Object Detection using Categorised 3D Edges

Lilita Kiforenko, Anders Glent Buch, Leon Bodenhagen and Norbert Krüger

Maersk Mc-Kinney Moller Institute, University of Southern Denmark

ABSTRACT

In this paper we present an object detection method that uses edge categorisation in combination with a local multi-modal histogram descriptor, all based on RGB-D data. Our target application is robust detection and pose estimation of known objects. We propose to apply a recently introduced edge categorisation algorithm for describing objects in terms of its different edge types. Relying on edge information allow our system to deal with objects with little or no texture or surface variation. We show that edge categorisation improves matching performance due to the higher level of discrimination, which is made possible by the explicit use of edge categories in the feature descriptor. We quantitatively compare our approach with the state-of-the-art template based Linemod method, which also provides an effective way of dealing with texture-less objects, tests were performed on our own object dataset. Our results show that detection based on edge local multi-modal histogram descriptor outperforms Linemod with a significantly smaller amount of templates.

Keywords: Object detection, pose estimation, edge detection

1. INTRODUCTION

We address the problem of 3D object detection and pose estimation for objects containing little texture and surface variation, a scenario often encountered in industrial settings, but also frequently in everyday environments (see Fig. 1). Many works on object detection are based on appearance or surface information from which distinctive image or shape feature descriptors are extracted, e.g. SIFT [1] or SHOT [2]. The disadvantage of those methods is that the object must have rich texture or contain surface variations, which can be captured in a local descriptor. On the other hand, objects with uniform

Further author information: E-mail: {lilita, anbu, lebo, norbert}@mmmi.sdu.dk

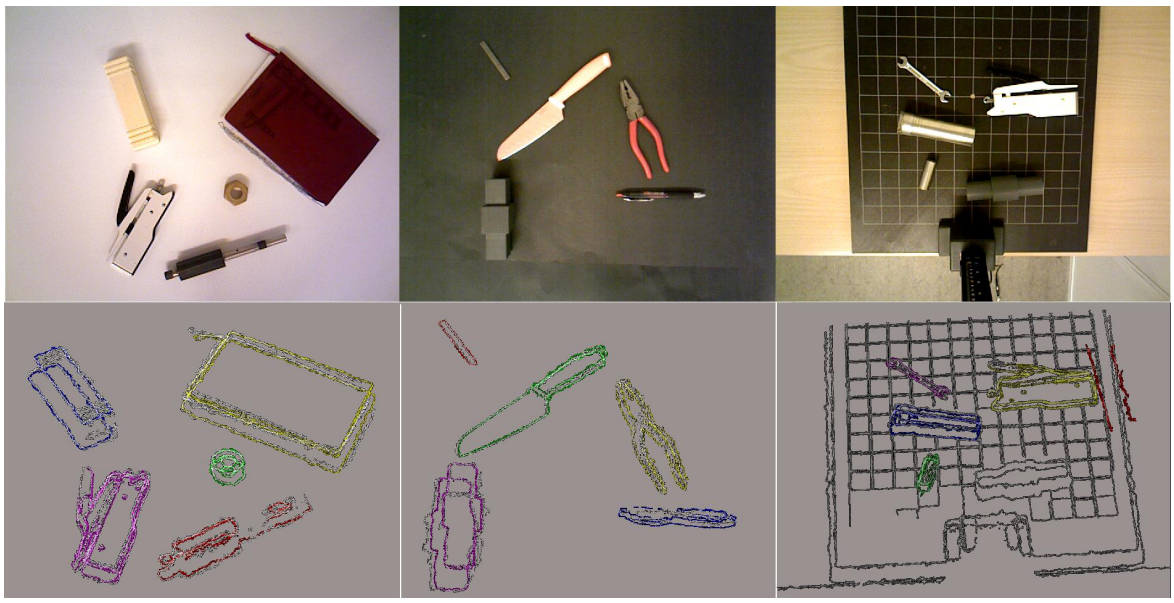

Figure 1. Example scenes used in our experiments. Top: different object types and varying backgrounds. Bottom: pose estimation results using all edges (categorised) for the top scenes.

