FACTA UNIVERSITATIS
Series: **Automatic Control and Robotics** Vol. 18, Nº 2, 2019, pp. 79 - 94
https://doi.org/10.22190/FUACR1902079C

DETECTION OF TEXTURE-LESS OBJECTS BY LINE-BASED APPROACH

UDC (004.722+(004.415.2:004.4'27))

Stevica Cvetković[1], Nemanja Grujić[1], Slobodan Ilić[2], Goran Stančić[1]

[1]University of Niš, Faculty of Electronic Engineering, Republic of Serbia
[2]Siemens AG Corporate Technology, Munich, Germany

Abstract. *This paper proposes a method for tackling the problem of scalable object instance detection in the presence of clutter and occlusions. It gathers together advantages in respect of the state-of-the-art object detection approaches, being at the same time able to scale favorably with the number of models, computationally efficient and suited to texture-less objects as well. The proposed method has the following advantages: a) generality – it works for both texture-less and textured objects, b) scalability – it scales sub-linearly with the number of objects stored in the object database, and c) computational efficiency – it runs in near real-time. In contrast to the traditional affine-invariant detectors/descriptors which are local and not discriminative for texture-less objects, our method is based on line segments around which it computes semi-global descriptor by encoding gradient information in scale and rotation invariant manner. It relies on both texture and shape information and is, therefore, suited for both textured and texture-less objects. The descriptor is integrated into efficient object detection procedure which exploits the fact that the line segment determines scale, orientation and position of an object, by its two endpoints. This is used to construct several effective techniques for object hypotheses generation, scoring and multiple object reasoning; which are integrated in the proposed object detection procedure. Thanks to its ability to detect objects even if only one correct line match is found, our method allows detection of the objects under heavy clutter and occlusions. Extensive evaluation on several public benchmark datasets for texture-less and textured object detection, demonstrates its scalability and high effectiveness.*

Key words: *object detection, object localization, image descriptor, line segments, histogram of oriented gradients.*

Received April 03, 2019
Corresponding author: Stevica Cvetković
Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Republic of Serbia
E-mail: stevica.cvetkovic@elfak.ni.ac.rs

1. INTRODUCTION

Object instance detection from RGB images is a widely studied problem in computer and robot vision. The advent of local keypoint detectors/descriptors such as SIFT [1] paved the way to the deployment of solutions able to reliably deal with this problem in case of highly textured objects. Computational efficiency has been successively addressed by techniques such as SURF [2], BRIEF [3] and ORB [4]. Nevertheless, one of the main limitations of such techniques is dealing with texture-less objects. These objects lack corners, blobs and local gradient information which are typically used for keypoint related detector and descriptors, which represents their major limitation when applied to texture-less objects. However, such objects are commonly present in many application scenarios, such as visual inspection, mobile visual search or robot guidance. While recent expansion of RGB-D sensors introduced an additional cue useful for the visual perception of texture-less objects [5, 26], restrictions still remains for application of such devices in outdoor environments.

Techniques specifically addressing the problem of texture-less object detection can be split into two main categories: template-based and line-based. Methods within the former category rely on template matching, where templates typically encode image gradient information [6, 7, 8]. The latter applies the feature detection and description paradigm specifically designed to embed the object shape using line segments [9, 10]. The main limitation of current template-based approaches is scalability with respect to the model database size, as well as their sensitivity to occlusions. Recently introduced line-based methods [9, 10] for detection of texture-less objects tackle scalability and occlusion issues, but cannot deal with textured objects, and shapes that are not naturally described by sets of straight lines. While significant amount of research has addressed each mentioned issue separately (texture-less objects, scalability, occlusion handling), addressing them simultaneously is still extremely challenging.

The goal of our work is to develop a novel technique that can deal with both texture-less as well as highly textured objects, tackling the issue of scalability with respect to the number of objects in the database. At the same time, we want our technique to be robust to typical nuisances such as noise, clutter and occlusion. To this end, we propose a method based on a novel descriptor - *LineHOG*, computed around pre-extracted line segments, having the characteristic of being highly distinctive even when the object surface lacks enough texture details (Fig. 1). The main concept behind the introduced descriptor is to encode spatially localized gradient information inside a patch defined by the location and orientation of the line segment. Being feature-based, it can be combined with efficient approximate nearest neighbor (ANN) indexing schemes, which yields logarithmic complexity with respect to the number of models in the object library. This descriptor is integrated in our efficient object detection procedure, specifically adopted to exploit the properties of line segments. We proposed several efficient techniques for homography estimation, inlier detection and matching, and multiple object reasoning; which are shown to significantly improve the overall detection accuracy.

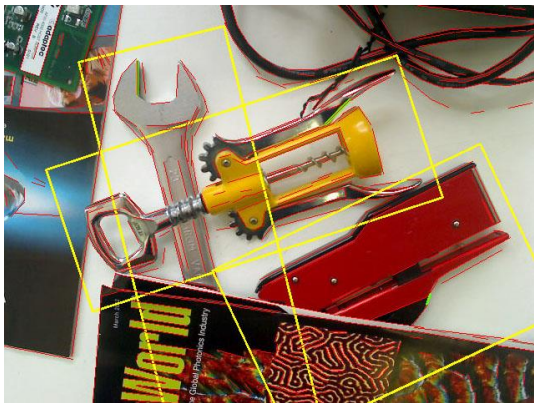

Fig. 1 Example of three correctly detected object instances in presence of clutter and occlusion, using the proposed *LineHOG* method. Extracted line segments are shown in red.

