades classifier converges earlier (300 cascades) because the first cascades are already very informative compared to the first trees of the Random Forest: 30% of accuracy for the first cascade and 10% for the first tree of the RF. Actually, the accuracy of RF does not seem to converge to a specific value as every new tree is informative. Our approach performs better than RF for $n_c \leq 1000$ and is slightly less efficient than SVMs.

Fig. 21 Localization dataset: Clockwise from top left: HumanEva, INRIA, movie sequences, lab sequence and CamVid.

4.2.3 Localization Results

To construct our localization dataset, we took images from several different datasets to compare each of the algorithms in different environments and at different subject scales. These images were taken from HumanEva [48], CamVid [12], INRIA [14], some images of pedestrians collected from movie sequences, and some images captured from a web camera in a simple lab environment. See Fig. 21 for some selected images from this dataset. We then manually annotated each of these 120 images so we could determine localization accuracy.

For a fair comparison, we have implemented our cascades, the multi-SVMs and RF classifiers in the same C++ framework with an efficient on-demand feature extraction system. In the original HOG implementation [14], multi-scale searches required the input image to be re-scaled at every searched scale and scanning window resolution before extracting HOG features for that scale. We believe that different scales can be explored by keeping the source image size fixed and reusing the integral histograms. The classifier window is rescaled to the equivalent scale (1/s) and the Integral HOG features are resized with respect to the new window size. Since the integral histogram sampling consists of only 4 coordinate samples for each cell considered, independent of the size of the HOG feature, the features can be scaled with respect to the new window size at no additional computation time, thus allowing different scales

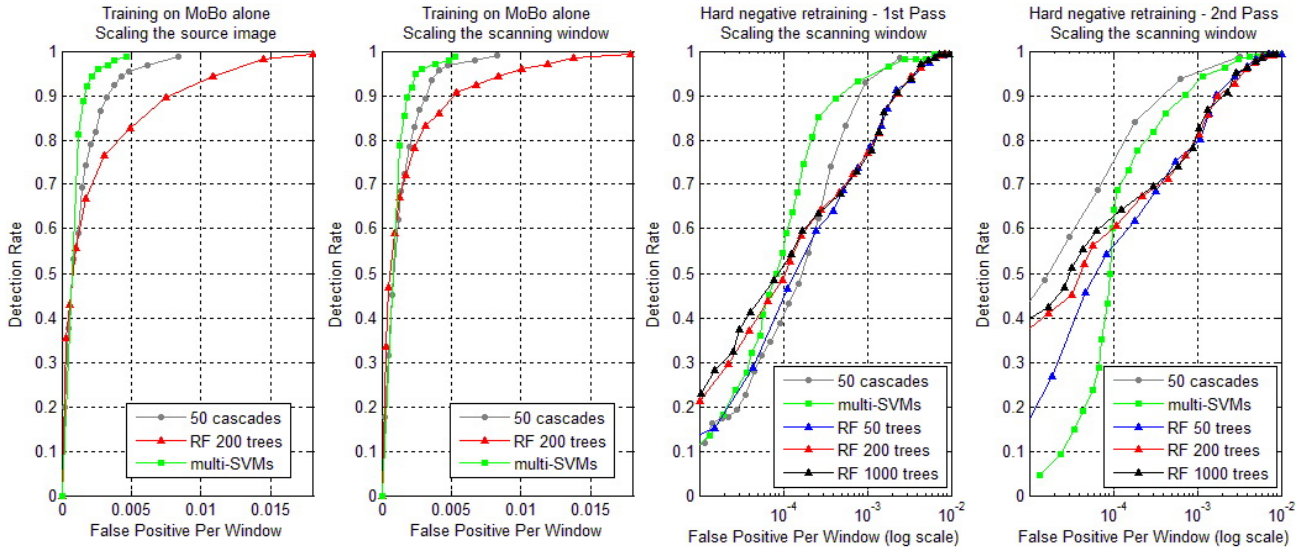


Fig. 22 Detection results on the combined localization dataset using different classifiers trained on Mobo images only (**a** and **b**), trained with a first pass of hard negatives from INRIA (**c**) and with a second pass (**d**). For the 1^{st} case, we present results for 2 different types of multi-scales scanning scheme: scaling the input image and keeping the size of the scanning windows fixed (**a**) and scaling the scanning window and keeping the size of the input image fixed (**b**), while the other 2 graphs (**c** and **d**) are obtained with the second scanning scheme. Note that a log scale is used for **c** and **d** to aid visualization.

to be explored at approximately the same computational cost (See Fig. 23).

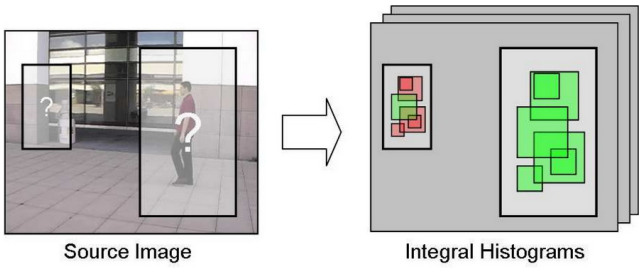


Fig. 23 HOG feature scaling: different scales are queried by rescaling the classifier window, and HOG feature boxes are rescaled in proportion to the new classifier window scale. The sampling coordinates for the HOG boxes can be rescaled at no extra cost in computation time.

