Learning Class-Specific Edges for Object Detection and Segmentation

Mukta Prasad[1], Andrew Zisserman[1], Andrew Fitzgibbon[2], M. Pawan Kumar[3], and P.H.S. Torr[3]

[1] University of Oxford, U.K.
[2] Microsoft Research, Cambridge, U.K.
[3] Oxford Brookes University, U.K.
http://www.robots.ox.ac.uk/∼{mukta, vgg}

Abstract. Recent research into recognizing object classes (such as humans, cows and hands) has made use of edge features to hypothesize and localize class instances. However, for the most part, these edge-based methods operate solely on the geometric shape of edges, treating them equally and ignoring the fact that for certain object classes, the appearance of the object on the "inside" of the edge may provide valuable recognition cues.

We show how, for such object classes, small regions around edges can be used to classify the edge into object or non-object. This classifier may then be used to prune edges which are not relevant to the object class, and thereby improve the performance of subsequent processing. We demonstrate learning class specific edges for a number of object classes — oranges, bananas and bottles — under challenging scale and illumination variation.

Because class-specific edge classification provides a low-level analysis of the image it may be integrated into any edge-based recognition strategy without significant change in the high-level algorithms. We illustrate its application to two algorithms: (i) chamfer matching for object detection, and (ii) modulating contrast terms in MRF based object-specific segmentation. We show that performance of both algorithms (matching and segmentation) is considerably improved by the class-specific edge labelling.

1 Introduction

There is a long tradition of using edge features in object recognition: dating back to the 1980s edges were used for recognizing specific objects [7,11,15]; and more recently edges have been used for recognizing object *classes* such as humans (e.g. in Gavrila's combination of chamfer matching and a template tree [6] or by shape context [1,17]), hands [21,23], and animals such as cows and horses [9,14,19].

In algorithms such as [6], recognition is performed while treating image edges equally regardless of their context. However, all edges are not equal. The edges on the boundary of an object from a specific class have the characteristic local colour or texture of that object class on one side (and can have anything else

P. Kalra and S. Peleg (Eds.): ICVGIP 2006, LNCS 4338, pp. 94–105, 2006.

on the other side). Similarly, class specific edges may also have a characteristic shape. The key idea of this paper is to learn a classifier for the object class of interest which can label an edge with the probability of it belonging to that object class or not.

Our objective is learn a classifier based on all the available local information around an edge – appearance, texture and shape. While conventional cue integration tends to occur later in the processing pathway, this "early vision" integration means that it is easy to modify existing applications to use our class-specific edges, offering the potential for improved performance across a range of applications.

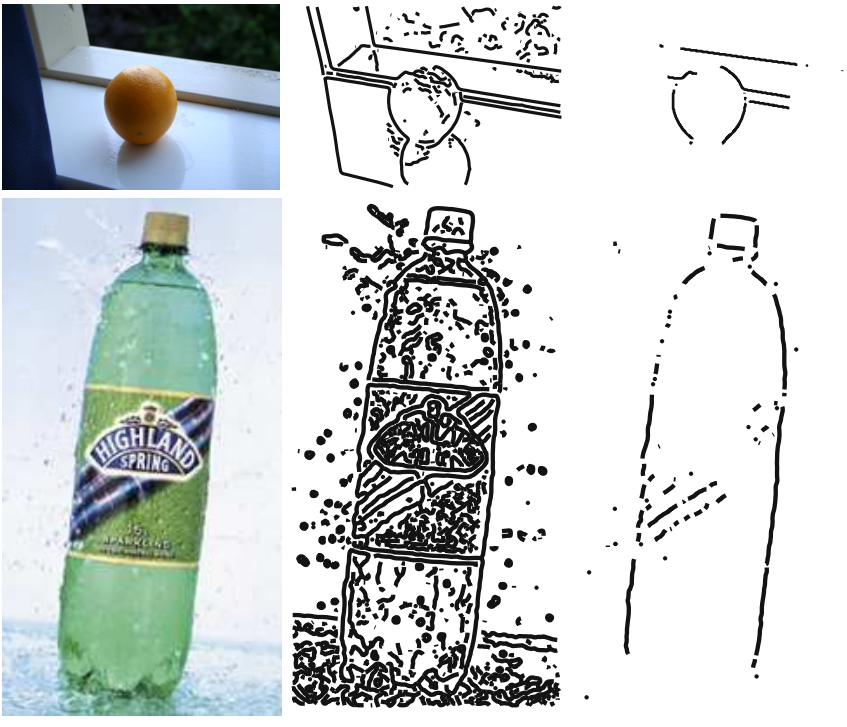
Previous research has considered classifying edges: Carmichael *et al.* [3] learnt edge shape (but not appearance) for mugs; McHenry *et al.* [12,13] built a classifier by hand for recognizing glass by combining a number of cues (e.g. specularities and the similarity between the image regions on either side of the edge); and Sidenbladh and Black [20] learn edge likelihoods for limbs for human detection in video sequences. The methods most closely related to ours are those of Shahrokni *et al.* [18] and Dollar *et al.* [4]. Both these approaches consider each pixel of the image independently and obtain its probability of being an (object class specific) edge. Due to the variation in negative examples, which include regions with no edges, they are forced to employ a large set of features. In contrast, our method classifies only the edges (i.e. not all pixels) which are provided by a standard detector (e.g. canny). This significantly reduces the variability in negative examples. For instance, homogeneous background regions are pruned away by canny. Class-specific edge detection is then obtained using simple local features together with a standard classifier such as the SVM [16,8] which guarantees a global minimum.

