Ordinal Random Forests for Object Detection

Samuel Schulter, Peter M. Roth, Horst Bischof

Institute for Computer Graphics and Vision
Graz University of Technology, Austria
{schulter,pmroth,bischof}@icg.tugraz.at

Abstract. In this paper, we present a novel formulation of Random Forests, which introduces order statistics into the splitting functions of nodes. Order statistics, in general, neglect the absolute values of single feature dimensions and just consider the ordering of different feature dimensions. Recent works showed that such statistics have more discriminative power than just observing single feature dimensions. However, they were just used as a preprocessing step, transforming data into a higher dimensional feature space, or were limited to just consider two feature dimensions. In contrast, we integrate order statistics into the Random Forest framework, and thus avoid explicit mapping onto higher dimensional spaces. In this way, we can also exploit more than two feature dimensions, resulting in increased discriminative power. Moreover, we show that this idea can easily be extended for the popular Hough Forest framework. The experimental results demonstrate that using splitting functions building on order statistics can improve both, the performance for classification tasks (using Random Forests) and for object detection (using Hough Forests).

1 Introduction

Random Forests (RF), as introduced by Amit and Geman [1] and further developed by Breiman [6], became quite popular in recent years due to its simplicity and low computational costs. They find many applications in various sub-domains of computer vision, like object detection [10], tracking [11, 12], and semantic image labeling [18]. Random Forests are ensembles of randomized decision trees, which separate the data via learned splitting functions and make predictions in the leaf nodes, *e.g.*, with estimated class-conditional probabilities for classification tasks. Random Forests are easy to implement and, due to the hierarchical tree structure, are very fast during both training and testing, making them a good choice for many computer vision applications.

In many of the aforementioned applications, Random Forests are adopted to suit the specific needs by changing the splitting functions. For instance, for fine-grained image categorization [21], max-margin classifiers are used to split the

The work was supported by the FFG projects Human Factors Technologies and Services (2371236) and Mobile Traffic Checker (8258408).

data in each node. Another example are Oblique Random Forests [14], which integrate random projections into the splitting functions. For object detection and semantic image labeling, both Hough Forests [10] and Texton Forests [18] rely on splitting functions based on the difference of two randomly selected pixels on small image patches. The splitting functions used in the latter two approaches consider partial ordering statistics, *i.e.*, the ordering of different feature dimensions according to their feature values. However, both approaches rely on the differences between the pixel values and not directly on the ordering and are limited to two pixels only. Furthermore, they neither set their focus on these splitting functions nor explore their functionalities in more detail.

A totally different approach, which explicitly explores partial order statistics, is that of Yagnik *et al.* [19]. They propose a non-linear transformation that randomly selects a subset of the feature dimensions and encodes the index of the maximum value of this subset. This is repeated several times to build a feature vector. They use this new data representation for several vision tasks (*e.g.*, image classification) and show the discriminative power of this non-linear transformation. However, for good performance, they need to generate a huge number of completely random non-linear codes (*i.e.*, yielding a quite high dimensional feature vector). As not all codes are discriminative, this increases the computational costs.

In this work, we show how such non-linear and discriminative transformations can be seamlessly integrated into the splitting functions of Random Forests. We can thus avoid the explicit calculation of such high dimensional data as in [19], because each node in the RF only selects the most discriminative codes. This can be seen as a feature selection process that results in lower computational costs. Nevertheless, we still maintain the properties of the non-linear transformation in each tree, thus making the Random Forest more discriminative. Similar to [19], the novel splitting functions neglect absolute values of single features but rather observe their partial ordering. Such statistics are particularly interesting in computer vision as they are independent from the illumination. That is, our splitting functions for RFs depend on several feature dimensions simultaneously and exploit their ordering statistics to split the data (see Sec. 3), rather than considering only a single dimension as in standard RFs. In the sequel, we denote this RF formulation as *Ordinal Random Forest (ORF)*.

We also show how these splitting functions can be integrated into the Hough Forests [10] (see Sec. 4), which is a popular and widely applicable framework used for object detection [10], tracking [11, 12], and action recognition [20]. This extension of Hough Forests, in the following denoted as *Ordinal Hough Forests (OHF)*, makes them more discriminative as we show in our experiments.