We evaluated our method in terms of its applicability to the presented issues, with a special emphasis on texture-less object detection in challenging environments. Experiments over several benchmark datasets for texture-less and textured object detection demonstrate the state-of-the art performance, showing particular characteristics in terms of robustness towards clutter and occlusion.

2. Related Work

There is a large number of research papers primarily considering textured object detection [1, 14, 15]. By contrast, only a limited amount of works is focusing on texture-less object detection. Since literature review of textured object detection exceeds the scope of this paper, we will focus only on related works for texture-less object detection. Detection of texture-less objects has just recently been addressed in [6-13]. The approaches to texture-less object detection can be divided into two broad categories: template-based and line-based.

Template-based methods for texture-less object detection. These methods associate an object of interest with a large number of templates, each encoding the appearance of the object as seen from a different viewpoint. The object of interest is detected in the image using a sliding window approach. The state-of-the-art in template-based methods is LINE-2D [12], which represents templates as binary strings using quantized gradient information. An extension has been proposed in [6], where, in addition to image gradients, surface normals are used in the similar way. The method demonstrated an impressive accuracy in extremely cluttered conditions, although several issues still remain: scalability, which is linear with the number of templates in the database, and sensitivity to occlusions.

The authors of [8] proposed to extend [6] by learning discriminative templates in just a few milliseconds, so to speed up detection by means of a cascaded classification scheme. Recently, a method appeared [13] proposing to tackle the occlusion handling limitation of [6] by means of a specific occlusion reasoning model which exploits prior knowledge to

estimate the statistics of the object portions appearing in a given environment. They demonstrate a significant performance improvement on a challenging occlusion dataset, but still without considering the scalability issue. A two-stage cascaded detection method is proposed in [11] for improved speed performance. The first stage prunes a vast majority of sliding windows, while the second stage uses the improved oriented chamfer score [16] for verification.

Although template-based methods achieve near real-time performance with a relatively small number of objects, real scalability to hundreds of partially occluded objects represented with the thousands of view dependent templates is still an issue for these approaches.

Line-based methods. Scalability issues have just recently been addressed by two methods [10], [9]. Relying on the SIFT-like detection pipeline, these methods introduced a specific line-based detector and a descriptor to avoid the sliding window approach typical for template matching. Damen et al. [10] introduced a local shape descriptor that computes relative orientations and distances between consecutive line segments. Although it has shown promising results, it lacks proper evaluation over standard challenging texture-less and textured datasets. Tombari et al. [9] proposed the BOLD descriptor, which encodes groups of neighboring line segments in a way that is invariant to rotation and scale changes. Besides excellent results on several texture-less datasets, detection accuracy of textured objects and curvilinear shapes is still limited.

By analyzing the performance of line-based methods, we observed that their accuracy on textured datasets is still inferior to SIFT-like methods. Conversely, our novel descriptor LineHOG significantly improves results of these methods by including appearance information around line segments in addition to just using line segments. We will start by giving a description of the proposed line based detector/descriptor called LineHOG. Afterwards, we will describe the entire pipeline for the generic and scalable detection procedure.

3. LINEHOG DETECTOR AND DESCRIPTOR

Instead of using salient keypoints to compute local appearance around them, our method relays on line segments which define a more global neighborhood around them. This seems to be a crucial concept for texture-less object detection. Although our method is independent of a specific line segment detector, repeatability of the extracted line segments is important for further processing. Among several line segment extraction algorithms recently proposed in the literature we have found that Line Segment Detector (LSD) [18, 19] provides high repeatability of the extracted segments, in line with what reported in [9]. Fig. 2 shows an example of LSD algorithm results. To overcome known limitations of the LSD algorithm in presence of noise and shadows, we extract line segments at multiple scales. Note that the number of extracted line segments is relatively small compared to the number of extracted keypoints in case of local feature based methods (SIFT, SURF, etc.), which additionally improves the overall efficiency of the method.

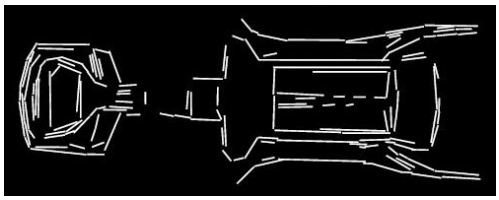

Fig. 2 An example of LSD (Line Segment Detector) algorithm results.

The LineHOG descriptor that we propose, captures the distribution of gradient orientations around a line segment relative to the segment's orientation. It is based on HOG [18], but contrary to the original HOG which is computed globally for the template defining the object at some view, the proposed descriptor is semi-global, as it is applied on an object subregion delimited by the line segment's length. Since it is computed relatively to the dominant orientation and length of the line segment, the descriptor is rotation and scale invariant.

In practice, we compute the LineHOG descriptor by firstly defining a rectangular region of interest around the line segment with the size determined by the length and the orientation of the segment. If we denote the length of the line as *L*, then the region used for LineHOG computation is of size *2L×2L*, centered around the line segment (Fig. 3). The edge orientation pattern in the area of the line segment is described by a HOG computed relatively to the orientation, position and length of the segment. More details about descriptor computation, along with parameters used, are given in section 5.