SVMs, RF and cascade classifiers trained on MoBo images in Sect. 4.2.2 were then run on the localization dataset over 6 discrete scale setting ($[0.3, 0.45, 0.75, 1.0, 1.5, 2.0]$) with a 4 pixels stride, thus exploring and classifying a total of $10,599,042$ sub-images. We used a state-of-the-art laptop with an Intel Core @ 1.73GHz. A non-maximum suppression step is used to merge nearby detections to one final hypothesis. In the first experiment, we compare accuracy and processing time using the 2 different scanning schemes, i.e. scaling the input image or scaling the sliding window. As can be seen in Fig. 22a and Fig. 22b, the results are very similar with the 2 different scanning methods that do not seem to affect the detection rate much. The target architecture for the experiments is an AMD/Intel x86 processor, and although the results show that scaling the window is much faster (see Table 3) on a single thread implementation on a system with local caches, this comparison is architecture dependent[14].

Table 3 Detection times (after hard negative retraining) for the experiments reported in Fig. 22.

Hard negatives	scaling	SVMs	RF (trees)			Cascades
			50	200	1000	$n_c = 50$
Pose only	image	15h03	-	8h37	-	19h38
	window	11h45	-	2h03	-	18h36
1^{st} Pass	image	20h30	7h19	7h42	8h12	12h48
	window	10h55	1h11	1h33	3h07	5h07
2^{nd} Pass	window	11h08	1h12	1h32	3h56	5h09

Each of the classifiers were re-trained on human subjects from the aligned Mobo dataset, and combined with hard background examples from the INRIA dataset [14]. To create the hard examples dataset, the classifiers trained on pose images only from Sect. 4.2.2 were run on

[14] As noted by Sugano et al [51] on a parallel SIMD (Single Instruction Multiple Data) implementations scaling the image can be more suitable for parallel processing and actually negate the additional impact of resizing the image to different scales.

negative examples from the INRIA dataset [14] (with a 4 pixels stride and the same 6 scales), and strong positive classifications were incorporated as hard examples. Hard background examples were included in the negative set T_b^- (see Fig 5) during the training of each rejectors of our cascades. They were added to SVMs and RF classifiers as a background class so that there are a total of 65 classes; 1 for background and 64 for pose. We repeat the process of hard negatives retraining several times to refine the classifiers. The detection rates for the three types of classifiers after hard negatives retraining are given in Fig. 22c and Fig. 22d while the corresponding classification times are shown in table 3. We can observe that the 1^{st} pass of hard negatives retraining helps to improve the accuracy as the FPPW rate (False Positive Per Window) is reduced by a factor of 3. Adding more trees to the RF classifier does not improve the performances (see Fig. 22c and Fig. 22d). Both SVMs and RF classifiers reach their best performances after one hard negative retraining while our cascade classifier outperforms them after the 2^{nd} pass (see Table 4). The reference point of 1×10^{-4} FPPW is arbitrary but is a reasonable comparison point given that it has been used in other detection works such as [14]. At this rate we achieve a better detection rate (73.5%) than both multi-SVMs and RF classifiers (62.79% and 62.53% respectively) at the same FPPW rate, but if a higher rate of 5×10^{-4} is acceptable, then we achieve over 90% accuracy as illustrated in Fig. 22d. However the Random Forests are still faster than this configuration of cascade classifier (see Table 3).

Table 4 Detection rates at 1×10^{-4} FPPW and 1×10^{-3} FPPW for the experiments reported in Fig. 22.

Hard negatives	FPPW	SVMs	RF (trees)			Cascades
			50	200	1000	$n_c = 50$
Pose only	1.10^{-4}	6.21	-	**12.11**	-	7.81
	1.10^{-3}	**62.04**	-	59.86	-	54.46
1^{st} Pass	1.10^{-4}	**54.89**	43.29	49.03	51.93	40.37
	1.10^{-3}	**94.12**	76.82	76.64	76.04	92.97
2^{nd} Pass	1.10^{-4}	62.79	56.13	60.31	62.53	**73.5**
	1.10^{-3}	92.35	78.21	79.67	80.80	**94.53**

The previous experiments have been carried out using an ensemble made of 50 cascades built from the best $n_h = 30$ HOG rejectors at each branch. In Fig. 24, we show how the number of cascades n_c and the number of HOG blocks n_h we sample from affect the detection and FFPW rates, while the corresponding classification times are shown in Fig. 26. If we consider a classifier with 10 or less cascades we can obtain a better and faster classification than with the best RF (less than an hour). Adding more cascades allows the hierarchical cascade classifier to reach higher detection rates but at a higher cost in terms of FPPW, while sampling from a smaller subset of HOGs (i.e. using a smaller n_h) makes the classifier more strict. Constraining the features that our cascades can randomly draw from impairs generalization performances and considerably affects pose classification rates as demonstrated in Fig. 19 and Fig. 20. A trade-off thus has to be made between the computational speed, the efficiency in detection and the pose classification accuracy. One approach to this is to localize using an ensemble built from a small pool of HOG blocks and then refine the pose classification of a few selected locations with additional cascades grown by sampling from a larger pool of rejectors.

