Superedge Grouping for Object Localization by Combining Appearance and Shape Information

Zhiqi Zhang[1], Sanja Fidler[2], Jarrell Waggoner[1], Yu Cao[1], Sven Dickinson[2], Jeffrey Mark Siskind[3], and Song Wang[1]*
[1]Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA
[2]Department of Computer Science, University of Toronto, Toronto ON, Canada
[3]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA

Abstract

Both appearance and shape play important roles in object localization and object detection. In this paper, we propose a new superedge grouping method for object localization by incorporating both boundary shape and appearance information of objects. Compared with the previous edge grouping methods, the proposed method does not subdivide detected edges into short edgels before grouping. Such long, unsubdivided superedges not only facilitate the incorporation of object shape information into localization, but also increase the robustness against image noise and reduce computation. We identify and address several important problems in achieving the proposed superedge grouping, including gap filling for connecting superedges, accurate encoding of region-based information into individual edges, and the incorporation of object-shape information into object localization. In this paper, we use the bag of visual words technique to quantify the region-based appearance features of the object of interest. We find that the proposed method, by integrating both boundary and region information, can produce better localization performance than previous subwindow search and edge grouping methods on most of the 20 *object categories from the VOC 2007 database. Experiments also show that the proposed method is roughly* 50 *times faster than the previous edge grouping method.*

1. Introduction

Object localization is an important task in computer vision. The major goal of object localization is to determine the exact position of an object of interest by assuming that at least one such object is present in an image. Many state-of-the-art object detection algorithms start with object localization, followed by a more complex recognizer that further verifies whether the object of interest is present at a localized position [23, 12, 15, 10, 26]. As in [14, 1, 27], this paper focuses only on addressing the object localization problem.

Object localization is a very challenging problem because the object to be localized usually defines a wide class of objects with varied appearance and shape. For example, in localizing a "dog" in an image, we actually consider all kinds of dogs, with varied species, size, color, and pose, together with varied view perspectives. The background of the image may also vary from one image to another. Therefore, in object localization, a supervised training algorithm is usually required to identify common features of the object of interest that are distinct from the background features. One state-of-the-art technique for this purpose is the bag of visual words (BOW) [20], which identifies a set of image features and then, with training, associates a score with each feature. This BOW score describes the likelihood of the feature being part of the object of interest.

Early object localization methods aimed to identify an optimal rectangular subwindow (with sides parallel to the image border) around the underlying object [14, 7, 4, 5, 9]. The most popular method may be the efficient subwindow search (ESS) algorithm [14]. By using the branch-and-bound technique, ESS can quickly find the optimal rectangular subwindow in which the covered image features show a maximum total BOW score. This algorithm has been extended to search for a fixed-shape polygonal subwindow in [25]. In [27], an edge grouping method is used for finding optimal free-shape windows that can localize the objects more tightly and accurately. In this method, a set of short edges are detected from the image and then a subset of them are identified and connected into a closed contour that encloses features with a maximum total BOW score. Edge grouping has been investigated by many researchers for other tasks in computer vision [24, 11, 13, 19, 6]. Grouping combining boundary and region information has also been studied by many researchers [22, 16].

The above BOW features mainly reflect the object appearance, but not the object shape. It is widely known that

*Corresponding author: songwang@cec.sc.edu.

both shape and appearance play an important role in human perception. The main goal of this paper is to develop an object localization method by combining both shape and appearance information. We achieve this goal by developing a new superedge grouping method. Differing from edge grouping, where grouping primitives are short edgels (short line segments) from edge detection and curve partitioning, we propose to use long, unsubdivided edges, which we call *superedges*, as grouping primitives. Clearly, such superedges bear richer shape information, with which we can more reliably measure their likelihood to be included in the desirable object boundaries. In addition, compared to edge grouping, the use of superedges substantially reduces the number of grouping primitives and this can increase the robustness against image noise and reduce the computational complexity of object localization.

We need to address a new gap-filling problem when using superedges for grouping: superedges may not be connected at their endpoints in the resulting contour and the resulting contour may include only part of a superedge. This further introduces the problem of encoding region information, such as the total BOW scores, into individual grouping primitives. In this paper, we address these problems by developing new strategies for gap filling and region information encoding. We finally introduce a few simple strategies to incorporate shape information into superedge grouping and show that they can improve the performance of object localization. We evaluate the proposed method for localization on 20 object categories from the PASCAL VOC 2007 database, with comparisons to the ESS algorithm [14] and the edge grouping method [27].

The remainder of the paper is organized as follows. Section 2 reviews edge grouping using ratio contour. Section 3 introduces superedge grouping and its application to object localization. Section 4 reports the experiments on VOC 2007 database. Section 5 concludes the paper.

