Superpixel Lattices

Alastair P. Moore Simon J. D. Prince* Jonathan Warrell* Umar Mohammed Graham Jones†

University College London
Gower Street, WC1E 6BT, UK
{a.moore,s.prince}@cs.ucl.ac.uk

† Sharp Laboratories Europe
Oxford, OX4 4GB, UK
graham.jones@sharp.co.uk

Abstract

Unsupervised over-segmentation of an image into superpixels *is a common preprocessing step for image parsing algorithms. Ideally, every pixel within each superpixel region will belong to the same real-world object. Existing algorithms generate superpixels that forfeit many useful properties of the regular topology of the original pixels: for example, the n^{th} superpixel has no consistent position or relationship with its neighbors. We propose a novel algorithm that produces superpixels that are forced to conform to a grid (a regular superpixel lattice). Despite this added topological constraint, our algorithm is comparable in terms of speed and accuracy to alternative segmentation approaches. To demonstrate this, we use evaluation metrics based on (i) image reconstruction (ii) comparison to human-segmented images and (iii) stability of segmentation over subsequent frames of video sequences.*

1. Introduction

Image *parsing* attempts to find a semantically meaningful label for every pixel in an image. Many vision problems that use natural images involve image parsing. Possible applications include autonomous navigation, augmented reality and image database retrieval.

Image parsing is necessary to resolve the natural ambiguity of local areas of an image [18]. One example of this ambiguity is depicted in Figure 1a. The small blue image patch might result from a variety of semantically different classes: sky, water, a car door or a person's clothing. However, given the whole image, we can see that when found *above* trees and mountains *and* alongside similar patches across the *top* of the image *and* in the absence of boats, people, roads *etc*., the correct class is *probably* sky.

Image parsing algorithms [19, 7, 20] combine segmentation, detection and recognition to attempt to resolve these ambiguities. It is common in the literature to use Markov Random Field (MRF) or Conditional Random Field (CRF) models to incorporate relational/spatial/temporal context. These are graph based models relating the observed image data to the hidden underlying object class at each position in an image. Normally, the probabilistic connection between nodes favors similarity between labels and therefore acts to smooth the estimated field of labels from data. Apart from some special cases, both exact and approximate inference methods in these models slow down as the number of nodes increases.

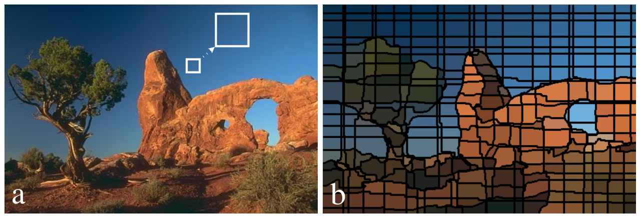

Figure 1. a) The semantic label associated with the small blue image patch (small white box, enlarged) is ambiguous in the absence of context. Image parsing algorithms can resolve this ambiguity. b) Over-segmentation is a common preprocessing step for image parsing. In this paper we propose a method for over-segmentation into a regular grid of superpixels. In this example we have reduced a 640 × 480 image to a regular 20 × 20 lattice of superpixels.

Consequently, inference can be slow in large images when MRF or CRF graphs have one node for every pixel. Moreover, the pixel representation is often redundant, since objects of interest are composed of many similar pixels. Computational resources are wasted propagating this redundant information. One possible solution to this is to use multi-scale and banded techniques [11, 15] to reduce memory requirements and limit the time complexity of solutions.

A different approach was suggested by Ren and Malik [16]. They proposed a preprocessing stage in which pixels were segmented into *superpixels*[1] thereby reducing the number of nodes in the graph. Their method uses the *normalized cut* criterion [17] to recursively partition an im-

*We acknowledge the support of the EPSRC Grant No. EP/E013309/1.

[1]A *superpixel* is a spatially-coherent, homogeneous, structure which preserves information over scales or sampling resolutions. We refer to algorithms that over-segment the image as *superpixel* algorithms.

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

age using contour and texture cues and has been used as a preprocessing step in contemporary image parsing schemes [7]. Other possible superpixel algorithms include [1, 3] and reducing the number of nodes using the watershed algorithm was demonstrated in [10].

Unfortunately, these superpixel algorithms forfeit several useful properties of the original pixel representation. First, pixels can be represented in arrays without the need for pointers. Second, it is easy to sub-sample pixels uniformly and use multi-scale methods. Third, the n^{th} pixel has a consistent, ordered, position in the image. Fourth, the n^{th} pixel has a consistent relationship with the $(n-1)^{th}$ pixel, allowing local neighborhood operations. Fifth, pixel representations of images of the same dimension $(row \times col)$ are isomorphic (a unique mapping from pixel to pixel in different images). These properties result from the regular topology of the original grid graph of pixels and are abandoned by contemporary superpixel algorithms.

