Fast and memory-efficient rigid object detection using a ternary node test fern

Kang-Kook Kong, Ki-Sang Hong
POSTECH
mjkkk@postech.ac.kr, hongks@postech.ac.kr

Abstract

In this paper, we propose a fast and memory-efficient detection algorithm for rigid objects. First, we propose a ternary node test fern classifier (called t-fern) which combines the robustness of local ternary patterns (LTP) with the discriminative power of a binary node test fern classifier (called b-fern). More specifically our proposed t-fern uses ternary node tests like the LTP and selects node test locations randomly in a larger patch like the b-fern. Then, we propose a rigid object detection algorithm which combines this t-fern with a Hough voting scheme. In the training phase, t-ferns learn the relationship between local patterns and voting information (voting locations and voting weights). In the testing phase, the identified local patterns through the t-ferns cast votes to the most probable locations of objects. In this way, our algorithm can achieve a high detection rate with high speed and memory efficiency. Experiments have been carried out with rigid object datasets such as the UIUC car, the TUD motorbike and the Weizmann horse datasets, showing state-of-the-art performance in the detection rate that is close to or better than that of existing methods. In addition, our algorithm seems to be ten times faster than other algorithms for both training and testing while using quite little memory, which is becoming more and more important in mobile environment.

1. Introduction

Object detection is the task that indicates the object center and the object scale using a bounding box in a given test image. To do this, the sliding window scheme [8, 9, 12, 22, 28, 37] and the Hough voting scheme [3, 14, 17, 19, 20, 21, 23, 24, 34, 38] are mainly used as the core components of detection algorithms.

The sliding window scheme usually uses Support Vector Machine (SVM) to learn object structures or appearance as a model and to match each subwindow of an input image to the learned model. The histogram of Oriented Gradients (HOG) [9], bag-of-visual-word (bovw) [16] and k adjacent segments (kAS) [37] are used as the feature descriptor of SVM. Because these approaches are somewhat slow, the Efficient Subwindow Search (ESS) method [15, 16, 18] has been proposed.

The Hough voting scheme converts an input image to a parameter space called a Hough image which consists of object location and scale. Each point in the Hough image corresponds to a hypothesis about the object of interest. The Implicit Shape Model (ISM) [20, 21] extracts feature descriptors from keypoints of the feature detector and matches these to predefined codewords and casts votes on the probable locations of objects. The Principled Implicit Shape Model (PRISM) [19] is a more advanced version of ISM, and Fast PRISM [17] combines PRSIM with ESS for speed. These algorithms are based on a keypoint detector and descriptor which require additional feature extraction time.

Recently Hough Forest [14, 34] has been proposed which gives the state-of-the-art detection rate. It combines random forests with the Hough voting scheme unlike the previous methods using a keypoint detector and descriptor. A random forest can directly map an image patch to a probabilistic vote, so all image patches of multi-channel images (for example, intensity, x-directional derivative or y-directional derivative channels) can be regarded as keypoint descriptors. Therefore it does not require additional feature extraction time, but it does require multi-channel extraction time for image patches. It also requires the node test optimization for random forests under the two constraints (offset uncertainty and class-label uncertainty). The robustness and the discriminative power obtained from heavy computations such as multi-channel extraction and node test optimization enable Hough Forest to handle general objects, both rigid and non-rigid.

By limiting the detection scope to rigid objects only, our algorithm aims at the speed of training and testing and memory efficiency as well as performance. Our goal is to develop a fast and memory-efficient detection algorithm for rigid objects. High speed and memory efficiency seem to be more and more important when considering the current mobile environment. To do this we propose a new local pattern classifier and combine it with the Hough voting scheme.

The proposed local pattern classifier is explained in Sec. 2. The training and the testing procedure of our algorithm are described in Sec. 3. Experimental results and implementation details are given in Sec. 4. The paper

concludes in Sec. 5.