In Sec. 5, we evaluate our novel splitting functions on two different tasks: (i) image categorization and (ii) object detection. We first apply the *ORF* framework for image categorization on the popular *Caltech101* and *USPS* data sets. Then, in the second experiment, we show object detection results with the *OHF* framework on widely used data sets (*i.e.*, TUD pedestrian, TUD crossing, and TUD campus). Both experiments show relative improvements of *ORF* and *OHF*

compared to standard Random Forest and Hough Forest, indicating the benefits of considering order statistics of several feature dimensions, rather than values of a single dimension.

2 Random Forests

Given labeled data $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^D$ is a D-dimensional feature vector, $y_i \in 1, \ldots, C$ is the corresponding class label, and N is the number of training samples, the goal of a machine learning algorithm is to learn a mapping from a given test sample $\mathbf{x}$ to the correct class label y. For Random Forests [6] (RF), this is realized by building an ensemble of T randomized decision trees $\mathcal{T}_t$, $t = 1, \ldots, T$. The literal purpose of RFs was to solve a supervised classification problem, although this learning approach was already extended to other tasks like regression, semi-supervised learning or density estimation [7]. However, in this work we focus on the classification problem.

During training, the algorithm tries to split the given training data $\{\mathbf{x}_i, y_i\}_{i=1}^N$ in each node, such that samples from different classes are separated. This recursive algorithm continues to split the data until either the maximum depth of the tree is reached, the subset of the data in a node is pure, or the number of samples is below a threshold. If any of these conditions is met, a leaf node is created and the class probability $p(y|\mathbf{x})$ is estimated by simply counting the number of samples that fell into this leaf node for each class y, normalized such that $\sum_{y=1}^{C} p(y|\mathbf{x}) = 1$.

In order to split the data in a single splitting node, a pre-defined number Γ of splitting functions

$$\xi_{d,\tau}^j(\mathbf{x}) = \begin{cases} 0, & \text{if } x_d < \tau \\ 1, & \text{otherwise} \end{cases} , \tag{1}$$

is drawn randomly, where $j = 1, \ldots, \Gamma$, x_d denotes the d-th dimension (randomly drawn) in $\mathbf{x}$, and τ is a random threshold, which split the data to the left or right child node. The number Γ of randomly sampled splitting functions is a hyperparameter, pre-defined by the user. The current node selects the function $\xi_{d,\tau}^j$ that best splits the data according to

$$I = -\frac{|L|}{|L| + |R|} \cdot H(L) - \frac{|R|}{|L| + |R|} \cdot H(R) \,, \tag{2}$$

where L and R are the left and right subsets of the data after splitting with $\xi_{d,\tau}^j$. In our implementation $H(\cdot) = -\sum_{y=1}^{C} p(y) \cdot (1 - p(y))$ is the Gini index, although the entropy can also be used; $p(y)$ is the class probability for class y. After having found the best splitting function $\xi_{d,\tau}^*$ among the Γ sampled ones, the data in that node is split and forwarded to the left and right child nodes, respectively, where the recursive algorithm continues.

During inference, a test sample $\mathbf{x}$ is traversed down all T trees and ends up in some leaf nodes. Each of these leaf nodes gives a prediction for a class label with

its learned class probabilities $p(y|\mathbf{x})$. The prediction of the complete ensemble of trees is then given as the average of those class probability estimates:

$$y^* = \arg\max_{y \in 1,\dots,C} \frac{1}{T} \sum_{t=1}^{T} p_t(y|\mathbf{x}) \,. \tag{3}$$

One property of Random Forests is that the generalization error can be estimated as a function of the strength of the individual trees and the diversity among them [6]. The generalization error is given as

$$GE \leq \bar{\rho} \frac{1 - s^2}{s^2} \,, \tag{4}$$

where $\bar{\rho}$ is the mean correlation between two pairs of trees and s is the strength of the trees in terms of prediction accuracy. Thus, strong but uncorrelated trees decrease the generalization error.

Random Forests have some desirable properties, especially for computer vision tasks. Due to their hierarchical structure, they are very fast during both training and testing. Furthermore, they can be easily parallelized (also on the GPU [17]) as the trees can be trained and tested independently from each other, making RFs even faster. RFs are inherently multi-class capable and do not rely on heuristic learning schemes like "one-vs.-all" or "one-vs.-one". Breiman [6] also states that RFs can handle significant amounts of noisy data. These reasons led to various applications for Random Forests [7], like semantic image labeling [18], object detection [10], object tracking [11, 12], or image classification [4, 15].