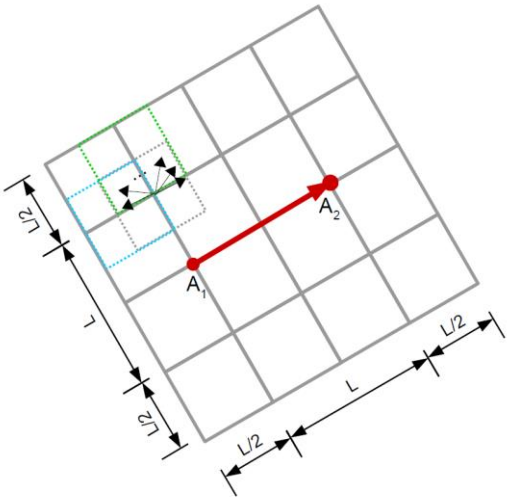

Fig. 3 Illustration of region over which our *LineHOG* descriptor is computed. A_1A_2 represents a line segment of length L. Surrounding region of size 2L×2L is divided into blocks of size L/2, with stride L/4. A histogram of edge orientations is computed for each block, and concatenated into a final descriptor.

4. LINE-BASED OBJECT DETECTION METHOD

Our object detection scenario assumes that multiple objects are given in a database, where each object is represented by a set of RGB images capturing the object seen from different viewpoints. Given a query image, the task is to locate all database objects that appear in the image. In practice, one or multiple objects from the database are present in the query image, but the size of the database, containing dozens of objects, extends beyond the number of objects usually present in one query image.

We achieve the scalability of the method by pre-processing images from the database, and building an efficient indexing data structure. This is done in an off-line phase, and does not affect run-time performance. The indexing data structure is later utilized in on-line detection phase to aid line segment matching. Concretely, for all images in the database, we extract line segments at several different scales and compute LineHOG descriptors, as previously described. Then, a randomized *kd-tree forest* [21] is built over all extracted LineHOG descriptors, which enables fast approximate NN search in detection phase.

On-line object detection phase should detect database objects that appear in the given query image. It starts by extracting line segments from the query image, and finding similar matches from the database. Each found match represents one object hypothesis, which is defined by an object that matched line segment belongs to. For all object hypothesis we compute a score, which represents an input to the multiple object detection phase, where we determine which hypothesis are promoted to object detections. Therefore, our object detection method consists of the following steps:

- Line segment extraction and matching.
- Object hypotheses generation.
- Object hypotheses scoring.
- Multiple object reasoning.

A detailed description of each step is given in the following.

4.1. Line segment extraction and matching

The goal of this step is to extract line segments from the query image, and to match them to the database. For each line segment from the query image, its match is line segment from the database with the most similar LineHOG descriptor. Matched line segment determines an exact view of an object, and is used to form an object hypothesis. To efficiently tackle descriptor matching we use a *kd-tree forest*, built in the off-line phase. To allow object detection at different scales the query image is downscaled several times before line segment extraction.

4.2. Object hypotheses generation

Each matched line segment defines one object hypothesis by its position, orientation and scale. We start by estimating a similarity transformation for every pair of matched line segments, and then refine it to estimate the full homography. Initial similarity transformation is used to transform line segments from the database image to the query image, and to detect inliers. Detected inliers are then used as new matches for estimating the full homography. By using this approach, we are able to generate hypotheses with only a single pair of matched line segments. This results in a significant reduction of possible hypothesis and

increased efficiency of the object detection procedure. Since our method performs exhaustive search over all possible hypothesis, it eliminates the randomness concept typical for the RANSAC based algorithms [22].

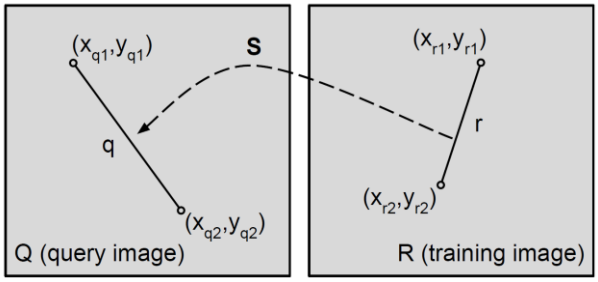

Fig. 4 Illustration of two matched line segments *q* and *r*. Similarity transformation *S* transforms line segment *r* to line segment *q*. The *S* can be obtained by solving a system of linear equations given by equation (2).

Estimation of similarity transformation

The matched line segments completely define a similarity transformation matrix $S_{3\times3}$ by their endpoints:

$$S = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \tag{1}$$

Such transformation is computed from 2 pairs of matched line endpoints which define 4 degrees of freedom (Fig. 4), as needed for *S*. Parameters of *S* can be computed by solving a system of linear equations:

$$\begin{bmatrix} x_{r1} & -y_{r1} & 1 & 0 \\ y_{r1} & x_{r1} & 0 & 1 \\ x_{r2} & -y_{r2} & 1 & 0 \\ y_{r2} & x_{r2} & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_{q1} \\ y_{q1} \\ x_{q2} \\ y_{q2} \end{bmatrix} \tag{2}$$

Since we match every line segment in the query image to the database, there are as many object hypotheses as line segments in the query image. The number of hypothesis does not depend on the number of images in the database, and that makes our method as scalable as the approximate nearest neighbor algorithm used (*kd-tree forest* [21]).