Seventh International Conference on Machine Vision (ICMV 2014), edited by Antanas Verikas, Branislav Vuksanovic, Petia Radeva, Jianhong Zhou, Proc. of SPIE Vol. 9445, 94450C

colour and/or surfaces are less likely to be handled well by such features, which poses great problems to such detection systems.

A common approach for describing texture-less objects is to use edge information. Edge information can be extracted from almost any kind of object (with/without texture, uniform, etc.) and—being sparse—edges allow for fast computation.

Edges arise from different properties and effects: colour and orientation discontinuities, textures, occlusions, borders etc. Using RGB-D observations, these different edge types can be reliably detected, and, as we show in Sec. 5, this information can be exploited to improve the description of an object used for detection.

In this paper, we use an existing edge detection and categorisation algorithm [3] for extracting edges from the RGB-D data. Particular for this approach is that a predefined set of edge categories are extracted, namely *boundary*, *occluding*, *occluded*, *high-curvature* and *rgb* (Canny [4]) edges (see Fig. 2). Based on this categorised edge information we create a local multi-modal edge descriptor using the 3D structure in the vicinity of an edge. We use distance and angle relations between an edge point and all neighbouring edge points, separated by category. Additionally we include a histogram with relations to neighbouring surface points in order to avoid losing information. We note that both the edge extraction method and our feature descriptor still require surface information (in the form of depth images) to be available; the distinction between our system and e.g. shape-based or template-based systems is that the surface information does not need to be discriminative, since the surface relations only represent a small part of our descriptor. We use our histogram descriptors for feature matching, followed by a RANSAC-based pose estimation algorithm to achieve robust detection results.

We evaluate the performance of our system against baseline edge-based methods, and a state-of-the-art multi-modal method for detecting texture-less objects, namely Linemod [5]. For our evaluations, we present a challenging dataset consisting of 15 objects and 225 test scenes. All scenes contain multiple objects in random configurations, and each object instance is annotated with a ground truth bounding box. We have carefully chosen the object set to represent a high variation of appearances, but also with many ambiguities and non-discriminative (uniform) surfaces. Examples of scenes are shown in Fig. 1.

In the remainder of the paper, we describe related work (Sec. 2), the proposed approach in details (Sec. 3) and show results by performing multiple quantitative experiments (Sec. 5), before drawing conclusions (Sec. 6).

2. RELATED WORK

Successful work has been done for the detection and—in some cases—pose estimation of objects using edge and/or surface information. In our comparisons we focus on feature- and template-based methods, and below we provide a non-exhaustive overview of recent literature.

In [6] a scale-invariant interest point detector and descriptor based on image contours is presented. In the detection stage, the descriptors of a test image is matched against a library of features computed in the training stage, and the final detection is inferred by mode seeking in the 2D scale space. Shotton et al. [7] further argued for the use of contours for multiclass visual object recognition using a sliding window detector. Based on a limited number of object templates, Liu et al. [8] uses an oriented chamfer matching in combination with a sliding window detector for detecting objects in images. In [9], a path that defines the relative direction between edge constellations is used as a primitive feature for indexing into a hierarchical hash table, which allows for recovering the homography of a training view. Tombari et al. [10] combine edge points into line segments and use geometric relationships between neighboring segments to create a local multi-scale descriptor for matching a single training view of an object with novel scenes. All of these methods are purely image-based, which allows for fast detections, but without obtaining the 6D pose of an object.

Another state of the art work that uses both image gradient and depth information for template matching in RGB-D images is Linemod [5]. This work shows great performance in recognition and speed. It also shows that usage of image data with depth information produces better results than image information alone. The disadvantage of the method is a high amount of required templates. Even though they propose a way to extract these templates from a CAD model [11], it is still rather complicated to acquire templates.