2. Edge Grouping for Object Localization

In this section, we review *ratio contour*, an edge grouping method that has been used for object localization [27], with better performance than the ESS algorithm. Like other edge grouping methods, ratio contour takes three steps to find an optimal closed contour that is expected to bound the object of interest tightly. First, a set of disjoint edge segments (or edgels) are detected from the image, by edge detection and subdivision, as illustrated in Fig. 1(b). Such segments are referred to as *detected* segments. Second, the detected segments are connected by constructing *gap-filling* segments between each pair of segment endpoints, as illustrated by the dashed lines in Fig. 1(c). Third, a grouping cost function is defined for each closed contour that traverses a subset of detected and gap-filling segments alternately, and a grouping algorithm is used to find the optimal contour with the minimum grouping cost, as illustrated in Fig. 1(d).

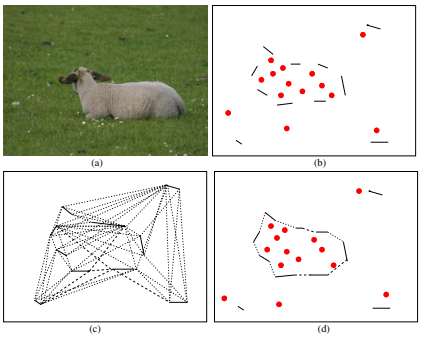

Figure 1. Three steps in applying edge grouping for object localization: (a) Input image, (b) detected segments, (c) gap-filling segments (dashed lines), and (d) the optimal contour for object localization. Red dots indicate BOW features.

To apply ratio contour to object localization, a 2D map M of the same size as the original image is constructed to describe BOW features and scores. If a BOW feature is located at pixel (x, y), $M(x, y)$ is its BOW score: a positive score indicates that this feature is likely to be on the object of interest, while a negative score indicates that this feature is likely to be from the background. If no BOW feature is detected at a pixel (x, y), $M(x, y)$ is set to 0. The grouping cost for object localization is then defined as

$$\phi(C) = \frac{|C_G|}{|\iint_{R(C)} M(x, y)dxdy|}, \tag{1}$$

where C_G is the total length of the gap-filling segments along the contour C and $R(C)$ is the region enclosed by the contour C. The optimal contour C with minimum grouping cost (1) can be found using a graph algorithm in polynomial time [24]. In the denominator of the grouping cost, $\iint_{R(C)} M(x, y)dxdy$ is the total of BOW scores in region $R(C)$, and if it is positive, this region term makes the grouping favor a contour with maximum covered total BOW scores. This is exactly the goal of ESS and other window-search based object localization methods. The numerator $|C_G|$ is a boundary term, which causes the grouping to favor a contour better aligned with detected segments, without limiting the contour shape.

3. Superedge Grouping by Ratio Contour

In most edge grouping approaches, including ratio contour, the grouping primitives are usually short edge segments, which bear no useful shape information. In this section, we extend edge grouping to superedge grouping, where the grouping primitives are long, unsubdivided *superedges*, *i.e.*, we use superedges as the detected segments for grouping. We construct the superedges using edge detection, followed by tracing the connected edge pixels to

form edge segments that are as long as possible. At a "Y" junction, we randomly pick an untraced branch to continue the edge tracing. An example of constructing superedges from a real image is shown in Fig. 2. This edge tracing step also produces many short edge segments, which we usually ignore in superedge construction. Note that the constructed superedges are disjoint and do not share any common edge pixels.

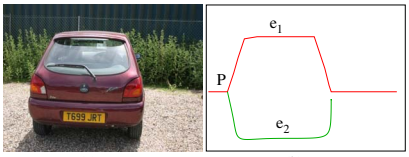

Figure 2. An illustration of superedge construction. (a) A real image, (b) two constructed superedges e_1 and e_2, which are separated at a "Y" junction P.

Using superedges for grouping and object localization has two advantages. First, a long, unsubdivided superedge may bear rich shape information, which facilitates the inclusion of object shape into localization. Second, with the edge subdivision, the number of superedges is usually much smaller than the number of short edge segments used in edge grouping. The complexity of many grouping algorithms, including ratio contour, is a function of the number of grouping primitives. We will show that using superedges as grouping primitives substantially reduces computation time.

There are several new problems in using superedges for grouping. First, with superedges as the detected segments, the construction of gap-filling segments becomes nontrivial. As shown in Fig. 3(b), we may not want a gap-filling segment to go through the endpoints of its connected superedges to obtain the desired contour. Second, the accurate encoding of the denominator of Eq. (1) into individual detected and gap-filling segments becomes a more difficult problem. Such an accurate encoding is important for finding the optimal contour. We will elaborate on this problem in Section 3.2. Third, in principle, shape information can be better included in superedge grouping, as described above. It is still a nontrivial problem to incorporate it into the grouping cost to improve object localization. In the following, we develop specific algorithms to address these three problems in the framework of ratio contour.