Inlier detection

The goal of this step is to determine line segments in the query image *Q*, which are inliers to the train image *R* (which contains matched segment), under similarity transformation *S*. Given the matched line segment in *R* and the transformation S, we apply the transformation to all other line segments from the train image *R* to detect inliers.

Here we define criteria for determining inliers. Let us denote line segments from the query image Q as q_i, and other line segments extracted on the object of interest (from the train image R) as r_j. Further, let $S(r_j)$ be the line segment r_j transformed by S. For every r_j, a set of possible inliers $Inl(r_j)$ consists of line segments from the image Q that simultaneously satisfy thresholds by position (T_c), orientation (T_φ), and length (T_l):

$$Inl(\mathrm{r_j}) = \{\mathrm{q_i} \in I : d_c(q_i, S(\mathrm{r_j})) < T_c \wedge d_\varphi(q_i, S(\mathrm{r_j})) < T_\varphi \wedge d_l(q_i, S(\mathrm{r_j})) < T_l\} \tag{3}$$

where $d_c(q,r)$ represents the distance between centers of line segments q and r, $d_\varphi(q,r)$ is a difference of the two line segment orientations, and $d_l(q,r)$ is a relative ratio of the line segment lengths:

$$\begin{aligned} d_c(q,r) &= \left\| q.center - r.center \right\|_2, \\ d_\varphi(q,r) &= \left| q.orientation - r.\mathrm{orientation} \right|, \\ d_l(q,r) &= \left| q.length - r.length \right| / r.length \end{aligned} \tag{4}$$

To find an inlier match $r_j^* \epsilon Q$ for every line segment $r_j \epsilon R$, we formulate our problem as the weighted bipartite matching (WBM) between the set of line segments $S(r_j)$ and the set of line segments q_i. In the bipartite matching graph, edges are added between the node that represent line segment $S(r_j)$, and nodes that represent line segments from the set of possible inliers (2). Edge weights are given by a composite distance measure:

$$d(q,r) = d_\varphi(q,r) + \alpha d_\varphi(q,r) + \beta d_l(q,r) \tag{5}$$

where α and β are orientation and relative length weighting parameters. The measure is inspired by the oriented Chamfer distance [14], and extended to be used with line segments.

To efficiently solve the WBM problem we are using a greedy algorithm, which gives suboptimal solution, but performs extremely fast. Other matching strategies may be employed as well, such as the Hungarian algorithm [23]. As a simplified alternative to the WBM formulation, inliers could be detected by finding the closest line segment in $Inl(r_j)$:

$$r_j^* = \underset{q_i \in Inl(r_i)}{\arg\min}\ d(q_i, S(r_j)) \tag{6}$$

The WBM approach prevents one line segment $q_i \epsilon Q$ to be counted as an inlier to multiple line segments $r_j \epsilon R$. The positive impact of WBM is shown in Table III.

Full homography estimation

A set of all inliers r_j^* is used to estimate the full homography transformation H. The center of each line segment r_j and the center of its inlier match r_j^*, is used as a pair to form a system of equations. Solving the system in the least square fashion, gives the full homography H between the image R and the image Q.

The full homography H is iteratively refined, by re-estimating the set of inliers r_j^* and the homography H in each iteration. During the iterative refinement, inlier thresholds (2)

are set to be twice as restrictive ($T_c/2, T_\varphi/2, T_l/2$), as proposed by Lowe [24]. The result of this step is the full homography H of the object hypothesis.

4.3. Object hypotheses scoring

For each object hypothesis, a similarity score *score(R;H)* is computed as the function of the number of inliers $N_{inl}(R;H)$, and the number of line segments in the train image R. Here we exploited the idea from [11] to compensate the bias toward simpler objects:

$$score(R,H) = \frac{N_{inl}(R,H)}{\lambda|R| + (1-\lambda)|R|_{avg}} \tag{7}$$

where $|R|$ is the number of line segments in a training image R, $|R|_{avg}$ is the average number of line segments over all training images, and $\lambda \in [0; 1]$ is a parameter that decreases the score for objects with fewer edge lines than the average. This score is then subject to a threshold T_{obj} used to decide whether the object is detected in the scene.
For $\lambda=1$, the score corresponds to the distance used in [14]. Lower values of λ decrease the score for objects with fewer line segments than average. This has a positive impact on the algorithm performance, as can be seen in Table 3.

4.4. Multiple object reasoning

A naive approach for multiple object reasoning would be to accept all object hypotheses that satisfy the threshold T_{obj}. That could lead to increasing of false positives, since multiple objects can be detected at the same position in the query image.

To reduce the number of false positive detections, we restrict a line segment to be an inlier in no more than one detected object. We examine all possible object hypotheses in a decreasing order of *score* in equation 7. Object hypotheses are processed in a greedy fashion using the priority queue for efficient computation. After an object hypothesis has been processed, its inliers $r_j^* \in Q$ are „banned“. This effectively means that banned inliers cannot be used by subsequent object hypotheses, which reduces their *score* in equation 7. This kind of inlier banning makes possible to avoid multiple object detections at the same position in the query image Q. Positive impact of inlier banning can be seen in Table 3.