The method presented in this paper uses local histogram descriptors based on 3D edge features, unlike many of the before mentioned methods which use image edges, line segments or contours. Similar to Linemod, we rely on RGB-D inputs to describe the object. However, where Linemod uses image gradients and surface normals, we use a combination of reconstructed 3D edges (both point and direction information) and surface normals in the vicinity of an edge.

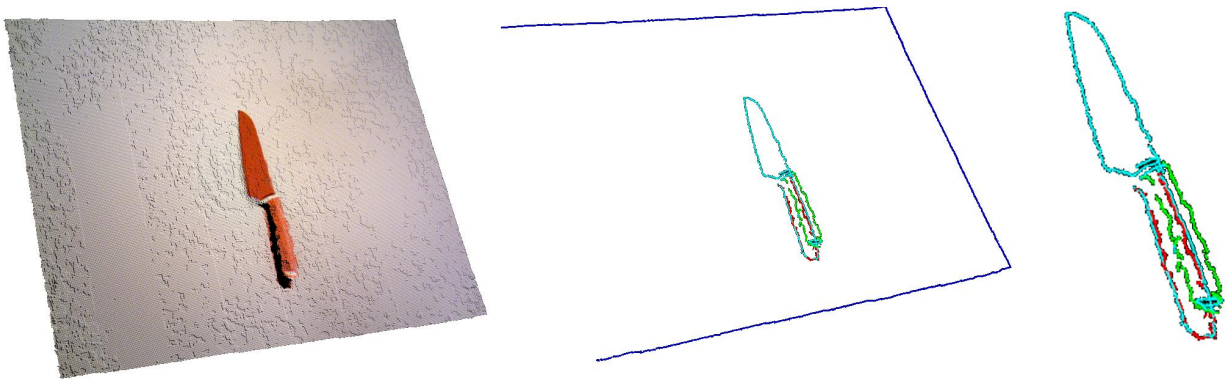

Figure 2. Edge detection example for a training view of a knife. Blue - represent *boundary* edges, green - *occluding*, red - *occluded* and cyan - *rgb* edges.

3. PROPOSED METHOD

The method proposed here uses edge feature descriptors to establish correspondences between a *query* and a *target* set, where the query consists of a training RGB-D view of an object, and the target is a novel scene captured by the sensor. For each view we perform two steps: i) extract 3D edges (Sec. 3.1) and ii) compute distance and angle histograms (Sec. 3.2). After that we compute correspondences by approximate nearest neighbor search [12] between query and target (Sec. 3.3), and finally we compute the object pose using a PCL [13] implementation of a RANSAC-based [14] correspondence rejection algorithm (Sec. 3.4). Finally, we refine the result using ICP [15] to get a more accurate pose estimate. If we use more than one object template, then we compute poses for all of templates and select the best pose by evaluating the amount of inliers that support the model. We describe the steps in details below.

3.1 3D Edge Extraction

For the extraction of edge information, we use the approach presented by Choi et al. [3] As input it uses RGB-D Kinect-like organized point clouds and it outputs 5 different edge types as separate point clouds: *boundary, occluding, occluded, high curvature* and *rgb* edges. The *occluding* and *occluded* edges are computed from depth discontinuities, where *occluding* points are on the foreground and *occluded* on the background. The *high curvature* edges are where the normals rapidly change and *rgb* edges are computed using the Canny edge detection algorithm [4]. The *boundary* edge represents the scene outer boundary. For training an object, we use a scene where the training object appears on a uniform background. Therefore the *boundary* edge is not used in our method, because it is not related to the object representation. An example of extracted edge types is shown in Fig. 2. For edge extraction we used the default parameters, except we changed threshold values for *rgb* edges by setting the low and high Canny thresholds to 30 and 70, respectively, in order to detect objects with non strong edge, such as *Ring* and *Magnet*.

