Real-time Learning and Detection of 3D Texture-less Objects: A Scalable Approach

Dima Damen[1]
damen@cs.bris.ac.uk
Pished Bunnun[2]
pished.bunnun@nectec.or.th
Andrew Calway[1]
andrew@cs.bris.ac.uk
Walterio Mayol-Cuevas[1]
wmayol@cs.bris.ac.uk

[1] University of Bristol
Bristol, UK
[2] National Electronics and Computer Technology Center
Bangkok, Thailand

Abstract

We present a method for the learning and detection of multiple rigid texture-less 3D objects intended to operate at frame rate speeds for video input. The method is geared for fast and scalable learning and detection by combining tractable extraction of edgelet constellations with library lookup based on rotation- and scale-invariant descriptors. The approach learns object views in real-time, and is generative - enabling more objects to be learnt without the need for re-training. During testing, a random sample of edgelet constellations is tested for the presence of known objects. We perform testing of single and multi-object detection on a 30 objects dataset showing detections of any of them within milliseconds from the object's visibility. The results show the scalability of the approach and its framerate performance.

1 Introduction

This paper concerns the detection of multiple rigid objects in live video streams. Our target application is the real-time analysis of workspaces, in which tools and components are first learnt and then are located under clutter and expected noise (Fig. 1). In most tasks, objects have little texture and adopt a wide range of 3D poses, thus the method we are targetting should be shape-based, occlusion-tolerant, scalable and fast - speed and performance should not degrade significantly as the number of objects being searched for increases.

Shape-based representations are based on edges or edge segments (edgelets). Edgelets are dense and quick to compute, and a constellation of edgelets characterises aspects of shape, either locally or globally, providing discrimination even in the presence of occlusion. A key and distinguishing element of the method is the use of *path tracing* for both training and testing. Each path defines the relative direction between the constellation's constituent edgelets. This introduction of paths is critical; as it limits the number of possible constellations and allows tractable generation of a library of descriptors. Note that our focus is on detection without tracking, i.e. we consider each frame independently. The method is tested on a dataset of 30 texture-less objects. It specifically trades recall for speed, testing a sample of edgelet constellations in each processed frame. At 7fps, recall of 50% (precision = 74%)

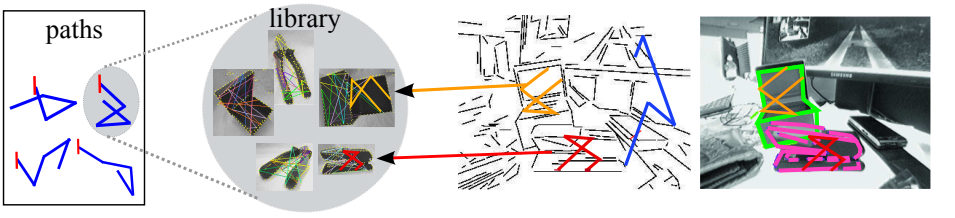


Figure 1: Spatial constellations of edgelets are traced out from the edge maps of reference views of multiple objects using defined paths, generating a library indexed by scale and rotation invariant descriptors. In test frames, sample edgelet constellations are traced out using the same paths to enable matching with views in the library.

was achieved when 30 objects were learnt (1433 views). This performance is suitable for intialising trackers for multiple objects in real-time. The method is generative and does not optimise the performance on a training set. It is not expected to outperform single-class classifiers, but for comparison, we also provide results on the ETHZ dataset showing acceptable performance.

2 Related Work

We first review methods aiming for shape-based detection of a single object or class **(I)**. Then, we review methods that target scalability (i.e. multi-class) **(II)**, and finally discuss works focusing on fast detection in video input **(III)**.

