Guided Informative Image Partitioning

Nathan Brewer[1,2], Nianjun Liu[2], and Lei Wang[1]

[1] College of Engineering and Computer Science, Australian National University,
Crnr North and Daley Roads, 0200, Canberra, Australia
[2] Canberra Research Laboratory, NICTA
Tower A, 7 London Circuit, 2600, Canberra, Australia
{nathan.brewer,lei.wang}@anu.edu.au, nianjun.liu@nicta.com.au

Abstract. Image partitioning separates an image into multiple visually and semantically homogeneous regions, providing a summary of visual content. Knowing that human observers focus on interesting objects or regions when interpreting a scene, and envisioning the usefulness of this focus in many computer vision tasks, this paper develops a user-attention adaptive image partitioning approach. Given a set of pairs of oversegments labeled by a user as "should be merged" or "should not be merged", the proposed approach produces a fine partitioning in user defined interesting areas, to retain interesting information, and a coarser partitioning in other regions to provide a parsimonious representation. To achieve this, a novel Markov Random Field (MRF) model is used to optimally infer the relationship ("merge" or "not merge") among oversegment pairs, by using the graph nodes to describe the relationship between pairs. By training an SVM classifier to provide the data term, a graph-cut algorithm is employed to infer the best MRF configuration. We discuss the difficulty in translating this configuration back to an image labelling, and develop a non-trivial post-processing to refine the configuration further. Experimental verification on benchmark data sets demonstrates the effectiveness of the proposed approach.

1 Introduction

As human observers, when we are observe the world around us there are things in our field of view that interest us more than others. We subconsciously pay more attention to these areas, taking in more information about them than their surroundings. In computer vision, partitioning an image into multiple visually and semantically homogeneous regions is an important step in scene understanding. It provides a useful summary of visual content and allows a complex visual recognition process to be decomposed to region-level subtasks. While image partitioning has been well studied and widely applied in many vision problems, existing segmentation techniques can not reflect this type of human focussing behavior. In this paper, we present a method for partitioning an image such that some user-defined interesting object(s) are preserved at higher resolution than uninteresting objects.

The idea that the human brain devotes more attention to some areas in the visual field has support from the field of psychology [1]. Such research has likened

E.R. Hancock et al. (Eds.): SSPR & SPR 2010, LNCS 6218, pp. 202–212, 2010.

perception to a spotlight, illuminating different players on a stage as attention is focussed on them. The rest of the stage remains visible, but more details stand out about the interesting actor. The partitioning that we propose is a good analogue of this model of human visual attention. It is important to note that we neither construct or utilize a specific attention model in our approach. Instead, we attempt to produce an informative image partitioning which expands on user input to produce a result which reflects this information by detailing interesting areas with a finer partitioning (placing more, smaller-sized segments) while abstracting uninteresting areas with coarser partitioning (placing fewer, larger-sized segments).

In this paper, we present a new framework for incorporating user-defined interestingness information in order to produce such an informative partitioning. Instead of segmenting an image at the object level, we split an image into a large number of oversegments, small-sized image patches containing similar visual information. A user expresses the relative interestingness of parts of the image by labelling a set of example pairs of oversegments as "should be merged" or "should not be merged", that is, as side information. Importantly, it is up to the user to define which areas should not be merged, allowing them to dictate objects that may otherwise be seen as background as interesting objects, or to define the focus of an image as uninteresting.

From this side information, an SVM classifier is learned to describe the relationship between pairs of oversegments. To take spatial context in account, an MRF model is employed to optimally infer the relationship between neighboring oversegments as "merge" or "not". Unlike the standard form, the nodes of the graph in our work denote the relationship of oversegment pairs rather than oversegments themselves, essentially producing an inversion of the edges and nodes in a typical image MRF. The graph-cut algorithm is employed to infer the best MRF configuration. However, we show that the result of a graph-cut with the Potts model, or any other local smoothness criterion cannot be readily used, and a non-trivial post-processing has to be developed to refine the above configuration further, reducing the impact of classification errors.

This distribution of differently sized segments across an image based on the interestingness of their contents provides a novel method for allocating resources to an image for whatever purpose the user has, particularly for classification and compression tasks. Our supervised approach allows us to learn a high quality system for producing this segmentation from a limited amount of side information. The modification we make to the standard MRF image representation to incorporate spatial relationships into our framework has potential application in many areas outside of this paper. We present experimental validation of the ideas presented in this paper, achieving excellent performance across a wide range of image classes.