We extend the edge extraction method by adding 3D edge directions to the categorised edge point clouds. We compute the direction of an edge point using the 2D image gradients $\begin{bmatrix} g_x & g_y \end{bmatrix}^T$ and the surface normals, which can be reliably estimated using the eigenvalue decomposition of the local scatter matrix of the surface points around an edge point. If we denote an edge point as p, its 3D surface normal as n and its in-plane 3D gradient as g, where:

$$g = \begin{bmatrix} g_x & g_y & 0 \end{bmatrix}^T \tag{1}$$

then the 3D edge direction d can be computed by the following cross products:

$$d = n \times (g \times n) \tag{2}$$

3.2 Distance and Angle Histograms

For each point in the edge map we compute a histogram descriptor based on both distance and direction information. That histogram contains a concatenation of 10 subhistograms, where the first 5 are reserved for distance relations and last 5

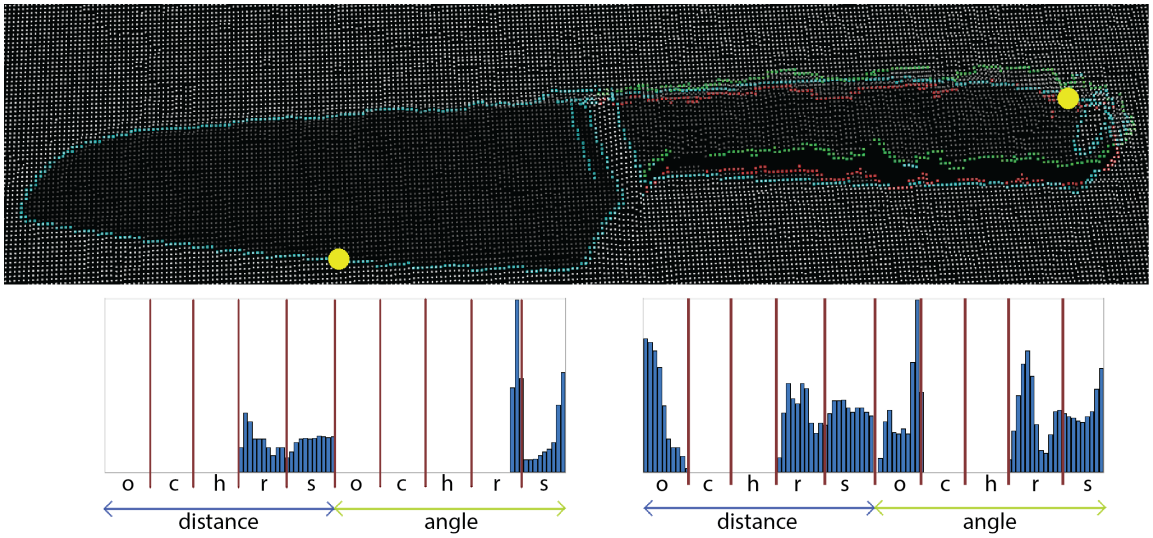

Figure 3. Two knife edge points histogram example. Symbol *o* corresponds to *occluding* edges, *c - occluded*, *h - high curvature*, *r - rgb*, *s - surface*.

for angle relations. For each edge point we use all points in a spherical neighbourhood of radius r to build the histogram. The radius r value was set empirically to 1 cm for a good trade-off between locality (low radius) and descriptiveness (high radius). Within the neighbourhood of an edge, up to 4 edge types are found, depending on the structure of the object (avoiding the *boundary* category). However, the surface points in the point cloud, not part of the edges, can also provide valuable information. Our initial tests have shown that including a subhistogram with relations to surface points improves performance. The *surface* type thus provides the 5th entry in our distance and angle subhistograms described below. See also Fig. 3.

For the distance relations we use the Euclidean distance from the edge point to each of its neighbours and bin the resulting distances into histograms. We use 5 histograms, where each histogram corresponds to the distance from the edge point considered to its neighbour of each type in the following order: *occluding*, *occluded*, *high curvature*, *rgb*, *surface*. Specifically, the direction information of a *surface* point is given by the normal vector n.