3.1. Gap Filling in Superedge Grouping

A superedge, without edge subdivision, may contain both desired boundary fragments of the object of interest and undesired fragments from other objects, textures, or background. The desired optimal contour may only contain part of a superedge, as shown in Fig. 3(b) and (c). We address this problem in the step of constructing gap-filling

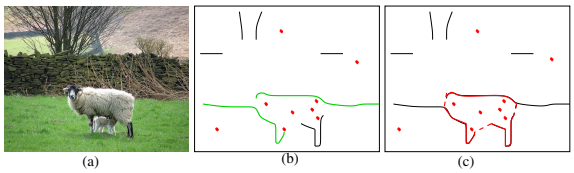

Figure 3. An illustration of the gap-filling problem in superedge grouping. (a) The input image, (b) constructed superedges, (c) the desired contour (shown in red).

segments. Specifically, for each gap-filling segment, we require one of its endpoints to be the endpoint of an adjacent detected segment (*i.e.*, superedge), but allow its other endpoint to be located at any point along the other adjacent detected segment. As shown in Fig. 4(a), we actually construct two gap-filling segments (in dashed lines) to fill the gap between endpoint P of superedge e_1 and endpoint Q of superedge e_2. The first gap-filling segment starts from P and ends at a point Q' on superedge e_2 where Q' is the point on e_2 with the shortest distance to P. Similarly, the second starts from Q and ends at a point P' on superedge e_1 where P' is the point on e_1 with the shortest distance to Q. We do not allow both endpoints of a gap-filling segment to slide freely along the adjacent superedges by minimizing the distance between them, because this will make gap filling completely uncontrollable and sensitive to noise, as shown in Fig. 4(b). In this paper, we use the same approach to construct a gap-filling segment between the two endpoints of the same detected segment, as illustrated in Fig. 4(c).

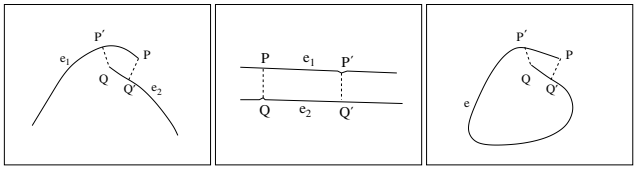

Figure 4. An illustration of constructing gap-filling segments between two superedges. (a) Two gap-filling segments are constructed between a pair of endpoints P and Q. (b) The problem of allowing both endpoints to slide freely on the adjacent detected segments (*i.e.*, superedges). (c) gap filling between the endpoints of the same detected segment.

In this paper, we simply construct gap-filling segments as straight lines, which might not be sufficient for an accurate image segmentation by localizing a contour that is fully aligned with the object boundaries. But it is usually sufficient for object localization, where we only need a certain region overlap between $R(C)$ and the region occupied by the object. As in edge grouping, we initially do not know which pair of superedges need to be included and connected in the desired contour. We need to construct gap-filling segments between every two possible endpoints, including two endpoints of the same superedge and two endpoints from different superedges.

3.2. Encoding Region Term into Individual Segments

The key step in the ratio contour algorithm is to encode both the boundary term (numerator) and the region term (denominator) of the grouping cost (1) into the detected and gap-filling segments in the form of two weight functions w_1 and w_2 associated with individual segments. Particularly, the requirement for this encoding is that the simple summation of the weight w_1 (or w_2) of the segments along a contour C is equal to the boundary term (or the region term) [24, 27]. In this section, we show that, by using the superedge-based gap-filling method described in Section 3.1, new problems occur in the region term encoding and we then develop new algorithms to address them. To better depict this problem, we first briefly review the boundary term and region term encoding process in edge grouping.

The boundary term encoding is straightforward. Weight w_1 for each segment takes the value of the segment length if it is a gap-filling segment and zero otherwise. This way, the total weight w_1 along the contour C is exactly the numerator of Eq. (1).