Clearly, this objective can be seen as an image segmentation problem. However, existing segmentation methods are unable to truly reflect the interest focussing behavior that we describe. Unsupervised segmentation methods such as [2,3] typically generate a segmentation of an image by identifying areas with

similar colour or texture, or by attempting to locate the most likely location of the boundary between different objects. These methods do not take the user's interest into account, and are unable to accommodate user input that expresses this interest. Supervised segmentation methods classify areas of an image based on a model learned from a large amount of training data into a set of pre-defined object classes. However, they are focused on how to accurately infer the class label of each pixel to achieve the best segmentation, rather than the resolution of this segmentation. Intuitively, we can choose to consider user attention for each segmented part *after* segmenting the whole image with an existing method. However, this is not efficient because i) Initial segmentation ignores user input when generating the segmentation, and as such much of the effort is wasted elegantly segmenting non-interesting areas; ii) it forces user attention to be expressed at the level of segmented areas; iii) the number of segments generally has to be pre defined. Alternatively, our technique can be seen as a region merging algorithm, such as that in [4]. However, existing region merging methods are typically unsupervised or take user input in the form of segment biases. They also do not allow for regions to be specifically not merged despite similarity, which is required to produce the partitioning we desire.

2 Generating an Informative Image Partitioning

2.1 Use of Oversegments and Oversegment Features

In order to reduce the size of the graph that we process, and to simplify the acquisition of human input, we use an oversegmentation of the input image in our process. The method used to generate this oversegmentation is largely arbitrary, however it is important for the process that the oversegments produced contain only a small-sized image patch with sufficiently homogeneous visual content.

Superpixel methods are a good way of generating the required oversegments, as they attempt to maximally follow the criteria that we have defined. Mori's superpixel generation method [5], based on the Normalized Cut algorithm [3], produces similarly sized typically homogeneous segments across an image. Superpixels are not constrained to any particular distribution across the image, and this lends itself well to ensuring that multiple image objects do not fall into the same oversegment.

The Superpixel Lattice method of Moore et al [6] provides a viable alternative to Mori's superpixels in our work, and the two methods can be used interchangeably. We did not investigate less homogeneous oversegmentation methods, such as the work of Felzenszwalb and Huttenlocher [2] and the classic watershed image transform [7], for generating our input, as it is difficult to control the number of segments obtained from these methods and they tend to produce very differently sized segments within the image.

We have used a simple feature set for classification in this paper. We use colour histograms in the CIE Lab colour space and a texture set. One ten bin, equal bin width histogram is generated for each colour channel. Each histogram is normalised by the size of the superpixel from which it is extracted to remove

the effect of oversegment size in comparison. We utilize a set of eight texture features, derived using the method presented by Varma and Zisserman in [8]. The average response of each pixel in an oversegment to each of these eight features is taken as a description of the texture of the oversegment. A feature vector $\mathbf{c}$ for each oversegment is constructed from this set of visual information.

2.2 Acquiring User Input

User input is taken in the form of pairs of oversegments and an instruction, either "Merge" or "Do Not Merge" these two oversegments. In this way, we do not require any knowledge of the number of classes of object in the image, nor do we require the user to directly assign labels to any part of the image. Input data can represent one of four things: In-class merges (merging similar oversegments), In-class no-merges (marking an object as interesting), interclass no-merges (Identifying that two oversegments belong to different objects) and finally interclass merges (Identifying that two oversegments belong to different objects, but the user would still like them to be merged).

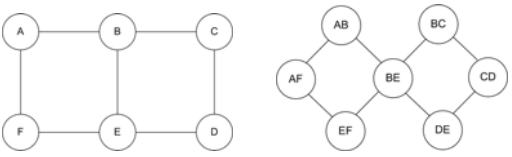
The amount of user input required depends largely on the complexity of the image. Simple problems can be solved with as few as 20 pairs, while more complex scenes can take up to 100 pairs to build a robust model. In class merges, interclass merges and in class no merges can be acquired with a brush input device, but interclass no merges require the user to specify the specific oversegments which should not be merged, as brush input in this case is unreliable.

3 MRF-Based Region Merging

3.1 MRF Model with Edge-to-Vertex Transformation

It is very common in the literature to represent an image as a Markov Random Field [9]. This MRF typically represents each pixel (or oversegment) as a vertex $\mathcal{V}$, with edges $\mathcal{E}$ between adjacent pixels, resulting in an MRF described by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Most applications seek to infer the optimal configuration of this MRF, that is, assigning class labels to each vertex by considering the spatial smoothness. In contrast, *our problem is not going to assign a class label directly to each vertex. Instead, we want to optimally infer the label ("merge" or "not merge") for each edge.* To achieve our goal, we have to conduct an edge-to-vertex transformation, as shown in Figure 1.