For the angle relations we have the same procedure as for the distance, but instead of computing Euclidean distances, we compute the absolute dot product between the edge point direction and the direction vector of each neighbour. We use absolute dot products due to the ambiguity in determining a consistent direction for edges, which are only defined up to a sign.

The choice of the number of histogram bins is a trade-off between high specificity (many bins), and robustness to noise (few bins). We have empirically found a good compromise with a total of 100 bins. We therefore use 10 bins per edge type, both for the distance and the angle part. Before concatenation, all 10-dimensional subhistograms are smoothed by a triangular kernel with 3 samples to diminish border effects.

3.3 Feature Correspondences

After computing histograms for each edge point in query and target, we search for nearest neighbor feature correspondences using our histogram descriptors. This process is prone to noise, which causes outlier matches. We use the semantic edge category labels to bound the search, so, e.g., *occluding* query edges are only matched against the *occluding* edges in the target, and so on for the other categories. This has positive effects on the feature matching, making the correspondence output more robust. For instance, the *Ring*, which has only *rgb* edges, is only matched to that type of edge features in a scene. This helps avoiding a large number of potential mismatches in irrelevant regions of the scene.

3.4 Pose Estimation

Having found feature correspondences, we use them in a RANSAC pose estimation algorithm. For this purpose we use the PCL library. It implements the original RANSAC, by sampling 3 random correspondences, calculating a transformation

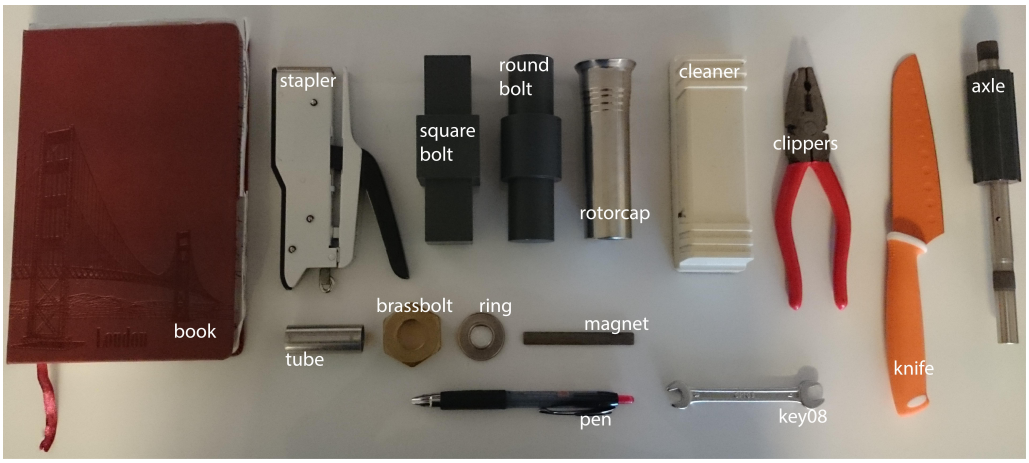

Figure 4. The 15 objects used in our evaluation dataset.

and computing the amount of inliers supporting the model. The point is counted as an inlier if, by transforming it, the resulting point is close than a defined distance threshold to its corresponding point. To improve inaccurate pose estimates due to either point noise or outliers, we use the ICP algorithm to achieve a more accurate pose.

4. EVALUATION DATASET

Our application is the detection of challenging objects with a small degree of appearance and surface variations, a problem often encountered in industrial applications. For this purpose, we introduce our own challenging RGB-D dataset for all our evaluations. To our knowledge there is no database that provides these data. For our dataset we have picked 15 objects to represent both a large variety in appearance and shape, but also with a significant amount of ambiguities and repeated structures. All the objects are shown in Fig. 4. For instance, the two black bolts (*Square bolt* and *Round bolt*) share both appearance and to some degree shape, while the *Brassbolt* has an edge profile similar to the *Ring*.