In this paper, we introduce a segmentation algorithm that is guaranteed to produce a regular grid of superpixels (a superpixel lattice). This is motivated by two concerns:

- First, a grid increases engineering efficiency and convenience. The architecture of common sensors means that most current vision algorithms implicitly assume a Cartesian grid and can be adapted easily for use in superpixel lattices without the need to resort to more general graph algorithms [5].
- There are some algorithms that are very difficult to adapt to a graph with non-regular topology. For example, there has been considerable recent interest in using higher order cliques relating labels over small regions of the image [6, 9]. However, it is not obvious how to learn joint statistics of labels over regions of superpixels if the topology relating superpixels changes in every image. Indeed, He *et al.* [6] bemoan this state of affairs, stating that "ideally the system we described would be applied to a higher level of representation [than pixels]. However, this requires a consistent and reliable method for extracting such representations from images." Our proposed algorithm is one such higher level of representation.

The contributions of the paper are as follows: in Section 2 we propose a superpixel algorithm that preserves the topology of a regular lattice, an example of which is shown in Figure 1b. In Section 3 we investigate the qualitative properties of the resulting segmentation and compare it to contemporary algorithms. In Section 4 we develop several quantitative measures that show that the topological constraint imposed by the regular lattice does not adversely affect segmentation, making our system a viable preprocessing step for image parsing algorithms. In Section 5 we show how superpixels can be merged to produce a coarser segmentation while retaining the qualitative advantages of our system. Lastly, in Section 6 we demonstrate that our algorithm produces stable segmentations across frames of video data.

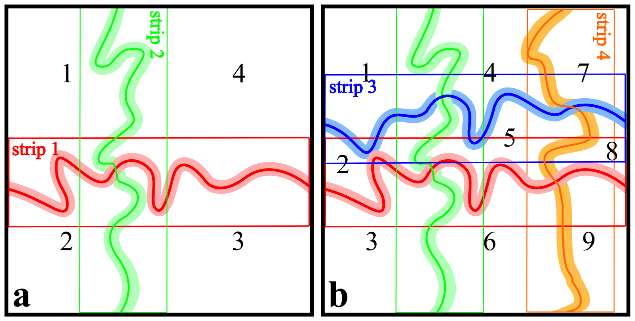

Figure 2. Incremental construction of superpixel lattice. a) The image is initially split left to right and top to bottom to form four regions. In each case we seek the optimal path within a predefined image strip. b) Adding one more vertical and horizontal path partitions the image into nine superpixels. Future path costs are modified in bands around previous paths (light colors) to prevent multiple-crossings and set a minimum distance of approach between paths.

2. A Greedy Regular Lattice

In this section we describe a greedy superpixel algorithm that maintains the regular topology of the grid graph of pixels. Most segmentation algorithms pose the question: "what properties would we like of individual segments?", and develop different metrics for segment homogeneity. Here, we consider "what relations would we like to hold *between* segments?". In particular, a regular topology is our goal.

The input to our algorithm is a *boundary map*. This is a 2D array containing a measure of the probability that a semantically meaningful boundary is present between two pixels. This problem is well studied in the literature, which in the simplest case results in the binary output of an edge detector but in more complicated schemes leads to an estimate of the probability of *natural* [13, 2] or *occlusion* boundaries [8] in an image. For convenience we invert and re-scale the boundary map to take a value of 0 where there is the most evidence for a boundary and 1 where there is no evidence. We term this the *boundary cost map*. Our goal is to segment the images in places where this boundary cost map is lowest, which we do by finding minimum weighted paths through the graph.

The construction of the superpixel lattice is incremental: initially we bipartition the image vertically and horizontally. Each path splits the image into two, to cumulatively produce four superpixels (see Figure 2a). At each subsequent step we add an additional vertical and horizontal path (Figure 2b). A regular lattice is guaranteed if we ensure that (i) each horizontal and vertical path only cross once (ii) no two horizontal paths cross and (iii) no two vertical paths cross.

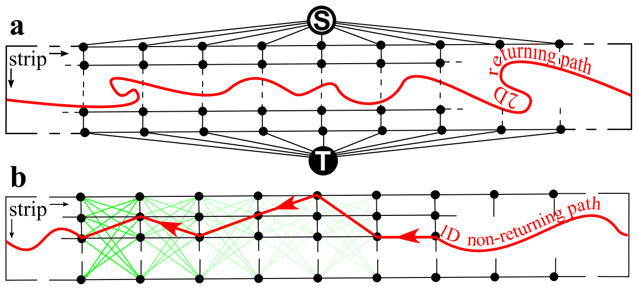

Figure 3. Estimation of optimal path through an image strip. a) Min-cut between *source and* sink. Arbitrary paths are allowed. b) Dynamic programming. Forward pass is **green**. Global optimal path of backwards pass is **red**. Only non-returning paths are obtained.

We first describe how to form each path, and then discuss how to ensure these constraints are maintained.