Edge-to-vertex transformation has been used in graph and network analysis. For example, the work in [10] applies this method to analyze a city traffic network, in which roads are mapped to vertexes and intersections to edges between vertexes. This transformation is taken by our approach to map the edges in the preceding MRF graph to the vertexes of a new graph. By doing so, each vertex in the new graph denotes the relationship between superpixels, and its label is either "merge" or "not merge". Formally, we define a new graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ with vertices $\mathcal{V}^*$ (corresponding to the edge $\mathcal{E}$ in the preceding graph) and edges



(a) (b)

Fig. 1. (a) Initial MRF representation of an image $\mathcal{G}$. (b) The transformed graph $\mathcal{G}^*$ in which we represent edges from $\mathcal{G}$ as nodes.

$\mathcal{E}^*$. α_i^* ($\alpha_i^* \in \{0,1\}$) is the label assigned to $\mathcal{V}_i^*$. Each vertex $\mathcal{V}_i^*$ retains the visual information c_i of the two vertices that it connects as an edge in the old graph. This formulation allows us to simplify the problem into a simple binary labelling, which allows us to employ the standard MRF inference method to find the optimal labels. Because semantic concepts of an image are formed by groups of oversegments and users often pay attention to meaningful objects, adjacent oversegments will often share the same merging (or non-merging) labels. This implies that the distribution of merge and non-merge labels will be distributed in a smooth fashion across the image, except at boundaries among interesting and uninteresting areas.

With the above MRF model, we represent our inference problem as minimisation of the cost function in equation (1), taking the form:

$$U(\alpha^*; \mathbf{c}^*) = \lambda \sum_{i \in \mathcal{V}^*} U_1(\alpha_i^*; c_i^*) + \sum_{ij \in \mathcal{E}^*} U_2(c_i^*; c_j^*)\delta[\alpha_i^* \neq \alpha_j^*] \tag{1}$$

Its data term U_1 represents the cost of assigning vertex i to $\alpha_i = 0$ (do not merge) or $\alpha_i = 1$ (do merge). Its local smoothing term U_2 represents the cost associated with assigning different labels to adjacent nodes. Derivation of the data term used is discussed in section 3.2. Using this form for our cost function allows us to use a graph cut algorithm to find an optimal labelling. Specifically, we use the max flow algorithm described in [11] to efficiently find a minimum cut over a graph where edge weights are defined as:

$$z_{ij} = U_2(c_i^*, c_j^*) = \frac{1}{||c_i^* - c_j^*|| + 1}; z_{i0} = \lambda \cdot U_1(\alpha_i^* = 0; c_i^*); z_{i1} = \lambda \cdot U_1(\alpha_i^* = 1; c_i^*) \tag{2}$$

where z_{i1} is the cost of assigning a node to merge, z_{i0} is the cost of not merging and z_{ij} is calculated based on the similarity between nodes. As shown, we reflect this smoothness by a simple Potts model, assigning the penalty for assigning different labels to adjacent nodes based on the difference between the features in the two nodes. Assigning smoothness in this way allows us to reduce the cost of assigning different labels when there is a significant difference between two nodes, as expected.

3.2 Learning the Data Term with a Support Vector Machine

The data term from Equation (1) is computed as follows. Let c^*_{ia} and c^*_{ib} denote the visual feature vectors of the two vertices that $\mathcal{V}^*_i$ connects when it is an edge in the initial graph. The problem is to estimate the possibility that $\mathcal{V}^*_i$ is labeled as 0 or 1 solely based on c^*_{ia} and c^*_{ib}. Since a user has labeled some example pairs of superpixels as merged or not, calculating the data term can be solved by a learning task.

An SVM classifier is trained as follows. Suppose a user provides a set of example pairs of superpixels. We stack two superpixels $\mathbf{c}^*_{ia}$ and $\mathbf{c}^*_{ib}$ in each pair as a long vector $\mathbf{c}^*_i$. Thus, a training set is obtained as $\{\mathbf{c}^*_1, \alpha_1\}, \{\mathbf{c}^*_2, \alpha_2\}, \cdots, \{\mathbf{c}^*_m, \alpha_m\}$, where $\mathbf{c}^*_i$ is the training vector and α_i is the corresponding label, 0 or 1. Note that whether two superpixels are to be merged or not is independent of its order presented in $\mathbf{c}^*_i$. Hence, for each $\mathbf{c}^*_i$ $(i = 1, \cdots, m)$, we can generate a "shadow" training vector $\overline{\mathbf{c}}^*_i$ in which the order of $\mathbf{c}^*_{ia}$ and $\mathbf{c}^*_{ib}$ is switched and its label remains. Thus, there are $2m$ training samples to be used for SVM training. We use a SVM classifier with a Gaussian RBF kernel of the form: $k(c^*_i, c^*_j) = \exp(-\gamma||c^*_i - c^*_j||^2)$, where γ is a nonnegative parameter which needs to be tuned for our specific application. A misclassification cost parameter C also needs to be tuned to produce optimal classification results. We use a grid search method with ten-fold cross validation on the training set to find optimal values for these parameters.