(I) There have been many previous approaches to shape matching. The approaches often extend beyond rigid objects to deformable categories. While textured-patches have proven successful in voting for object centres for planar [7, 11] as well as 3D objects [20, 23], edge segments are not discriminative enough for similar voting. Instead, neighbouring edge responses at various radii [2], contour fragments [19, 24] and non-adjacent edgelet constellations [5, 15] have been used. Selecting discriminative contour fragments that best distinguish a category of deformable objects was proposed in [19, 24] within a boosting framework. In [17], probabilistic weights of discriminative parts are learnt in a max-margin framework. In [3], discriminative selection of lines and ellipses results in a class-specific shape structure that can be used for detection in real-world images. In [5], consistent constellations of edgelets over examples of a deformable category are learned from weakly-labelled images. The most consistent pair of edgelets in the learnt model is selected as the 'aligning pair' and is exhaustively compared to all pairs of edgelets in the test image. Instead of a pair of edgelets, a fully connected clique was used in [15]. The clique is described using a highly-redundant descriptor and exhaustive search detects an object in the image by maximising the matching score between the learnt clique and test edgelets. Grouping edgelets into mutually-independent parts before voting for the object's centre, was proposed in [27] using iterative optimization. These discriminative voting approaches are state-of-the-art in single-view class detection. When multiple objects are being considered, detectors are applied in sequence resulting in linear scalability. Despite their recently-increasing focus on faster detection, discriminative learning requires offline processing, and the shape representations are not rotation- or scale-invariant. During detection, the approaches are tested on multiple scales. Rotation-variance remains acceptable for natural images where vehicles, animals and humans are standing upright.

(II) When targetting sub-linear scalability (often referred to as multi-class detection), hi-

erarchies of views or objects were proposed as the number of models increases. For views of a single 3D object, [26] organises the views into a hierarchy of shapes. The test image is searched for views in the top level, and branches are explored when a match is found. Using a fast similarity measure, an object is detected within 300-900ms. Building a hierarchy of detectors was also achieved in [10]. Arrangements of edge fragments are learnt in an unsupervised statistical manner from training images, and are later combined into a hierarchy of detectors. While the approach theoretically allows finding transformation-invariant parts, the rotation invariance was removed for tractable inference. The multi-class hierarchy in [10] speeds up detection over methods that test models sequentially - 120 single-view classes were detected within multiple seconds ($>$ 5sec).

Scalability and transformation invariance were addressed in earlier works in the 80s and 90s by indexing and geometric hashing, similar in form to the library look up that we use in our method. Examples include early work on points [14] and surfaces [12], and later on edges [1, 21, 22] where edges are grouped and represented by a transformation-invariant descriptor. The descriptors are indexed for all training views, then edges are grouped in the same manner in test images and the descriptor is computed for each pair, and compared to the library. For example, in [22], the contour segment between two consecutive bi-tangent edge points is mapped to a canonical frame, which is a projective-invariant descriptor. In test images, bi-tangent points are detected and the descriptor is computed for each pair, and compared to the library. While this approach is efficient, it is only applicable to concave curved boundaries. In [1], straight edges are grouped if co-terminating or parallel, and the descriptor encodes the relative angles and the relative lengths of the grouped edges. The descriptors for all groups are indexed using a best-bin-first k-d tree. In the test image, lines are similarly grouped, and the descriptors for m such groups are calculated. The nearest k examples from training images are found and a probabilistic approach decides on the best detections out of $k.m$ possible explanations. The method is not designed for frame-rate detection, and is sensitive to the detection of lines of the same lengths in training and test images.

(III) Recently, fast performance for detecting 3D objects from a video input has been tackled, mainly focusing on efficient algorithms. In [16] chamfer distance matching is improved and made faster by using 3D distance transforms and directional integral images. Detection time of 710ms given 300 reference views is reported, although search time increases linearly with more views and more objects. In [13], patches represented by histograms of dominant orientations followed by efficient bitwise matching enable detection within 80ms of one object using 1600 reference views. Similar to our purpose, [13] aims at real-time learning and detection of objects in workspaces. However, the representation is not rotation- or scale-invariant (hence the need for large numbers of reference views) and complexity increases with multiple objects, with detection time increasing to 333ms for 3 objects [1].

We can conclude that most shape-based detectors either require offline training or scale linearly as more objects are being searched for, or commonly both. To address speed and scalability in learning and testing, this paper proposes the use of pre-defined paths that specify the relative direction between edgelets, and importantly, make the search tractable for real-time operation. The traced edgelets are represented by a simple to compute transformation invariant descriptor, that is used as an index to similarly stored descriptors (in a way that revisits geometric hashing). The usage of paths was previously introduced in our earlier work [4] but is extended here by providing a method for path selection, and using it for real-time learning. The method is described in detail in the following section.