At each stage in the algorithm we seek the optimal path across the whole image. The optimal path is determined by the values in the boundary cost map along the path (we aim to follow image boundaries). However, we also apply regularizing constraints that prevent the path from wandering arbitrarily. One such constraint is to restrict each path to fall within a predefined strip across the image (see Figure 2). This prevents the formation of paths that run diagonally across the whole image and hence restrict the placement of subsequent paths. It also forces a quasi-regular grid and reduces the computation at each step by limiting the number of paths considered.

We present two solutions. The *s-t min-cut method* produces paths of arbitrary topology. We also used the *dynamic programming method* that produces paths that are non-returning (every subsequent point on the path is closer to the other side of the image and the path cannot turn back on itself). In general we expect the former method to follow boundaries more closely, but the latter method to be faster and to exhibit more stability. Examples of the min-cut method are shown in Figures 1 and 4. An example of the dynamic programming method is shown in Figure 5.

Method 1 - s-t min-cut: For this method, we define a graph $G_{MC} = \{V, E\}$ as depicted in Figure 3a. In this graph there is one cost associated with each edge (ν_i, ν_j) between neighboring pairs of nodes. All nodes on one side of the strip are connected to the *source*. Nodes on the opposite side are connected to the *sink*. The costs for edges connecting pixels is determined by the boundary cost map so that the path is more likely to pass along boundaries. In addition, we add a constant value to the costs for cuts perpendicular to the strip direction. This controls the *tortuosity* (the degree to which the path deviates from a straight line). The min-cut algorithm [4] finds the minimum cost cut between source and sink and hence defines the path.

Method 2 - dynamic programming: We define a second, different, graph $G_{DP} = \{V, E\}$ over the image pixels in the strip as shown in Figure 3b. Dynamic programming is a well known algorithm that finds the 1D path that minimizes the total cost of edges and nodes. The edge cost is zero for paths that move straight across the image (left to right in Figure 3). The cost for diagonal paths is a parameter and affects the tortuosity of the path.

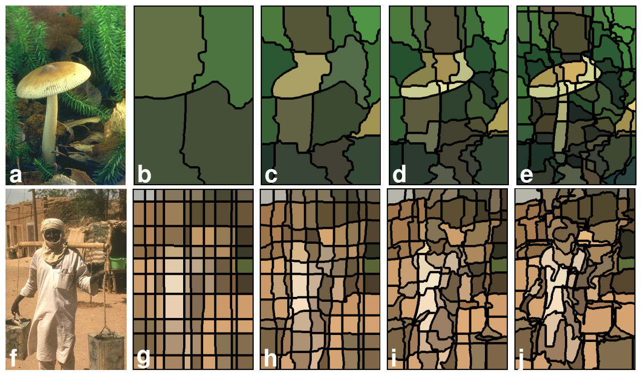

Figure 4. Two image sequences to demonstrate tuneable parameters of a greedy regular lattice. a) Original image. b)-e) Superpixel lattices with increasing superpixel resolution; 2×2, 4×4, 6×6 and 10×10, respectively. f) Original image. g)-j) 10×10 superpixels. Increasing tortuosity results in transition from straight grid to superpixels that conform to natural image boundaries.

Strips for adjacent paths must overlap, to preserve boundaries in the image, so this in itself is insufficient to prevent parallel paths from intersecting. To ensure that this does not happen, forcing the correct topology, we update the boundary cost map after generating each path. Values along the path are allocated a fixed large cost. This prevents subsequent parallel paths crossing. Perpendicular paths are forced by geometry to cross once, but the high cost prevents them turning back on themselves and crossing again. In addition we increase the costs of a band of neighboring nodes to each path to prevent paths becoming very close to one another. This is undesirable because (i) it produces very small superpixels and (ii) close paths often follow the same real-world boundary, making the subsequent semantic interpretation of the intervening superpixel difficult.

2.1. Parameters of Greedy Lattice

There are three important parameters that control the final superpixel lattice. First, the *resolution* determines the total number of superpixels, which is indirectly determined by the number of paths. In Figure 4a-e we demonstrate increasing resolution. Second, the *width* of each image strip constrains the chosen path. Together the width and resolution determine the overlap of the strips. These must overlap so that a real-world boundary may be followed from one strip to the next using different paths. Third, the *tortuosity* of the path determines the degree to which the curve deviates from a straight line. The effect of varying this parameter can be seen in Figure 4f-j. As tortuosity increases, the paths are slowly allowed to conform to the costs in the boundary map. Increasing this parameter does not indefinitely improve results because, at very high levels, the algorithm produces a meandering solution that attempts to assimilate all the image boundaries into a single path.