2. The proposed local pattern classifier

The widely used local pattern classifier is Local Binary Patterns (LBP) [29, 30, 41], which is invariant to monotonic gray-scale change and has low computational complexity. The main application area of LBP is texture analysis, and it shows good performance to classify textured patches. LBP at pixel location **x** is defined as:

$$LBP_{D,N}(\mathbf{x}) = \sum_{d=0}^{D-1} b\left(p(\mathbf{x}), \mathbf{p}_d, \mathbf{q}_d\right) 2^d, \quad (1)$$

$$b\left(p(\mathbf{x}), \mathbf{p}_d, \mathbf{q}_d\right) = \begin{cases} 1, & if \;\; p(\mathbf{x}, \mathbf{p}_d) - p(\mathbf{x}, \mathbf{q}_d) > 0 \\ 0, & otherwise \end{cases}, \text{ and} \quad (2)$$

$$\mathbf{p}_d = \mathbf{x}, \;\; \mathbf{q}_d \in L(\mathbf{x}, N), \quad (3)$$

where local pattern (patch) $p(\mathbf{x})$ is an $N \times N$ patch centered at **x** of the intensity channel image, D denotes the number of node test (depth), the function $b(\cdot)$ is binary node test, and $p(\mathbf{x}, \mathbf{q})$ is the intensity value at pixel location **q** in $p(\mathbf{x})$. The set $L(\mathbf{x},N)$ denotes circularly symmetric neighbor sets in $p(\mathbf{x})$, and pixel locations ($\mathbf{p}_d$, $\mathbf{q}_d$) of node test. The node test locations of LBP are symmetric about the center of the patch and depth D varies according to the patch size N. Although LBP is suitable for texture analysis, it is limited in that it is sensitive to image noise and small variations in patches because it uses a binary node test.

To overcome this limitation, Local Ternary Patterns (LTP) [28, 40] has been proposed as the extended version of LBP. By using a ternary node test LTP can reduce the effect of image noise and small variations. LTP at pixel location **x** is defined as:

$$LTP_{D,N}(\mathbf{x}) = \sum_{d=0}^{D-1} t\left(p(\mathbf{x}), \mathbf{p}_d, \mathbf{q}_d, T\right) 3^d, \quad (4)$$

$$t\left(p(\mathbf{x}), \mathbf{p}_d, \mathbf{q}_d, T\right) = \begin{cases} 1, & if \;\; p(\mathbf{x}, \mathbf{p}_d) - p(\mathbf{x}, \mathbf{q}_d) > T \\ 2, & if \;\; p(\mathbf{x}, \mathbf{p}_d) - p(\mathbf{x}, \mathbf{q}_d) \le -T \\ 0, & otherwise \end{cases}, \text{ and} \quad (5)$$

$$\mathbf{p}_d = \mathbf{x}, \;\; \mathbf{q}_d \in L(\mathbf{x}, N), \quad (6)$$

where the function $t(\cdot)$ is the ternary node test and T is its threshold that determines the sensitivity to small variations in patches. Although LTP is robust to small variations, it also has some limitations. When the widely used $LTP_{8,3}$ is applied to rigid object detection, the main disadvantage is the lack of the discriminative power, because 3×3 patch size is too small to deal with all possible patches of objects. To increase the discriminative power, the LTP with larger patch size such as $LTP_{16,5}$ and $LTP_{24,7}$ can be used, but in this case LTP requires too many node tests, which causes too large memory space.

To keep simplicity and robustness and to increase the discriminative power, we propose a ternary node test fern classifier (called t-fern). Our t-fern combines the robustness of LTP with the discriminative power of a binary node test fern classifier (called b-fern) which comes from a random fern. Originally, random fern-based descriptors were used for fast keypoint recognition [7, 25, 26]. In those papers, a keypoint matching problem is regarded as keypoint recognition using binary node test fern classifiers. In our paper, a random fern is used to classify local patterns like LBP and LTP. The t-fern is more robust to image noise than the b-fern due to the use of ternary node tests and more discriminative than the LTP due to the selection of node test locations randomly in the larger patch. We define b-fern and t-fern at pixel location **x** as:

$$bFern_{D,N}(\mathbf{x}) = \sum_{d=0}^{D-1} b\left(p(\mathbf{x}), \mathbf{p}_d, \mathbf{q}_d\right) 2^d, \quad (7)$$