3 Ordinal Split Functions for Random Forests

In this section, we introduce our novel splitting function for Random Forests that is based on partial order statistics. We thus modify the splitting function in Eq. (1) with the goal to neglect absolute values of single dimensions in a feature vector $\mathbf{x}$ and observe order statistics between several features instead.

We define our novel splitting function as

$$\psi^j_{\mathbf{D},\delta}(\mathbf{x}) = \begin{cases} 0, & \text{if } \arg\max_{d \in \{0,\dots,K-1\}} x(\mathbf{D})_d = \delta \\ 1, & \text{otherwise} \end{cases} \,, \tag{5}$$

where $\mathbf{D}$ is a vector of length K containing randomly sampled indices in the range $0, \dots, D-1$; $x(\mathbf{D})$ is the K-dimensional vector containing the feature dimension indexed by the vector $\mathbf{D}$; δ is an integer value in the range $[0, \dots, K-1]$. That is, to define such a splitting function, we have to construct the random vector $\mathbf{D}$ that selects K dimensions of the data sample $\mathbf{x}$, yielding the vector $x(\mathbf{D})$. Then, the response r of this data sample is defined as the index of the maximum element in $x(\mathbf{D})$, which is in the range $[0, \dots, K-1]$. If the response matches δ, the sample is forwarded to the left child node, otherwise to the right child node.

Let us now consider this splitting function in more detail. The standard splitting function given by Eq. (1) only considers a single feature dimension and

thresholds the values of this dimension. Samples $\mathbf{x}$ having a higher/lower value in a single dimension x_d than a threshold τ are forwarded to the left/right child nodes. After having split the data, the only commonality of the samples in one of the child nodes is that the values in one dimension are above/below τ. On the other hand, considering our novel splitting functions $\psi^j_{\mathbf{D},\delta}$ in Eq. (5), the (enforced) commonality of split data is much higher, thus making the splitting functions stronger. Considering samples in the left child node after splitting, all samples within this node fulfill $K-1$ inequalities. Namely, one of the randomly selected feature dimensions has a higher value than $K-1$ other feature dimensions. Although samples in the right child node only fulfill a single constraint, namely that the selected feature dimension is not the maximum, the overall (enforced) commonality of split data is still higher than in standard RFs, thus yielding stronger decision trees.

However, making single trees in an ensemble method stronger can be harmful, as often the diversity is reduced. As mentioned above (see Eq. (4)), to reduce the generalization error, we have to increase the strength of single trees, while still keeping the diversity across the forest. The new splitting functions $\Psi^j_{\mathbf{D},\delta}$ make the trees stronger, but we keep the diversity as the functions are still drawn complete randomly. Additionally, the number of available ordinal splitting functions is now $\binom{D}{K}$, instead of only D in standard splitting functions. Thus, the number of available splitting functions increases with K, as long as $K \leq \frac{D}{2}$.

Finally, we have to note that such transformations [19] or splitting functions like in Eq. (5) are only meaningful if the distributions of data samples from different classes do not share the same covariance information. Otherwise, relations between feature dimensions are not discriminative at all. That is, pure machine learning or artificial data is often not suitable for such splitting functions, however, when working with vision data, partial order statistics make sense for most data representations (*e.g.*, consider simple pixel tests [10, 18]).

4 Extension to Hough Forests

We now describe the extension of our splitting functions, introduced in Sec. 3, to the Hough Forests (HF) [10]. First, we briefly review HF and then integrate our novel splitting functions into the original formulation of [10].

Hough Forests work on small patches $\mathcal{P}_i$ extracted at random locations within a given bounding box from positive and negative training images of an object. Each patch is described with several features, termed channels. Positive samples additionally store an offset vector $\mathbf{o}_i$ pointing to the center of the bounding box. HFs then try to separate positive from negative patches and simultaneously cluster together similar positive patches according to their offset vectors $\mathbf{o}_i$.