The organization of this paper is as follows. In §2, we describe our method for edge classification. We then give two illustrative examples of its use. First for object detection based on chamfer matching in §3, and then for object segmentation in §4 using the ObjCut algorithm of [10].

2 Classifying Edges for an Object Class

In this section we describe how local information can be learnt to classify detected edges into those arising from the boundaries of an object class or not. Our objective is to separate class-specific boundary edges from other image edges — those arising from internal discontinuities, specularities, and background clutter. We illustrate our method on two classes here, oranges and bottles.

We follow the standard machine learning procedure and assemble a set of images which are used to train and evaluate the classifier. We assemble a database for each class of about 100 images. Each dataset is split into half for training and testing. The images cover a wide range of scale, pose and illumination conditions, and include multiple object instances, partial occlusions, and background clutter. Examples are shown in figures 1–3. Edges are obtained using the Canny edge detector with hysteresis. To simplify ground truth annotation, edges are



(a) A simple image (b) Canny edges (c) Validated edges

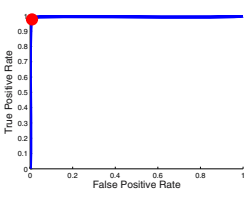
Fig. 1. Overview. The background and internal gradients in (b) throw off chamfer matching. (c) The class-specific validated edges help in removal of edges from clutter and internal gradients. Template matching works better on this edgemap. Note that most of the non-class edges have been suppressed, greatly simplifying subsequent processing such as object detection or class specific segmentation.

linked into chains automatically according to their spatial proximity as shown in figure 1(b), and all edge chains are manually annotated so they are positive if they lie on the boundary of an object instance; all other edge chains are negative.

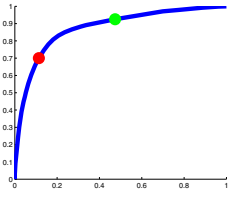
There is then the question of how to represent the appearance (e.g. colour distribution), texture and shape of the edges. At the simplest level we could simply extract a patch around each edge point and use a feature vector consisting of the ordered colour pixels – this would implicitly capture the shape (since the edge boundary runs through the patch). It would also capture the texture since Varma and Zisserman [25] have shown that a simple patch feature is sufficient to classify texture in monochrome images. On the other hand, with such a simple representation we are not explicitly recording that the distributions on each side of the edge may be different. To deal with this point, we do use a simple patch centred on the edge, but rotate the patch so that its x-axis is aligned with the edge tangent (derived from the Canny operator and edge chains). In detail, we

choose a $m \times n$ patch around the edge as our feature. This is rotated so that the edge chain runs horizontally through its centre, giving a rotationally invariant image descriptor. We also record the colour values of each pixel to represent the appearance ([25] only used grey values).