3. Qualitative Evaluation

In Figure 5 we qualitatively compare the regular lattice (top) to two other superpixel algorithms (middle and bottom) for the same image. The segmentations produced by our algorithm have a number of desirable properties:

(i) Consistent pixel positions: For a fixed resolution, each of our superpixels is always at roughly the same position in the image. This facilitates the definition of spatially varying priors over image classes as in [6]. For example, we can impose the information that superpixel 1 in the top-left of the image tends to be part of the sky. In other segmentation schemes, we would have to first establish the spatial position of superpixel 1, which may be ambiguous, and then relate this to a spatial prior defined over the original image.

(ii) Consistent spatial relations: In image parsing we want to learn the probabilistic relations between labels; for instance the frequency with which sky appears *above* the ground (*e.g.* [7]). While such relations can be defined *ad hoc* on any segmentation it results in a graph isomorphism problem: is the relationship between nodes in this graph the same as that encountered on other graphs during learning? A regular lattice means there is a bijection between segmentations (a one-to-one correspondence between segments), resulting in a consistent and unambiguous relationship between superpixels. This can be seen in Figure 5d. In contrast, in Figure 5e, which numbered region is to the left of region 244: 67,71 or 65? Which is under region 208: 73 or 67? Learning label distributions under this segmentation is ambiguous or involves imposing a new mapping.

(iii) Conservatism: The segments in the superpixel lattice are of regular size and never greedily select huge image regions. This limits the possibility of erroneously grouping large semantically different regions such as sky and sea causing drastic 'leaking' between classes. Other algorithms can produce extended regions (*e.g.* sky in Figure 5c).

(iv) Natural Scale Hierarchy: It is common to solve random field models using multi-scale techniques (e.g [15]). Regular lattices easily accommodate such methods as they have the same multi-scale relations as the original pixels: each superpixel decomposes into four smaller child superpixels, since happens between Figure 4b and c.

(v) Graph Isomorphism: For a given resolution, superpixels in every segmentation have the same relationships with one another. This allows the development of algorithms that learn of the relationships between the labeling of groups of superpixels (*i.e.* higher-order cliques).

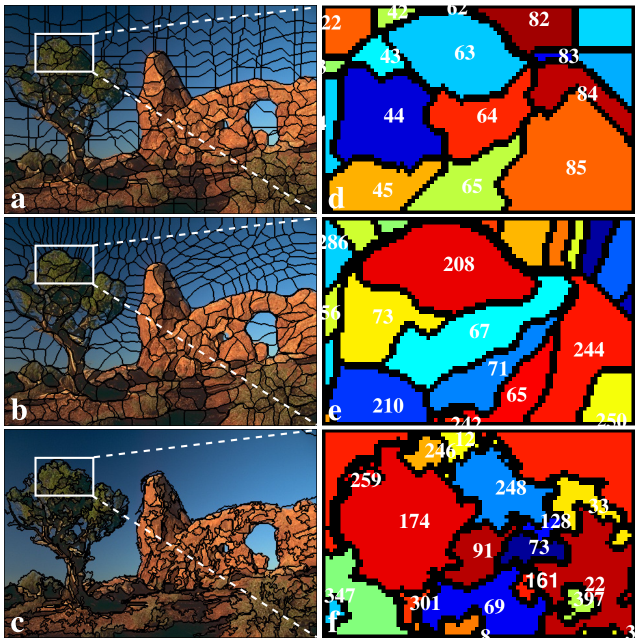

Figure 5. Comparing superpixel properties. a) Our algorithm. b) Superpixel algorithm [17] that provides state-of-the-art performance against human labeled ground truth (see Section 4). c) Superpixel algorithm [3] that provides efficient segmentation benchmark (see Section 4). d) Our algorithm maintains all the useful pixel properties that result from a regular lattice. e) Superpixels have a variable number of neighbors in varying spatial configurations. f) Superpixels with varying topologies *e.g.* some superpixels can exist completely inside others.

4. Quantitative Evaluation

In this section we demonstrate that our algorithm produces useful segmentations despite the added topological constraint of being forced to create a grid. Our evaluation is based on 11 grayscale test images, equally spaced on a ranked list[2] for performance, from the Berkeley Segmentation Database Benchmark (BSDB) [12].

We investigate three choices of boundary map: The *Pb* boundary map [13] generated using gradient/texton cue integration that provides good performance against human labeling. The fast *BEL* boundary map [2] generated using a boosted classifier to learn natural boundaries. This algorithm is efficient and produces a good f-measure score [12] on the BSDB. We contrast with a simple edge map generated using the absolute value of the Sobel operator at four orientations.

We use two metrics for comparing the performance of superpixel algorithms: explained variation and accuracy.

