Object Recognition Using Junctions

Bo Wang[1], Xiang Bai[1], Xinggang Wang[1], Wenyu Liu[1], Zhuowen Tu[2]

1. Dept. of Electronics and Information Engineering,
Huazhong University of Science and Technology, China
{wangbo.yunze,wxg}@gmail.com, {xbai,liuwy}@hust.edu.cn
2. Lab of Neuro Imaging, University of California, Los Angeles
ztu@loni.ucla.edu

Abstract. In this paper, we propose an object detection/recognition algorithm based on a new set of shape-driven features and morphological operators. Each object class is modeled by the corner points (junctions) on its contour. We design two types of shape-context like features between the corner points, which are efficient to compute and effective in capturing the underlying shape deformation. In the testing stage, we use a recently proposed junction detection algorithm [1] to detect corner points/junctions on natural images. The detection and recognition of an object are then done by matching learned shape features to those in the input image with an efficient search strategy. The proposed system is robust to a certain degree of scale change and we obtained encouraging results on the ETHZ dataset. Our algorithm also has advantages of recognizing object parts and dealing with occlusions.

1 Introduction

Recent progress for object detection/recognition has been mostly driven by using advanced learning methods [2–6] and designing smart feature/object descriptors [7–9]. A detector is often trained on either a large number of features [2] or SIFT like features in a bounding box[4]. Most of the resulting algorithms, however, only tell whether an object is present or not in a bounding box by sweeping an input image at all locations and different scales. Besides the successes the field has witnessed for detecting rigid objects, such as frontal faces, detecting non-rigid objects remains a big challenge in computer vision and most of the systems are still not practical to use in general scenes [10].

Another interesting direction is using deformable templates [11] through matching-based approaches. Typical methods include generalized Hough transform [12], shape contexts [13], pyramid matching [14], pictorial structures [15], codebook-based approaches [16, 17], and hierarchical shape representations [18–20]. These algorithms not only locate where an object appears in an image, they also recognize where the parts are, either through direct template correspondences or part representations. However, the performances of these algorithms are still not fully satisfactory, in terms of both efficiency and accuracy.

Marr [21] laid out a path to object recognition with a series of procedures including: (1) generic edge detection, (2) morphological operators such as edge

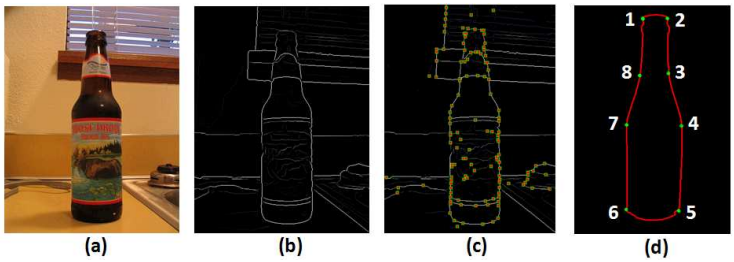

Fig. 1. (a) is the original image, (b) is an edge map by [1], (c) shows automatically detected junctions of (b), and (d) is our model with 8 junctions.

linking and thinning, (3) shape matching on edges and object boundaries. This direction recently becomes unpopular because it largely relies on obtaining high quality edges; in addition, its object descriptors are too simplistic to handle the level of complexity in natural images. It is now accepted that perfect edge/feature detection does not exist [22] and it is hard to strictly separate the high-level recognition process from the low-level feature extraction stage. Nevertheless, these type of traditional methods still offer many appealing perspectives compared to modern approaches for being simple, generic, and without heavy learning.

In this paper, we take a rather traditional route by performing junction extraction first, followed by shape matching using a new set of feature descriptors. Note that from the remainder of this paper, we refer to junctions as corner points with more than one-degree connection. Given an object template described by its boundary contour, we annotate several corner points of high curvature with their order information; we then design two types of shape-context like features for describing the junction points. Note that these features are different from the standard shape context [13] since we only take into account the relevant junctions on the boundary. This means that, in the detection stage, we need to perform explicit search to exclude the background clutter. Fig. (1) shows an example of an object template with its corresponding junction points. To detect/recognize an object, we first apply a recently developed algorithm [1] to extract junction points from cluttered images; we then apply a pre-processing procedure to clean the edges and junctions; shape matching is then performed between the templates and the extracted junctions with an efficient search strategy. The proposed system spends about 1 or 2 minutes on an image to recognize an object (excluding another 1 or 2 minutes for extracting junctions). Our algorithm also has advantages of recognizing object parts and dealing with occlusions. The strength of this paper lies in: (1) the design of a new set of shape descriptors, (2) the development of a promising matching-based object detection/recognition system, (3) the achievement of significantly improved results on non-rigid objects like those in the ETHZ dataset.