[1] Personal correspondence with the authors

3 The Method

For a 3D object, we refer to the edge map as seen from one point on the viewing sphere as a view of that object. Thus, a **view** ω is a set of edgelets $\{e_i\}$ where an **edgelet** is a short straight segment, represented by its centre point and orientation. A constellation of edgelets is an *n-tuple* of edgelets $c = (e_1,..,e_n)$. These edgelets could be nearby or distant, thus constellations characterise both local parts and global shape. Potentially, an exponential number of constellations is present in each view. To manage the matching complexity, we use paths that trace out constellations from the edge map. A **path** Θ is a sequence of angles $\Theta = (\theta_0,..,\theta_{n-2})$. From any starting edgelet e_1, the base angle θ_0 specifies the direction of a tracing vector v_1, initially with unspecified length, relative to the orientation of the starting edgelet. If this tracing vector intersects with another edgelet e_2 in the edge map, then the edgelet is added to the constellation. The next tracing vector v_2 then has the direction θ_1 relative to v_1, i.e. $cos(\theta_1) = (v_1 \cdot v_2)/(|v_1||v_2|)$. Note that the direction of v_2 is not dependent on the orientation of e_2, but only on the incoming vector v_1. This process continues until the constellation has n edgelets. Note that starting from any edgelet, zero or more constellations can be found using the same path (within a tolerance ε in the angles). Using a pre-defined path limits the number of considered constellations while maintaining sufficient variability in the configurations of these constellations (Fig. 2).

For a traced constellation c_i, the descriptor $f(c_i) = (\phi_1,...,\phi_{n-1},\delta_1,...,\delta_{n-2})$ specifies the relative orientations and distances between the consecutive edgelets in the constellation's tuple, where $\phi_i = \widehat{e_i,e_{i+1}}$ is the relative orientation of consecutive edgelets ($1 \le i \le n-1$), and $\delta_i = |v_{i+1}|/|v_i|$ is the relative distances between the edgelets ($1 \le i \le n-2$). The descriptor is of size $2n-3$, and is translation-, rotation- and scale-invariant. By keeping a comprehensive library of descriptors for all constellations guided by one path Θ from all starting edgelets, it is sufficient to extract one constellation using the same path from the object in the test image to produce a candidate detection that is verified using the rest of the view edgelets (Fig. 3).

In a test image, constellations are traced out using the same path. When a constellation c_t is found, the descriptor is calculated $f(c_t)$ and is compared to the library. If a match is found, an affine transformation H is estimated from the corresponding edgelets in the matched tuples. The homography H transforms all the view edgelets to the test image, and iterative closest edgelet [28] is used to refine the homography, where the distance between two edgelets $d(e_i,e_j)$ assesses the similarity in orientation (*ori*) and spatial position (*pos*) [24]

$$d(e_i,e_j) = |e_i.pos - e_j.pos|_2 + \lambda\,|e_i.ori - e_j.ori| \tag{1}$$

In Eq. 1, λ weights the orientation term. If $H(e_i)$ is the transformation of edgelet e_i under the affine-transformation H, and $\tau(e_i,H)$ is the closest edgelet in the test image to the transformed edgelet e_i using the distance in Eq. 1, then the cost of the detection E is the scaled

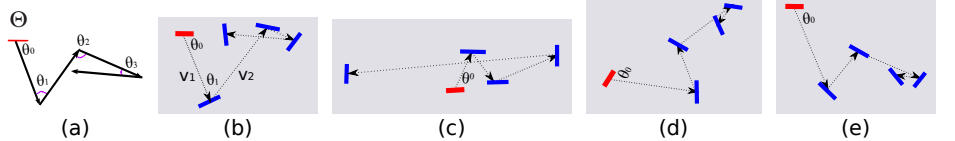


Figure 2: For a given path that is defined by a sequence of angles (a), four constellations are shown traced out by the same path from different starting edgelets (in red). Tracing vectors are shown (dotted) along with the constellation's edgelets. (d) and (e) have the same relative distances but differ in the relative orientations of the edgelets.