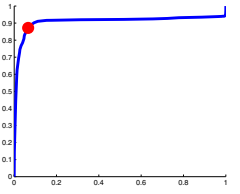
Classification. A Support Vector Machine (SVM) [16] is used to learn an edge classifier for the patch features. The parameters we need to learn are: the size of the patch; and for the SVM: the kernel type (we compare linear, RBF and



(a) Orange

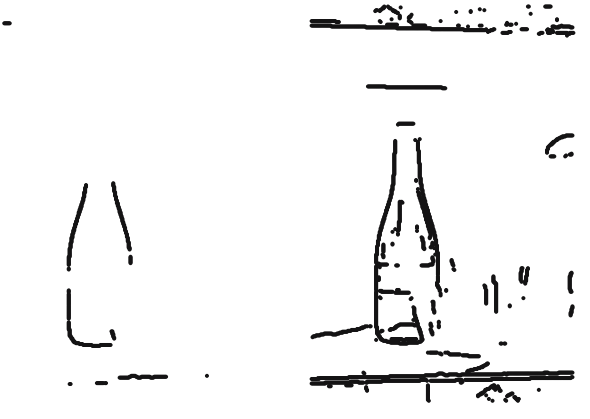


(b) Bottle



(c) Banana

(e) Image



(f)Lower false positive rate (g)Higher false positive rate

Class	Accuracy	Precision	Recall
Orange	98.48%	99.39%	97.57%
Bottle	82.01%	90.03%	72.00%
Banana	90.37%	92.79%	87.53%

Fig. 2. Edge Classification Results. The ROC curve plots the True Positive Rate against the False Positive Rate as the threshold is varied for classification between the minimum and maximum values of the SVM output. (f),(g) show edge classification with a variation in the operating point for the bottle image of (e). In (b) the operating point towards the left results in lower false positives as seen in (f) and a change to the green operating point on the right results in a higher false positive rate (g). The red points on (a),(b) and (c) show the operating point used for the datasets. The classification results at these operating points are given in the table.

polynomial kernels of degree $1, 2$ and 3) and the slack variables. Optimizing over the test data we find that the best performance for the orange and banana datasets is achieved with a polynomial kernel of degree 2, and with patches of size $m = 11$ (in the y direction) and $n = 11$ (in the x direction). For the bottle dataset, the RBF kernel shows superior performance, the size of the patches being the same.

Flip ambiguity. Our patches extracted from the images are rotationally invariant up to a flip factor. The object region can lie either on the top or bottom half of the patch. In the presence of dominant characteristics such as colour and texture, the patches can be flipped to remove this ambiguity. For the orange and banana classes, effective gaussian mixture models of colour characteristics are built from training data for this purpose.

For bottles the colour and texture is much more variable and hence gaussian mixture models for colour will not be helpful in disambiguation. For such categories the classifier will have to handle the ambiguity by choosing the appropriate support vectors and slack variables.

Alternatively, we can try to handle this ambiguity at the kernel level. We experimented with modifying the RBF kernels to internally flip the patches and choose the one which best optimizes the cost. For example, using the kernel

$$k(x, x') = \max\left(\exp\left(-\gamma\|x - x'\|^2\right), \exp\left(-\gamma\|x - \text{flipud}(x')\|^2\right)\right)$$

where $\text{flipud}(x)$ flips the patch vertically, with the intuition that the correct alignment of the patches (flipped or not) will have lower cost due to more consistency of the object region. However, upon experimentation, we find that this kernel (and similar modifications of linear kernels) have slightly inferior performance compared to the standard polynomial and RBF kernels. This difference is heightened if the category has a strong colour model.

For our three classes, the performance of the classifier is summarized in the table of figure 2. The accuracy, recall and precision are defined as

$$\begin{array}{l|l} \text{Accuracy} = \left(\frac{tp+tn}{tp+fp+fn+tn}\right) & \text{tp = True Positive} \\ \text{Precision} = \left(\frac{tp}{tp+fp}\right) & \text{tn = True Negative} \\ \text{Recall} \quad = \left(\frac{tp}{tp+fn}\right) & \text{fp = False positive} \\ & \text{fn = False negative} \end{array} \tag{1}$$

The models are fairly well learnt as can be seen from the receiver operator characteristic curves in the top row of figure 2. In the case of bottles, the lack of one distinctive colour or texture reduces our accuracy. For such object classes, (lacking distinctive colour or texture) other representations may be necessary. For example, texton distributions or separate local colour histograms for each side of the boundary. Other classifiers, such as Adaboost, may also be employed.