The splitting functions in the HF framework are defined as follows:

$$\Omega(\mathcal{P}_i; a, p, q, r, s, \tau) = \begin{cases} 0, & \text{if } \mathcal{P}_i^a(p,q) < \mathcal{P}_i^a(r,s) + \tau \\ 1, & \text{otherwise} \end{cases}, \tag{6}$$

where a indicates the feature channel, $\mathcal{P}_i^a$ is the calculated feature response of feature type a for patch $\mathcal{P}_i$, (p, q) and (r, s) define two pixel locations within $\mathcal{P}_i^a$ and τ is a threshold. That is, each node in the HF randomly selects a feature channel and two pixels within the patch $\mathcal{P}_i$ and calculates the difference of the feature values. This difference is then thresholded to determine which patches are forwarded to the left or the right child node. This is already quite similar to our previously defined splitting functions (Eq. (5)) based on partial order statistics, however, only two pixels are considered and a different thresholding is applied.

To integrate the ordinal splitting functions into the Hough Forest framework, we also build on only one feature channel but select $K \geq 2$ pixels. Then, we split the data in the same way as described in Sec. 3 by forwarding all patches $\mathcal{P}_i$ to the left child, if the maximum pixel index is equal to the "threshold" δ (an integer between 0 and $K-1$). Hence, we forward patches $\mathcal{P}_i$ to the left child node if all $K-1$ inequalities (*i.e.*, simple pixel tests) are fulfilled. All other patches are forwarded to the right child node. Then, the standard growing procedure of HFs continues by selecting the best randomly sampled splitting function according to the optimization objective (see [10] for more details).

In contrast to the works of Gall and Lempitsky [10] (and Shotton *et al.* [18]), we define splitting functions solely based on partial ordering of feature dimensions, not limited to two features (or pixels).

5 Experimental Evaluation

To demonstrate the benefits of our ordinal splitting functions, we perform two different experiments. First, image classification using Random Forests and, second, object detection using Hough Forests.

Table 1. Classification accuracy on the *Caltech101* data set, where the columns correspond to different forest sizes T. The rows show the evaluated methods, RF and ORF, where different values for the parameter K of ORF are tested. Best performing methods are marked **bold** for each forest size.

Method	$T = 1$	$T = 5$	$T = 10$	$T = 20$	$T = 50$	$T = 100$
RF	.161	.256	.347	.412	.461	.492
ORF, $K = 2$	.257	.363	.405	.445	.486	.496
ORF, $K = 3$	.262	.370	.402	.435	.483	.499
ORF, $K = 4$	.262	.372	.399	.443	.489	.508
ORF, $K = 5$	.266	.369	**.422**	**.460**	.488	.507
ORF, $K = 6$	**.273**	**.387**	.411	.455	.497	.511
ORF, $K = 7$	.269	.379	.421	.440	**.498**	**.520**

5.1 Image Classification

We first evaluate the novel ordinal splitting functions on two image classification tasks and compare it to the standard Random Forest framework [6]. We use the popular *Caltech101* object categorization data set [9] and the *USPS* digit categorization data set [13][1]. The *Caltech101* data set contains images from 101 categories (we leave out the additional background class), where we use 15 images per class for training and 10 images for testing. This results in a training set of 1515 images and a test set of 1010 images. As feature representation, we used *PHOW* features [5][2], for gray and color channels, each with a codebook size of 600 (computed with k-means), ending up in a 1200-dimensional feature vector.

The *USPS* data set consists of 7291 training samples and 2007 test samples from 10 different classes, *i.e.*, hand-written digits from 0 to 9. The 256 dimensions of the feature vectors in this data set contain the raw pixel values from a 16×16 gray scale image capturing the handwritten digits (the feature values are normalized in the range $[0\,,1]$).

Table 2. Classification accuracy on the *USPS* data set, where the columns correspond to different forest sizes T. The rows show the evaluated methods, RF and ORF, where different values for the parameter K of ORF are tested. Best performing methods are marked **bold** for each forest size.