[2]id(rank): 42049(1), 189080(10), 182053(20), 295087(30), 271035(40), 143090(50), 208001(60), 167083(70), 54082(80), 58060(90), 8023(100)

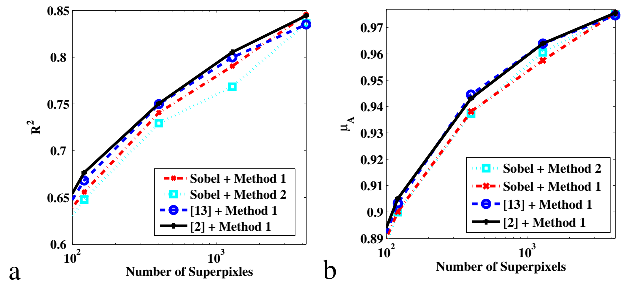

Figure 6. Performance of superpixel algorithm using alternative boundary maps. a) Explained variation metric (R^2) b) Mean accuracy (μ_A). The region of 100-2000 superpixels represents a region around ∼1% of the original pixels.

4.1. Explained Variation

We are interested in measuring how well the data in the original pixels is represented by superpixels, for which we introduce *explained variation*

$$R^2 = \frac{\sum_i (\mu_i - \mu)^2}{\sum_i (x_i - \mu)^2} \tag{1}$$

where we sum over i pixels, x_i is the actual pixel value, μ is the global pixel mean and μ_i is the mean value of the pixels assigned to the superpixel that contains x_i. An example of using the superpixel mean can be seen in Figure 1b. This metric describes the proportion of image variation that is explained when the detail within the superpixels is removed.

The explained variation metric R^2 will take the maximum possible value of 1 as the number of superpixels increases and we recover the original pixels. It takes the minimum possible value of 0 when there is only one superpixel (the image mean). This is not a perfect metric for evaluating performance as it penalizes superpixels that contain consistent texture with large pixel variance. However, our intent is to provide a human independent metric.

In Figure 6a we investigate the performance of our algorithm using explained variation. For all boundary maps expected variation increases with the number of superpixels as we gradually return to the original pixel image. We achieve similar performance using boundary maps [13] and [2]. This is important as [13] takes of the order of a minute to compute which prohibits its use as a preprocessing step. This is not a limitation for [2] since it uses a boosted classifier on the Canny edge mask. Unsurprisingly both of these algorithms are superior to just using the Sobel operator. We also compare the min-cut and dynamic programming formulations for the Sobel operator. The min-cut formulation is superior: it is better to allow paths that double back on themselves even though our control over the smoothness of the path is diminished.

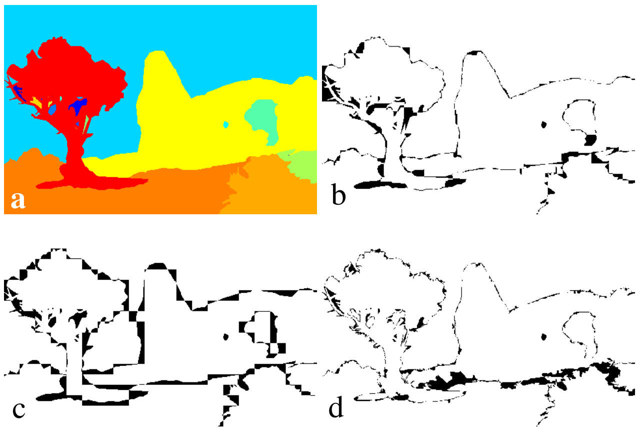

Figure 7. Accuracy of superpixels against human-labeled ground truth. a) Human-labeled ground truth regions. b) An example of the errors associated with the 20×20 lattice presented in Figure 1b when compared to the ground truth in (a). Pixels in black will be misclassified by an ideal classifier as they lie in superpixels which are dominated by a different object class. c) Errors for uniform sampling. This can only produce piece-wise linear approximation to image boundaries. d) Errors for [3] with 400 superpixels.

4.2. Mean Accuracy

We use human-labeled ground truth data from the BSDB to test the accuracy of superpixel boundaries. Each image in the data set contains independently labeled ground truth from multiple human subjects. Example ground truth data can be seen in Figure 7a. We use the mean accuracy, μ_A, over subjects, where Accuracy[3] is the agreement between our superpixel and the pixel ground truth s_i.

We set the j pixels assigned to the n^{th} superpixel to the mode class (most frequently occurring) of the ground truth data. This can be interpreted as using an ideal classifier. As with R^2, the mean accuracy μ_A inevitably tends to a maximum value of 1 as the resolution of the superpixels increases to that of individual pixels. An example of the errors associated with the 20×20 lattice presented in Figure 1b can be seen in Figure 7b, together with competing methods.

With respect to the choice of boundary map, the pattern is the same as for the explained variation metric: the Sobel method is inferior to the more sophisticated boundary finding methods. However, in contrast with the previous metric, performance is improved with the dynamic programming method relative to the min-cut algorithm. This effect results from the implicit smoothing properties of the dynamic programming algorithm, which are important with a weak boundary cost map.

It is instructive to consider the absolute figures in the graph. The original images were 640×480 pixels, so if we reduce the number of superpixels to 1/500th of this number (*i.e.* ∼600 superpixels) we expect to incur a 5% penalty in

[3]Accuracy is $(TruePositive + TrueNegative)/Total$.