$$tFern_{D,N}(\mathbf{x}) = \sum_{d=0}^{D-1} t\left(p(\mathbf{x}), \mathbf{p}_d, \mathbf{q}_d, T\right) 3^d, \text{ and} \quad (8)$$

$$\mathbf{p}_d, \;\; \mathbf{q}_d \in R(\mathbf{x}, N), \quad (9)$$

where the set $R(\mathbf{x},N)$ denotes pixels within $p(\mathbf{x})$ located at **x** among which pixel locations ($\mathbf{p}_d$, $\mathbf{q}_d$) for node test are selected randomly with Gaussian distribution, because it gives more stable results than other node test schemes [6].

In summary our proposed t-fern can be regarded as a general version of local pattern classifiers. For example, $LBP_{8,3}(\mathbf{x})$ is the special case that T is 0, D is 8, $p(\mathbf{x})$ is a 3×3 patch centered at **x** in the intensity channel, the set of first pixel locations of node test $\{\mathbf{p}_d\}$ is always the same as **x**, and the set of the second pixel location $\{\mathbf{q}_d\}$ is the 8-neighboring pixels of **x**. Likewise, LTP and b-fern can be regarded as a special case of t-fern.

3. The proposed algorithm

Our goal is to develop an efficient algorithm for detecting rigid objects. To do this, we combine the proposed t-fern with the Hough voting scheme. In the training phase, t-ferns learn the relationship between local patterns and voting information (voting locations and voting weights). In the testing phase, the identified local patterns through t-ferns cast votes on the most probable locations of objects in the Hough image. From the next sub-section, we explain the training and the testing phases of our algorithm.

3.1 The training phase

The training phase is to construct t-ferns and to learn voting information (voting locations and voting weights) of the t-ferns.

First, t-ferns are constructed for a chosen depth D, patch size N, and threshold of ternary node test T. By choosing D node tests (D depths), each t-fern has 3^D leaf nodes. Therefore the decimal integer value of $tFern(\mathbf{x})$ is in the range of [0, 3^D-1]. The value of $tFern(\mathbf{x})$ itself can be regarded as a codeword index of the patch. In the aspect of local pattern classification, t-fern is likely to map similar local patterns to the same codeword.

The next step is to map all patches to codewords. Codewords of patches are obtained through the t-ferns using Eq. (8). All pixels of training images are regarded as keypoints (centers of patches). Therefore feature detectors are not used at all in our algorithm. Then, offset vectors of all patches of positive training images are obtained.

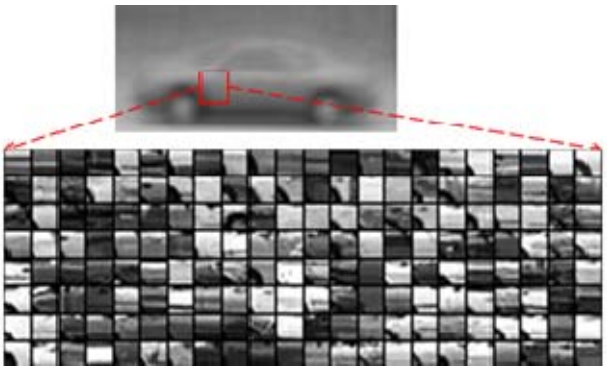

Figure 1 TOP: The averaged image of positive samples in the UIUC car training dataset. BOTTOM: Examples of patches of positive samples at the same relative position. All patches are from the UIUC car data set [1].

Because we assume that the object centers of positive samples are already known, the relative location to object center (offset vector) can easily be calculated.