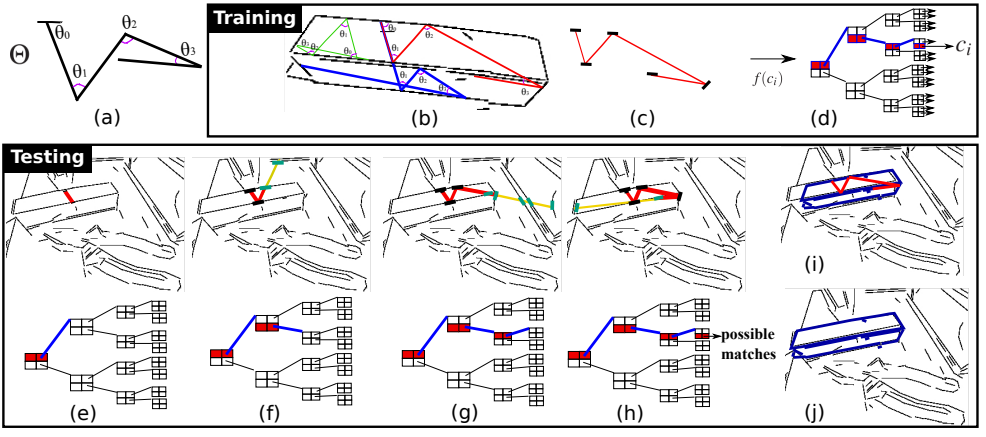


Figure 3: For a given path (a), edgelet constellations are traced out from training views. Tracing vectors from the same starting edgelet (red and blue) and from a different starting edgelet (green) are shown (b). During training, all constellations from all starting edgelets are found, and the descriptor for each constellation $f(c_i)$ is inserted into a quantised hierarchical hash table (d). During testing, constellations are traced out using the same path. The relative orientation and distance are calculated and the corresponding hash bin is located (e). When a constellation is completed (h), possible matches are located and a homography maps the view's edgelets to the test image (i). Iterative closest edgelet then refines the match (j).

average of the distance measures between corresponding edgelets.

$$E(\omega,H) = \frac{\sum_i \min(d(H(e_i),\tau(e_i,H)),\beta)}{|\omega|} \frac{|\{\tau(e_i,H) : d(H(e_i),\tau(e_i,H)) \leq \beta\}|}{|\{e_i : d(H(e_i),\tau(e_i,H)) \leq \beta\}|} R \qquad (2)$$

In Eq. 2, the distance measures are averaged along with a penalty measure β for missing correspondences when $d(H(e_i),\tau(e_i,H)) > \beta$, and the scale is estimated by the number of matched edgelets in the test image to the number they correspond to from view edgelets. The term R is the ratio of the lengths of view edgelets to test edgelets when different edgelet lengths are used. An object is detected at the test edgelets $\{\tau(e_i,H); d(H(e_i),\tau(e_i,H)) \leq \beta\}$ if $E(\omega,H) < \alpha$, where α is the acceptance threshold.

To speed detection, four techniques were used. First, the relative orientation, direction and position for all pair of edgelets in the test image are pre-calculated. Accordingly, all pairs of edgelets with relative positions that satisfy the base angle θ_0 (within the tolerance ε) are found from the pre-calculated data. From these, pairs are chosen at random, and constellations are completed by performing further lookups in the pre-calculated data. Thus, vectors are not actually traced in the image, but constellations are found from the pair calculations. Second, only one constellation is completed for each considered pair of edgelets. Given that the library contains a comprehensive list of all possible view constellations traced out by path Θ, the risk of skipping a pair before pursuing all the possibly-exponential number of constellations starting with that pair is acceptable, and proves sufficient during the experiments. Third, a quantised hierarchical hash table is used so the descriptor is incrementally calculated and compared to the corresponding level in the hash table (Fig. 3). When the accumulated descriptor cannot match any descriptor in the library, the search is prematurely stopped, and

another pair is pursued. Fourth, when a test constellation matches a view constellation with an error $E(\omega, H) < \alpha$ (Eq. 2), the corresponding test edgelets are greedily removed from any further searches to speed the detection of multiple objects present in the same image.

Several paths are used and a separate library is built for each chosen path. The choice of paths is discussed in Sec. 4.1. When all pairs are tested, another path Θ_2 is used. For k paths, the worst case is $O(k \cdot p^2)$ where p is the number of edgelets in the test image. The search is stopped when the maximum search time is reached. Note that it is possible to parallelise the path searches, but this is not implemented in the results presented next.