Using these objects, we have generated all in all 225 test scenes in a systematic way. We pick 5 objects and place them in random configurations on a table 3 times. We do this for 3 different backgrounds: white, black, and a more difficult chessboard background (see Fig. 1). For one distinct set of 5 objects, we thus get 15 test scenes, 5 with each background type. We now replace one of the 5 objects with a new object, and repeat the procedure until we have used all objects. This way, each object appears in $5 \cdot 15 = 75$ scenes with varying background and cluttering objects. For all test scenes, we have manually annotated each object instance with a ground truth bounding box in the RGB image. We have chosen this rather than generating ground truth 6D poses, since many objects contain symmetries, giving an infinite amount of correct poses. Additionally, resolving the rotation axis/axes of the objects is not possible, since we provide captured training views where the intrinsic object coordinate frames are not known. For immediate comparisons, the Linemod implementation also outputs a bounding box in 2D coordinates.

All training views and test scenes were recorded using the Primesense Carmine 1.09 sensor. When using PCL functionality in combination with this sensor, organized point clouds are readily captured, in a similar way as the Microsoft Kinect for Windows. We compared the results obtained using the Carmine sensor and the Kinect. Carmine, having a stronger IR emitter, showed much better performance for objects with reflective surfaces, where Kinect was not able to capture any depth data.

4.1 Training

Regarding the training phase, there are fundamental differences between the way this should be done for our method and for Linemod, which we will compare against. In the best case, Linemod should be provided many templates, covering all viewpoints around the object as densely as possible. However, as all our test scenes are captured from a fixed distance above the table, we can significantly limit the template acquisition for Linemod. For the training, we thus place the object on the table, and perform small incremental in-plane rotations of the object, covering a high number of rotations. Many

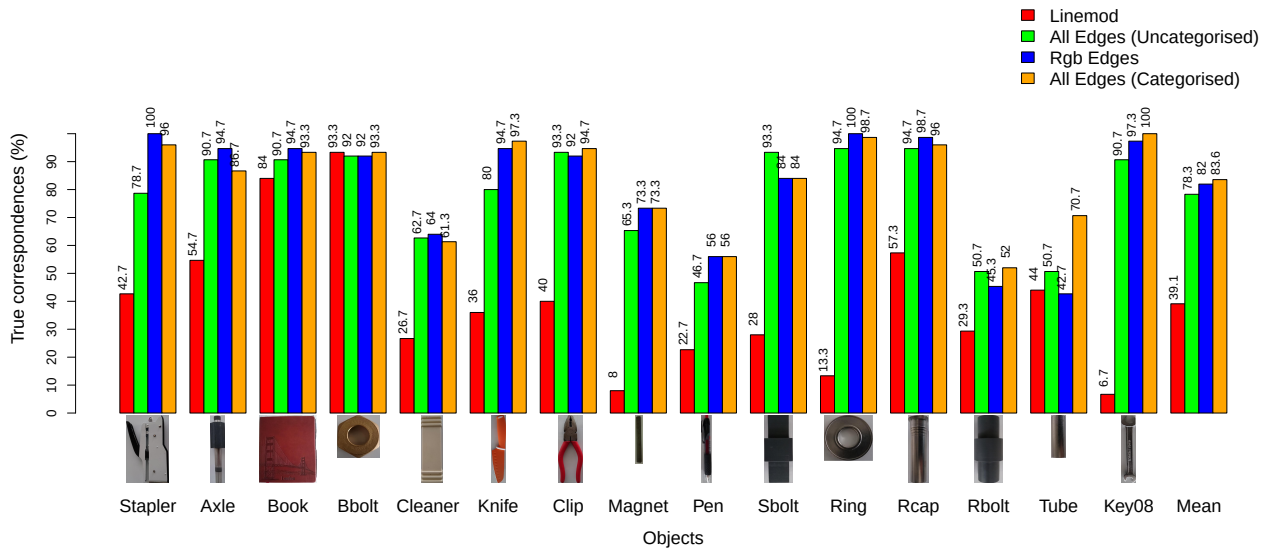

Figure 5. True positive detection rates for Linemod (red bars) and our method using all edges (uncategorised) (green bars), all edges (categorised) (orange bars) and only *rgb* edges (blue bars) for the 225 test scenes in our dataset. *Bbolt* stands for *Brassbolt*, *Clip* for *Clippers*, *Rbolt* for *Round bolt*, *Rcap* for *Rotorcap* and *Sbolt* for *Square bolt*.

