“Lattice Cut” - Constructing superpixels using layer constraints

Alastair P. Moore  Simon J. D. Prince*  Jonathan Warrell†

University College London  Gower Street, WC1E 6BT, UK
{a.moore, s.prince}@cs.ucl.ac.uk
† Oxford Brookes University  Oxford, OX3 3JX, UK  jwarrell@brookes.ac.uk

Abstract

Unsupervised over-segmentation of an image into superpixels is a common preprocessing step for image parsing algorithms. Superpixels are used as both regions of support for feature vectors and as a starting point for the final segmentation. Recent algorithms that construct superpixels that conform to a regular grid (or superpixel lattice) have used greedy solutions. In this paper we show that we can construct a globally optimal solution in either the horizontal or vertical direction using a single graph cut. The solution takes into account both edges in the image, and the coherence of the resulting superpixel regions. We show that our method outperforms existing algorithms for computing superpixel lattices. Additionally, we show that performance can be comparable or better than other contemporary segmentation algorithms which are not constrained to produce a lattice.

1. Introduction

Over-segmentation has become a standard preprocessing step in many computer vision pipelines [13, 14, 16, 25] for two reasons: first, small regions of the image, or superpixels, can be used as regions of support for feature vectors. In this regard, superpixels are superior to fixed rectangular windows as they conform to the local structure of the image and so are less likely to span more than one object. Second, reducing the number of primitives from $\sim250000$ pixels to $\sim400$ superpixels allows modeling long range interactions [2] and speeds up subsequent inference about the scene.

Recent work has investigated using a regular lattice of superpixels [23, 29]. A lattice confers several benefits. First, most existing vision algorithms work with a regular Cartesian grid of pixels and can be adapted easily for use with a regular grid of superpixels without the need for more general graph algorithms [12]. Second, some algorithms exploit the special properties of a regular lattice either for efficiency [26] or tractability [7]. Third, and most importantly, a regular grid allows us to learn higher-order relations between object labels in nearby superpixels: this is very difficult if the segmentation topology changes.

Unfortunately, current over-segmentation algorithms are inadequate, even when they are not constrained to produce a lattice: the resulting superpixels do not always conform to the edges of real world objects. This has forced some authors to use multiple segmentations as a starting point for further scene interpretation [13, 16, 20, 28].

One weakness of current methods is that they either use pixel color statistics [8, 11] or are based on a pre-computed

---

*We acknowledge the support of the EPSRC Grant No. EP/E013309/1.
The data terms $D_p(x_p)$ represent the penalty for pixel $p$ having label $x_p$ and the regularization terms $V_{pq}(x_p, x_q)$, defined on a neighborhood system $\mathcal{N}$, encourage spatial smoothness.

$$E(x) = \sum_{p \in P} D_p(x_p) + \sum_{p,q \in \mathcal{N}} V_{pq}(x_p, x_q)$$  \hspace{1cm} (1)
We demonstrate the main ideas with a simple example. Consider two pixels at sites \( p \) and \( q \), each of which can take one of three labels \( x_p, x_q \in \{0, 1, 2\} \). In the associated graph construction (Figure 3a), the two columns represent the two pixels \( p \) and \( q \). For each column there are three downward pointing links, each of which represents one of the three possible labels at that pixel. In a minimum cut that separates source from sink we must cut exactly one of the three downward links at each pixel and this choice determines the assigned label.

For the Potts model with two labels the energy in Equation 1 can be minimized with a single minimum-cut on a certain graph but in general the multi-label case is NP-hard [5].

However, recent work on multi-label MRFs has used novel graph constructions to find MAP solutions with a single graph-cut. For instance, Ishikawa’s construction [15] allows for the exact solution to a linearly ordered label set with convex priors. Our graph construction is similar to Ishikawa’s modified to impose constraints 1-3.

**3.1. Example graph construction**

We weight each downward link with the associated unary term. For example, the link representing label 2 at pixel \( p \) is weighted by \( D^2_p \) (the cost at pixel \( p \) for assigning label 2). If this link is part of the final cut we pay this unary cost.