4 Experiments

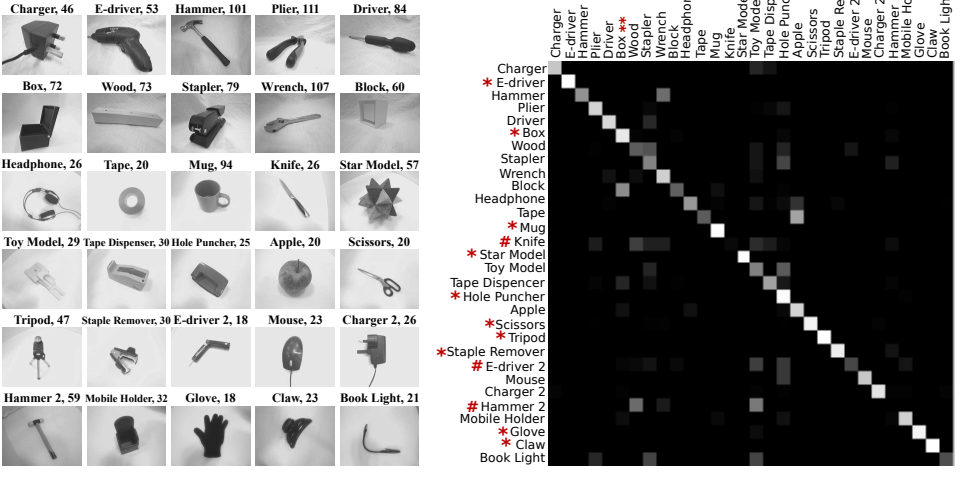


Figure 4: Thirty texture-less objects in the dataset (name, # of views) along with the confusion matrix from 10 runs. Each cell indicates the percentage of times an object of type row was classified as the type column. Ten objects achieved recall > 90% (*), two of them with precision < 90% (**). Three objects are difficult to detect with recall < 30% (#).

We have tested the method on a dataset of 30 objects (Fig. 4). Training uses a video around the viewing hemisphere on a clear background using a hand-mounted camera, and views are automatically sampled from the video (total 1433 views). There is no need to cluster similar views, and the descriptors of all constellations over the chosen paths are indexed in real-time. To extract edgelets, we evaluate two line detectors. The first is the line-segment detector (LSD) from [25]. The second uses the canny edge map, then traces the edge pixels to form straight line segments. During training, edgelets of 10 pixels length are sampled from the lines (ε = 0.02). During testing, edgelet length is increased to 15 pixels (thus, in Eq. 2, $R = 2/3$). Based on preliminary results, the length of the constellation tuple n was set to 5, β to 30 and λ to 2. Shorter tuples have a higher chance of hallucinating detections while longer tuples decrease the recall. Each level in the hash table was quantised into 64 bins - non-leaf nodes are binary, signifying the presence of matches.

4.1 Selecting Paths

We choose paths based on their ability to find constellations in a subset of views, prior to training. Instead of exhaustively searching the space of possible paths, we sample 100 random paths, and rank them by the number of edgelet constellations they can find in sampled training views. Recall on a ground-truthed sequence drops steadily (58.8% for rank 1,

33.0% for rank 15, 16.3% for rank 30 and 12.3% for rank 90). Figure 5 shows the top-six paths from three runs on different training objects. Interestingly, the best path is similar, and several paths are similar between the independent runs. Figure 6 shows the average number of processed paths as the specified maximum search time decreases. At 1fps, 5 paths were tested on average. At this rate, we test the contribution of the ordered paths in finding objects using different permutations of the top 6 ranked paths (Table 1). The table shows that 90% of the detections on average are found using 3 paths. One run on sampled views from the first 10 objects in the dataset results in the top-6 paths used in all our experiments on both the tools and ETHZ dataset. These proved suitable for various unseen object shapes.

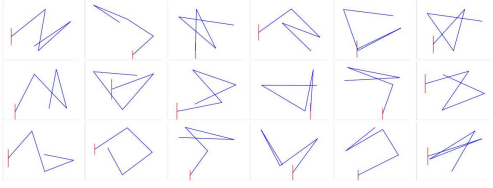


Figure 5: The top 6 paths by the number of extracted edgelet constellations, from three attempts. Notice that the first path is similar in the three attempts.

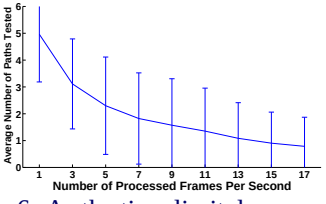


Figure 6: As the time limit decreases from 1s to 59ms, the avg. number of paths tested (out of 6) is shown.

Order of paths	Acc. % of detections after *n* paths					
	1	2	3	4	5	6
(1,3,4,5,6,2)	75.61	90.42	91.04	94.13	98.45	100
(2,3,4,5,6,1)	51.84	82.61	89.3	94.65	96.99	100
(3,1,2,4,5,6)	61.07	86.26	87.28	90.33	95.67	100
(4,3,5,1,6,2)	78.12	90.89	95.45	98.19	99.41	100
(5,1,2,4,6,3)	80.79	88.90	89.50	91.6	94.9	100
(6,5,4,3,2,1)	67.91	84.70	88.06	95.90	99.24	100
Avg.	69.22	87.30	90.10	94.13	97.44	100

Table 1: For 6 permutations of the top 6 paths, the accumulated percentage of detections is shown. The table shows that on average 69% of the detections are found by the first path, and more than 90% of the detections are found using 3 paths.