The veracity of classification on several example images is shown in figure 3. In subsequent sections we will use both the classification and also the distance from the decision boundary as a confidence measure. The occurrence of false positives and the degree of suppression of true negatives, can be controlled by

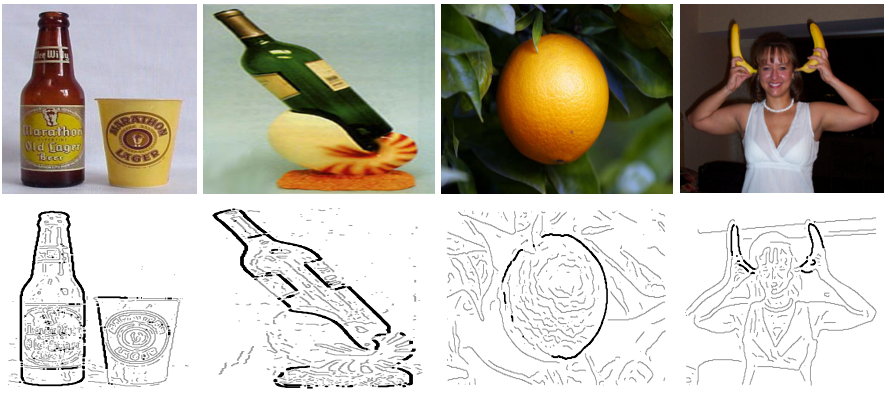


Fig. 3. Edge classification. Example images and the class based edge classifications. A large number of edges from clutter, specularity and internal gradients that confuse template matching are discarded by this classification.

varying the operating point along the ROC curve. Figure 2 shows the varying result of suppression of false edges with the variation of the operating point.