The final step is to learn voting information for each t-fern from the relationship between codeword and offset vector obtained above. The voting weight $w(\mathbf{s},v)$ of codeword v at offset vector $\mathbf{s}$ is defined as:

$$w(\mathbf{s},v)=\frac{P(\mathbf{s},v)/P_T(\mathbf{s})}{P(\mathbf{s},v)/P_T(\mathbf{s})+N(v)/N_T}, \quad (10)$$

$$P_T(\mathbf{s})=\sum_v P(\mathbf{s},v),\ \ N_T=\sum_v N(v), \quad (11)$$

where $P(\mathbf{s},v)$ is the number of patches of positive samples at relative location $\mathbf{s}$ with codeword v, and $N(v)$ is the number of patches of negative samples with codeword v. For aligned rigid objects, all patches of positive samples at the same relative position to object center would have similar appearances (local patterns) as shown in Figure 1 and the t-fern is likely to map similar local patterns to the same codeword, so all patches in the same relative position should have the same codeword with high probability. This can be the cue for rigid-object detection and is reflected in the voting weight. After voting weight $w(\mathbf{s},v)$ is calculated, voting locations $\mathbf{t}_{vj}$ (j=1, … , J) are obtained which are offset vectors with j-th highest voting weight among offset vectors $\{\mathbf{s}\}$ in codeword v. Therefore each leaf node with its own codeword v has J sets of voting locations $\mathbf{t}_{vj}$ and voting weights $w(\mathbf{t}_{vj},v)$ (j=1, … , J) as shown in Figure 2.

3.2 The testing phase

For a given test image I, the testing phase predicts the center and scale of objects from the Hough image. We assume that the aspect ratio of objects is fixed and known.

First, codewords of all patches are obtained through the t-ferns prepared during the training phase using Eq. (8). All pixels of the testing image are regarded as keypoints (centers of patches). The identified codeword yields voting information (voting weights and voting locations) which was also learned in the training phase.

The next step is to cast votes on the probable locations in the Hough image. The Hough image $H^k(\mathbf{x}|I)$ from each t-fern is computed as:

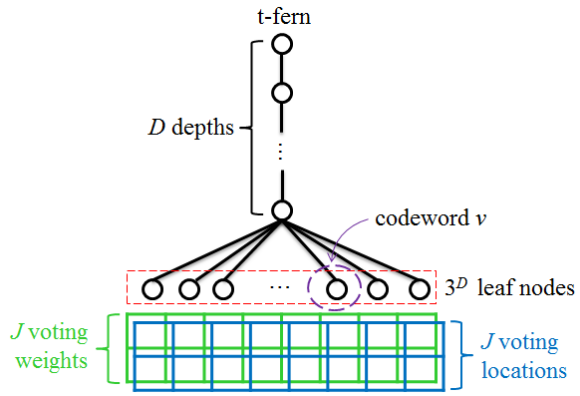

Figure 2 The t-fern configuration. Each t-fern has 3^D leaf nodes (codewords). Each leaf node has J voting locations together with corresponding weights.

$$H^k(\mathbf{x}|I)=\sum_{\mathbf{y}\in I}\sum_{v}^{V}\left(\delta\left(v=tFern^k(\mathbf{y})\right)\sum_{j}^{J}w(\mathbf{t}_{vj},v)G\left(\mathbf{x}-\left(\mathbf{y}-\mathbf{t}_{vj}\right)\right)\right), \quad (12)$$

where $\mathbf{x}$ is a location of the Hough image and $tFern^k$ denotes the k-th t-fern, $\mathbf{y}$ is a center of local pattern (patch) in the test image I, $\mathbf{y}$-$\mathbf{t}_{vj}$ is the probable object location, V is the number of codewords, and J is the number of votes which corresponds to that of voting locations and voting weights in codeword v. The function $G(\cdot)$ is a Gaussian function with zero mean and is used to compensate for the uncertainty of voting location. The total Hough image $H(\mathbf{x}|I)$ is obtained by summing the Hough images from each t-fern, $H^k(\mathbf{x}|I)$.

$$H(\mathbf{x}|I)=\sum_k^K H^k(\mathbf{x}|I), \quad (13)$$

where K is the number of used t-ferns. In our algorithm, a single t-fern (K = 1) gives adequate performance, as shown in Table 4, but it is possible to use many t-ferns to increase the detection rate.

For multi-scale objects we repeat the single scale detection procedure for each testing scale by resizing the test image. After the Hough image is obtained for each scale, the object's center is predicted by finding the maximum voting score in 3D space (location and scale). There are many methods to find the maxima, including the mode seeking algorithm and the mean-shift algorithms [20, 21, 24]. We simply find the maxima in each scale, and compare the voting score across the scale. Finally, objects are detected by thresholding the voting score.