The region term encoding process is illustrated in Fig. 5. By assuming that each detected and gap-filling segment is a short line and that they are connected only through their endpoints, w_2 for each segment e with two endpoints P and Q is defined by considering the direction and the relative position between two endpoints. For e, two mirrored weights $w_2(P \to Q)$ and $w_2(Q \to P)$ are defined with an identical magnitude, but opposite signs. In particular, $w_2(P \to Q) = -w_2(Q \to P) = \iint_{R(e)} M(x,y)dxdy$ if P is located to the left of Q, and $w_2(P \to Q) = -w_2(Q \to P) = -\iint_{R(e)} M(x,y)dxdy$, otherwise, where $R(e)$ is the region enclosed by e and its projection to the bottom side of the image, as shown in Fig. 5(a). Note that if P and Q are on the same vertical line, the weight w_2 is zero and the sign is not important. This way, as shown in Fig. 5(b), following the clockwise direction for contour C, the total weight w_2 along C is exactly the region term $\iint_{R(C)} M(x,y)dxdy$, *i.e.*,

$$\iint_{R(C)} M(x,y)dxdy = \sum_{i=1}^{5} w_2(P_i \to P_{i+1}) + w_2(P_6 \to P_1).$$

This completes the region term encoding.

In superedge grouping, the above region information encoding technique cannot be directly applied to detected segments (*i.e.*, superedges) because a superedge may be partially included in the final contour C. In addition, different points on a superedge may be identified when it is connected to different superedges. For example, when constructing a gap-filling segment e_4 between e_1 and e_2 in Fig. 6(a), the fragment between A and B should be left out in calculating $w_2(e_1)$. When constructing a gap-filling segment e_5 between e_1 and e_3 in Fig. 6(a), the fragment between A and C should be left out in calculating $w_2(e_1)$. This means that we may not be able to define a unique weight $w_2(e_1)$ and the region term $\iint_{R(C)} M(x,y)dxdy$ cannot be correctly encoded into individual segments.

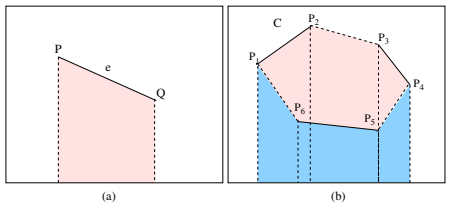

Figure 5. An illustration of encoding the region term to individual segments in edge grouping. (a) Weight w_2 of a segment reflects the total BOW scores below this segment in the image. (b) The region term is equal to the total weight w_2 along the contour C, by following a clockwise direction.

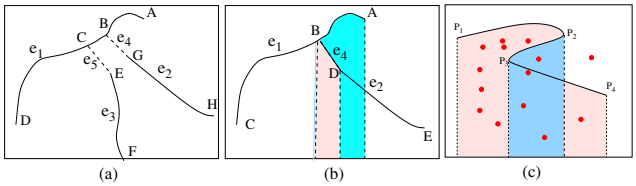

Figure 6. An illustration of the region term encoding in superedge grouping. (a) Two different points B and C are identified from superedge e_1 when connecting to different superedges e_2 and e_3, and they prevent the use of the initial region term encoding technique in edge grouping. (b) Updating the w_2 for gap-filling segments to address the problem shown in (a). (c) Defining weight w_2 for a superedge with horizontal turnings.

To solve this problem, we update the weight w_2 to w_2' for gap-filling segments by negatively including the left-out fragments of its adjacent detected segments. As shown in Fig. 6(b), we define a new weight w_2' for gap-filling segment e_4 as

$$w_2'(B \to D) = -w_2'(D \to B) = w_2(B \to D) + w_2(A \to B),$$

where w_2 is of the same definition as described above. For detected segment e_1, its weight w_2 is defined as in edge grouping without leaving out any fragments. This way, when adding over e_1, e_2 and e_4, the contribution from the fragment between A and B will be canceled out as

$$\begin{aligned} & w_2(C \to A) + w_2'(B \to D) + w_2(D \to E) \\ = \; & w_2(C \to B) + w_2(B \to D) + w_2(D \to E). \end{aligned}$$

Another issue is that a superedge is not straight and there may be horizontal turnings as shown in Fig. 6(c), which confounds the definition of w_2. We solve this problem by finding these turnings and define its weight w_2 by summing

up the contributions from the fragments defined by these turning points. For example, for a superedge e with endpoints P_1 and P_4 as shown in Fig. 6(c), we define its weight

$$w_2(P_1 \to P_4) = -w_2(P_4 \to P_1) = \sum_{i=1}^{3} w_2(P_i \to P_{i+1}).$$

3.3. Considering Object Shape in Superedge Grouping

Shape plays an important role in defining an object or an object class. Unlike previous edge grouping methods used in object localization [27] which dealt with short nondistinctive edge segments, superedges enable the inclusion of object shape in grouping in a natural way. For example, given their length and complexity, we can compare each superedge, or each pair of superedges (identified by the gap-filling segment between them), to template shapes of an object of interest to define their similarity. If this similarity is high, we can intentionally weight this superedge (or superedge pair) to make it more likely to be included in the optimal contour. If this similarity is very low, we can do the opposite: weight it to make it less likely to be included in the optimal contour. Such a shape comparison can be formulated as a partial shape matching problem [21, 18, 2, 3, 8]. However, when the shape of the objects in the same category varies substantially, such as those shown in the VOC datasets, it is very difficult to construct a template shape or shape space for each object category.