Algorithm	400	1296
[2] + Method 1	0.750	0.805
[3]	0.808	0.874
[13] + [14]	0.792	0.819

Table 1. Comparison of algorithms using explained variation. Our method is relatively poor at reconstructing images because it distributes the superpixels roughly evenly over the frame, rather than having many superpixels in highly textured areas and few elsewhere.

terms of classification. Given that most image parsing algorithms currently exhibit error rates of several times this magnitude, this is an acceptable price to pay for the convenience of reducing the number of unknown parameters by a factor of 500.

4.3. Comparison to other algorithms

We compare our algorithm to two other methods: the normalized cuts (NC) code made available by [14] and the agglomerative method (FH) of Felzenszwalb and Huttenlocher [3]. These algorithms represent either ends of a spectrum of segmentation algorithms: the NC algorithm provides state-of-the-art performance against human-labeled ground truth data, whereas the FH algorithm sets the bench mark for efficiency. The implementation of the NC algorithm [14] includes post-processing to remove small segments and break superpixels into homogeneous sizes. We apply the NC algorithm to the boundary map [13] to achieve a bench mark of maximum performance. This algorithm is too slow to be a viable preprocessing component of a vision algorithm pipeline but serves as a upper bound of segmentation performance. We post-process the FH algorithm by removing regions $<10\%$ of the size of the image area divided by the number of superpixels. This improves performance at low resolutions and makes results comparable to our algorithm, which has natural regularizing constraints. We perform no post-processing on our algorithm.

In Table 1 we compare our algorithm to these competing methods using the explained variation metric. We compare for segmentations of 400 and 1296 superpixels, since these values roughly bracket the useful range of a superpixel algorithm: the region in which we get a large compression ratio without too great a sacrifice in the segmentation quality. The best performance is achieved with the algorithm of [3], followed by [13] + [14], and then our algorithm. It is unsurprising that our method performs worse than these other approaches because of the topological restrictions. However, it is surprising that normalized cuts ([13] + [14]) does not perform best. Closer examination of the results reveals that the algorithm of [3] tends to describe textures with a large number of very small regions. This is good for reconstructing the image, but these regions have no seman-

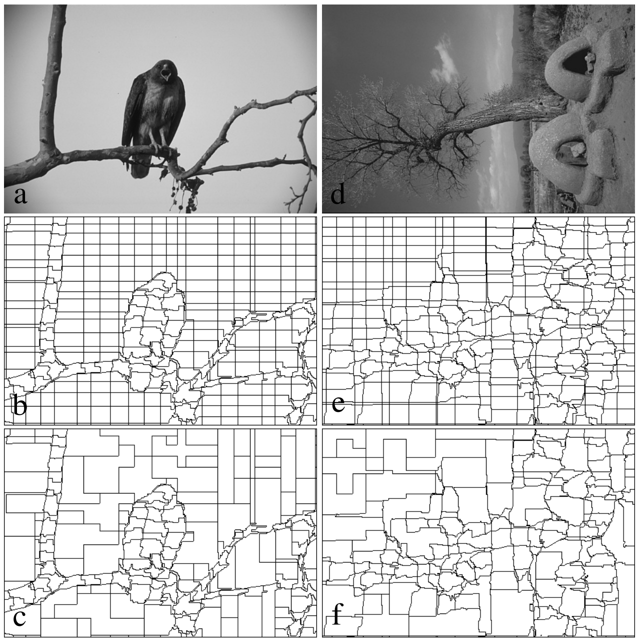

Figure 8. Merged lattices using [2]. This reduces the number of superpixels while the lattice preserves structure as well as accuracy. a) Original high ranked image (easy). b) 20×20 regular lattice. c) Merged lattice reduced to 200 superpixels. Accuracy reduces from 0.956 to 0.951. d) Original low ranked image (difficult). e) 20×20 regular lattice. f) Merged lattice reduced to 200 superpixels. Accuracy reduces from 0.923 to 0.902.

tic meaning, which is unhelpful for the final goal of image parsing. Moreover, this tendency to over-model the image results in a lack of stability. This is investigated further in Section 6.

In Table 2 we compare algorithms using the mean accuracy metric where a different pattern emerges. Our algorithm produces almost identical results to that of [3]. This can be seen qualitatively in Figure 7. The normalized cuts algorithm [17] produces slightly better results, with an average 1% difference in the region of interest. Unsurprisingly all methods perform considerably better than simply reducing the resolution of the image (see Figure 7). We conclude that our algorithm produces segmentations comparable to those of [3] despite having an additional topological constraint.

Algorithm	400	1296
[2] + Method 1	0.942	0.964
[3]	0.948	0.964
[13] + [14]	0.957	0.968

Table 2. Comparison of algorithms using the mean accuracy metric. Remarkably, our algorithm produces almost identical results to that proposed by [3] despite being topologically restricted to segment into a lattice.