of the objects in our set have rotational symmetries, so in these cases we can use fewer templates (10 for the *Ring* which is ∞-fold rotationally symmetric). The maximal number of templates (43) is generated for the *Stapler* object. For our method, we use much fewer training views with the object placed on the table in a canonical position, from which we generate our object representation in terms of edge histograms. The objects used in our dataset are shown in Fig. 4. The simplicity of the training procedure highlights the advantage of using invariant 3D features, and even with one template, we achieve superior detection performance, as we will demonstrate in the following sections.

5. EXPERIMENTS

In this section, we describe the object detection results on our recorded dataset. We perform the quantitative comparison between our method using all edges with/without categorisation, using only *rgb* edges and using Linemod. For all of our methods, we use the same number of bins (10 bins per edge type) and include the *surface* type histogram in the end. For *rgb* edges, we look only into *rgb* edges, discarding all other types. For all edges (uncategorised), we use all extracted edges, but do not store them into separate histogram bins, and total the bin size is the same as the total number of bins used for all edges with categorisation. For Linemod we use the available OpenCV implementation. In the end of the section we also show the performance of our method using up to 10 templates.

We run our detector and the Linemod detector on all scenes in our dataset. For validating a Linemod detection, we require that the bounding box of the detected template overlaps at least 50 % with the ground truth bounding box by the union over intersection criterion. For our own method, which generates a full 6D pose of the edges of the training view, we apply the resulting pose to the training edges, project the bounding volume of the aligned edges to the test image of the scene, and then use the same overlap criterion as for Linemod. For this test we used 4 templates per object. We chose the number 4 to avoid noisy and broken templates. We are working in an edge domain and edges are not very stable and depend on many things (illumination, reflection, etc.).

The results of this experiment are shown in Fig. 5. We choose to show the true positive rate or recall, since both our method and Linemod outputs at most 1 detection per object per scene. There are thus very few false positives, and the recall thus makes it easier to assess relative performances. The results of this figure clearly show that our method is much better at recovering the instances of the objects in the scenes compared to Linemod. Some objects with very little surface

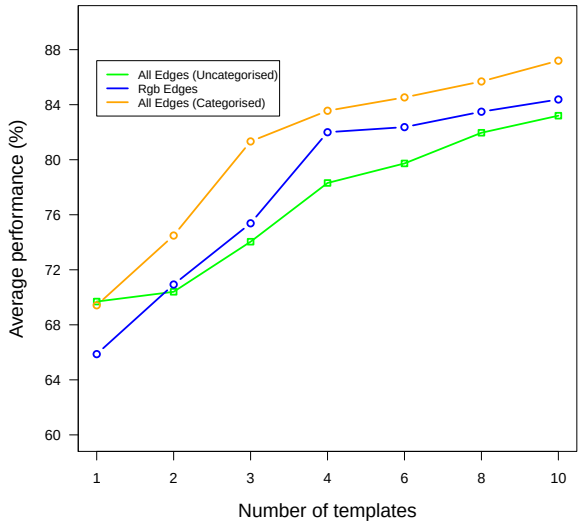

Figure 6. Performance comparison using different amount of templates.

variation (*Magnet* and *Key08*) are hardly detectable by Linemod, while our method shows very high recall. The recognition rates for *Pen* and *Round bolt* are smaller. Possible explanations for this are similar shapes of *Rotorcap*, *Round bolt* and *Square bolt* and missing depth values for *Pen*. The results show that *rgb* (Canny) edges are the most important edge for this dataset. If we use just all edges without categorisation we still get a high performance, but it is better if we use the labelling of edges.