Method	$T = 1$	$T = 5$	$T = 10$	$T = 20$	$T = 50$	$T = 100$
RF	.820	.888	.917	.926	.936	.938
ORF, $K = 2$	**.851**	.924	**.939**	**.946**	.947	**.950**
ORF, $K = 3$	.841	**.926**	.936	.944	**.949**	.949
ORF, $K = 4$	.849	.917	.930	.941	.944	.949
ORF, $K = 5$	.834	.911	.929	.938	.945	.947
ORF, $K = 6$	.828	.904	.925	.938	.941	.943
ORF, $K = 7$	.816	.899	.927	.933	.939	.941

We compare our novel Ordinal Random Forest (ORF) framework with standard Random Forests[3]. We evaluate our approach for different numbers of trees in the forest and, for *ORF*, also for different values of the window size K. As suggested in [6], we set the number of randomly sampled splitting functions per node in the standard RF to $\sqrt{D}$, where D is the feature dimensionality. As already mentioned in Sec. 3 we have to increase the number of sampled splitting functions for our approach as the feature space is increasing when sampling more than one feature for a single splitting function. In fact, the number of possible combinations of K feature dimensions is $\binom{D}{K}$, resulting in $\sqrt{\binom{D}{K}}$ sampled

[1] `http://www.cs.nyu.edu/~roweis/data.html`
[2] We use the publicly available toolbox from `http://www.vlfeat.org/`
[3] We use public code from `http://code.google.com/p/randomforest-matlab/`

splitting functions for *ORF*. However, to keep the computational effort on a reasonable level, we reduced the number of sample tests to 500, which has been shown to be enough for our purposes. We depict our results for both data sets in Tables 1 and 2, respectively. As can be seen from both tables, compared to the standard RF, *ORF* improves the classification scores for all sizes of the forest and different values of the parameter K. Furthermore, we note that best results are obtained with higher values of K on the *Caltech101* data set and with lower values on the *USPS* data set. A reason for this could be the large amount of classes in *Caltech101* in combination with the rather small amount of training data, thus requiring stronger splits.

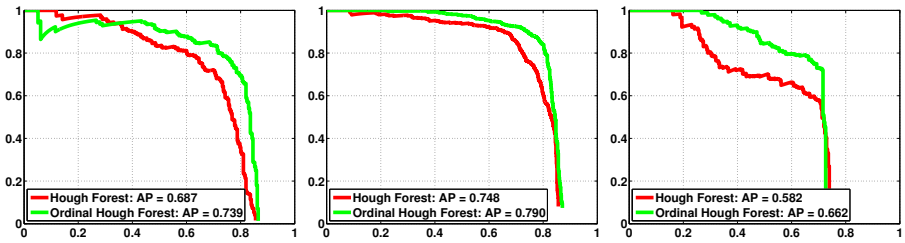

Fig. 1. (a-c) Precision-recall curves of OHF and HF on the three data sets, *i.e.*, *TUD-pedestrian*, *TUD-crossing*, and *TUD-campus*.

5.2 Object detection

In the second experiment, we evaluate our extension of HFs [10] with ordinal splitting functions (see Sec. 4) and compare it with the standard HF implementation on three popular pedestrian detection benchmarks, namely the *TUD-pedestrian*, *TUD-crossing*, and *TUD-campus* data sets [2], which were often used for evaluating Hough Forest-based approaches [3, 16].

The *TUD-pedestrian* data set contains 400 bounding-box annotated persons and 250 test images, capturing 311 pedestrians. The other data sets, *TUD-crossing* and *TUD-campus*, are smaller and contain only test sets with 201 and 71 images, respectively. We thus use the training images from the *TUD-pedestrian* data set as training data for all 3 test sets. For training, we uniformly extracted the patches from all training images, such that the total amount of positive training patches is around 10000; we use the same amount of negative patches, randomly cropped from the background.

For all our experiments we used 15 trees and a maximum depth of 15, similar to [10]. We use the publicly available implementation[4] for the Hough Forests,

[4] `http://www.vision.ee.ethz.ch/~gallju/projects/houghforest/houghforest.html`

where we also build our extensions on. Due to computational reasons, we set the number of randomly sampled tests for both methods to 2000, although our splitting functions would theoretically require more, as soon as $K > 2$. Nevertheless, we also evaluate the influence of the number of randomly drawn splitting functions (see Table 3). For all other evaluations we set $K = 4$.

We depict our results as precision-recall curves in Figure 1 and also report the average precision values [8]. As can be seen from the figures, the novel splitting function can boost the performance on all three data sets. All results are averaged over 3 independent test runs to compensate for the randomness in the training procedure of the forests.

Table 3. Average precision values on the *TUD-crossing* data set for different parameter values. The columns correspond to different numbers of randomly sampled splitting functions Γ. The rows show the evaluated methods, HF and *OHF* (with different values for K). Best performing methods are marked **bold** for each Γ.