There are several other related methods worth mentioning. Shotton et al. [23] describes the shape of the entire object using deformable contour fragments and their relative positions. Since their distance measure using improved Chamfer Matching is sensitive to the noise, many training samples are required for boosting the discriminative shape features. G. Heitz et al. [24] uses probabilistic shape to localize the object outlines. Our method is different from [25]. (1) We design two types of SC-like features of junctions and edges on actively searched contours whereas [25] uses geometric features of connected contours; (2) we emphasize a sparse representation on junctions whereas [25] uses dense points for object detection. Other works [26, 27, 18, 28] decompose a given contour of a model shape into a group of contour parts, and match the resulting contour parts to edge segments in a given edge image.

2 Junction features

We use junction points as the basic elements to describe the object contour, and thus, our shape model could be considered as a simplified polygon with junction points being the vertices. In general, a majority of the junctions are the high curvature corner points of degree 2. There are also some junction points with degree 3 or 4, depending upon the image complexity. However, there are rarely junctions with degree higher than 4. We adopt a recently developed algorithm [1] to detect the junction points. and Fig. (1.c) shows an example. In Fig. (1.d), an object template with 8 junction points is displayed. As we can see, due to the presence of image clutter, it is not an easy task to match the template to the object even with reliable low-level features.

Given a contour C of n junction points, we denote $C = (J_1, J_2, ..., J_n)$, where J_i is the i^{th} junction on C. Note that we preserve the clockwise order of each junction as the ordering is important in our model. In our current implementation, we assume multiple templates for each object type have the same number of junctions. However, clustering can be used to obtain different clusters of the same object type.

2.1 Junction descriptors

We design two types of features, which are called F_1 and F_2 respectively. For each junction point J_i, we compute the feature $F_1(J_i)$ based on its connected contour segments using a shape context like approach. Unlike the traditional shape context approaches [13] where all the points within the context radius are taken into account, we only use those points on the contour segments.

The two contour segments $e_{i-1,i}$ and $e_{i,i+1}$ between (J_{i-1} and J_i) and (J_i and J_{i+1}) respectively are called **path** to J_i, denoted as $P(J_i)$. We then use path $P(J_i)$ to characterize J_i and compute the corresponding feature $F_1(J_i)$. Fig. (2.a) gives an example. We sample 10 points at equal space on $P(J_i)$ and call them path points as $(p_1^{(i)}...p_{10}^{(i)})$ (see green points in Fig. (2.b)). Here t is the index along the path from J_{i-1} to J_{i+1}. Note that these 10 points are on the

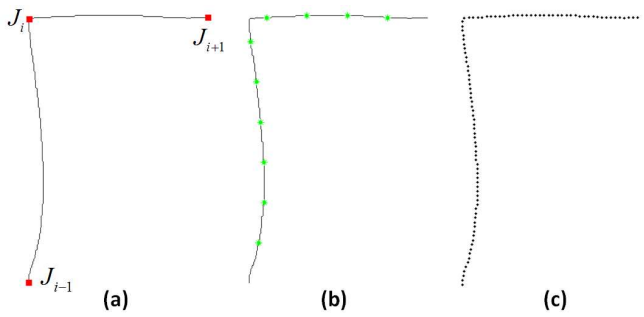

Fig. 2. The illustration for the feature (F_1) for characterizing the junction points. The red ones in (a) are the junction points. The green dots in (b) are sampled path points on which shape-context like features are computed. (c) shows the densely sampled points for the green dots in (b) to compute shape context information.

path altogether and $e_{i-1,i}$ and $e_{i,i+1}$ may not have 5 points each since they do necessarily have the same length. For each path point p_t, we compute its feature $h(p_t)$ based on 50 densely sampled points on path $P(J_i)$ at equal space. Fig. (2.c) gives an illustration. The parameter setting for computing the histogram of shape context is the same as that in [13]: 5 distance scales and 12 angle scales. Thus, each $h(p_t)$ can be viewed as a feature vector of length 60. Finally, we are ready to describe $F_1(J_i)$ as:

$$F_1(J_i) = \left(h(p_1^{(i)}), ..., h(p_{10}^{(i)})\right)^T, \tag{1}$$

which is of length $60 \times 10 = 600$.