4. Experimental Results

The experimental result section consists of three parts. First, parameter sensitivity is shown, and then the detection performance of various local pattern classifiers mentioned above is shown. Finally the performance of our algorithm is compared with that of existing detection methods. Our algorithm is implemented on an Intel® i7 CPU operating at 3.4GHz using Matlab. The effect of random selection of node test location is considered by repeating experiments with the same parameter settings for 30 predefined t-ferns,

and performance results are averaged. Performance is measured by Equal recall precision Error Rate (EER) and Average Precision (AP). Detected results are accepted as correct if the intersection area of the detected bounding box and the ground-truth bounding box exceeds 50% of the union of the two boxes, which is the PASCAL Challenge criterion.

4.1 Parameter sensitivity

In order to show the parameter sensitivity of our algorithm, it is applied to the TUD motorbike dataset [10]. The parameters used in our algorithm can be divided into two sets. One set is the parameters related to t-fern such as depth (D), patch size (N) and threshold of ternary node test (T) in Eq. (8). The other is the parameters related to the Hough voting scheme such as sampling density (Q) and the number of votes (J) in Eq. (12). The default parameter setting is a single t-fern (K = 1, D = 9, N = 11, T = 20) with the Hough voting scheme (Q = 1, J = 3).

The parameter sensitivities of depth (D), patch size (N) and threshold of ternary node test (T) are shown in Table. 1. Usually, as depth of t-fern increases, the performance increases. Eight or nine depths seem efficient from the aspect of the detection performance and memory requirement. Also, the middle patch size (9~13 pixels) gives good performance. Too small or too large patch size decreases the performance. A threshold of ternary node test in the range of 15~25 gives good performance for the intensity value of an image in the range of [0, 255].

The parameter sensitivity of sampling density (Q) and the number of votes (J) is shown in Table. 2. Sampling density means how many pixels in a test image are used in the testing process. For example, sampling density 1 means all pixels are used and sampling density 1/4 means one out of 2×2 pixels is used. It is natural that dense sampling gives better performance than sparse sampling. The number of votes means how many offset vectors among all possible offset vectors are used for voting. In the training phase, J voting locations are selected from offset vectors with the highest voting weight. We found that a small number of votes are enough to give good performance as shown in Table 2.

4.2 Performance of the local pattern classifiers

The performance between various local pattern classifiers (LBP, LTP, b-fern and t-fern) was also compared for the TUD motorbike dataset. For this comparison, $LBP_{8,3}$ (D = 8, N = 3), $LTP_{8,3}$ (D = 8, N = 3, T = 20), $bFern_{9,11}$ (D = 9, N = 11) and $tFern_{9,11}$ (D = 9, N = 11, T = 20) are used. In all cases, a single local pattern classifier is used and the experiment is done in the default setting (Q = 1, J = 3). As shown in Table 3, t-fern gives the best performance.

Table 1 The parameter sensitivity of depth (D), patch size (N) and threshold of ternary node test (T) in the TUD motorbike dataset. Performance is measured by EER (%) and AP (%).

D	EER	AP	N	EER	AP	T	EER	AP
5	60.6	62.3	5	82.7	85.2	0	48.0	35.7
6	70.6	73.9	7	86.9	89.5	5	81.6	85.2
7	78.3	81.7	9	88.7	91.6	10	87.2	89.6
8	85.9	88.2	11	89.0	91.2	15	88.4	90.9
9	89.0	91.2	13	88.6	91.2	20	89.0	91.2
10	87.5	90.8	15	86.7	91.4	25	88.9	91.8
			17	85.3	90.1	30	87.5	90.5

Table 2 The parameter sensitivities of sampling density (Q) and the number of votes (J) in the TUD motorbike dataset. Performance is measured by EER (%) and AP (%).