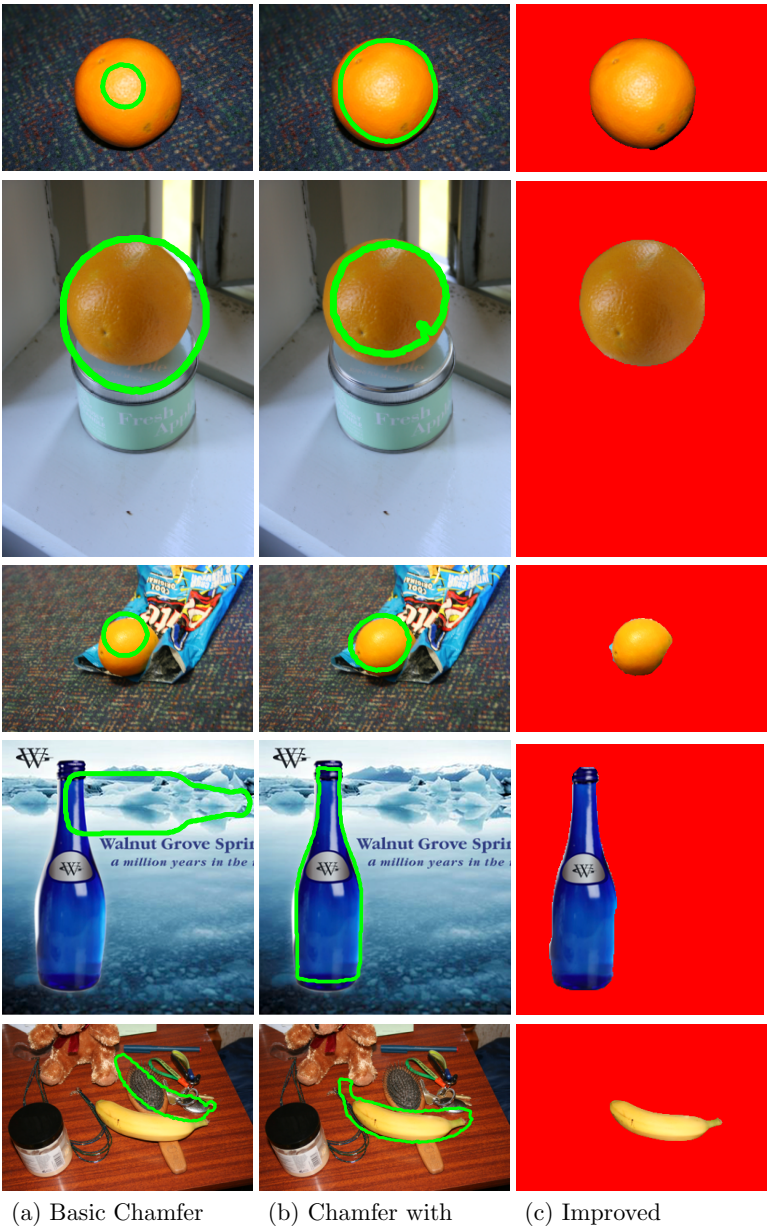
3 Chamfer Matching

In this section we illustrate how edge specific classification can be used to improve the performance of an object detection algorithm based on chamfer matching. In chamfer matching a set of learnt object templates are matched to the detected edges in the image using a distance transform. The position at which the convolution of the template with the distance transform of the feature image (capped at a certain threshold for stability) is minimal, determines the match. Chamfer can be made more robust by taking orientation at edges into account.

In practice, an artificial template database is created by geometrical transformations of exemplar templates, and a hierarchical tree is built to enhance the search speed [6]. Given a test image, hierarchical chamfer matching is then used to fit the object model over the test image. With the occasional modification, this is a standard algorithm for object detection. However, this algorithm applied to the simple orange object class gives numerous and classic mismatches (figure 4 (a)).

3.1 Class Based Chamfer Matching

We use our edge classifier to determine the relevant edges for the current object of interest (see figure 3). Performing chamfer matching on only the positive edges from the classifier output results in a tremendous improvement (figure 4(b)) compared to the original truncated oriented chamfer matching (figure 4(a)) . This can be used to improve any algorithm that uses template matching, such as ObjCut.



(a) Basic Chamfer (b) Chamfer with class-specific edges (c) Improved ObjCut

Fig. 4. Improving Chamfer for object localization. (a) Chamfer Matching using all image edges. The matches latch on to irrelevant edges corresponding to internal gradients and specularities (first row), and clutter (circular lid, second and third row). **(b)** Matching on *class* edges and texture. This leads to better matching – compare with the confusions arising from the problems in (a). **(c)** Modified ObjCut results are much more accurate.

4 Class Based Segmentation — ObjCut

In this section we illustrate how edge specific classification can be used to improve the performance of an object segmentation algorithm. In particular we modify the ObjCut algorithm of Kumar *et al.* [10]. ObjCut is a Bayesian method for class based binary segmentation using Object Category Specific Markov Random Field (MRF) and Pictorial Structures. In practice it is implemented in two stages:

1. Initialization–the edges and texture features in the image are used to determine the object's position in the image (as in §3)
2. Segmentation–the match found is used to initialize the image segmentation. Graph cuts are used to optimize the energy over an Object Category Specific MRF.

We change this method to affect two terms in the ObjCut energy function using class-based edge classification: (i) As in §3 we modify the initialization by chamfer matching to only use class specific edges; and, (ii) We modify the boundary term in the MRF to encourage segmentation along high contrast regions, *but* only if they are relevant to the object class. This is described in more detail below.

4.1 The Boundary Term

Following [2] the MRF used in ObjCut has a contrast dependent prior. This means that a segmentation which introduces a change of state between pixels (i.e. a change from foreground to background in this case) adds a cost to the energy, *but* this cost is diminished if there is a strong gradient between the pixels (as measured in the image/data). The inclusion of data-dependent pairwise terms for pixels in a clique gives a substantial improvement in segmentation quality, and the resulting MRF can still be minimized using graph cuts as described in [2].

We are given an image $\mathbf{D}$ containing an instance of the object. The label at each pixel x is denoted by m_x. We want the algorithm to consider only those edges that are relevant to the object class. Therefore, only those edges of the MRF which coincide with object boundaries are weakened. Our edge classification (§2) gives us a likelihood $edge(x)$, for every pixel x (+ve for valid

Table 1. Average number of misclassified pixels per image

Object class	ObjCut	ObjCut + modified MRF	ObjCut + modified MRF + chamfer matching
Orange	2947	2457	256
Bottle	8121	8064	4077