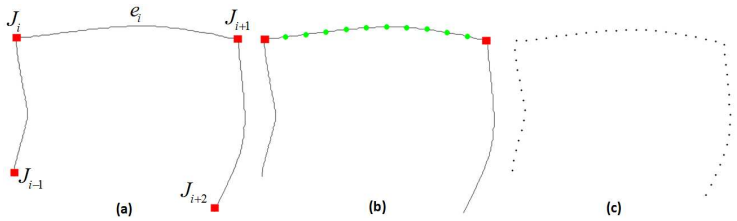

Fig. 3. The illustration for the feature (F_2) for characterizing the contour segments between junction points. The red ones in (a) are the junction points. The green dots in (b) are those on which shape-context like features are computed. (c) shows the densely sampled points for the green dots in (b) to compute shape context information.

Next, we show how to compute feature F_2 to characterize the shape information about a contour segment $e_{i,i+1}$. The approach is similar to the way F_1

is computed. We sample 10 segment points at equal space on $e_{i,i+1}$ and denote them as $(p_1^{(i,i+1)}...p_{10}^{(i,i+1)})$; for each $p_t^{(i,i+1)}$, we compute its shape context feature based on 50 equally sampled points on $e_{i-1,i}$, $e_{i,i+1}$, and $e_{i+1,i+2}$ altogether; the parameter setting for computing the shape context is the same as that in computing F_1. This means that the features for $e_{i,i+1}$ also takes into account its immediate neighboring segments. Thus,

$$F_2(e_{i,i+1}) = \left(h(p_1^{(i,i+1)}), ..., h(p_{10}^{(i,i+1)})\right)^T, \tag{2}$$

which is also of length $60 \times 10 = 600$. Fig. (3) shows an illustration.

2.2 Junction descriptors for edge maps

Due to the background clutter in natural images, the low-level edge/junction detection algorithms are always not perfect. We briefly describe some pre-processing steps in our algorithm. First, standard edge linking methods [29] are applied on extracted edge maps [1] using morphological operators. Fig. (4) gives an illustration. Fig. (4.a) shows the original edge segments by [29], which removes many background clutters. The remaining edges are used to connect the junction points also by [1].

Fig. 4. The linking process for the segments around a junction

Given an input image I in the detection stage, we use method in [1] to extract the edges and junction points, and apply a software package [29] to perform edge linking. We call a junction point detected in a test image, J'. Next, we discuss how to compute its corresponding feature, $F_1(J')$. The idea is to search for the other two most plausible junctions J'_- and J'_+ for J' to be adjacent on the object contour. The junctions on the template are selected based on the guideline to have high curvature; the search strategy echoes this but without using any shape matching strategy at this stage.

We first discuss the case where the degree of J' is 2. The problem is that the nearest junctions to J', $J'_-(0)$ and $J'_+(0)$ from the low-level edge/junction extraction process, might not be the desirable ones. We propose a simple deterministic procedure to perform the search:

1) Given a junction J', we find its nearest junctions $J'_-(0)$, $J'_+(0)$ along the edges.

2) Let $S = \{s_1, s_2, ..., s_{|S|}\}$ denote the adjacent junctions of $J'_-(t)$ on the edge map (the junctions on the path between J' and $J'_-(t)$ are not included in S).

Let $\mathbf{x}$ and $\mathbf{x}_-(t)$ be the coordinates of J' and $J'_-(t)$ respectively. Let $\mathbf{x}_l$ denote the coordinates of s_l. We compute the angle as:

$$\theta_l = arccos(\frac{(\mathbf{x} - \mathbf{x}_-(t)).(\mathbf{x}_-(t) - \mathbf{x}_l)}{|\mathbf{x} - \mathbf{x}_-(t)||\mathbf{x}_-(t) - \mathbf{x}_l|}). \tag{3}$$

3) Then we find a l^* that satisfy:

$$l^* = \arg \min_{l=1,2,...,|S|} \theta_l. \tag{4}$$

If θ_{l^*} is smaller than a given threshold $\xi = 0.175$, let $t = t + 1$, set s_{l^*} as $J'_-(t)$ and go back to step 2). Else, output the $J'_-(t)$ as the final J'_-.

The above procedures determine the junction J'_-, and the procedures to determine J'_+ are the same. Fig. (5) shows an example when the degree of junction is 2. In Fig. 5(a), point J'_1 is a junction and the proposed procedure searches for the most plausible J'_-/J'_+, then the path between point 6 (J_-) and point 1 is chosen for computing feature F_1. Fig. (5.b) shows the path between 1 and 6.