The easiest way to incorporate the shape of the superedges into ratio contour is to update the weight w_1 of each superedge by considering its likelihood to belong to the object category of interest. However, during the grouping process, we also want to avoid linking those superedges such that the shape of two connected superedges is unlikely to come from the object. We can address this problem by further updating the weight w_1 of each gap-filling segment that defines the connection of two superedges. Specifically, we can update w_1 by adding a penalty term that reflects the unlikelihood that these two connected superedges belong to the boundary of the desirable object. The proposed superedge grouping method for object localization can be summarized in Algorithm 1.

As an instantiation of this idea, we implemented two simple strategies to incorporate shape information that is suitable for localizing certain classes of objects. The basic idea is to learn from a training set whether long straight-line superedges and parallel superedge pairs, which are typical shape features of the man-made objects as shown in Fig. 7, are likely to be from the object boundary, and then use this learned knowledge to re-weight the superedges or superedge pairs accordingly to achieve better object localization. For each object category, our learning procedure consists of the following steps: 1) in the training set, count

Algorithm 1 Input image I

1: Construct the BOW map M from the image I.
2: Construct the detected segments (*i.e.*, superedges) from I.
3: Construct the gap filling segments between detected segments.
4: Encode boundary and region terms to individual segments, in the form of weights w_1 and w_2.
5: Update w_1 by considering shape information as described above.
6: Run the ratio-contour algorithm to find the optimal contours as the localized objects.

the total number of long straight-line superedges that are from the desired object boundaries and background, respectively; 2) in the training set, count the total number of parallel superedge pairs that are from the desired object boundaries and background, respectively. For both of them, we check the percentage of the counted numbers that are from the object boundaries. If these percentages are low for an object category, we should add a penalty term to the edge weight w_1 in connecting straight-line superedges or parallel superedge pairs in object localization.

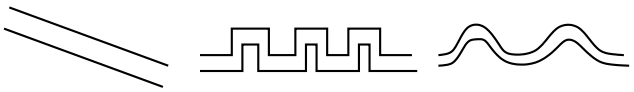

Figure 7. Pairs of long parallel curves that are unlikely to be drawn from the boundary of an animal.

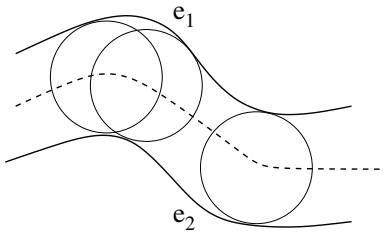

Figure 8. An illustration of the method for evaluating the parallelism of two curves.

For each detected segment or a pair of detected segments connected by a gap-filling segment, we check its likeliness to be a long, straight line. For the latter, it actually represents a longer curve that combines the two adjacent detected segments (excluding the left-out fragment) and the gap-filling segment in between. Specifically, we use the line-fitting algorithm to identify the straight lines. If the fitting error is less than a given threshold, we consider it to be a straight line. We scale the length of this straight line and add it to the weight w_1 of the involved gap-filling segment. For all our experiments, we choose the threshold of 1.5 pixels (maximum deviation from the straight line) for

	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow
Total # Straight Lines	539	486	284	341	334	437	1435	511	890	108
# Lines from Boundary	86	11	1	60	9	69	64	28	30	1
% Lines from Boundary	0.16	0.023	0.004	0.176	0.027	0.158	0.0446	0.055	0.034	0.0093
Total # Parallel Curves	4638	3835	2861	2929	3421	4409	15831	6047	11270	756
# Parallels from Boundary	102	4	7	26	21	73	101	1	48	3
% Parallels from Boundary	0.021	0.0010	0.0024	0.0089	0.006	0.0166	0.0064	0.00017	0.0043	0.0040
	Dining Table	Dog	Horse	Motorbike	Person	Potted plant	Sheep	Sofa	Train	TV/Monitor
Total # Straight Lines	348	448	334	381	2995	508	55	460	721	661
# Lines from Boundary	21	10	4	3	34	1	1	54	133	98
% Lines from Boundary	0.063	0.022	0.012	0.0079	0.011	0.0020	0.018	0.117	0.18	0.14
Total # Parallel Curves	4662	4092	4991	3401	31677	6575	334	6100	5231	9367
# Parallels from Boundary	10	17	9	7	74	9	1	36	71	135
% Parallels from Boundary	0.0021	0.0042	0.0018	0.0021	0.0023	0.0014	0.0030	0.0059	0.014	0.014