5. Merging Superpixels

It is possible to improve the performance of our algorithm by merging adjacent superpixels. In other words, we can apply a further segmentation algorithm to our computed superpixel grid. In Figure 8 we show examples of merging superpixels based on greedily removing boundaries with the smallest cost. This merging process need not eliminate the desirable properties of the lattice structure. In the context of a message passing algorithm, we consider the merged regions as groups of constituent superpixels: they maintain their usual relationships with neighbors outside the group, but the MRF/CRF costs are designed so an infinite penalty is incurred if the group members take different values. Many current inference algorithms can operate in these circumstances. The graph can still be represented on a regular grid with all the advantages this provides. The relationships between the new regions are inexpensive to calculate as we operate in the 20×20 grid of superpixels, rather than the 640×480 domain of the original images.

We investigated segmenting initially into 784 superpixels (28×28 grid) and merging superpixels until we reach 400 superpixels. Under these circumstances, the performance of our algorithm increases substantially (see Table 3). In terms of explained variation our algorithm is superior to [13] + [14] and comparable to that of [3]. However, the superpixels are qualitatively more interpretable in our case sice they do not attempt to model individual texture elements. More importantly, using mean accuracy our performance is better than both competing algorithms and may be attributed to reducing the *small cut* behavior [17] by overpartitioning and merging.

Algorithm	Explained Variation	Mean Accuracy
[2] + Method 1	0.803	0.963
[3]	0.808	0.948
[13] + [14]	0.792	0.957

Table 3. Comparison of algorithms using both metrics after merging superpixels. Our algorithm explains the variation in the image almost as well as the best competing algorithm [3] and produces a better mean accuracy than both competing algorithms.

6. Stability

The use of greedy algorithms means that the solutions are unstable for consecutive frames of video. To deal with the temporal aspects of segmenting sequences of video footage we extend the s-t min-cut solution presented in Section 2 to 3D. In this new graph source and sink nodes are connected to the edge of each strip of the image over several frames.

However, this is only half a solution as either memory or time constraints will limit the number of available frames

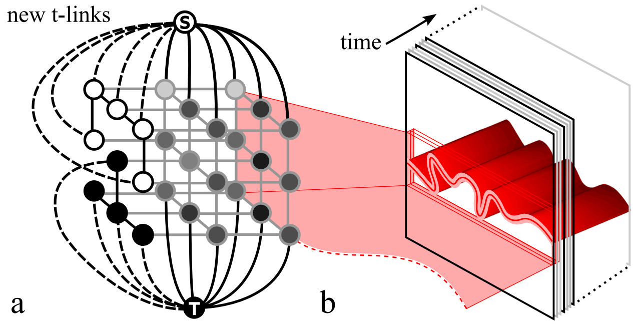

Figure 9. A seeded 3D lattice. a) A batch of two frames with additional t-link edges from the solution of previous frame (black outline). Additional nodes colored white and black from previous s-t min-cut solution. b) Cut surface representing the solution to 3D lattice. Performing the s-t min-cut over several frames results in a cut that is stable between consecutive frames.

segmented at any given instance. Given such constraints, each batch (set of frames) would be temporally consistent but there would be a large jump at the transition between batches. It is therefore necessary to impose temporal stability on the next batch of frames by utilizing the greedy solution from the previous batch of frames.

One solution to achieve a stable greedy lattice is to alter the nodes and t-links in the 3D grid graph. The first frame of each new batch is connected (seeded) to the nodes in the graph of the last frame in the previous batch. Membership of nodes to source and sink in the s-t min-cut solution from this last frame are used to connect new t-links in the first frame of the new batch. An example of this can be seen in Figure 9. Additional nodes mean that the new s-t min-cut solution will be influenced by the solution to the last batch.

We took 50 frames from a video sequence of a shop entrance in a mall. We segmented this sequence using both [3] and our method but do not compare with normalized cuts because this is too slow for useful application to video.

We quantify stability by comparing adjacent pairs of images in the sequence. As [3] does not produce segmentations that are isomorphic we find the nearest superpixel in the consecutive frames where proximity is determined by the Euclidean distance between cluster centers. We then count the proportion of pixels from the superpixel in the first image that are in the matching superpixel in the second image. We normalize this by the total number of pixels to provide a number that varies from 0 (totally unstable) to 1 (completely stable). As the scene is static, viewed with a static camera, the ground truth stability is 1.

Our algorithm applied to each frame produces a mean stability of 0.73 compared to 0.69 for [3]. However it rapidly increases to 0.96, 0.97 and 0.98 using batches of 1,2 and 5 frames respectively. There is a trade off between the stability and the smoothness of the solution which will be more apparent in dynamic scenes or changing camera viewpoint and this is left for further work.