When multiple views of an object are presented in the training database, our method can find the most similar view with as much as one correct line segment match. The standard methods based on RANSAC typically require at least three feature matches to be found in the same view to detect an object [25]. This restricts the accuracy of such methods because matches could be spread through similar views. Being able to detect an object with only one correct line segment match makes our method naturally applicable to scenarios when multiple views of an object are given.

5. EXPERIMENTAL EVALUATION

We performed the extensive evaluation of the proposed method on four publicly available datasets and compared it to the state-of-the-art methods for texture-less (LINE-2D [12], BOLD [9], Damen [10], etc.) and textured (SIFT [1], SURF [2], ORB [4]) object detection. The overview of the four used datasets is given in Table 1, while examples of objects and scenes from these datasets are shown in Fig.5.

Table 1 Test datasets overview including: number of objects (#obj), number of training images (#train) and number of test images (#test).

Dataset	textured	#obj	#train	#test
D-Textureless [9]	no	9	9	54
Obj30 [10]	no	30	1433	1220
Caltech Covers [9]	yes	80	80	50
CMU-KO8 Multi [13]	no	8	400	800

5.1. Parameters

For line segment extraction in all datasets we used default parameters of the LSD algorithm provided by [18]. After extraction, line segments with a length less than the predefined threshold are discarded (in our case set to 10 pixels). A scale pyramid is constructed by successive resizing of the image by the factor of 0.8 using bicubic interpolation. By default, we used 2 scales for line segment extraction (one original and one reduced), except for Damen's Obj30 dataset [10] where we needed 5 scales.

LineHOG descriptors are computed by first warping a patch around the oriented line segment to the canonical rectangular region. Then the rectangular region is divided into 4×4 blocks with strides equals to the half of the block size. A histogram of 13 gradient orientations (12 orientation bins + 1 non-oriented) is computed for each block and no block grouping into cells was used for normalization. This results in a 637 dimensional descriptor for the line segment.

Approximate descriptor matching was done using 4 kd-trees inside FLANN [19] framework. Homography estimation parameters were set to $T_c = 20$; $T_\varphi = \pi/8$; $T_l = 0.5$; $\alpha = 50$; $\beta = 20$, and score parameter $\lambda = 0.75$. For all descriptors comparison we used the standard Euclidean distance. Experiments with other distance measures including Bhattacharya's $d(x,y) = 1 - \sqrt{xy}$ [27] and χ^2 distance $d(x,y) = 1 - 2xy/(x+y)$, have shown no gain in performance. All result curves are produced by varying the T_{obj} parameter. Results of other methods used for comparison are taken from their respective publications.

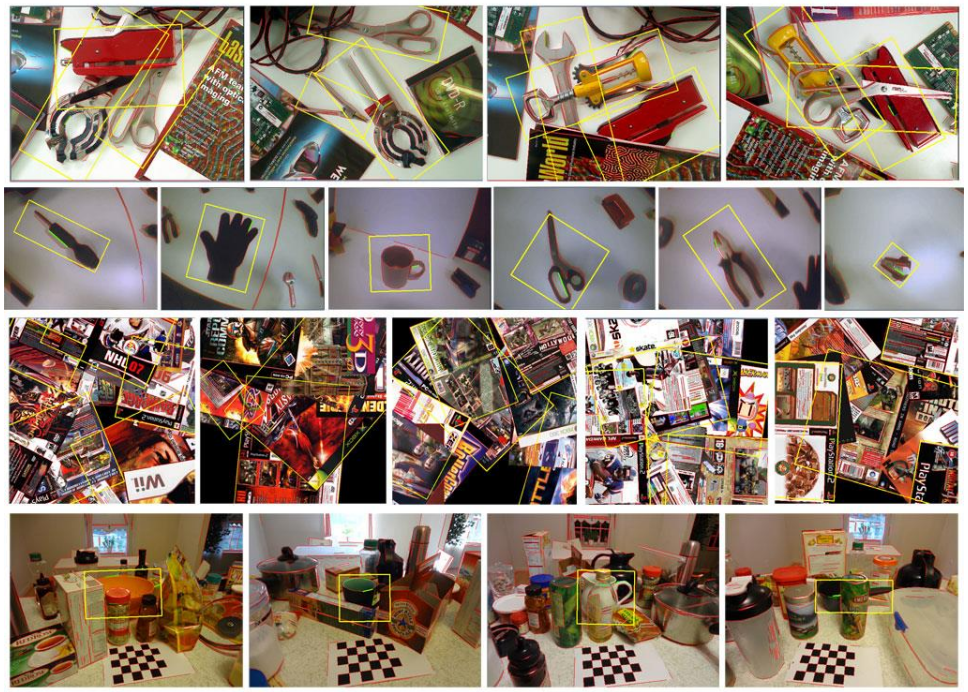

Fig. 5 Preview of our LineHOG detection results on four test datasets. First row shows results on D-Textureless [9], second row on Obj30 [10], third row on Caltech Covers subset [9], and fourth row on CMU-KO8 Multivew [13] dataset. Note that in the Obj30 dataset only one central object should be detected, while in the Caltech Covers not all covers in the test image are presented in the database.