Table 1. Simple statistics on the VOC2007 training set to identify the object categories for applying Strategies 1 and/or 2 for incorporating shape information.

the fitting error. Rows $1-3$ in Table 1 show the number of the straight-line superedges that are from the desired object boundaries and the background, respectively, by processing all the images in the VOC 2007 training set. Based on the percentages of such superedges that are from the object boundaries, we re-weight edge weight w_1 to penalize straight-line superedges when localizing all the categories of objects except for "Aeroplane", "Boat", "Bus', "Sofa", "Train" and "TV/monitor". In our experiments, we add a penalty of $0.7 \cdot l$ where l is the length of a considered superedge (or superedge pairs with gap filling) to w_1 of the involved gap-filling segment when the considered superedges (or superedge pairs with gap filling) are identified as long straight lines. In this penalty, the parameter 0.7 is selected to maximize the localization rate on the training and validation sets. In the following, we refer to this re-weighting procedure as Strategy 1 for incorporating shape information.

For every pair of superedges, we check their parallelism by finding their inscribed circles with centers along the medial axis, as shown in Fig. 8. We denote the largest radius among these circles to be $r_{\max}$ and the smallest radius among these circles to be $r_{\min}$. If $\frac{r_{\max}-r_{\min}}{r_{\max}} < T$, we claim they are parallel curves. In our experiments, we always set $T = 0.1$. Rows $4-6$ in Table 1 show the numbers of parallel superedge pairs that are from the desired object boundaries and the background, respectively, by processing all the images in the VOC 2007 training set. Based on the percentages of such superedge pairs that are from the object boundaries, we re-weight edge weight w_1 to penalize superedge pairs when localizing all the categories of objects except for "Aeroplane", "Boat", "Bottle", "Bus', "Car", "Sofa", "Train" and "TV/monitor". In our experiments, we scale the parallel-curve length and add it to weight w_1 of the involved segments. In the following, we refer to this re-weighting procedure as Strategy 2 for incorporating shape information.

4. Experiments

In the experiments, we use the proposed method to localize the 20 categories of objects from the PASCAL VOC2007 database and compare its performance with the free-shape subwindow search method [27] and the ESS algorithm [14]. VOC 2007 contains $9,963$ images and $24,640$ object instances. Images in the database are divided into three sets for training, validation and testing and this experiment reports the localization rates on the testing set. For all these methods, we only allow the localization of one instance of each object class per image to make it a fair comparison. The localization rate for each object class is the ratio between the correctly localized instances of this object and the total instances of this object in the whole test set [14, 27]. As in [27], we use the PASCAL intersection-over-union 0.5 criterion to decide whether an object is correctly localized and we take the tightest rectangular box over the optimal contour found by the proposed method for evaluation.

To obtain the BOW map M, we extract SIFT features from the training and validation images, and randomly select $150,000$ SIFT features and quantize them into $3,000$ visual words by the K-means clustering algorithm, as in [27]. We use the features located inside the ground-truth object boundaries as positive training samples and use the features in the background as the negative training samples to train the BOW score using a support vector machine (SVM). The ground-truth object boundaries are obtained by manual labeling. We apply the Berkeley edge detector [17] as used in other edge grouping methods [27] to construct the superedges, using its default settings. All the edges that are shorter than 30 pixels after the edge tracing are ignored.

We test two versions of the proposed method: "Proposed-1" which implements all the steps in Algorithm 1 and "Proposed-2" that disables the shape-incorporation step (Step 5 of Algorithm 1). The row of "Shape Strategies"

Method	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow
Shape Strategies	-	S1+S2	S1+S2	-	S1	-	S1	S1+S2	S1+S2	S1+S2
Proposed-1	-	0.240	**0.240**	-	**0.0384**	-	0.306	**0.614**	**0.077**	**0.250**
Proposed-2	**0.379**	0.229	0.229	**0.205**	0.036	**0.394**	**0.308**	0.573	0.075	0.242
Edge Grouping [27]	0.322	0.216	0.212	0.164	0.021	0.305	0.292	0.433	0.064	0.217
ESS [14]	0.327	**0.255**	0.163	0.148	0.032	0.347	0.277	0.422	0.046	0.176
Method	Dining Table	Dog	Horse	Motorbike	Person	Potted plant	Sheep	Sofa	Train	TV/Monitor
Shape Strategies	S1+S2	S1+S2	S1+S2	S1+S2	S1+S2	S1+S2	S1+S2	-	-	-
Proposed-1	0.306	**0.499**	**0.454**	**0.379**	**0.125**	**0.063**	**0.149**	-	-	-
Proposed-2	0.267	0.479	0.402	0.363	0.119	0.056	0.136	0.406	**0.482**	**0.218**
Edge Grouping [27]	0.238	0.419	0.398	0.308	0.093	0.058	0.132	**0.469**	0.447	0.188
ESS [14]	**0.354**	0.389	0.388	0.369	0.102	0.040	0.095	0.460	0.443	0.149