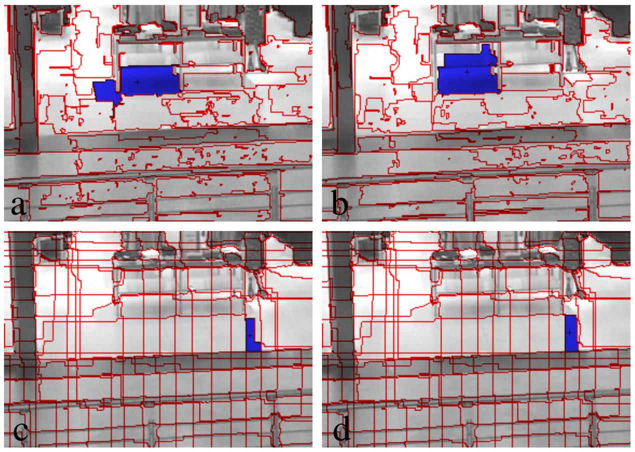

Figure 10. Superpixel Stability. Example superpixels (blue) have stability of 0.70 with the centroid marked with a cross (black). Note the change in superpixel boundaries (red) between consecutive frames. a) and b) Example frames i and $i+1$ using [3], stability 0.69. c) and d) Example frames i and $i+1$ using a greedy regular lattice, stability 0.97.

7. Discussion and Conclusions

In this paper, we have introduced a novel algorithm to segment the image into a regular grid of superpixels[4]. We have argued that the regular grid confers a number of useful properties. We have also demonstrated that despite this topological constraint, we can achieve segmentation performance comparable with contemporary algorithms.

The current gold standard method for segmentation is the normalized cuts algorithm [17]. For a square $N \times N$ image, with N^2 pixels, this involves solving an eigenproblem of size N^2 using m iterations of the Lanczos method. It can take of the order of minutes to converge per image. The computation in [3] is dominated by the need to sort the edge strengths (of which there are roughly $2N^2$) yielding $O(N^2 log N^2)$ complexity and in practice can work at near frame rate for reasonable sized images. Our method is also fast and scales well as image size increases when applied to overlapping strips of length N and width S. Ignoring overlap there are approximately 2N/S of these strips. For the min-cut method, the cost for processing each strip is $O(N^2 S^2 \log NS)$, giving an overall complexity of $O(N^3 S \log NS)$ which runs at about 2fps for 20×20 superpixel lattices on 321×481 images.

In conclusion, our algorithm is fast, accurate, stable and produces a segmentation with favorable topological properties. Moreover, if we are prepared to abandon a fixed topology it produces more accurate segmentations than the current gold standard. In future work, we intend to develop our algorithm to spatiotemporal segmentation in dynamic scenes and investigate learning higher order MRF and CRF models using the superpixel lattice as a basis.

[4]Code will be made publicly available from http://pvl.cs.ucl.ac.uk/

References

[1] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis Machine Intell.*, 24(5):603–619, 2002.

[2] P. Dollr, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. *CVPR*, 2:1964–1971, 2006.

[3] P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *IJCV*, 2(59):167–181, 2004.

[4] A. V. Goldberg and R. E. Trajan. Solving minimum cost flow problems be successive approximation. *ACM Symposium on the Theory of Computing*, 36:388–397, 1988.

[5] L. J. Grady. *Space-variant computer vision: a graph theoretic approach*. PhD thesis, Boston University, 2004.

[6] X. He, R. D. Zemel, and M. A. Carreira-Perpinan. Multiscale Conditional Random Fields for Image Labeling. *CVPR*, 2:695–702, 2004.

[7] X. He, R. S. Zemel, and D. Ray. Learning and incorporating top-down cues in image segmentation. *ECCV*, 1:338–351, 2006.

[8] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. *ICCV*, 1:1–8, 2007.

[9] P. Kohli, M. P. Kumar, and P. H. Torr. P3 and beyond: Solving energies with higher order cliques. *CVPR*, 2007.

[10] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graphics*, 23(3):303–308, 2004.

[11] H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. *ICCV*, 1:1550–5499, 2005.

[12] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *ICCV*, 2:416–423, 2001.

[13] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004.

[14] G. Mori. Guiding model search using segmentation. *ICCV*, 2:1417–1423, 2005.

[15] P. Perez and F. Heitz. Restriction of a markov random field on a graph and multiresolution statistical image modeling. *Trans. Information Theory*, 42(1):180190, 1996.

[16] X. Ren and J. Malik. Learning a classification model for segmentation. *ICCV*, 1:10–17, 2003.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *PRMI*, 22(8):888–905, 2000.

[18] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2):169–191, 2003.

[19] Z. Tu, X. Chen, A. Yuille, and S.-C. Zhu. Image parsing: Unifying segementation, detection and recognition. *IJCV*, 2(63):113–140, 2005.

[20] J. Winn and J. Shotton. The Layout Consistent Random Field for Recognizing and Segmenting Partically Occluded Objects. *Proc. CVPR*, 1:37–44, 2006.