4.2 Results on the Tools Dataset

A video sequence of 1300 frames containing all the 30 objects was tested. Figure 7 plots recall against precision for three objects as well as all the objects (the PASCAL 50% overlap criterion is used [6]). In both cases, the search is for all the 30 objects. The figure shows that recall of 50% was achieved at 7fps (precision = 74%) using LSD. For a system that runs

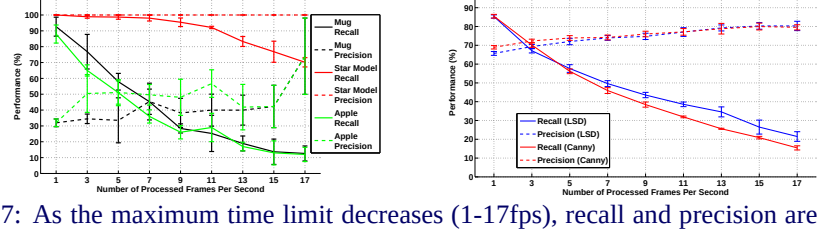


Figure 7: As the maximum time limit decreases (1-17fps), recall and precision are plotted for three objects (left) and for all the 30 objects (right). The performance varies for objects depending on their shape's distinctiveness. Results are similar for both LSD and Canny edge detectors, showing the method's resilience to the choice of edge detector.

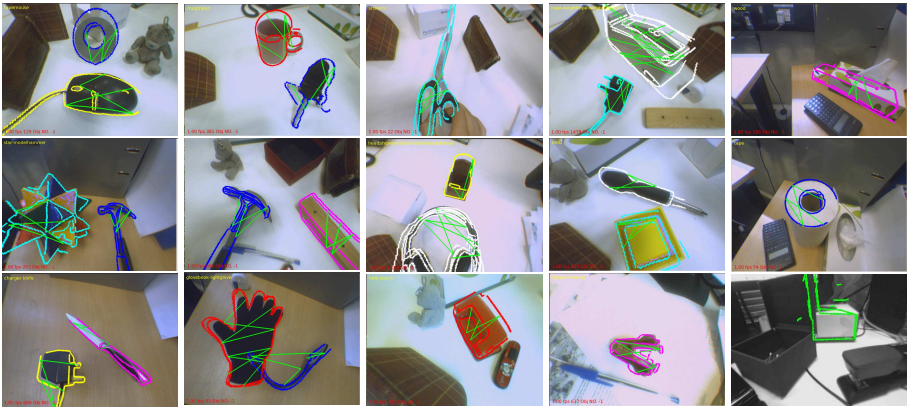


Figure 8: True-positive (col 1-4) and false-positive (col 5) cases from the 30-objects dataset using canny. False positive cases (wood, tape and box) could be resolved using appearance or depth information.

Figure 9: Detections on a sample of frames processed at 7fps using LSD. The last column shows test edgelets and the corresponding view edgelets marked in blue dots. Red dots indicate missing edgelets.

on a stream of live images, it is affordable if an object is missed in one frame as long as it can be detected in a few subsequent frames. True-positive and false-positive examples are shown in Fig. 8. We further assess the ability of the descriptor to distinguish between the 30 different objects. We run the detector for 10 times at 7fps (using LSD), and accumulate the detections. We report the results as a confusion matrix (Fig. 4). Three of our 30 objects are very difficult to distinguish (Fig. 4 #). This is because the descriptors of most constellations derived from these objects match descriptors extracted from other objects.

We also test the method on a 300-frame sequence (Fig. 9) containing 6 objects with surrounding clutter. Fig. 9 shows equally-spaced frames from the video sequence. Again, recall of 51% was achieved on this video at 7fps (precision = 86%) using LSD. The figure also shows correctly labelled occluded/missing view edgelets.