Table 2. The object localization rate of the proposed method, the free-shape subwindow search method using edge grouping [27] and the ESS algorithm [14] on VOC 2007 database. In the row of "Shape Strategies", S1 stands for Strategy 1, and S2 stands for Strategy 2 described in Section 3.3. Note that the classes with label "-" are those we apply neither shape strategies according to the results in Table 1.

in Table 2 lists the shape-incorporation strategies used in "Proposed-1" for each object category, as discussed in Section 3.3. Table 2 shows the localization rate for the 20 categories of objects in VOC 2007 database. For comparison, Table 2 also includes the localization rates from two available methods: the free-shape subwindow search method using edge grouping [27] and the ESS algorithm [14], using the same testing data, the same BOW features and scores, and the same evaluation criteria. We can see that "Proposed-1" outperforms ESS on 17 categories out of 20 categories and the edge grouping method on 19 categories out of 20 categories. Even without the shape-incorporation step, "Proposed-2" still outperforms ESS and the edge grouping method on 16 and 18 categories, respectively.

According to Table 1, for 14 object categories (bicycle, bird cat, chair, cow, diningtable, dog, horse, sheep, pottedplant, motorbike, person, car and bottle) we need to penalize either long straight lines, parallel curves or both, as described in Section 3.3. Table 2 shows that, for 13 out of these 14 categories, "Proposed-1" outperforms "Proposed-2" and this clearly shows the effectiveness of the proposed simple shape-incorporating strategies discussed in Section 3.3. Figure 9 shows the sample localization results from the proposed method, the edge grouping method and the ESS algorithm. For the proposed method and the edge grouping method, we show the detected optimal contour. But be reminded that the rectangular bounding box around these contours is used for evaluating whether the localization is correct or not, and the localization rates are reported in Table 2.

Finally, we compare the running time of the proposed superedge grouping method and the edge grouping method [27]. Specifically, for the proposed method, we take "Proposed-1" since it is more time-consuming than "Proposed-2", by incorporating shape information. From Table 3, we can see the proposed method is 50 times faster than the edge grouping method [27]. The major reason for this speedup is that the number of constructed superedges is much smaller than the number of constructed edges, as shown in Table 3. The reported CPU time is obtained on a 2GHz Linux workstation with 8 GB of RAM. For both methods, the reported CPU time includes both detected/gap-filling segment construction and grouping. Note that the proposed method is primarily implemented in Matlab and the running time can be substantially reduced by using C or C++ implementations.

5. Conclusion

In this paper, we introduced a new superedge grouping method for more robust object localization. Like the previous edge grouping method [27], this new method can find a free-shape subwindow that localizes the object of interest more tightly and accurately. In particular, superedges bear much richer shape information and can facilitate the inclusion of object shape into its localization. We developed two strategies to incorporate the object shape information into the proposed superedge grouping method. In addition, this proposed method substantially reduces the number of grouping primitives when compared to the edge grouping methods. Experiments showed that the proposed method outperforms the previous edge grouping method on 19 out of 20 object categories and the ESS algorithm on 17 out of 20 object categories from the PASCAL VOC2007 dataset. Experiments also showed that the proposed method is roughly 50 times faster than the previous edge grouping method. In the future, we plan to develop more effective shape learning approaches to decide how likely a superedge (or a superedge pair) comes from the boundary of a class of objects.

Acknowledgments

This work was supported, in part, by AFOSR FA9550-11-1-0327, NSF IIS-0951754, NSF IIS-1017199, and

Method	cat	cow	horse	sheep	dog	bird	person
Proposed-1 (CPU time per Image)	**23.1s**	**16.3s**	**31.5**s	**9.95s**	**16.5s**	**22.9s**	**43.2s**
Average Number of **Superedges** per Image	**65.0**	**52.5**	**65.1**	**47.6**	**53.8**	**59.8**	**68.8**
Edge Grouping [27] (CPU time per Image)	1130.5s	775.3s	1605.3s	523.3s	990.2s	1896.3s	2300.6s
Average Number of Edges per Image	314.2	339.0	376.6	300.7	321.8	373.2	386.2

Table 3. Running time and the number of grouping primitives of the proposed superedge grouping method and the free-shape subwindow search method using edge grouping [27].