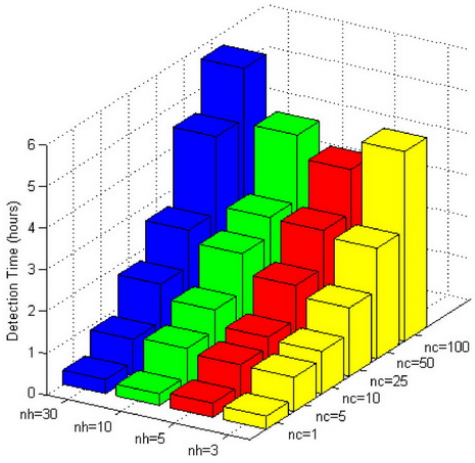
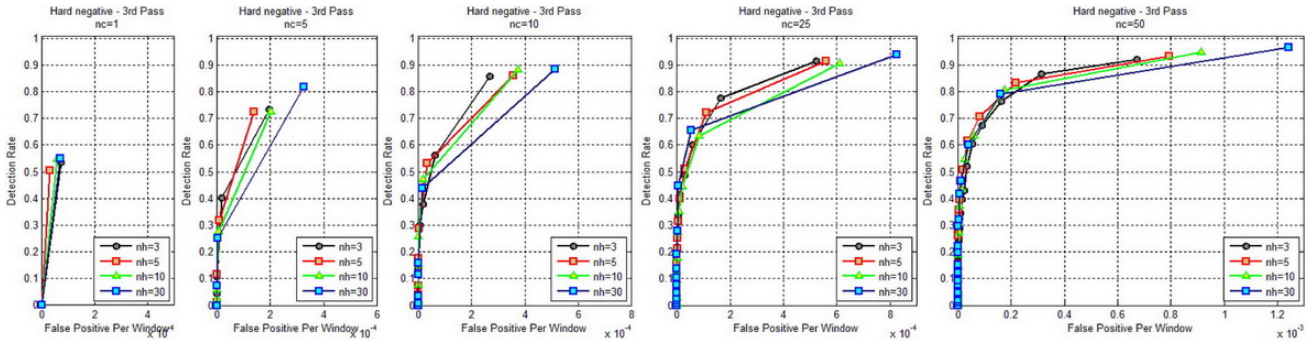


Fig. 26 Detection time when scanning the entire localization dataset for different configurations of the cascades classifiers after a 3^{rd} pass of hard negative retraining (experiments reported in Fig. 24).

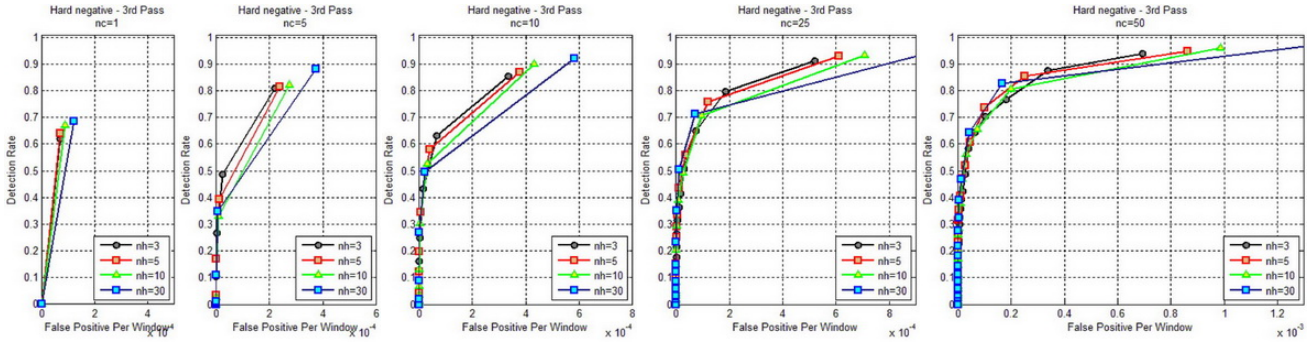
4.2.4 On-line Randomization and Filtering

In addition to the advantages stated in previous sections, our classifier has 2 very interesting properties that have not been evaluated yet. First, the on-line randomization of Φ_k which allows the generation of a different classifier $r_k(I_N, \Phi_k, \mathcal{S}, \mathcal{B})$ at each location considered (i.e. for each sub-image to be classified) at no extra-cost. The qualitative effects of the on-line randomization on the saliency map can be appreciated in Fig. 27 where we show the result of a dense scan (1-pixel stride) with several different cascades classifier. A 1-cascade classifier (Fig. 27a) produces few responses which are grouped and more likely to be missed at a higher search stride while using a different cascade at each location (Fig. 27b) produces a cloud of responses that favors a good detection at a higher search



(a) (b) (c) (d) (e)

Fig. 24 Detection results for different configurations of the cascades classifiers with a 3^{rd} pass of hard negative retraining: we present the results with respectively 1 (**a**), 5 (**b**), 10 (**c**), 25 (**d**) and 50 cascades (**e**). For each case we vary the number n_h of HOG blocks we sample from at each branch of the tree. The corresponding classification times are shown in in Fig. 26.