We weight the horizontal links with the pairwise costs \( V^{10}_{pq}, V^{10}_{pq}, V^{12}_{pq}, V^{21}_{pq} \) as shown in figure 3a. Here the notation \( V^{kk}_{pq} \) denotes the pairwise cost for assigning pixel \( p \) to label \( k \) and pixel \( q \) to label \( l \). If we assign both pixels to the same label these links remain intact: we pay no penalty (i.e. the pairwise terms \( V^{kk}_{pq} \) are zero) and this encourages smoothness in the label field (Figures 3b-c). If the assigned labels differ by one, then we must cut exactly one of the horizontal links: there are four possible ways the labels can differ by one (01, 10, 12 and 21) and each has an associated horizontal link and cost (examples in Figures 3d-e).

Unfortunately, this graph also assigns a finite cost for configurations where the labels differ by more than one. For example the solution where \( p \) takes label 2 and \( q \) takes label 0 (figure 3f) incurs a pairwise costs of \( V^{21}_{pq} \) and \( V^{10}_{pq} \) with constraint links of infinite cost. This gives labelings like that in figure 3f an infinite cost.

Unfortunately, this graph also assigns a finite cost for configurations where the labels differ by more than one. For example the solution where \( p \) takes label 2 and \( q \) takes label 0 (figure 3f) incurs a pairwise costs of \( V^{21}_{pq} \) and \( V^{10}_{pq} \) with constraint links of infinite cost. This gives labelings like that in figure 3f an infinite cost.

Figure 4a shows how to prevent this happening. We add a diagonal links \( \{q_2, p_1\} \) and \( \{p_2, q_1\} \) with infinite weights and we term these **constraint edges**. Now any solution that assigns label 2 to pixel \( p \) and label 0 to pixel \( q \) must also cut the link \( \{q_2, p_1\} \) and hence has an infinite cost (figure 4b). Similarly, the link \( \{q_2, p_1\} \) prevents the solution where the pixel \( p \) takes label 0 and pixel \( q \) takes label 2.

We have now constructed a graph in which neighboring pixels must change sequentially (constraint 1). To further constrain them to monotonically increase (constraint 2) we make the pairwise costs \( V^{10}_{pq} \) and \( V^{21}_{pq} \) infinite (figure 4c). Now solutions in which the label decrements as we move from pixel \( q \) to pixel \( p \) have infinite cost. Equivalently, we can leave \( V^{10}_{pq} \) and \( V^{21}_{pq} \) as they are and add infinitely expe-
ensured by limiting the possible labels to a subset of \(|L|\) possible labels, we build a graph with \(|L| - 1\) layers of \(X \times Y\) nodes (Figure 5a). The topmost layer connects to the source and the bottom most layer connects to the sink. Viewed from above, the final cut on this graph will look like a set of stepped layers very much like Figure 1e.

Links between layers in the graph (and the source and sink) have capacities given by the unary terms \(D_k\). Links between neighboring pixels \(p\) and \(q\) within layer \(l\) have capacities given by pairwise terms \(V_{pq}\).

Horizontal constraint links with infinite capacity between nodes at \(\{x, y, l\}\) and \(\{x + 1, y, l\}\) ensure that the labels increase monotonically as we move from left to right. Diagonal constraint links with infinite capacity between the node at \(\{x, y, l\}\) and \(\{x - 1, y - 1, l - 1\}\) prevent non-sequential labels at neighboring pixels.

These diagonal constraint links effectively force the minimum width of each superpixel to be one. In fact it is possible to include additional diagonal constraint links to all nodes in the layer above within some radius. This has the effect of constraining the width of each superpixel to be at least equal to this radius (not shown).

We must also ensure that all of the labels are included in the final solution. We modify the unary terms for pixels at the left edge of the image so that there is an infinite cost for any label other than 0. In a similar manner, we ensure that the pixels at the right edge of the image take label \(|L|\).

Finally, we would like the superpixels to be compact: the vertical bands should be distributed roughly evenly over the image and the superpixels limited in size. This can be ensured by limiting the possible labels to a subset \(K \subset L\) at

3.2. Full graph construction

The full graph construction to guarantee a solution as in figure 2a is as follows. For an image with \(X \times Y\) pixels and \(|L|\) possible labels, we build a graph with \(|L| - 1\) layers of \(X \times Y\) nodes (figure 5a). The topmost layer connects to the source and the bottom most layer connects to the sink. Viewed from above, the final cut on this graph will look like a set of stepped layers very much like figure 1e.