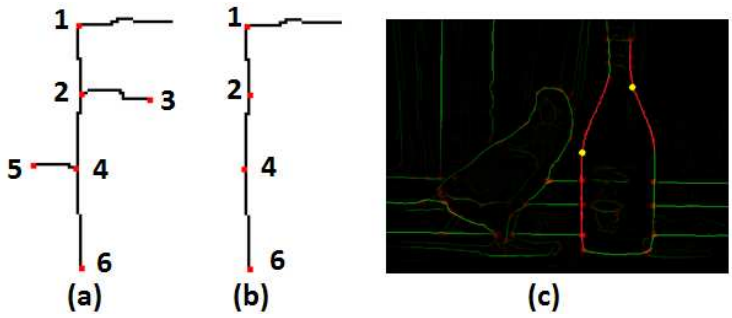

Fig. 5. An illustration for finding the path from junction J'_1 (point 1) to its J'_- (point 6).

Once J'_- and J'_+ are determined, we then obtain a path $P(J')$ from J'_- to J'_+ (passing J') that is used for computing the feature $F_1(J')$. Fig. 5(c) gives two examples on a real image: the two points in yellow are two junctions, and the red segments denotes the paths used for computing F_1 of them separately. We can also view our algorithm as designed for finding the salient contour segments, which might be useful in other vision applications.

When the degree of J' is higher than 2, we then compute multiple features F_1 for J' corresponding different possible paths passing through J'. Let $d(J')$ denote the degree of J'. There are $d(J')$ junctions that are adjacent to J', which

means there are $D = C^2_{d(J')}$ paths P_q $(q = 1, ..., D)$ that will pass J' (D pairs of J_-/J_+ can be estimated). In order to keep the same with the 2-degree case, we consider J' as D different 2-degree junctions $J'_{(q)}(q = 1, ..., D)$ with the same position and different paths (different F_1 feature):

$$F_1(J'_{(q)}) = F_1(P_q). \tag{5}$$

Fig. 6 shows an example when the degree of a junction is 3. Point 1 in Fig. (6.a) can have three possible paths as separately shown in Fig. (6.b,c,d).

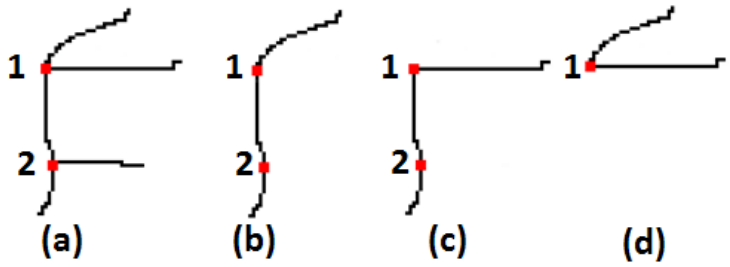

Fig. 6. An illustration for junctions with degree higher than 2.

3 Detection

Once the junctions are determined, we then proceed to the detection/recognition stage by matching the features on the junctions and segments to those in the templates. A two-layer detection framework is proposed: In the first layer, we classify all the junctions in a edge map $\mathcal{M}$ using a kNN classifier; based on the junction classification results, we use the shortest path to find the order of these junctions along the contour on $\mathcal{M}$ in the second layer; then we localize the object position. Our goal is to to find a sequence of junctions most similar to the training sequence, which is similar to shape matching with Dynamic Programming.

3.1 Junction classification

Recall that for each object type, all templates have the same number of junction points. For example, for the bottle templates, there are 8 junctions. Given a set of training templates, we compute the corresponding $F_1(J)$ for each junction. The problem of computing how likely a junction J' in a test image belongs to a specific junction on the bottle becomes a classification problem. Each $F_1(J)$ is a 600 dimension feature and we simply learn a kNN classifier to classify J' int $\{1, 2, ..., 8\}$ classes of junction points. In training a kNN classifier, the most important thing is to define the distance measure:

Let f denote a vector value of F_1, then we define the distance function dis in the same manner of SC [13]:

$$dis(F_1(J), F_1(J')) = \frac{1}{2}\sum_{i=1}^{600}\frac{(f_i - f'_i)^2}{f_i + f'_i}. \tag{6}$$

The class label L^* corresponding to the maximum is output by the algorithm:

$$L^* = \arg\max_{i=1,\ldots,n} p(L_i|F_1(J')) \tag{7}$$

3.2 Graph model

On the edge/junction map $\mathcal{M}(\mathcal{I})$ of image I, we classify all the junctions into n groups G'_i, based on the trained kNN classifier. Our next goal is to localize the object boundary using a polygon with junctions as the vertices, which can be solved by finding the shortest path on a graph. As shown in Fig. (8), we construct a connected graph model (V, E) in which the vertices V represent junctions in a test image. Let $e_{(j,k)}$ denote the edge between two junction nodes J'_j, J'_k from adjacent groups G'_i and G'_{i+1} respectively. Let $w_{j,k}$ denote the weight of the edge $e_{(j,k)}$. We set two dummy node N_s and N_e (in red) as the source node and the target node respectively. The weights of the edges connecting with the two dummy points are set as zero. The intuition is that all the critical junctions on the object should lie on the shortest path between N_s and N_e. We use the shortest path algorithm to solve this problem.