(a) (b) (c) (d) (e)

Fig. 25 Detection results for different configurations of the cascades classifiers with a 3^{rd} pass of hard negative retraining and on-line randomization: we present the results with respectively 1 (**a**), 5 (**b**), 10 (**c**), 25 (**d**) and 50 cascades (**e**). For each case we vary the number n_h of HOG blocks we sample from at each branch of the tree.

stride. With a 5-cascade classifier, the benefit of the randomization is less immediately obvious (Fig. 27c and Fig. 27d). Though this property is less useful with a 100-cascade classifier (Fig. 27e and Fig. 27f). To evaluate quantitatively the benefit of this property, we run the same test on the localization dataset, varying the number n_c of cascades and number n_h of HOGs, but this time randomizing the vectors Φ_k at each location. The results are presented in Fig. 25. As expected, if we compare with Fig. 24, we can see that the on-line randomization has a great influence for classifiers with few cascades: the maximum detection rate increases by about 10% for a 5-cascade classifier while it increases by over 15% for a single cascade classifier.

Single hierarchical cascades have good approximate localization performance (see Fig. 24), but the disadvantage is that they may misclassify locations that would have otherwise been correctly classified using more cascades. This leads to the second property of our algorithm: we can adapt the number of cascades in the classifier on-line, and thus, each sub-image can be classified using a different number of cascades.

Locations that yield a detection using a few cascades can be classified with more trees until the detection result converges. We will leave for future work the study and optimization of the convergence criteria for the *cascade thresholding* approach as the detection of this convergence requires a minimum number of cascades to be applied. However, since the goal is to create a classifier that can make a very fast detection using as few cascades as possible, we instead exploit the *adaptive cascade filtering* property for efficient localization: we will use the first n_f cascades of the classifier as a filter and classify with the rest of the cascades only if a positive vote has been made. A qualitative analysis can be made from Fig. 27: in the presented example, we can see that filtering with the first (g) or the first 5 cascades (h) of a 100-cascade classifier leads to a decent distribution, almost as good as the ones obtained without any filtering (e and f) but it is much faster (between 5 and 10 times faster for that example).

We have evaluated the combination of these 2 properties by running a series of experiments to select the best cascade classifier: we evaluated detection rates and

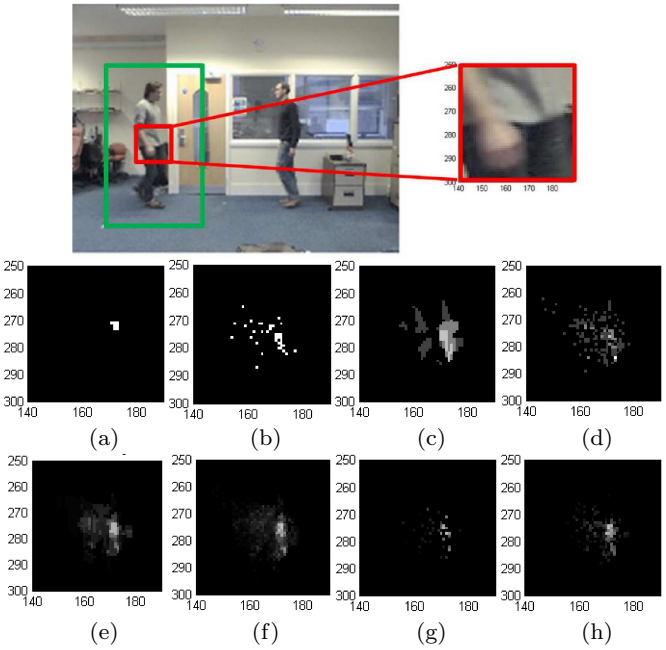


Fig. 27 On-line randomization and filtering: the picture shown in the upper part with a ground truth box (green) is scanned at a 1-pixel stride. We present the saliency maps for a region of interest around the ground truth location (red box) obtained with different cascade classifiers: 1-cascade (**a**), 1-cascade randomized(**b**), 5-cascade (**c**), 5-cascade randomized (**d**), 100-cascade (**e**), 100-cascade randomized (**f**), 100-cascade randomized and filtered with 1st cascade (**g**) and 100-cascade randomized and filtered with the first 5 cascades (**h**). Processing times of the entire image are respectively: 13.27s (**a**), 13.25s(**b**), 31.66s(**c**), 28.53s(**d**), 114.52s(**e**), 103s(**f**), 9.34s(**g**) and 23s(**h**).