Each horizontal position so that at the left of the image only the first few labels can be chosen and at the right only the last few (Figure 5b). To this end, we prune the nodes from that graph that represent undesirable combinations of pixels and labels. The result, a “sliced layered cake”, appears as a diagonal strip in cross-section (Figure 5b). The pruning also has the beneficial effect of reducing the computational cost and storage requirements of the algorithm.

3.3. Solution

Following [9] we can consider the constraint links as adding interaction terms to the cost function

\[
E(\mathbf{x}) = \sum_{p \in P} D_p(\mathbf{x}_p) + \sum_{p,q \in N} V_{pq}(\mathbf{x}_p, \mathbf{x}_q) + \sum_{p,q \in \mathcal{M}} W_{pq}(\mathbf{x}_p, \mathbf{x}_q)
\]

where \(W\) represents the interaction terms that restrict the set of feasible solutions on the graph over a set of labels \(L\) with neighborhood system \(\mathcal{M}\).

With the particular interaction terms (constraints) we proposed above, this cost function can be minimized exactly in polynomial time by finding the minimum cut in our graph. Not only does this formulation impose the required constraints in the label field, but it is guaranteed to find the optimal solution. This contrasts with off-the-shelf multilabel algorithms like alpha-expansion which don’t guarantee to find a global minimum or impose the constraints.

As a toy example, we take a label field that obeys the constraints (Figure 6a) and simulate measurements for each pixel (Figure 6b). These are drawn from different (but known) Gaussian distributions for each label. Solutions based on the unary terms only (Figure 6c) and alpha-expansion (Figure 6d) are very noisy. However, our solution (Figure 6f) is reasonable.

\[1\]These additional edges preserve order in 4-connected sense and are trivially extended to higher connectivities.
Algorithm 1 Lattice Cut

1: $\mathcal{P} \leftarrow \text{image, } \mathcal{B} \leftarrow [6, 10, 21]$
2: $\mathcal{S} = \{1, 2, \ldots M \times N\} \leftarrow \text{Uniform labeling}$
3: $E(\hat{x}) \leftarrow D + V, E(\hat{x}) \leftarrow \infty, \text{iter} \leftarrow 0$
4: while $E(\hat{x}) < E(x)$ do
5: $E(x) \leftarrow E(\hat{x})$
6: if mod(iter) = 2 then
7: $[S, E(\hat{x})] \leftarrow \text{LayerCut}(\mathcal{P}, \mathcal{B}, \mathcal{S}, M)$ //Horizontal
8: else
9: $[S, E(\hat{x})] \leftarrow \text{LayerCut}(\mathcal{P}, \mathcal{B}, \mathcal{S}, N)$ //Vertical
10: end if
11: iter + +
12: end while
13: return $S$

Algorithm 2 Layer Cut

1: for all $i \in S$ do
2: //Fit colour model to current superpixel data
3: $\mu_i \leftarrow \text{argmax} \log(Pr(\mathcal{P}_i|\mu_i)Pr(\mu_i))$
4: end for
5: $D \leftarrow -\log(Pr(\mathcal{P}, \mu))$ //Unary terms - Figure 7
6: $V \leftarrow \mathcal{B} \cap \text{constraint 3} //\text{Pair terms}$
7: $(S, E(\hat{x})) \leftarrow \text{maxflow}(D, V, W) //\text{Do cut}[4]$
8: return $S, E(\hat{x})$

4. Lattice Cut

The full lattice-cut algorithm\(^2\) is summarized in Algorithms 1 and 2, for a set of $M \times N = |S|$ superpixels, pixel data $\mathcal{P}$ and boundary data $\mathcal{B}$. To complete our description, we discuss how to impose the grid structure along with the choice of boundary map and local color models.

Grid Structure (Constraint 3)

We must ensure that layer boundaries at orthogonal orientations cross only once (see Figure 2b-c). Following [23] we modify the graph at each iteration to impose a punitive cost at every edge on a layer boundary orthogonal to the current cut. This cost must inevitably be paid once between each orthogonal layer, but the minimum cut necessarily avoids any additional high cost edges. For instance, the cost of intersecting paths in Figure 2b is at least three times greater than in Figure 2c, as a result of the number of crossing points.