The edge weight $w_{j,k}$ is computed with dissimilarity between the edge $e_{(j,k)}$ and the edges $e^t_{i,i+1}(t = 1, ..., M)$ from the training templates. We use F_2 feature to measure this dissimilarity:

$$w_{j,k} = \frac{1}{M}\sum_{t=1}^{M} dis(F_2(e_{(j,k)}), F_2(e^t_{i,i+1})) \tag{8}$$

Notice that the way for computing F_2 feature on a edge map is different from the case for training template, since we do not know the adjacent junctions on a edge map. For junctions J'_j and J'_k, we can obtain their related paths $P(J'_j)$ and $P(J'_k)$ (as shown Fig. 7 (a)) respectively using the search algorithm proposed in Section 2.2 firstly. Then we sample the straight segment between J'_j and J'_k into ten points $p^{(j,k)}_t(t = 1, .., 10)$ at equal space (see Fig. 7 (b)); For each $p^{(j,k)}_t$, we compute its shape contexts feature on 50 equally sample points (see Fig. 7 (c)) on $P(J'_j)$ and $P(J'_k)$ together. Finally, the $F_2(e_{(j,k)})$ is described as:

$$F_2(e_{(j,k)}) = \left(h(p^{(j,k)}_1), ..., h(p^{(i,k)}_{10})\right)^T, \tag{9}$$

For the shortest path on our graph model, the linear programming has the special property that is integral. A * search algorithm [19] which uses heuristics to try to speed up the search can be applied to solve the optimization problem. The confidence for a detection is the sum of all the edge weights on the shortest path, which is used for the categories classification.

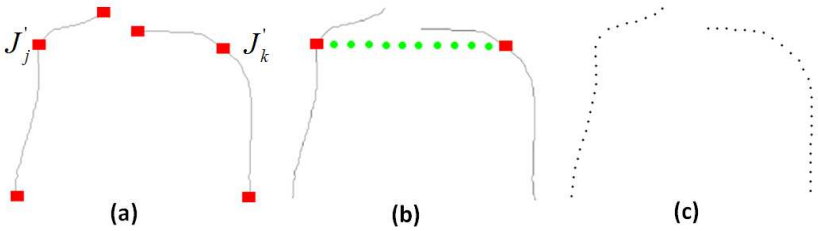

Fig. 7. The illustration for computing F_2 feature on a edge map

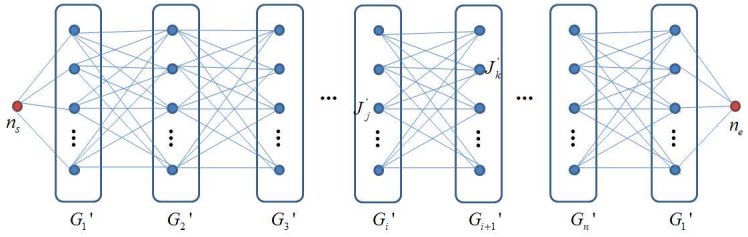

Fig. 8. The illustration for the graph model

4 Experiments

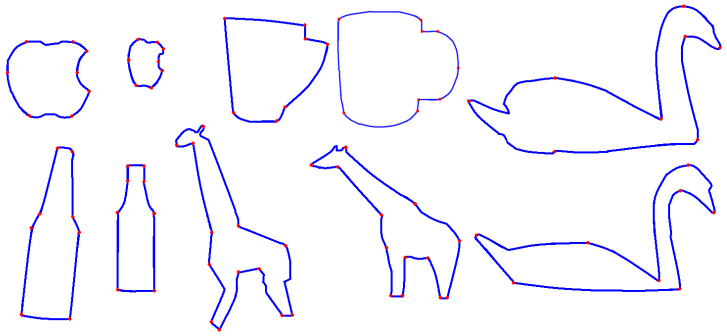

Fig. 9. Two training templates for each class from the ETHZ dataset [16]

We tested the proposed method on ETHZ shape dataset [16], which contains 5 different shape-based classes (apple logos, bottles, giraffes, mugs, and swans) with 255 images in total. Each category has significant variations in scale, intra-class pose, and color which make the object detection task challenging. To have a fair comparison with [28], we use 1/3 positive images of each class as training

samples, same to [28]. Fig. 9 shows a few training contour templates (the red points denote the junctions of each contour), which are extracted from the binary masks of the ETHZ dataset.