processing times for different search strides (1, 2, 4, 8 and 16 pixels) using a 100-cascade classifier filtered with the first, the first 5 and the first 10 cascades. Quantitative results are reported in Fig. 28a and Fig. 28b. As expected, using a denser search (i.e. using a smaller stride) considerably improves the detection rate but it also increases the processing time up to 3 or more hours. The same observation can be made concerning the number of cascades used for filtering: more cascades also increase the detection rates for any search strides but at a higher computation cost. Classifying every 4 pixels and filtering with the first 5 cascades offers a reasonable compromise between accuracy (maximum detection rate of 90%) and computational cost (1h01 i.e 35.5 seconds per image). Our selected cascade classifier performs slightly better than the best multi-SVMs classifier but is 10 times faster. It is also slightly faster than the best RF classifier (1h11) but considerably outperforms it in classification (see Fig. 29a).

A detailed analysis of the performances of our classifier (see Fig. 29b) shows that the best detection rates are obtained for the HumanEVA and the Lab subsets:

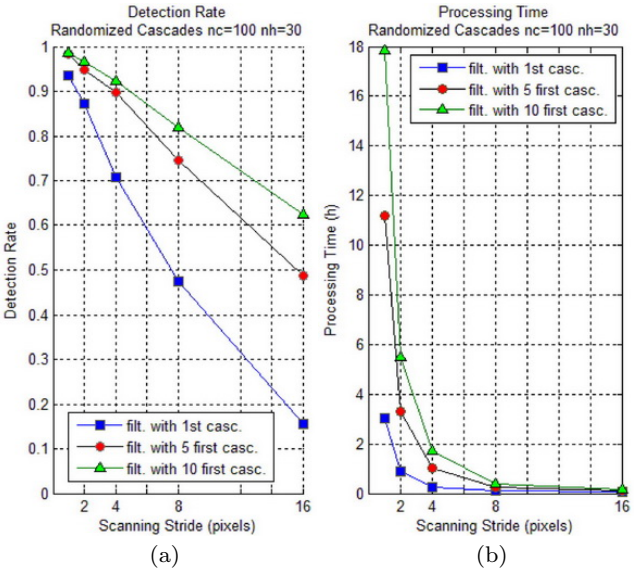


Fig. 28 Selection of the best cascade classifier: maximum detection rate varying search stride and number of cascades used for filtering (**a**), corresponding processing times are given in (**b**). Classifying every 4 pixels and filtering with the first 5 cascades offers a reasonable compromise between accuracy (maximum detection rate of 90%) and computational cost (1h01).

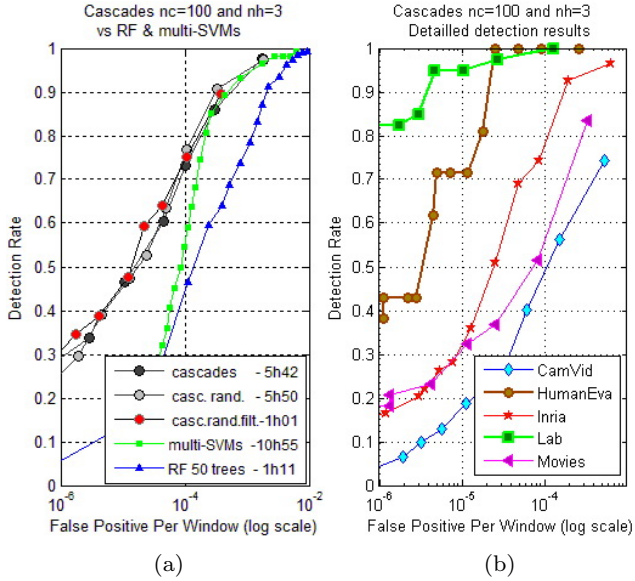


Fig. 29 Detection rate of the selected cascade classifier (100 cascades filtered with the first 5 cascades) compared to other classifiers for a 4-pixels search stride (**a**) and corresponding detailed detection rates over the 5 different types of images in our localization dataset (**b**).

for a FPPW rate of 1×10^{-4} or higher there is no misdetection. This makes perfect sense since our MoBo training data has been captured in a similar lab environment and the background is less complex for this 2 subsets. The drop in the classifier performances with