Boundary Map

We compute edges using the Canny [6], Pb [21] or BEL [10] boundary detectors (see Table 1). Each gives an estimate of the strength $B_{pq} \in [0, 1]$ of a boundary at each pixel $p$. The pairwise cost between neighboring pixels $p$ and $q$ is set to $C(1 - 0.5B_p - 0.5B_q)$ so the pairwise cost $C$ is reduced when there is local evidence for a boundary.

\(^2\)Code publicly available from http://pvl.cs.ucl.ac.uk/

5. Results

Figure 8 shows four examples of over-segmentations computed using our algorithm. In each case, the grid adheres to the contours of the image and coincides with most of the edges of the objects. In Figure 9 we demonstrate...
segmentation of a single image with lattices of increasing resolution. As the number of superpixels increases, more objects in the scene are separable.

In figure 1a-d we show four iterations of our algorithm. The evenly spaced grid gradually conforms to the contours of the images. Figure 10a shows the global cost function decreasing with each iteration of the algorithm. Figure 10b-e shows how the color model for superpixel 11 evolves as the grid changes. At the first iteration, the superpixel contains both the skirt and water and there are two clear modes in the distribution. As the segmentation improves the superpixel conforms to the shape of the skirt and the colour model becomes unimodal and more concentrated.

Figure 1 also illustrates a limitation of our method: constraint 2 means that the vertical component labels always increase as we move across the image and likewise the horizontal component labels always increase as we move down the image. This restricts the possible shape of the resulting superpixels as their edges can never turn back on themselves. This can be seen in superpixel 7 of Figure 1d. This superpixel cannot expand upwards under the arm region or the black boundary between pixels \{6, 7\} and \{10, 11\} will turn back on itself. This will create scanlines with the illegal label ordering 0-1-2-1-2-3.

5.1. Quantitative Evaluation

We quantify performance using the Berkeley segmentation dataset [22]. We first assign each superpixel to the mode class of the ground truth data. This is equivalent to using an ideal classifier at each superpixel. We compare the resulting segmentation \(S\) to the ground truth segmentation \(S'\) using the segmentation covering metric.

The overlap \(O(R, R')\) between a region \(R\) in our segmentation and a region \(R'\) in the ground truth is defined as

\[
O(R, R') = \frac{|R \cap R'|}{|R| \cup |R'|}
\]

This returns a number that is zero if the regions do not overlap at all and one if the regions exactly match. The covering of a segmentation \(S\) by \(S'\) is then given by

\[
C(S' \rightarrow S) = \frac{1}{N} \sum_{R \in S} |R| \cdot \max_{R' \in S'} O(R', R)
\]

where \(N\) denotes the total number of regions. In other words, we find the best overlapping region \(R'\) in the ground truth for each region \(R\) in our segmentation, and calculate the overlap score. The segmentation cover is the average of all of these overlap scores. For the results in this paper we chose \(|K| = 5\) and a minimum superpixel diameter of 2 pixels with image sizes of 321 × 481. The speed of our algorithm is dominated by the graph cut and this took approximately 10 seconds with these parameters. We used six iterations (three horizontal and three vertical cuts) giving a total computation time of roughly one minute per image.

Additionally, we briefly investigate the relative weighting of the edge and region terms using a validation set of 10 images (Figure 11). In each case we divided the image

![Figure 11](image-url)

Figure 11. The same image segmented using a) only pairwise terms (edges) b) both pairwise and unary terms c) only unary terms (regions).
into a grid of $10 \times 10$ superpixels. For these images, the segmentation cover score was 0.62 when only the pairwise (edge) terms were used and 0.78 when only the unary (region) terms were used. With equal weighting of the edge and region terms, the segmentation cover score was 0.92. We conclude that both parts of our model contribute to the final result, but did not optimize the relative weighting for subsequent experiments.

5.2. Comparison to other methods

In Table 1 we compare lattice-cut to the greedy lattice algorithm [23]. We compare our results for superpixel lattice resolutions of $10 \times 10$, $14 \times 15$, $18 \times 19$ and $24 \times 25$ giving totals of 100, 210, 342 and 600 superpixels respectively. Note that the ground truth data across multiple subjects include on average only 20 separate regions per image so this number of superpixels can be considered to span parameter settings where there are potentially multiple superpixels for each image region.