Initially, we extracted the gPb-based edge maps and junctions [1]. In an image of average complexity, there are on average 100 junctions. We take the binary mask annotation as the training templates. The results under PASCAL criterion of our method are reported in Fig. (10) with precision vs. recall curve. We also compare it to the latest results in [28, 26] . Fig. (10) shows P/R curves for the Kimia's method based on skeletal shape model [28] in red and for contour selection [26] in blue. Our method significantly outperforms [28] on the four categories of apple logos, bottles, mugs and swans, and a little better than [28] in the category of giraffe. This demonstrates that our junction model can well capture the intra-class variations of objects. Our result is also better than [26] on all the categories. We also compare the precision at the same recall to [28, 26, 30].

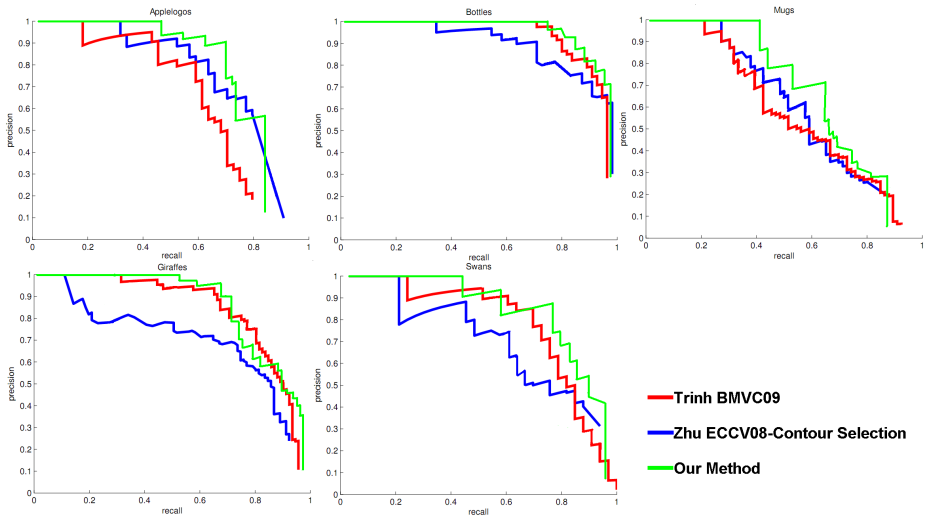

Fig. 10. Precision/Recall curves of our method compared to [28] and [26] for 5 classes of ETHZ dataset

As Table 1 shows, our method works better than [28] and [30], particularly in the category of apple logos. This is because our junction features take into consideration of both local and global structures. Even though our method is just slightly better than [28] in the category of giraffe, our method does not need multi-scale shape skeletons as our method is based on junctions that are more-or-less scale-invariant. Fig. (11) shows some detection results by the proposed method. The points and segments in red are the junction points and the poly-

	Apple logos	Bottles	Giraffes	Mugs	Swans
Our method	**52.9/86.4**	**69.8/92.7**	**82.4/70.3**	**28.2/83.4**	**40.0/93.9**
Zhu et al. [26]	49.3/86.4	65.4/92.7	69.3/70.3	25.7/83.4	31.3/93.9
Trinh&Kimia [28]	18.0/86.4	65.1/92.7	80.0/70.3	26.3/83.4	26.3/93.9
Ferrari et al. [30]	20.4/86.4	30.2/92.7	39.2/70.3	22.7/83.4	27.1/93.9

Table 1. Comparison of the precision at the same recall.