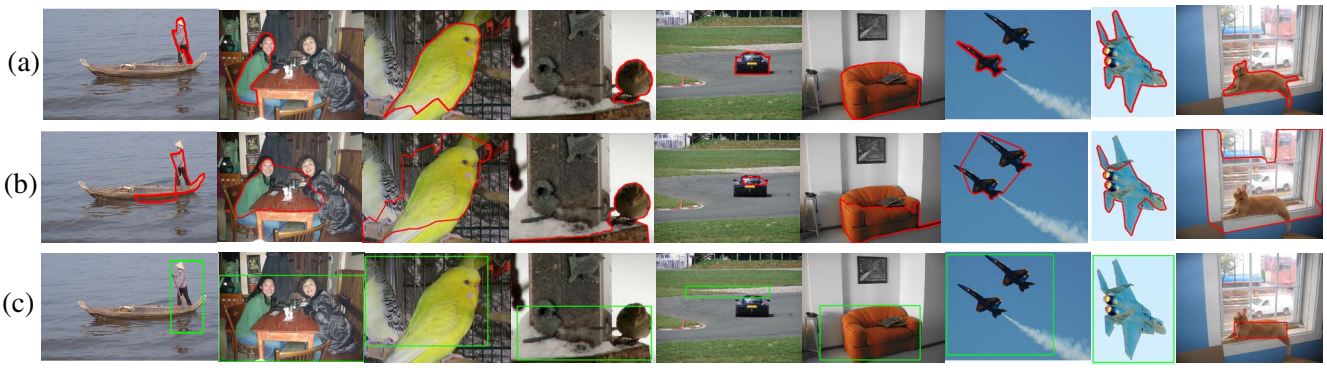

Figure 9. Localization results on sample images using (a) the proposed superedge grouping method ("Proposed-1"), (b) the free-shape subwindow search method using edge grouping [27], and (c) the ESS algorithm [14].

ARL under Cooperative Agreement Number W911NF-10-2-0060. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either express or implied, of AFOSR, NSF, ARL or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

References

[1] S. An, P. Peursum, W. Liu, and S. Venkatesh. Efficient algorithms for subwindow search in object detection and localization. In *CVPR*, 2009.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape maching and object recognition using shape contexts. In *PAMI*, 2002.

[3] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. In *PAMI*, 1988.

[4] O. Chum and A. Zisserman. An exemplar model for learning object class. In *CVPR*, 2007.

[5] N. Dalal and B. Triggs. Histogram of oriented gradients for human detection. In *CVPR*, 2005.

[6] J. Elder and S. Zucker. Computing contour closure. In *ECCV*, pages 399–412, 1996.

[7] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *PAMI*, 30(1):36–51, 2008.

[8] S. Fidler, M. Boben, and A. Leonardis. A coarse-to-fine taxonomy of constellations for fast multi-class object detection. In *ECCV*, 2010.

[9] M. Fritz and B. Schiele. Decomposition, discovery and detection of visual categories using topic models. In *CVPR*, 2008.

[10] J. Gemert, J. Geusebroek, C. Veenman, and A. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, 2008.

[11] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *IJCV*, 20(1):113–133, 1996.

[12] H. Harazllah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.

[13] D. Jacobs. Robust and efficient detection of convex groups. *PAMI*, 18(1):23–27, 1996.

[14] C. Lampert, M. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008.

[15] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[16] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Optimal contour closure by superpixel grouping. In *ECCV*, 2010.

[17] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *CVPR*, 2008.

[18] M. Marsza and C. Schmid. Accurate object localization with shape masks. In *ICCV*, jun 2007.

[19] S. Sarkar and P. Soundararajan. Supervised learning of large perceptual organization: Graph spectral partitioning and learning automata. *PAMI*, 22(5):504–525, 2000.

[20] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[21] P. Srinivasan, Q. Zhu, and J. Shi. Many-to-one contour matching for describing and discriminating object shape. In *CVPR*, 2010.

[22] J. Stahl and S. Wang. Edge grouping combining boundary and region information. *IEEE TIP*, 16(10):2590–2606, 2007.

[23] A. Vedaldi, V. Gulsha, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *CVPR*, 2010.

[24] S. Wang, T. Kubota, J. Siskind, and J. Wang. Salient closed boundary extraction with ratio contour. *PAMI*, 27(4):546–561, 2005.

[25] T. Yeh, J. Lee, and T. Darrell. Fast concurrent object localization and recognition. In *CVPR*, 2009.

[26] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, 73(2):213–238, 2007.

[27] Z. Zhang, Y. Cao, D. Salvi, K. Oliver, J. Waggoner, and S. Wang. Free-shape subwindow search for object localization. In *CVPR*, 2010.