5.2. Results on different datasets

D-Textureless dataset: We first evaluated the algorithm on the D-Textureless dataset [9]. The dataset contains 9 textureless objects, and 55 query images with moderate clutter and occlusions. We were detecting all 9 objects simultaneously in each scene. The ROC curve presented in Fig. 6 shows that LineHOG outperforms all other methods, including methods specifically designed for textureless objects (BOLD, LINE-2D and Halcon). As expected, these methods get significantly better results than SIFT, SURF and ORB. A reason for lower performance of LINE-2D and Halcon methods lies in their limitation for occlusions handling.

Obj30 dataset: We also evaluated our method on the Obj30 dataset proposed by Damen [10], which is characterized by a large variety of shapes and a low image quality. It contains 30 texture-less objects recorded by a hand-mounted camera, with 40-50 views per object for training. For testing, 1220 images are used, where each scene contains one dominant object from the database. To be able to compare our method with reported results, we search for all 30 objects and detect the object yielding the best score. Since other authors reported precision at 50% recall, we gave comparison of the same measure in Table II. Other results in Table II are taken from [8]. All compared methods except our

LineHOG and Damen's are template-based and generate thousands of training templates that restrict their scalability. While Damen's method is scalable to some extent, its precision is inferior to our results.

Table 2 Comparison of results on *Obj30* dataset [10].

Method	Precision at 50% recall
LineHOG (our)	92%
DTT-OPT [8]	90%
SelEdge [11]	86%
LINE-2D [6]	80%

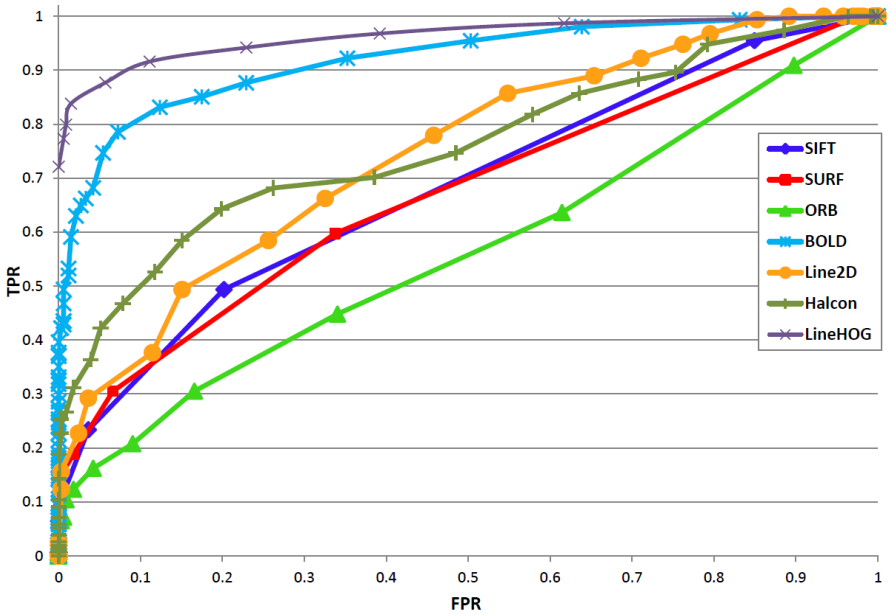

Fig. 6 Comparasion of results on *D-Textureless* dataset. Note the advantage of our method at lower FPR values, where TPR of other methods drops drastically, while we retain TPR>0.72, even for FPR = 0.

Textured Caltech Covers dataset: To test LineHOG performance for textured object detection in a heavily cluttered environment with significant occlusions, we used a subset of Caltech Game Covers dataset provided by [9]. This dataset contains 80 objects and 50 query images synthetically built to simulate clutter and occlusion up to 90%. During tests, we were looking for all 80 models in each query image. Fig.7 shows that our method achieves comparable performance to that of SIFT and SURF, although these methods should be dominant when dealing with textured objects.

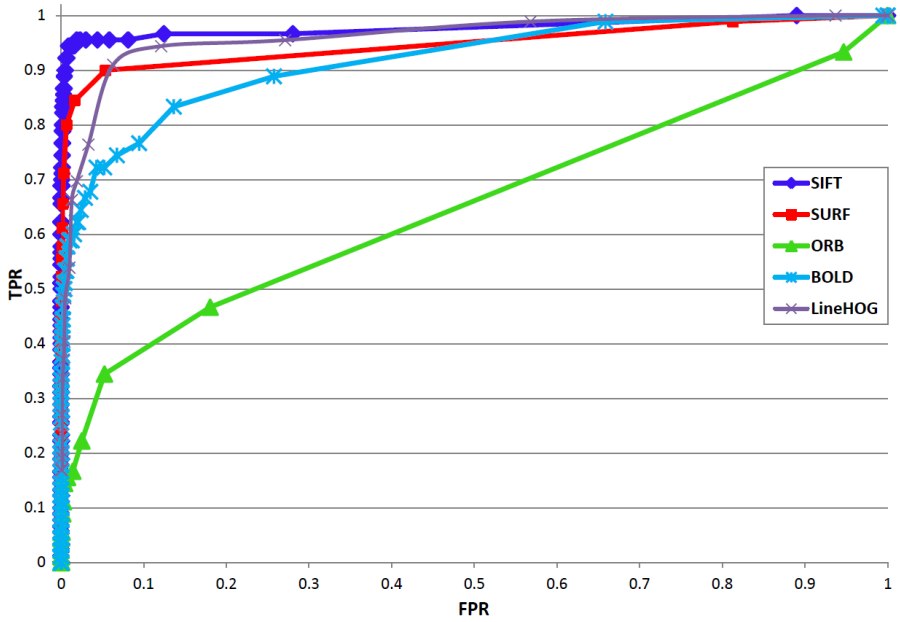

Fig. 7 Comparasion of results on *Caltech Covers* dataset of highly textured objects under large clutter and occlusion.