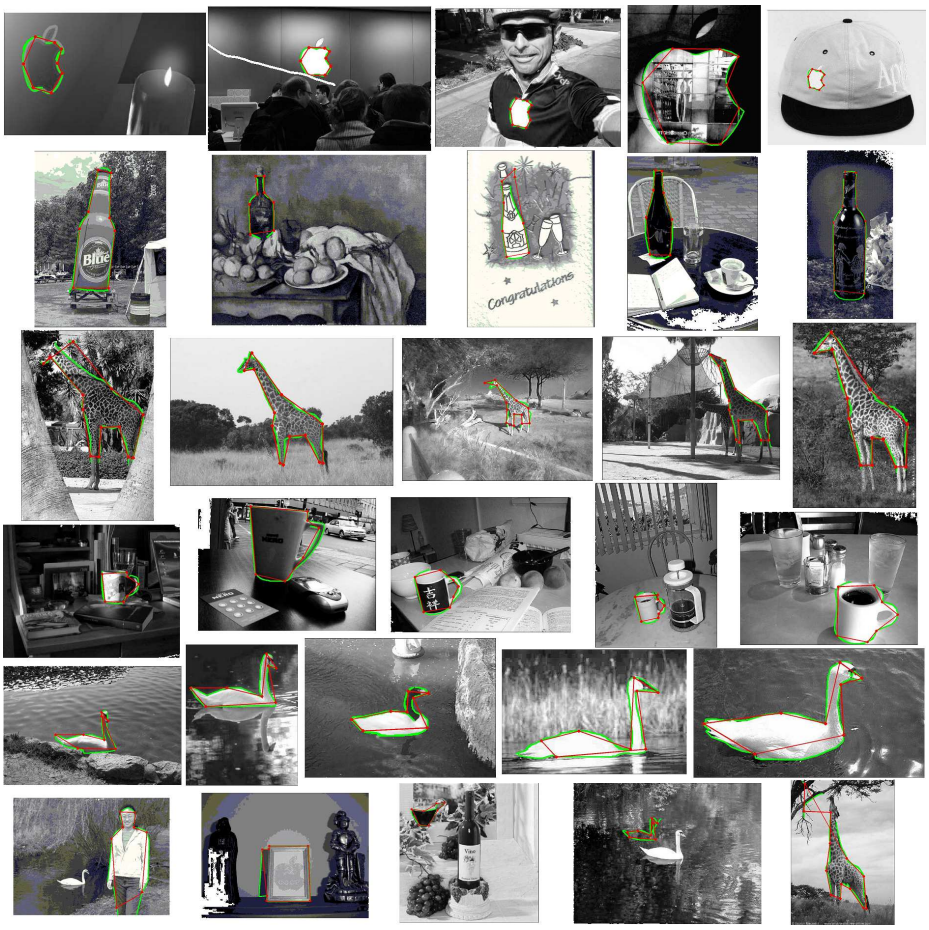

Fig. 11. The detection results of ETHZ dataset

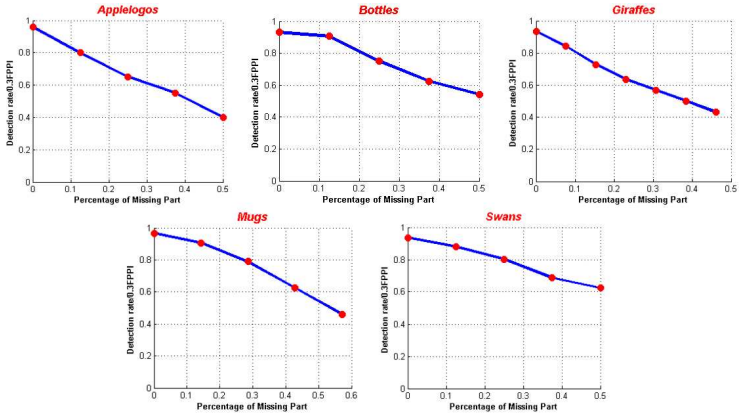

Fig. 12. The curves about detection rate (DR) at 0.3 FPPI vs. the percentage of miss contours for 5 classes of ETHZ dataset

gons that use the detection junctions as the vertices. We also show the contour segments (in green) related to each junction in these images; we observe that our method not only can detect the object position robustly but also have good localization of the object contour, benefiting from the junctions.

The last row in Fig. (11) shows a few false detections. It's very interesting that we detected a girl when detecting a bottle in the first image (last row); in the second image, we detected a photo frame when detecting a mug; in the third image, we detected a mug when detecting a swan; the fourth and fifth images (last row) show two examples about false positives. Notice that even we could not detect a swan in the fourth image, the segments detected out are very similar to a swan, which is a graceful failure.

Our method is not limited to detect the whole contour of objects. It can also be used to detect object parts. For detecting a contour part, we only use a group of consecutive junctions from J_i^t to $J_{i+m}^t (m < n)$ on the training templates. We randomly choose the start junction and end junction with a fixed length percentage for training, and to make a clear evaluation of performance, we use detection rate vs false positive per image(DR/FPPI). Fig.12 reports the average detection rate at 0.3 FPPI for five classes of ETHZ dataset. In Fig. 12, we observe that our detection rates can still reach above 0.4 at 0.3 FPPI when 50% of the training contours are missing. This demonstrates that the proposed junction features are stable and effective for recognizing shapes in clutter images. Fig. 13 shows a few detection results with only parts detected.