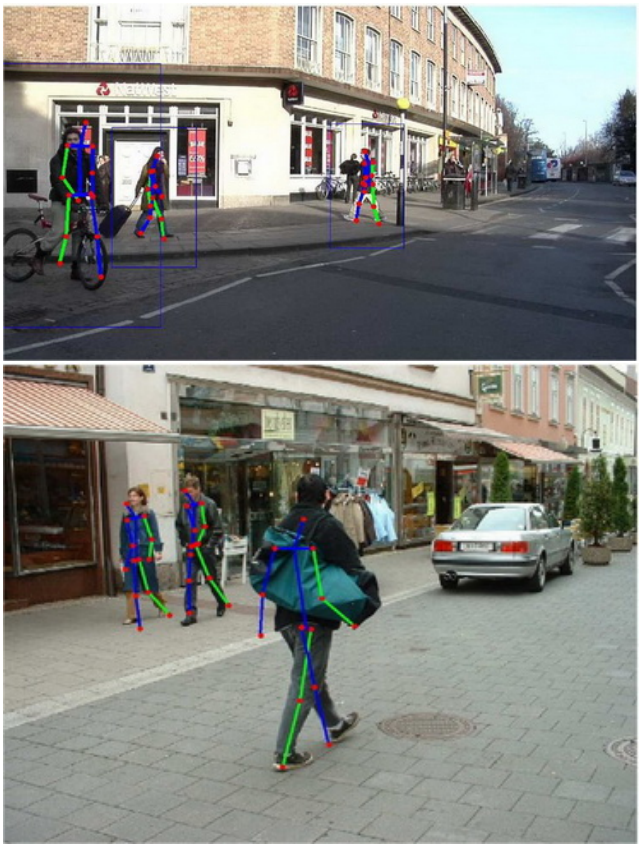


Fig. 30 Detection results: we present 2 examples of multiple multi-scale detections in the same image.

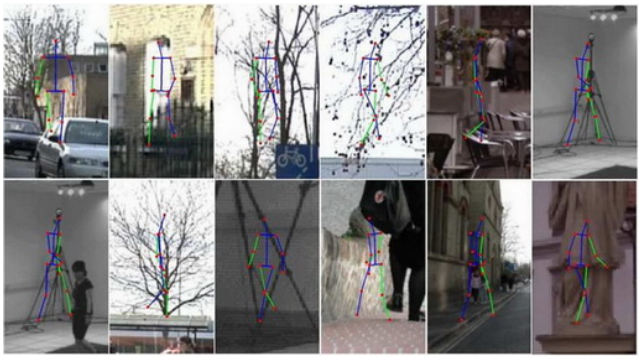


Fig. 31 Cascades detection errors - False Positives: we can observe that the presence of many edges in textured areas and trees make these regions difficult to classify as background.

Fig. 32 Cascades detection errors - False Negatives: we can observe that the clothing, occlusions and low resolution make the detection harder for some subjects in Camvid and Movie subsets.

INRIA images can be explained by their richer backgrounds, and are more difficult to correctly classify. The lower performances on the remaining 2 subsets, the Camvid and Movies, can be explained by the fact that the images also present richer backgrounds and have been captured with a moving camera. More importantly, the humans are much more difficult to detect than in the other subsets. Indeed, the Camvid and Movies subsets present images with street views where people wear coats, dresses, hats and all sorts of clothing completely absent in our lab-type training dataset. Some severe occlusions also occur in these uncontrolled environments and the resolution is often not fine enough to give accurate results. All these observations are visually confirmed with the examples of pose detection errors: some False Positives are presented in Fig. 31 while some False Negatives are shown in Fig. 32.

Finally, we present some examples of good detections for the 5 different subsets in Fig. 33. For each detection, the represented pose corresponds to the "winning class" i.e. the highest peak in the pose distribution. As we can see, a correct detection does not always returns a correct pose classification. In some cases, the pose is not well-recognized because it does not belong to our training dataset and does not correspond to any of our classes (e.g. walking with hands in pockets). In other cases, we believe that a proper analysis of the pose distribution would probably improves the accuracy in terms of pose but we will leave it for future work.

5 Conclusions and Discussions

In this work, an efficient method to jointly localize and recognize the pose of humans is proposed, using an exemplar based approach and fast search technique: the randomized hierarchical cascades classifier. Unlike most previous works, this pose detection algorithm is applicable to more challenging scenarios involving extensive viewpoint changes and moving camera, without any prior assumption on an available segmented silhouette or normalized input bounding box.

Cascade approaches are efficient at quickly rejecting negative examples, and we exploit this property by learning multi-class hierarchical cascades. By randomly sampling the feature, each cascade uses different sets of features to vote, it adds some robustness to noise that helps to prevent over fitting. Moreover, each cascade can vote for one or more class so the ensemble of

Fig. 33 Cascades pose detections - True Positives: Each row corresponds to one of the testing sequences (*from top to down*): Movies, INRIA, Camvid, Lab and HumanEVA. For each sequence, we show 5 examples of correct pose classification (*left*) and 3 examples of wrong pose classification (*right*). For each presented frame, we present the normalized 96×160 image corresponding to the highest values of the saliency map obtained when applying the pose detector. The pose corresponding to the "winning class" i.e. the highest peak in the distribution is represented on top of the cropped image.

random cascade classifiers outputs a distribution over possible poses that can be useful when combined with tracking algorithms for resolving ambiguities in pose.