Q	EER	AP	J	EER	AP
1	89.0	91.2	1	87.7	89.8
1/4	87.8	90.7	2	88.4	90.7
1/9	85.3	88.3	3	89.0	91.2
1/16	83.0	86.7	4	89.0	91.3
1/25	78.8	82.9	5	89.1	91.5

Table 3 Performance comparison of the local pattern classifiers (LBP, LTP, b-fern and t-fern) in the TUD motorbike dataset. Performance is measured by EER (%) and AP (%).

method	EER	AP
$LBP_{8,3}$	41.6	32.7
$LTP_{8,3}$	78.4	82.1
$bFern_{9,11}$	48.0	35.7
$tFern_{9,11}$	89.0	91.2

4.3 Performance comparison

In order to evaluate the performance of our algorithm and compare it with existing detection methods, we apply our method to three datasets (the UIUC car [1], the TUD motorbike [10] and the Weizmann horse [35] datasets). Each dataset contains the objects from a fixed viewpoint and pose. The test images are resized by a factor of 0.8 to detect larger objects. Table 4 shows experimental results of the state-of-the-art detection methods and our algorithm for each dataset.

The UIUC car dataset [1] contains images of side views of cars. The training set consists of 550 car images and 500 non-car images of a fixed size (100×40 pixels). There are two test datasets. One is the UIUC-Single dataset which consists of 170 images containing 200 cars of almost the same size (100×40 pixels). The other is the UIUC-Multi dataset which contains 108 images containing 139 cars of various sizes from 89×36 to 212×85 pixels. Both sets include partially occluded cars, multiple car instances, and cluttered backgrounds under challenging illumination. It is interesting to notice that a single t-fern is enough to get comparable performance to Hough Forest that uses 15 trees and the maximum 15 depths for each tree. Moreover Fast

Table 4 Summary of experimental results of Hough Fern and Hough LBP on the five datasets (**UIUC-S**ingle, **UIUC-M**ulti, **CalTech** car rear, **TUD m**otorbike and **TUD p**edestrian). AP (%) is also shown in the parentheses.

Methods		UIUC-Single	UIUC-Multi	TUD motorbike	Weizmann horse
ESS [16]		98.5%	98.6%	-	-
ISM [21]		97.5%	95.0%	87.0%	-
PRISM [19]		-	-	83.0%	-
Fast PRISM [17]		-	97.8%	81.0%	-
Hough Forest [34]		98.5%	98.6%	-	91.0%
t-fern	$K = 1$	98.5% (99.7%)	98.1% (99.7%)	89.0% (91.2%)	86.1% (90.2%)
	$K = 3$	99.2% (99.8%)	99.3% (100%)	90.0% (91.5%)	87.5% (91.4%)
	$K = 5$	99.2% (99.9%)	99.6% (100%)	90.1% (91.6%)	87.9% (91.6%)

PRISM takes about 800ms [17] using Matlab implementation, but our algorithm takes only 76ms using Matlab implementation.

The TUD motorbike dataset is a part of the 2005 PASCAL dataset [10]. For training, we use 153 motorbike side view images from the CalTech database [11], and use the INRIA person negative dataset [9] as a negative training set. Each training image is equally resized to 200 pixel width. The test set consists of 115 images containing 125 side views of motorbikes of different scales with cluttered background and occlusion. In this dataset, our algorithm shows that the single t-fern is enough to give the best performance. Moreover, our algorithm takes only 63ms using Matlab implementation while PRISM [19] takes 920ms (570ms for SURF feature extraction time and 350ms for voting and search time) with similar EER.

The Weizmann horse dataset [35] contains images of the near-side views of horses in natural scenes. This dataset is more challenging from the viewpoint of rigid object detection because of varying poses. We follow the training-testing split procedure, that is, 100 horse images and 100 background images are used for training, and 228 horse images and 228 background images are used for testing as suggested in [36]. Our algorithm shows comparable performance with Hough Forest, although it is not best. However, notice that our algorithm uses the intensity channel only and a small number of t-ferns.