5 Conclusions and future work

In this paper, we have introduced a shape-based object detection/recognition system and showed its advantage on detecting rigid and non-rigid objects, like

Fig. 13. The detection results for partial contour detection with the proposed method

those in the ETHZ dataset. Our method follows the line of template matching by defining contour templates with a set of junction points. We found the designed shape descriptors to be informative and our system outperforms many contemporary approaches using heavy learning and design. We anticipate junction features to be useful for other vision tasks. In the future, we plan to combine the shape features with appearance information to provide more robust results.

Acknowledgement

We thank Michael Maire for nicely providing us the edge images with junctions of ETHz dataset. This work was jointly supported by NSFC 60903096, NSFC 60873127, ONR N000140910099, NSF CAREER award IIS-0844566, and China 863 2008AA01Z126.

References

1. Maire, M., Arbelaez, P., Fowlkes, C., Malik, J.: Using contour to detect and localize junctions in natural images. In: CVPR. (2008)
2. Viola, P.A., Jones, M.J.: Robust real-time face detection. IJCV **57** (2004) 137–154
3. Kumar, S., Hebert, M.: Discriminative random fields: a discriminative framework for contextual interaction in classification. In: ICCV. (2003)
4. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: CVPR. (2008)
5. Dollár, P., Babenko, B., Belongie, S., Perona, P., Tu, Z.: Multiple component learning for object detection. In: Proc. of ECCV. (2008)
6. Maji, S., Malik, J.: Object detection using a max-margin hough transform. In: CVPR. (2009)
7. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60** (2004) 91–110

8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proc. of CVPR. (2005) 886–893
9. Gu, C., Lim, J.J., Arbelaez, P., Malik, J.: Recognition using regions. In: CVPR. (2009)
10. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: A benchmark. In: Proc. of CVPR. (2008)
11. Yuille, A.L., Hallinan, P.W., Cohen, D.S.: Feature extraction from faces using deformable templates. IJCV **8** (1992) 99–111
12. Ballard, D.H.: Generalizing the hough transform to detect arbitrary shapes. Pattern Recognition **13** (1981) 111–122
13. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. PAMI **24** (2002) 509–522
14. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: Proc. of ICCV. (2005)
15. Felzenszwalb, P.F., Huttenlocher, D.P.: Pictorial structures for object recognition. IJCV **61** (2005) 55–79
16. Ferrari, V., Fevrier, L., Jurie, F., Schmid, C.: Groups of adjacent contour segments for object detection. PAMI **30** (2008) 36–51
17. Opelt, A., Pinz, A., Zisserman, A.: A boundary-frament-model for object detection. In: ECCV. (2006)
18. Gavrila, D.: A bayesian, exemplar-based approach to hierarchical shape matching. IEEE Trans. Pattern Anal. Mach. Intell. **29** (2007) 1408–1421
19. Felzenszwalb, P.F., Schwartz, J.D.: Hierachical matching of deformable shapes. In: CVPR. (2007)
20. Zhu, L., Chen, Y., Ye, X., Yuille, A.L.: Structure-perceptron learning of a hierarchical log-linear model. In: Proc. of CVPR. (2008)
21. Marr, D.: Vision. W.H. Freeman and Co. San Francisco (1982)
22. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, colour and texture cues. PAMI **26** (2004) 530–549
23. Shotton, J., Blake, A., Cipolla, R.: Multi-scale categorical object recognition using contour fragments. PAMI **30** (2008) 1270–1281
24. Heitz, G., Elidan, G., Parker, B., Koller, D.: Loops: Localizing object outlines using probabilistics shape. In: NIPS. (2008)
25. Ferrari, V., Jurie, F., Schmid, C.: Accurate object detection combining recognition and segmentation. In: CVPR. (2007)
26. Zhu, Q., Wang, L., Wu, Y., Shi, J.: Contour context selection for object detection: A set-to-set contour matching approach. In: ECCV. (2008)
27. Lu, C., Latecki, L., Adluru, N., Yang, X., Ling, H.: Shape guided contour grouping with particle filters. In: ICCV. (2009)
28. Trinh, N., Kimia, B.: Category-specific object recognition and segmentation using a skeletal shape model. In: BMVC. (2009)
29. Kovesi, P.D.: MATLAB and Octave functions for computer vision and image processing. School of Computer Science & Software Engineering, The University of Western Australia (2008) Available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>.
30. Ferrari, V., Fevrier, L., Jurie, F., Schmid, C.: From images to shape models for object detection. In: Technical Report 6600, INRIA. (2008)