Other algorithms require that the full feature space be available from an image during training. This can lead to very high dimensional feature vectors being extracted from an image for large configurations of features and potentially leads to memory problems for training sets with a large number of examples. Our algorithm selects the features it considers to be the most informative during training, and can build a smaller more useful feature space by sampling a small set of features from a much larger configuration of feature descriptors. This approach is computationally efficient as cascades learning takes around 2 hours while SVM and RF approaches took respectively 17h30 and 21h45 with the same training set. We have validated our approach with a numerical evaluation for 3 different levels of analysis: human detection/localization, pose classification and pose estimation (with body joints localization).

If the search space (locations in the image and scales) can be reduced, e.g. using a tracking algorithm or limiting the distance to the camera in a Human-Computer Interface, then our method can reach real-time performances. A computationally efficient implementation of the HOG descriptor, for example, using a GPU implementation could speed up the detection even further.

Our method finds a distribution over exemplars, an intriguing direction for future work would be to combine this with a kernel regression to see if this could produce a method that is both computationally efficient and more accurate in terms of estimated pose.

Finally, this work opens several other interesting lines for future work: for instance, we could try to efficiently combine different types of features (color, depth, etc) inside our cascade classifier and extend the algorithm to wider range of motions and actions, or apply the algorithm to general machine learning problems.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable comments and constructive suggestions which helped to improve the quality of the paper. They are also grateful to Dr Srikumar Ramalingam for his contribution on the initial version of this paper.

Bibliography

1. Agarwal A, Triggs B (2006) Recovering 3d human pose from monocular images. IEEE Trans Pattern Anal Mach Intell 28(1):44–58
2. Andriluka M, Roth S, Schiele B (2009) Pictorial structures revisited: People detection and articulated pose estimation. In: CVPR
3. Andriluka M, Roth S, Schiele B (2010) Monocular 3d pose estimation and tracking by detection. In: CVPR, pp 623–630
4. Bergtholdt M, Kappes JH, Schmidt S, Schnörr C (2010) A study of parts-based object class detection using complete graphs. International Journal of Computer Vision 87(1-2):93–117
5. Bissacco A, Yang MH, Soatto S (2006) Detecting humans via their pose. In: NIPS, pp 169–176
6. Bissacco A, Yang MH, Soatto S (2007) Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. In: CVPR
7. Bookstein F (1991) Morphometric Tools for Landmark Data: Geometry and Biology. Cambridge Univ. Press
8. Bosch A, Zisserman A, Munoz X (2007) Image classification using random forests and ferns. In: ICCV
9. Bourdev L, Malik J (2009) Poselets: Body part detectors trained using 3d human pose annotations. In: ICCV
10. Breiman L (1996) Bagging predictors. Machine Learning 24:123–140
11. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
12. Brostow GJ, Shotton J, Fauqueur J, Cipolla R (2008) Segmentation and recognition using structure from motion point clouds. In: ECCV, pp 44–57
13. Collins R, Liu Y (2003) On-line selection of discriminative tracking features. ICCV
14. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: CVPR, vol 2, pp 886–893
15. Datar M, Immorlica N, Indyk P, Mirrokni V (2004) Locality-sensitive hashing scheme based on p-stable distributions. In: Proc. of the 20th annual symposium on Computational geometry, pp 253–262
16. Deselaers T, Criminisi A, Winn JM, Agarwal A (2007) Incorporating on-demand stereo for real time recognition. In: CVPR
17. Dimitrijevic M, Lepetit V, Fua P (2006) Human body pose detection using bayesian spatio-temporal templates. Comput Vis Image Underst 104(2):127–139
18. Elgammal AM, Lee CS (2009) Tracking people on a torus. IEEE Trans on PAMI 31(3):520–538
19. Felzenszwalb PF, Huttenlocher DP (2005) Pictorial structures for object recognition. International Journal of Computer Vision 61(1):55–79
20. Felzenszwalb PF, Girshick RB, McAllester DA (2010) Cascade object detection with deformable part models. In: CVPR, pp 2241–2248
21. Ferrari V, Marn-Jimnez MJ, Zisserman A (2008) Progressive search space reduction for human pose estimation. In: CVPR
22. Fossati A, Dimitrijevic M, Lepetit V, Fua P (2007) Bridging the gap between detection and tracking for 3d monocular video-based motion capture. In: CVPR
23. Gall J, Rosenhahn B, Brox T, Seidel HP (2010) Optimization and filtering for human motion capture. Int Journal of Computer Vision 87(1-2):75–92
24. Gavrila DM (2007) A bayesian, exemplar-based approach to hierarchical shape matching. IEEE Trans Pattern Anal Mach Intell 29(8):1408–1421
25. Gross R, Shi J (2001) The cmu motion of body (mobo) database. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA
26. Jaeggli T, Koller-Meier E, Gool LJV (2009) Learning generative models for multi-activity body pose estimation. International Journal of Computer Vision 83(2):121–134
27. Kanade T, li Tian Y, Cohn JF (2000) Comprehensive database for facial expression analysis. In: FG,