Method	$\Gamma = 500$	$\Gamma = 1000$	$\Gamma = 2000$	$\Gamma = 3000$
HF	.757	.753	.761	.733
OHF, $K = 2$	.781	.776	.779	.767
OHF, $K = 3$	**.784**	**.782**	**.784**	.772
OHF, $K = 4$	.762	**.782**	.761	.769
OHF, $K = 5$	.771	.764	.769	.772
OHF, $K = 6$	.770	.763	.776	**.776**

We also evaluate the influence of the parameter K and the number of sampled random tests per node, Γ, on the *Ordinal Hough Forest*. For comparison, we also added the performance of the standard Hough Forests (HF) for different values of Γ. Table 3 shows the results as average precision values. As can be seen from the table, *OHF* outperforms HF for all parameter settings. Furthermore, to get good performance, the number of randomly sampled splitting functions Γ has to be increased with K. This is somehow clear, as also the possible number of available splitting functions increases with K.

6 Conclusion

Partial order statistics, which neglect absolute values of features and only consider relative differences and orderings, have recently shown good discriminative power; especially, when dealing with high dimensional data. In this work, we adopted these ideas and proposed a novel, more discriminative splitting function for Random Forests. We integrated this splitting function into both, the Random Forest and the Hough Forest frameworks. To demonstrate the benefits of the proposed approach, we applied it for two different tasks, *i.e.*, image categorization and object detection. In both cases the discriminative power can

be increased, and we are able to show that using the new splitting function the standard implementations (*i.e.*, for Random Forests and Hough Forests) can be outperformed. Hence, the approach could also be used for other computer vision tasks. Furthermore, future work also includes the combination of both kinds of splitting functions (absolute feature values and order statistics), getting the best of both worlds.

References

1. Amit, Y., Geman, D.: Shape Quantization and Recognition with Randomized Trees 9(7), 1545–1588 (1997)
2. Andriluka, M., Roth, S., Schiele, B.: People-Tracking-by-Detection and People-Detection-by-Tracking. In: CVPR (2008)
3. Barinova, O., Lempitsky, V., Kohli, P.: On Detection of Multiple Object Instances using Hough Transforms. In: CVPR (2010)
4. Bosch, A., Zisserman, A., Munoz, X.: Image Classification using Random Forests and Ferns. In: ICCV (2007)
5. Bosch, A., Zisserman, A., Munoz, X.: Representing Shape with a Spatial Pyramid Kernel. In: CIVR (2007)
6. Breiman, L.: Random Forests. ML 45(1), 5–32 (2001)
7. Criminsi, A., Shotton, J.: Decision Forests for Computer Vision and Medical Image Analysis. Springer (2013)
8. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. IJCV 88(2), 303–338 (2010)
9. Fei-Fei, L., Fergus, R., Perona, P.: One-Shot Learning of Object Categories. PAMI 28(4), 594–611 (2006)
10. Gall, J., Lempitsky, V.: Class-Specific Hough Forests for Object Detection. In: CVPR (2009)
11. Gall, J., Razavi, N., Van Gool, L.: On-line Adaption of Class-specific Codebooks for Instance Tracking. In: BMVC (2010)
12. Godec, M., Roth, P.M., Bischof, H.: Hough-based Tracking of Non-rigid Objects. In: ICCV (2011)
13. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer (2009)
14. Menze, B.H., Kelm, B.M., Splitthoff, D.N., Koethe, U., Hamprecht, F.A.: On Oblique Random Forests. In: ECMLPKDD (2011)
15. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: NIPS (2006)
16. Razavi, N., Gall, J., Van Gool, L.: Backprojection Revisited: Scalable Multi-view Object Detection and Similarity Metrics for Detections. In: ECCV (2010)
17. Sharp, T.: Implementing Decision Trees and Forests on a GPU. In: ECCV (2008)
18. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: CVPR (2008)
19. Yagnik, J., Strelow, D., Ross, D.A., Lin, R.s.: The Power of Comparative Reasoning. In: ICCV (2011)
20. Yao, A., Gall, J., Gool, L.v.: A Hough transform-based voting framework for action recognition. In: CVPR (2010)
21. Yao, B., Aditya, K., Fei-Fei, L.: Combining Randomization and Discrimination for Fine-Grained Image Categorization. In: CVPR (2011)