Our results are consistently better across all superpixel resolutions. The main difference is that our approach uses both region and edge information whereas [23] only uses edges and our approach uses sequential global optimization strategies whereas [23] uses a purely greedy approach. This improvement is despite the fact that the algorithm of [23] can produce non-returning boundaries. Additionally, we tested different boundary detectors. It is interesting to note that our performance is best with simple boundary detectors (Canny). We interpret this result as evidence that the inclusion of a unary term fulfills some of the role of texture or feature discrimination in learned approaches, but is additionally image specific.

Table 2 provides a comparison with other segmentation algorithms. Although each of these algorithms can be used to over-segment images, they do not naturally return a fixed number of superpixels. To facilitate comparison, we ran each algorithm at eight parameter settings each of which provides an increasing number of superpixels. We find the parameter settings that produce numbers of superpixels that bracket the desired quantity and estimate the segmentation constraint more favorable.

### Table 1. Comparison to other lattice algorithms: our method improves on the greedy approach of [23]. For reference, we also compare to a uniform grid.

<table>
<thead>
<tr>
<th>Algorithm (Boundary map)</th>
<th>Number of superpixels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td>Uniform</td>
<td>0.777</td>
</tr>
<tr>
<td>SP Lattices [23] (Canny)</td>
<td>0.799</td>
</tr>
<tr>
<td>SP Lattices [23] (Pb)</td>
<td>0.810</td>
</tr>
<tr>
<td>SP Lattices [23] (BEL)</td>
<td>0.818</td>
</tr>
<tr>
<td>Lattice-Cut (Canny)</td>
<td>0.860</td>
</tr>
<tr>
<td>Lattice-Cut (Pb)</td>
<td>0.848</td>
</tr>
<tr>
<td>Lattice-Cut (BEL)</td>
<td>0.859</td>
</tr>
</tbody>
</table>

### Table 2. Comparison to competing methods. ‘*’ denotes interpolated results. Despite the additional constraints our method has comparable or better performance than competing methods, particularly at lower resolutions.

<table>
<thead>
<tr>
<th>Algorithm (Boundary map)</th>
<th>Number of superpixels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td>Uniform</td>
<td>0.777</td>
</tr>
<tr>
<td>Mean Shift [8] *</td>
<td>0.795</td>
</tr>
<tr>
<td>Felzen-Hutten [11] *</td>
<td>0.738</td>
</tr>
<tr>
<td>NormCuts [31] (Pb) *</td>
<td>0.839</td>
</tr>
<tr>
<td>Arbelaez [1] ((19)) *</td>
<td>0.850</td>
</tr>
<tr>
<td>Lattice-Cut (Canny)</td>
<td>0.860</td>
</tr>
<tr>
<td>Lattice-Cut Merged</td>
<td>0.863</td>
</tr>
</tbody>
</table>

<ref>Arbelaez [1] ((19)) * denotes interpolated results.</ref>
6. Discussion

In this paper, we have presented a method for unsupervised segmentation of an image into a regular grid. We alternately find global solutions for the vertical and horizontal layers. One limitation of our method is that paths between superpixels must be non-returning: the order constraints prevent paths that turn back on themselves and this occasionally introduces flaws in the segmentation. One possibility is to postprocess locally to remove these flaws without drastically changing the solution.

Our graph construction preserves order in the lattice using a set of constraint edges, and is based on techniques for minimizing multi-label problems using a binary graph cut [15, 30]. Other authors have also considered ordering constraints: Liu [18] introduced “order preserving moves” that improves upon α-expansion in the presence of order constraints. Li et al. [17] introduced a similar graph construction to ours that preserved order constraints but did not include unary terms. Finally, Delong et al. [9] impose topological constraints on unordered label sets using a graph construction that also limits the set of feasible solutions.

Ordering constraints may find future application in segmenting objects with ordered parts. A label map that denotes parts of the face (eyes, mouth, nose etc.) contains ordered structure and it may be possible to find a global segmentation of the face given these constraints. Prior statistical knowledge of the scene can be used to adapt the distribution of the superpixels [24]. Lastly, sampling from the MRF, or directed alternatives, could be used to generate hypotheses in a similar manner to work exploring the use of multiple segmentations during inference [20].

References