To test the method's scalability, we expect that as the number of objects and views increases, the detection time scales graciously. For one test image containing the object, Fig. 10 plots the detection time as more objects are learnt. The detection time is the elapsed time until the object is correctly detected without specifying a time limit for the search. The increase in detection time results from comparing to a larger number of descriptors in the hashtable, as

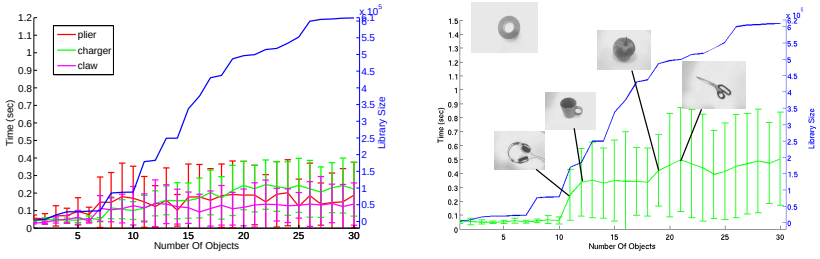
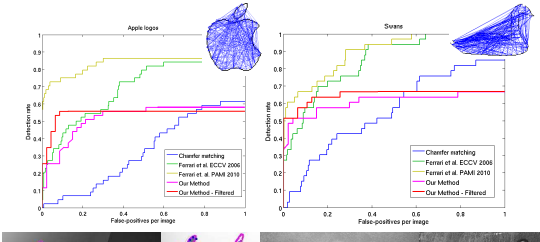


Figure 10: As the number of objects increases from 1 to 30, the library size increases by more than 150x, while the avg. detection time increases by 3.3x (plier), 4.8x (claw) and 5.5x (charger). For the object tape (right), the avg. detection time increases by 10x, particularly when objects with a circular shape are learnt (headphone, mug, apple and scissors).

well as assessing ambiguous matches. From the figure, adding non-ambiguous objects does not affect the average detection time much. Average detection time of 200ms was reported when 30 objects are learnt, compared to 60ms for a single object. For the ambiguous object 'tape' (Fig. 10 right), the average detection time increased by 10 folds when 30 objects are being searched for. This is because the circular shape of the tape is present within the shape of a few other objects in the dataset. Notice that the verification stage using all the view's edgelets will ensure that objects like the scissor cannot be detected when the tape is present, but the detection time is nevertheless affected by the descriptors' ambiguity.

4.3 Comparison with Off-line Learning on ETHZ dataset

With our method geared towards both learning and simultaneous multi-object detection in real-time, we do not expect to outperform methods that have the luxury of off-line process-



	Apple Logo	Swan	Bottle	Giraffe	Mug
[8]	83.2	75.4	83.2	58.6	83.6
[27](v)	84.0	76.7	93.1	79.5	67.0
[27](v+f)	95.8	94.1	96.3	84.1	96.4
[10]	87.3	80.0	87.6	83.5	86.1
[5]	73.0	63.5	86.9	80.3	81.6
Ours	73.2	66.1	68.97	72.4	60.9

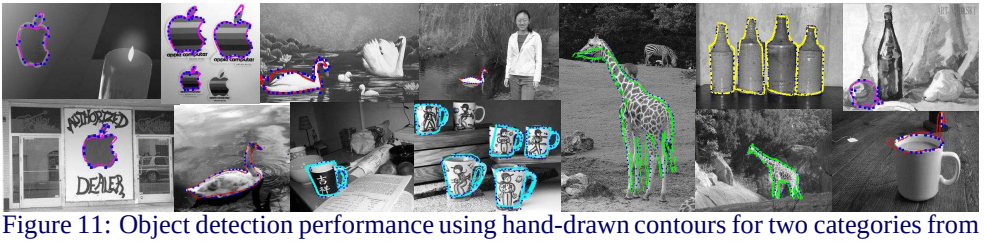


Figure 11: Object detection performance using hand-drawn contours for two categories from the ETHZ dataset compared to [8, 9] (left) as well as using training images for all categories at 0.4 FPPI from 5 runs (right). For comparison, we filter non-vertical detections as in [5]. Our real-time non-exhaustive generative method is compared to discriminative methods including the state-of-the-art on the dataset [27] that uses voting (v) followed by verification (f). True and false positive examples are shown.

ing and class-tuned discriminative classifiers. However, we are interested in evaluating how the proposed approach performs when compared to such systems. Here we present comparable results using the ETHZ dataset [9]. We use the Berkeley edge maps provided with the dataset [18], and the 50% PASCAL overlap criterion [6]. The system is run at 1 second per image - though many images were searched within less time. Fig. 11 shows sample results and detection performance on hand-drawn contours and training images. All training examples are considered as different views, and edgelet constellations are extracted using one path (Fig. 5 top-left). The results show competitive performance on this standard dataset.