CMU Kitchen Occlusion dataset: We complemented our evaluation by testing on the extremely challenging CMU Kitchen Occlusion dataset [13], consisting of 8 texture-less household objects in real cluttered environments under high level of occlusion. This dataset is divided into two subsets intended to test object detection algorithms trained with, respectively, a single view and multiple views of an object. Since in practice, one would only detect objects under multiple views, we evaluated our algorithm only on this subset denoted as CMU-KO8 Multi. Following the experimental protocol proposed in [13], we present the results in terms of Recall versus False Positives Per Image (*fppi*) averaged across all 8 objects, where an object is correctly detected if the Intersection-over-Union (IoU) of the predicted bounding box and the ground-truth one is greater than 0.5. LineHOG results are compared to those reported in [13] and [9]. From Fig. 8 it can be observed that LineHOG outperforms other general methods that operate without any a-priori knowledge of the scene (LINE-2D, BOLD). It can be observed that for lower *fppi* values LineHOG delivers dominant results compared to all other methods including template based methods tuned for occlusion reasoning [13].

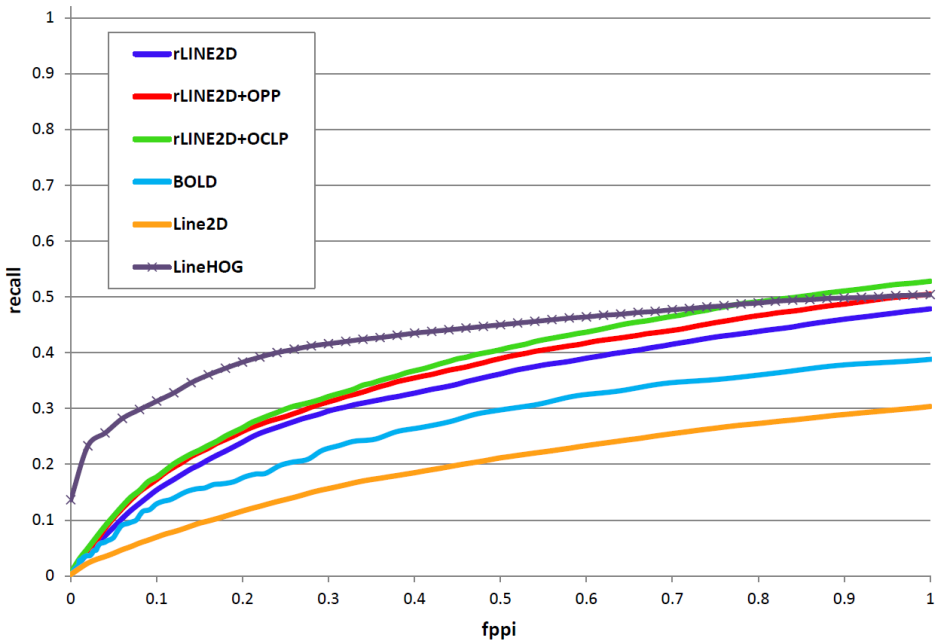

Fig. 8 Average detection rate (recall) on challenging *CMU-KO8 Multiview* dataset.

5.3. Impact of different object detection techniques

To analyze the importance of several techniques described in Section 4, we tested their impact on overall detection results. Specifically we measured impact of following techniques: Weighted Bipartite Matching (WBM), compensating the bias toward simpler objects (equation 7), and inlier banning. In Table 3 we report results that show the crucial importance of these techniques for reaching high quality results. It can be observed that all presented techniques consistently improve results on all datasets.

Table 3 Mean Average Precision (MAP) over 4 datasets, showing the impact of different object detection techniques. (a): The full proposed LineHOG method. (b): LineHOG method without biasing towards simpler object ($\lambda = 1$). (c): Represents (b) where WBM is replaced with (5). (d): Represents (c) with naive approach to multiple object reasoning (i.e. without inlier banning).

Dataset	(a)	(b)	(b)	(d)
D-Textureless [9]	0.94	0.911	0.894	0.882
Obj30 [10]	0.705	0.668	0.644	0.644
Caltech Covers [9]	0.611	0.591	0.542	0.416
CMU-KO8 Multiview [13]	0.443	0.397	0.360	0.359

5.4. Runtime performances

Finally we give runtime performances of the implemented algorithm in Table 4. The implementation was done in C++ utilizing OpenCV, and all tests were run on a 3.6 GHz Intel Core i7 computer. We made only straightforward multi-threaded parallelization of the HOG-like descriptor, assuming that an implementation on GPU would significantly improve time performances. Note that the runtime is directly dependent on the number of extracted line segments in the test image, while being independent of the number of objects in the database (as described in Section 4).