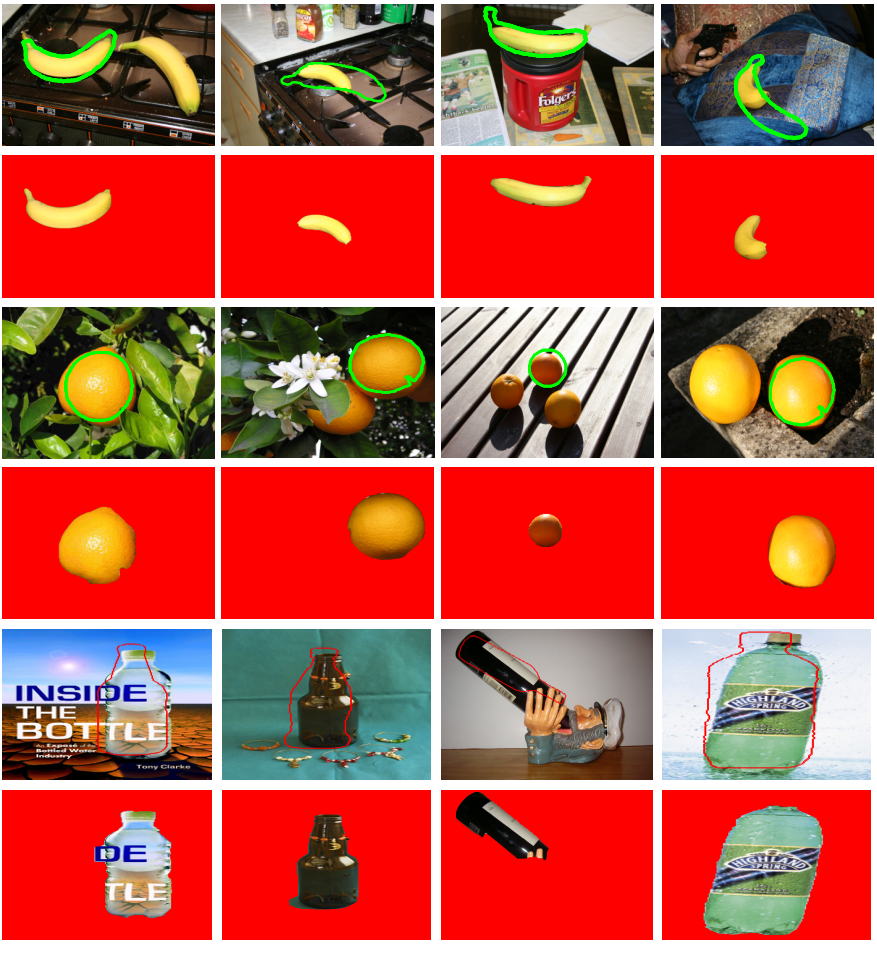


Fig. 5. More results. The performance of our method on some examples is shown on the banana, orange and bottle datasets. This dataset, has a wide range of challenges from pose, scale, clutter and lighting. In the presence of multiple instances, we use the best Chamfer match as shown in Row 1. The segmentation using the initialization from Row 1, 3 and 5, by the improved ObjCut is shown on Row 2, 4 and 6.

boundary, −ve otherwise). A new boundary term is defined, which adds to the category specificity of the MRF:

$$\zeta(\mathbf{D}|m_x, m_y) = \begin{cases} \lambda * \exp(-edge(x)) & \text{if edge exists} \\ \text{constant} & \text{for no edge at } x \end{cases} \quad (2)$$

λ is the parameter controlling the influence of this boundary term.

4.2 Implementation

For ObjCut with the modified category specific MRF and shape model, the optimal parameters must be found. A large number of parameters (around 20) are identified for the model to be learnt. $5-6$ control the relative weights between the colour likelihoods, shape likelihoods, prior and the boundary terms, and are most crucial for performance and strongly interrelated. A simple, gradient descent is performed to optimize performance on the ground truth labelling over this subset of important parameters. Subsequently, the other parameters can be individually optimized in a similar manner. We start with large step sizes for gradient descent and reduce them as we refine our estimates.

4.3 Results

The performance is measured by the number of misclassified pixels in the test data with respect to the manually segmented ground truth. Table 1 summarizes the results for two object classes.