5 Conclusion

This paper proposes a method for real-time learning and detection of multiple 3D textureless objects. The method uses paths as a tractable way to extract edgelet constellations. Constellations are represented by a transformation-invariant descriptor, which is used as an index to candidate detections. The method is both fast and scalable, and is tested at frame rates varying from 1 to 17 fps detecting multiple objects out of 30 three-dimensional textureless objects. As the number of objects in the library increases from 1 to 30, the increase of detection time is dependent on the shape's ambiguity rather than the number of objects. Videos and dataset are available online at http://www.cs.bris.ac.uk/Publications.

References

[1] J Beis and D Lowe. Indexing without invariants in 3D object recognition. *IEEE Transactions on Pattern And Machine Intelligence (PAMI)*, 21(10), 1999.

[2] O Carmichael and M Hebert. Object recognition by a cascade of edge probes. In *British Machine Vision Conference (BMVC)*, 2002.

[3] A Chia, S Rahardja, D Rajan, and M Leung. Object recognition by discriminative combinations of line segments and ellipses. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[4] D Damen, A Gee, A Calway, and W Mayol-Cuevas. Detecting and localising multiple 3D objects: A fast and scalable approach. In *IROS Workshop on Active Semantic Perception and Object Search in the Real World (ASP-AVS-11)*, 2011.

[5] O Danielsson, S Carlsson, and J Sullivan. Automatic learning and extraction of multi-local features. In *International Conference on Computer Vision (ICCV)*, 2009.

[6] M Everingham and J Winn. The PASCAL visual object classes challenge (VOC2007) development kit. Technical report, 2007.

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

[8] V Ferrari, T Tuytelaars, and L Van Gool. Object detection by contour segment networks. In *European Conference on Computer Vision (ECCV)*, 2006.

[9] V Ferrari, F Jurie, and C Schmid. From images to shape models for object detection. *International Journal of Computer Vision (IJCV)*, 87(3), 2009.

[10] S Fidler, M Boben, and A Leonardis. A coarse-to-fine taxonomy of constellations for fast multi-class object detection. In *European Conference on Computer Vision (ECCV)*, 2010.

[11] J Gall and V Lempitsky. Class-specific Hough forests for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[12] W Grimson and D Huttenlocher. On the sensitivity of geometric hashing. In *International Conference on Computer Vision (ICCV)*, 1990.

[13] S Hinterstoisser, V Lepetit, S Ilic, P Fua, and N Navab. Dominant orientation templates for real-time detection of texture-less objects. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[14] Y Lamdan and H Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *International Conference on Computer Vision (ICCV)*, 1988.

[15] M Leordeanu, M Hebert, and R Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[16] M Liu, O Tuzel, A Veeraraghavan, and R Chellappa. Fast directional chamfer matching. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[17] S Maji and J Malik. Object detection using a max-margin Hough transform. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[18] D Martin, C Fowlkes, and J Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. on Pattern and Machine Intelligence (PAMI)*, 26(5), 2004.

[19] A Opelt, A Pinz, and A Zisserman. A boundary-fragment-model for object detection. In *European Confernece on Computer Vision (ECCV)*, 2006.

[20] M Ozuysal, M Calonder, V Lepetit, and P Fua. Fast keypoint recognition using random forests. *IEEE Trans. on Pattern and Machine Intelligence (PAMI)*, 32(3), 2010.

[21] S Procter and J Illingworth. Foresight: fast object recognition using geometric hashing with edge-triple features. In *International Conference on Image Processing (ICIP)*, 1997.

[22] C Rothwell, A Zisserman, D Forsyth, and J Mundy. Canonical frames for planar object recognition. In *European Conference on Computer Vision (ECCV)*, 1992.

[23] S Savarese and L Fei-Fei. 3D generic object categorization, localization and pose estimation. In *International Conference on Computer Vision (ICCV)*, 2007.

[24] J Shotton, A Blake, and R Cipolla. Contour-based learning for object detection. In *International Conference on Computer Vision (ICCV)*, 2005.

[25] R Grompone von Gioi, J Jakubowicz, J Morel, and G Randall. LSD: A line segement detector. *IEEE Trans. on Pattern and Machine Intelligence (PAMI)*, 32(4), 2010.

[26] C Wiedemann, M Ulrich, and C Steger. Recognition and tracking of 3D objects. In *DAGM Symposium on Pattern Recognition*, 2008.

[27] P Yarlagadda, A Monroy, and B Ommer. Voting by grouping dependent parts. In *European Conference on Computer Vision (ECCV)*, 2010.

[28] Z Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision (IJCV)*, 13(2), 1994.