Table 4 Execution time of our algorithm depending on the average number of extracted line segments per image (including all scales).

Dataset	num. scales	num. lines	runtime
D-Textureless [9]	2	633	156 ms
Obj30 [10]	5	214	118 ms
Caltech Covers [9]	2	2446	1157 ms
CMU-KO8 Multiview [13]	2	952	216 ms

6. Conclusion

We presented a method for object instance detection that is generic, scalable, fast and highly accurate. The key results we achieved are scalable object detection, high accuracy for both textured and texture-less objects, and robustness to clutter and occlusion. We noticed several possible improvements in the line segment extraction phase that could make it more resistant to blur, shadows and scale changes. We also plan to include an object verification step that will compare a global color gradient descriptor over the region delimited by all inliers in the query image. It allows fast rejection of false positive object detections and improves overall results.

Acknowledgement: *The research presented in this paper is financed by the Ministry of Education, Science and Technological Development of the Republic of Serbia under the project 43012.*

References

[1] D. Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.

[2] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in Computer Vision-ECCV 2006, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3951, pp. 404–417, 2006.

[3] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in Proc. ECCV, pp. 778–792, 2010.

[4] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2564–2571, 2011.

[5] C. A. Luna, C. Losada-Gutierrez, D. Fuentes-Jimenez, A. Fernandez-Rincon, M. Mazo, and J. Macias-Guarasa, “Robust people detection using depth information from an overhead time-of-flight camera,” Expert Systems with Applications, vol. 71, pp. 240 – 256, 2017.

[6] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 5, pp. 876–888, May 2012.
[7] M. Ulrich, C. Wiedemann, and C. Steger, "Combining scale-space and similarity-based aspect graphs for fast 3d object recognition," PAMI, vol. 34, no. 10, pp. 1902–1914, 2012.
[8] R. Rios-Cabrera and T. Tuytelaars, "Discriminatively trained templates for 3d object detection: A real time scalable approach," in IEEE International Conference on Computer Vision (ICCV), 2013, pp. 2048–2055, 2013.
[9] F. Tombari, A. Franchi, and L. Di Stefano, "Bold features to detect texture-less objects," in IEEE International Conference on Computer Vision (ICCV), 2013, pp. 1265–1272, 2013.
[10] D. Damen, P. Bunnun, A. Calway, and W. Mayol-Cuevas, "Real-time learning and detection of 3d texture-less objects: A scalable approach," in Proceedings of the British Machine Vision Conference, pp. 23.1–23.12, 2012.
[11] H. Cai, T. Werner, and J. Matas, "Fast Detection of Multiple Texture-less 3-D Objects," in International Conference on Computer Vision System, 2013.
[12] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for real-time detection of texture-less objects," in EEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, June 2010, pp. 2257–2264, 2010.
[13] E. Hsiao and M. Hebert, "Occlusion reasoning for object detection under arbitrary viewpoint," in Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), ser. CVPR '12, Washington, DC, USA, 2012, pp. 3146–3153, 2012.
[14] J. Shotton, A. Blake, and R. Cipolla, "Contour-based learning for object detection," in Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005., vol. 1, pp. 503–510 Vol. 1, 2005.
[15] C. Akinlar and C. Topal, "Edlines: A real-time line segment detector with a false detection control," Pattern Recognition Letters, vol. 32, no. 13, pp. 1633 – 1642, 2011.
[16] S. Cvetković, M. B. Stojanović, S. V. Nikolić, "Multi-channel descriptors and ensemble of Extreme Learning Machines for classification of remote sensing images," Signal Processing: Image Communication, vol. 39, pp.111-120, 2015.
[17] S. Cvetković , M. B. Stojanović, S. V. Nikolić, "Hierarchical ELM ensembles for visual descriptor fusion," Information Fusion, Elsevier, vol. 41, pp. 16-24, 2018.
[18] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: a Line Segment Detector," Image Processing On Line, vol. 2, pp. 35–55, 2012.
[19] R. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 32, no. 4, pp. 722–732, April 2010.
[20] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01, ser. CVPR '05. Washington, DC, USA, pp. 886–893, 2005.
[21] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in International Conference on Computer Vision Theory and Application VISSAPP'09, pp. 331–340, 2009.
[22] R. I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.
[23] H. Kuhn, "The hungarian method for the assignment problem," in 50 Years of Integer Programming 1958-2008, M. Junger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, Eds. Springer Berlin Heidelberg, pp. 29–47, 2010.
[24] D. Lowe, "Object recognition from local scale-invariant features," in IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157, 1999.
[25] D. Lowe, "Local feature view clustering for 3d object recognition," in IEEE Conference on Computer Vision and Pattern Recognition 2001, pp. 682–688, 2001.
[26] D. Ristic-Durrant, G. Gao, and A. Leu. "Low-level sensor fusion-based human tracking for mobile robot." Facta Universitatis, Series: Automatic Control and Robotics, pp. 17–32, 2016.
[27] R. Arandjelovic and A. Zisserman, "Three things everyone should know to improve object retrieval," in IEEE Computer Vision and Pattern Recognition, 2012.