We choose 23 images with single oranges for optimization with respect to ground truth. The basic ObjCut (optimized for performance) yields segmentation over 22 out of the 23 images with an average misclassification of 2947.2 pixels per image. (Note: Each image has an average of $90,000$ pixels). ObjCut with a modified MRF (with the boundary term using only relevant edges) yields segmentation over 22 images with an average misclassification of 2457.3 pixels per image. The final ObjCut with modifications at both the Chamfer matching (top) and MRF (low) levels yields segmentations over all of the 23 images over which we are optimizing. The per image error reduces drastically to 255.5 pixels per image. Note: Each image has $90,000$ pixels on an average. For the orange class, we get visually correct segmentations for 47 out of 50 images. For the bottle class, we get 57 correct segmentations out of 90 images. The banana dataset is our most challenging dataset, owing to the wide shape variations and image clutter. We get good segmentations of around 37 out of 60 images. While both our edge based modifications improve ObjCut, the use of relevant edges in chamfer matching makes the more significant difference (see figures 4,(c) and 5).

5 Conclusion

We have demonstrated the advantages in using class specific edges both for Chamfer matching and segmentation. However, the implications of such class specific edge labelling are many fold — since any algorithm for object classes using edges can now be improved. Examples include tracking, segmentation [22] and recognition [5,14,19]. Of course, the performance of the classifier can be improved and we are currently investigating other feature vectors and classifiers such as boosted trees [24]. We are also interested in finding efficient methods for parameter optimization. This is important for optimal results with algorithms like ObjCut and for experimentation with new kernels.

References

1. S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *Proc. ICCV*, volume 1, pages 454–461, 2001.
2. Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV*, pages 105–112, 2001.
3. O. Carmichael, S. Mahamud, and M. Hebert. Discriminant filters for object recognition. Technical Report CMU-RI-TR-02-09, Carnegie Mellon University, Mar 2002.
4. P. Dollar, T. Zhuowen, and S. Belongie. Supervised learning of edges and object boundaries. In *Proc. CVPR*, 2006.
5. V. Ferrari, T. Tuytelaars, and L.J. Van Gool. Object detection by contour segment networks. pages III: 14–28, 2006.
6. D. Gavrila. Pedestrian detection from a moving vehicle. In *Proc. ECCV*, pages II: 37–49, 2000.
7. D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. ICCV*, pages 102–111, 1987.
8. T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.
9. M. P. Kumar, P. H. S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *Proc. BMVC.*, 2004.
10. M.P. Kumar, P.H.S. Torr, and A. Zisserman. OBJ CUT. In *Proc. CVPR*, pages I: 18–25, 2005.
11. D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987.
12. K. McHenry and J. Ponce. A geodesic active contour framework for finding glass. In *Proc. CVPR*, 2006.
13. K. McHenry, J. Ponce, and D. A. Forsyth. Finding glass. In *Proc. CVPR*, pages 973–979, 2005.
14. A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual alphabet. In *Proc. CVPR*, 2006.
15. C. Rothwell, A. Zisserman, D. Forsyth, and J. Mundy. Canonical frames for planar object recognition. In *Proc. ECCV*, LNCS 588. Springer-Verlag, 1992.
16. B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.
17. E. Seemann, B. Leibe, K. Mikolajczyk, and B. Schiele. An evaluation of local shape-based features for pedestrian detection. In *Proc. BMVC.*, 2005.
18. A. Shahrokni, F. Fleuret, and P. Fua. Classifier-based contour tracking for rigid and deformable objects. In *Proc. BMVC.*, Oxford, UK, 2005.
19. J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *Proc. ICCV*, pages I: 503–510, 2005.
20. H. Sidenbladh and M. Black. Learning image statistics for bayesian tracking. In *Proc. ECCV*, 2001.
21. B. Stenger, A. Thayananthan, P.H.S. Torr, and R. Cipolla. Hand pose estimation using hierarchical detection. In *CVHCI04*, pages 105–116, 2004.
22. J. Sun, W. Zhang, X. Tang, and H. Shum. Background cut. In *Proc. ECCV*, 2006.

23. A. Thayananthan, B. Stenger, P.H.S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proc. CVPR*, pages I: 127–133, 2003.
24. Z. Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *Proc. ICCV*, pages II: 1589–1596, 2005.
25. M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *Proc. CVPR*, volume 2, pages 691–698, Jun 2003.