In summary, our algorithm shows state-of-the-art performance in the detection rate, close to or better than existing methods. Additionally our algorithm has good detection speed and memory efficiency. Our algorithm is very fast in testing. As mentioned above, our algorithm is 10~15 times faster than PRISM and fast PRISM. In our algorithm, only the intensity channel is used and patch itself is used as a feature, so it does not require multi-channel extraction time and feature extraction time. Moreover our algorithm uses a small number of t-ferns and voting operations, which lead to a significant speed gain. Our algorithm is also very fast in training. Actually, our learning is usually completed in less than a minute, whereas Hough Forest, which needs the optimization process, takes much longer (hours). Our algorithm is also memory-efficient in testing. Because our algorithm uses the intensity channel only, it does not need integral images used in ESS and multi-channel images used in Hough Forest. Our algorithm also needs little memory to save voting information. For example, the single t-fern ($K = 1$, $D = 9$, $J = 3$) requires about 230KB ($K \times 3^D \times J \times 4$Bytes). This is much smaller than that of Hough Forest, which uses 15 trees and maximally 15 depths.

5. Conclusion

The well-known LBP and LTP, and the recently proposed b-fern can be regarded as a special version of our proposed t-fern. We show that our algorithm using t-fern combined with the Hough voting scheme is very efficient for rigid object detection in terms of detection rate, computation speed and memory requirement.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0016669)

References

[1] Agarwal S, Awan A, Roth D (2004) Learning to detect objects in images via a sparse, part-based representation. IEEE Trans Pattern Anal Mach Intell 26 : 1475-1490

[2] Amit Y, Geman D (1997) Shape quantization and recognition with randomized trees. Neural Computation 9 : 1545– 1588

[3] Barinova O, Lempitsky V, Kohli P (2010) On detection of multiple object instances using Hough transforms. In: Computer Vision and Pattern Recognition(CVPR), pp 2233-2240

[4] Bay H, Ess A, Tuytelaars T, Van Gool L (2008) SURF: Speeded Up Robust Features, Comput Vis Image Underst(CVIU) 110 : 346-359

[5] Breiman L (2001) Random forests, Machine Learning 45(1): 5–32

[6] Calonder M, Lepetit V, Fua P (2010) BRIEF: Binary Robust Independent Elementary Features. In: European Conference on Computer Vision(ECCV) 6314 : 778-792

[7] Calonder M, Lepetit V, Fua P, Konolige K, Bowman J, Mihelich P(2009) Compact Signatures for High-Speed Interest Point Description and Matching. In: International Conference on Computer Vision (ICCV), pp 357-364

[8] Chum O, Zisserman A (2007) An Exemplar Model for Learning Object Classes. In: Computer Vision and Pattern Recognition(CVPR), pp 1-8

[9] Dalal N, Triggs B (2005) Histograms of Oriented Gradients for Human Detection. In: Computer Vision and Pattern Recognition(CVPR) 1: 886-893

[10] Everingham M, Zisserman A et al (2006) The 2005 PASCAL visual object classes challenge. In: *First PASCAL Challenge Workshop* 3944:117-176

[11] Fergus R, Perona P. Zisserman A (2003) Object class recognition by unsupervised scale-invariant learning. In: Computer Vision and Pattern Recognition(CVPR) 2:264-271

[12] Ferrari V, Jurie F, Schmid C (2007) Accurate object detection with deformable shape models learnt from images. In: Computer Vision and Pattern Recognition(CVPR), pp:1-8

[13] Fritz M, Leibe B, Caputo B, Schiele B (2005) Integrating representative and discriminant models for object category detection. In: International Conference on Computer Vision (ICCV) 2:1363-1370

[14] Gall J, Lempitsky V (2009) Class-Specific Hough Forests for Object Detection. In: Computer Vision and Pattern Recognition(CVPR), pp:1022-1029

[15] Lampert C.H, Blaschko M.B, Hofmann T (2008) Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. In: Computer Vision and Pattern Recognition(CVPR),pp:1-8

[16] Lampert C.H, Blaschko M.B, Hofmann T (2009) Efficient Subwindow search: A branch and bound framework for object localization. IEEE Trans Pattern Anal Mach Intell 31:2129-2142

[17] Lehmann A, Leibe B, Van Gool L (2011) Fast PRISM: Branch and Bound Hough Transform for Object Class Detection. Int J Comput Vis 94:175-197

