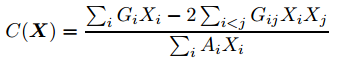
Optimal Contour Closure by Superpixel Grouping

To find the contour closure of our object in each picture, we use the algorithm proposed by Alex et al. In stead of finding a circle of disconnected contour fragments that separate an object from its background, this method focuses on finding subsets of superpixels whose collective boundary has strong edge support in the image. It develops a cost function, which is a ratio of a novel learned boundary gap measure to area, to promotes spatially coherent sets of superpixels. Moreover, its properties support a global optimization procedure using parametric maxflow.

**Problem Formulation**

The algorithm develops a framework that reduces grouping complexity by restricting closure to lie along superpixel boundaries. Given a contour image, it first segments it into N superpixels using a modified version of the superpixel segmentation method of Mori et al. [25] (uses the Pb edge detector [26] while it uses globalPb [24]). Let be a binary indicator variable for the i-th superpixel, the vector yields a full labeling of the superpixels of as figure (1) or ground (0). Since the goal is to select a maximal set of superpixels which have high spatial coherence and whose boundary has strong contour support in the image, drawing on Stahl and Wang [4], it defines the closure cost to be , where is the boundary gap along the perimeter of (the “on” superpixels of) X, and is its area. Boundary gap is a measure of the disagreement between the boundary of X and is defined to be , where is the perimeter of X and is the “edginess” of the boundary of X. Out of the total number of pixels along the boundary of edginess is the number of edge pixels, with the edginess of image boundary pixels defined to be 0.

In order to facilitate the optimization of this cost using an optimal graph cut-based approach, it decomposes the cost function into unary and pairwise terms of the variables in X. Let be the perimeter length of superpixel i and let be the length of the shared edge between superpixels i and j. Similarly, let be the edginess of superpixel i’s boundary, and be the edginess for the shared boundary between superpixels. Let the superpixel and superpixel edge gaps be and respectively. Finally, let be the area of superpixel i. Our closure cost becomes:



The denominator in the above ratio simply adds the individual areas of all the superpixels that were selected. Normalization by area not only promotes spatial coherence but also promotes compactness; given two possible paths (with strong edge support) a closure may take, it will prefer a compact path over one with deep concavities. The numerator in the above cost is more complicated. To compute the gap along the perimeter, it first adds the individual gaps of all the selected superpixels. However, for selected superpixels that share boundaries, adding individual superpixel gaps would add gaps that are not on the boundary of the selection. For every internal boundary, the gap over that boundary was counted twice (once for each of the superpixels that share the boundary). Therefore, it subtracts the gap twice for all internal boundaries. Note that if two superpixels do not have a shared boundary, then both and (and thus ) will be 0. The figure below gives an example of gap computation over a simple superpixel graph.

**Learning the Gap Measure**

Most approaches to detecting contour closure typically define gap as simply the length of the missing contour fragments, i.e., the length of that portion of the closure for which no image edges exist. In order to ground the gap measure using image evidence, as well as incorporate multiple contour features for gap computation, this method chooses to learn the gap from ground truth. For a pair of superpixels i and j, the gap on the edge between them is . Specifically, if is the set of pixels on the superpixel edge (i, j), then and , where is an edge indicator for pixel p (P (·) is a logistic regressor and f p is a feature vector for the pixel p). is a necessary threshold to measure the edginess. Since the distribution of edges in the training set is not necessarily the same as that for test images, this parameter controls the contribution of weak edges. Decreasing it results in many smaller structures being detected and causes more potential solutions to be generated.

Given a pixel p on the superpixel boundary, the feature vector is a function of both the local geometry of the superpixel boundary and the detected image edge evidence in the neighborhood of the superpixel boundary pixel. This feature vector consists of four features:

1. Distance to the nearest image edge; closer edges provide stronger evidence.

2. Strength of the nearest image edge; stronger edges provide stronger evidence.

3. Alignment between the tangent to the superpixel boundary pixel and the tangent to the nearest image edge; aligned edges provide stronger evidence.

4. Squared curvature of the superpixel edge at a point.

Given a dataset of images with manually labeled figure/ground, it maps the ground truth onto superpixels. The training set is composed of all the pixels falling on superpixel boundaries and is used to train a logistic classifier over a feature vector . In addition to learning from all four of the above features, it tried learning from subsets of the features. The thickness of each superpixel edge corresponds to the average edge probability of its superpixel boundary pixels.

Using all four features results in the best performance, in terms of retaining object boundary edges while suppressing other edges.

**Optimizing Framework**

In the case of binary variables, ratio minimization can be reduced to solving a parametric maxflow problem. Kolmogorov et al. [5] showed that under certain constraints on the ratio R(x), the energy E(x, λ) is submodular and can thus be minimized globally in polynomial time using min-cuts. Converting our closure cost C(X) to a parametrized difference results in a submodular cost C(X, λ), making the method in [5] applicable for minimizing the ratio C(X). In fact, the method in [5] does not simply optimize the ratio R(x), but finds all intervals of λ (and the corresponding x) for which the solution x remains constant. The interval boundaries are called breakpoints, and while the smallest breakpoint corresponds to the optimal ratio R(x), consecutively larger breakpoints λ1, λ2, . . . are also related to ratio optimization. Kolmogorov et al. show that the optimal solution x∗ of E(x, λ) in the interval [λi, λi+1], is also an optimal solution of , where T = Q(x∗). In case of optimizing the closure cost, using parametric maxflow results in a multiscale set of optimal closure solutions under increasing area thresholds.