pp 46–53
28. Laptev I (2009) Improving object detection with boosted histograms. Image Vision Comput 27(5):535–544
29. Lee CS, Elgammal AM (2010) Coupled visual and kinematic manifold models for tracking. International Journal of Computer Vision 87(1-2):118–139
30. Lepetit V, Fua P (2006) Keypoint recognition using randomized trees. IEEE Trans Pattern Anal Mach Intell 28(9):1465–1479
31. Lin Z, Davis LS (2010) Shape-based human detection and segmentation via hierarchical part-template matching. IEEE Trans Pattern Anal Mach Intell 32(4):604–618
32. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2):91–110
33. Ma Y, Ding X (2005) Real-time multi-view face detection and pose estimation based on cost-sensitive adaboost. Tsinghua Science & Technology 10(2):152–157
34. Moosmann F, Nowak E, Jurie F (2008) Randomized clustering forests for image classification. IEEE Trans on PAMI 30(9):1632–1646
35. Mori G, Malik J (2006) Recovering 3d human body configurations using shape contexts. IEEE Trans on Pattern Anal and Mach Intell 28(7):1052–1062
36. Navaratnam R, Thayananthan A, Torr P, Cipolla R (2005) Hierarchical part-based human body pose estimation. In: BMVC
37. Okada R, Soatto S (2008) Relevant feature selection for human pose estimation and localization in cluttered images. In: ECCV, pp 434–445
38. Okada R, Stenger B (2008) A single camera motion capture system for human-computer interaction. IEICE TRANSACTIONS on Information and Systems 91(7):1855–1862
39. Orrite C, Gañán A, Rogez G (2009) Hog-based decision tree for facial expression classification. In: IbPRIA, pp 176–183
40. Roberts T, McKenna S, Ricketts I (2004) Human pose estimation using learnt probabilistic region similarities and partial configurations. ECCV pp 291–303
41. Rogez G, Orrite C, Martínez J (2008) A spatio-temporal 2d-models framework for human pose recovery in monocular sequences. Pattern Recognition
42. Rogez G, Rihan J, Ramalingam S, Orrite C, Torr PH (2008) Randomized trees for human pose detection. CVPR pp 1–8
43. Sabzmeydani P, Mori G (2007) Detecting pedestrians by learning shapelet features. In: CVPR07
44. Shakhnarovich G, Viola P, Darrell R (2003) Fast pose estimation with parameter-sensitive hashing. In: ICCV
45. Shotton J, Johnson M, Cipolla R, Center T, Kawasaki J (2008) Semantic texton forests for image categorization and segmentation. In: CVPR
46. Shotton J, Fitzgibbon A, Cook M, Sharp T, Finocchio M, Moore R, Kipman A, Blake A (2011) Real-time human pose recognition in parts from single depth images. In: CVPR
47. Sigal L, Black MJ (2010) Guest editorial: State of the art in image- and video-based human pose and motion estimation. International Journal of Computer Vision 87(1-2):1–3
48. Sigal L, Balan AO, Black MJ (2010) Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. International Journal of Computer Vision 87(1-2):4–27
49. Sminchisescu C, Kanaujia A, Metaxas DN (2006) Learning joint top-down and bottom-up processes for 3d visual inference. In: CVPR (2), pp 1743–1752
50. Stenger B (2004) Model-based hand tracking using a hierarchical bayesian filter. PhD thesis, Department of Engineering, University of Cambridge
51. Sugano H, Miyamoto R (2007) A real-time object recognition system on cell broadband engine. In: Proc. of the 2nd Pacific Rim conference on Advances in image and video technology, pp 932–943
52. Thayananthan A, Navaratnam R, Stenger B, Torr PHS, Cipolla R (2006) Multivariate relevance vector machines for tracking. In: ECCV (3), pp 124–138
53. Toyama K, Blake A (2002) Probabilistic tracking with exemplars in a metric space. Int J Comput Vision 48(1):9–19
54. Villamizar M, Sanfeliu A, Andrade-Cetto J (2009) Local boosted features for pedestrian detection. IbPRIA pp 128–135
55. Viola P, Jones M (2002) Robust real-time object detection. Int Journal of Computer Vision
56. Viola P, Jones MJ, Snow D (2005) Detecting pedestrians using patterns of motion and appearance. Int J Comput Vision 63(2):153–161
57. Wu B, Nevatia R (2005) Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In: ICCV, pp 90–97
58. Zehnder P, Koller-Meier E, Van Gool L (2005) A hierarchical system for recognition, tracking and pose estimation. MLMI pp 329–340
59. Zhang J, Zhou S, McMillan L, Comaniciu D (2007) Joint real-time object detection and pose esti-

mation using probabilistic boosting network. In: CVPR, pp 1–8
60. Zhang Z, Zhu L, Li S, Zhang H (2002) Real-time multi-view face detection. In: Proc. Intl Conf. Automatic Face and Gesture Recognition, pp 149–154
61. Zhu Q, Avidan S, Yeh MC, Cheng KT (2006) Fast human detection using a cascade of histograms of oriented gradients. In: CVPR, pp 1491–1498