[18] Lehmann A, Leibe B, Van Gool L (2009) Feature-Centric Efficient Subwindow Search. In: International Conference on Computer Vision (ICCV), pp:940-947

[19] Lehmann A, Leibe B, Van Gool L (2009) PRISM: Principled implicit shape model. In: British Machine Vision Conference(BMVC)

[20] Leibe B, Schiele B (2003) Interleaved object categorization and segmentation. In: British Machine Vision Conference(BMVC),pp 759-768

[21] Leibe B, Leonardis A, Schiele B (2008) Robust Object Detection with Interleaved Categorization and Segmentation. Int J Comput Vis 77: 259-289

[22] Maji S, Berg A.C, Malik J (2008) Classification Using Intersection Kernel Support Vector Machines Is Efficient. In: Computer Vision and Pattern Recognition(CVPR),pp:1-8

[23] Maji S, Malik J (2009) Object detection using a max-margin hough transform. In: Computer Vision and Pattern Recognition(CVPR), pp:1038-1045

[24] Opelt A, Pinz A, Zisserman A (2008) Learning an alphabet of shape and appearance for multi-class object detection. Int J Comput Vis 80:16-44

[25] Ozuysal M, Fua P, Lepetit V (2007) Fast Keypoint Recognition in Ten Lines of Code. In: Computer Vision and Pattern Recognition(CVPR),pp:1-8

[26] Ozuysal M, Calonder M, Lepetit V, Fua P(2010) Fast Keypoint Recognition using Random Ferns. IEEE Trans Pattern Anal Mach Intell 32: 448-461

[27] Quinlan J.R (1986) Induction of decision trees. *Machine Learning*, 1: 81–106

[28] Tan X, Triggs B (2007) Enhanced local texture feature sets for face recognition under difficult lighting conditions. In: International conference on Analysis and modeling of faces and gestures(AMFG), pp 168–182.

[29] Mu Y, Yan S, Liu Y, Huang T, Zhou B (2008) Discriminative local binary patterns for human detection in personal album. In: Computer Vision and Pattern Recognition(CVPR),pp 1-8

[30] Ojala T, Pietik¨ainen M, Harwood D (1996) A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition* 29:51-59

[31] Andriluka M, Roth S, Schiele B (2008) People-tracking-by-detection and people-detection-by-tracking. In: Computer Vision and Pattern Recognition(CVPR)

[32] Seemann E, Schiele B (2006) Cross-articulation learning for robust detection of pedestrians. In: Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM) 4174 : 242-252

[33] Wang Xiaoyu, Han Tony X, Yan Shuicheng(2009)An HOG-LBP Human Detector with Partial Occlusion Handling. In: International Conference on Computer Vision (*ICCV*), pp 32-39

[34] Gall J, Yao A, Razavi N, Van Gool L, Lempitsky V(2011) Hough Forests for Object Detection, Tracking, and Action Recognition, IEEE Trans Pattern Anal Mach Intell (PAMI) 33:2188-2202

[35] Borenstein E and Ullman S (2002) Class-specific, top-down segmentation, In: European Conference on Computer Vision(ECCV) 2351:639-641

[36] Shotton J, Blake A, Cipolla R (2008) Multiscale categorical object recognition using contour fragments, IEEE Trans Pattern Anal Mach Intell (PAMI) 30(7): 1270–1281

[37] Ferrari V, Fevrier L, Jurie F, Schmid C (2008) Groups of adjacent contour segments for object detection, IEEE Trans Pattern Anal Mach Intell (PAMI) 30(1):36–51

[38] Ferrari V, Jurie F, Schmid C (2010) From Images to Shape Models for Object Detection, Int J Comput Vis 87:284-303

[39] Jurie F,Schmid C(2004) Scale-invariant shape features for recognition of object categories, In: Computer Vision and Pattern Recognition(CVPR) 2: II-90 - II-96

[40] Tan X,Triggs B(2010) Enhanced Local Texture Feature Sets for Face Recognition Under Difficult Lighting Conditions. IEEE Trans Image Process 19(6):1635-1650

[41] Ojala T, Pietikäinen M, Mäenpää T (2002)Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7) :971-987