Fig. 6 shows the performance of our method using from 1 template up to 10 templates per object. We evaluate on the same dataset as before. The results show an improvement of more than 10 % if we use more then 3 templates per object. The additional number of views allows for a higher robustness towards noisy edge points, which then leads to a better matching quality.

6. CONCLUSIONS

We have presented a new method for object detection that uses edge categorisation in combination with a local multi-modal histogram descriptor. For a thorough evaluation, we have generated a large dataset consisting of 15 objects of varying appearances and shapes and 225 test scenes for evaluating detection performance. The average object recognition rate of our method on our dataset is 83.6 % using all edges (categorised), 82 % using only *rgb* edges and 78.3 % using all edges (uncategorised). Our methods significantly outperforms Linemod, which average recognition rate is 39.11 %.

The results show that the algorithm based on edge information introduced in this paper — at least for the object classes presented here — outperforms significantly a state-of-the-art method based on template information. The difference in performance is expected to be lower for object classes in which texture is more prominent. However, our results show the general importance of edge information for the discrimination of certain object classes. Moreover, we show in our paper that the use of a categorisation of edges in the matching process improves performance compared to a non-categorised use. Canny edges turned out to be the most important individual edge category for the data set used in this paper, but we expect for objects with a more varying depth profile an even higher difference in performance between the different variants of the edge based algorithm we have introduced here. This we plan to investigate in our future research.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Communitys Seventh Framework Programme FP7/2007-2013 (Programme and Theme: ICT-2011.2.1, Cognitive Systems and Robotics) under grant agreement no. 600578, ACAT and by Danish Agency for Science, Technology and Innovation, project CARMEN.

REFERENCES

[1] Lowe, D. G., "Distinctive image features from scale-invariant keypoints," *International journal of computer vision* **60**(2), 91–110 (2004).

[2] Tombari, F., Salti, S., and Di Stefano, L., "Unique signatures of histograms for local surface description," in [*Computer Vision–ECCV 2010*], 356–369, Springer (2010).

[3] Choi, C., Trevor, A. J., and Christensen, H. I., "Rgb-d edge detection and edge-based registration," in [*Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*], 1568–1575, IEEE (2013).

[4] Canny, J., "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **PAMI-8**(6), 679–698 (1986).

[5] Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V., "Gradient response maps for real-time detection of textureless objects," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **34**(5), 876–888 (2012).

[6] Jurie, F. and Schmid, C., "Scale-invariant shape features for recognition of object categories," in [*Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*], **2**, II–90, IEEE (2004).

[7] Shotton, J., Blake, A., and Cipolla, R., "Multiscale categorical object recognition using contour fragments," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **30**(7), 1270–1281 (2008).

[8] Liu, M.-Y., Tuzel, O., Veeraraghavan, A., and Chellappa, R., "Fast directional chamfer matching," in [*Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*], 1696–1703, IEEE (2010).

[9] Damen, D., Bunnun, P., Calway, A., and Mayol-Cuevas, W. W., "Real-time learning and detection of 3d texture-less objects: A scalable approach.," in [*BMVC*], 1–12 (2012).

[10] Tombari, F., Franchi, A., and Stefano, L. D., "Bold features to detect texture-less objects," in [*Computer Vision (ICCV), 2013 IEEE International Conference on*], (2013).

[11] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N., "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in [*Computer Vision–ACCV 2012*], 548–562, Springer (2012).

[12] Muja, M. and Lowe, D. G., "Fast approximate nearest neighbors with automatic algorithm configuration.," in [*VISAPP (1)*], 331–340 (2009).

[13] Rusu, R. B. and Cousins, S., "3d is here: Point cloud library (pcl)," in [*Robotics and Automation (ICRA), 2011 IEEE International Conference on*], 1–4, IEEE (2011).

[14] Fischler, M. A. and Bolles, R. C., "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM* **24**(6), 381–395 (1981).

[15] Besl, P. and McKay, N. D., "A method for registration of 3-d shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **14**, 239–256 (Feb 1992).