Feature components in the training set are scaled to the range (0,1) based on the full range of values that are present in the image, not only the training set. Using the popular LibSVM tool [12], we are able to directly determine (with the switch -b in LibSVM) a probability for each vertex $\mathcal{V}^*_i$ being assigned label $\alpha_i = 0$. We define this value as $\Theta[f_{SVM}(c^*_i)]$. From this, we can obtain $U_1(\alpha^*_i = 1, c^*_i) = \Theta[f_{SVM}(c^*_i)]$ and $U_1(\alpha^*_i = 0, c^*_i) = 1 - U_1(\alpha^*_i = 1, c^*_i)$.

3.3 Inverse Edge-to-Node Transformation

After manipulating the graph to incorporate side information in this fashion, we aim to determine whether adjacent nodes in the original graph $\mathcal{G}$ should be merged or not, based on the label assigned in the transformed graph $\mathcal{G}^*$. Clearly, to realise this, we must convert from the node labelling in $\mathcal{G}^*$ back to a labelling on the initial graph $\mathcal{G}$. First, each oversegment is assigned a unique label. Then, we are able to sequentially apply the merges described by $\mathcal{G}^*$, eliminating labels as we merge oversegments. Once all merges have been performed, this labelling corresponds directly to the partitioning of the image.

Unfortunately, the merging behavior described by $\mathcal{G}^*$ cannot always be directly applied, as doing so can produce unresolvable ambiguities in the labelling, in which two nodes should both be merged and not merged.

As such, an additional translation between the edge graph and a partitioning of the image is required. A global translation method is described here with two possible solutions: merge a set of pairs that should otherwise not be merged, which converts some zeros into ones, or split a set of merges by converting ones in the graph into zeros until there are no conflicts.

Applying the first solution is quite simple. One needs only to apply all merges present in $\mathcal{G}^*$. Given the sequential merging of labels that we conduct to generate our partitioning, this will resolve conflicts by allowing merges that our graph cut classifier would otherwise prevent. The second solution prioritises the "do not merge" instruction from $\mathcal{G}^*$ over the merge instruction. This indicates that if $\alpha_i^* = 0$ in the transformed graph, then under no circumstances should the oversegments represented by vertex $\mathcal{V}_i^*$ be present in a single partition, even if there exists a path that would otherwise allow these segments to merge. The location of such splits can be arbitrarily decided, but we are also able to make use of a hierarchical clustering type approach to produce a partitioning in more structured fashion.

3.4 Hierarchical Clustering

Hierarchical clustering is a commonly used clustering method in which the best available merging of two existing clusters is found and applied repetitively, either until a set number of clusters is reached or the best merge passes below some threshold of quality. This lends itself naturally to our application, allowing us to organically grow the partitioning to produce more consistent results.

Cluster Distance and Constrained Hierarchical Clustering. While we do not have a direct measure of the distance between oversegments, the SVM model learned above provides an estimate of the likelihood that any two oversegments should be merged. This can be used to determine the pair of oversegments most likely to be merged, and gives us a non-metric distance measure.

As we aim to preserve the "do not merge" instructions from our edge labelling, we require that any pair of oversegments marked this way remain in separate clusters. This can be taken into account by adding a significant cost to the merging of any two clusters that would break this restriction. Furthermore, we require a spatial restriction such that oversegments and clusters that are not adjacent to one another cannot be merged. This is done by applying a similar cost to non-neighboring clusters as to disjoint clusters.

Constrained Clustering Algorithm. To perform the hierarchical clustering, we first use our SVM to determine the distance between each pair of oversegments in our image. We store these distances as elements in a symmetrical $n \times n$ *distance* matrix, D, where n is the number of oversegments. We then construct from the edge graph a binary *disjoint* matrix X, which is set to one at X_{ij} if oversegments i and j have a "do not merge" instruction between them. Finally, a binary *neighborhood* matrix N is constructed with element $N_{ij} = 1$ if i and j share an edge, and zero otherwise.

We initialise each cluster to contain a single oversegment, then perform hierarchical clustering using a single-link approach, as described in Algorithm 1. The single-link methodology is used here rather than the full-link methodology based on experimental performance. This is likely a result of our non-metric distance function together with the spatial and disjoint constraints we apply.

```
Data: Matrices D, X and N of size n × n(see text)
Result: Set of Clusters
set number of clusters c to n
set cluster contents c_i(s) to cluster number
while more possible merges exist do
    set high value for bestCdist for i ← 1 to c do
        for j ← i to c do
            foreach element a in cluster i and element b in cluster j do
                if (there exists X_ab = 1) or (∀N_ab = 0) then
                    Break
                else
                    minCdist = min D_ab
                end
            end
            if minCdist < bestCdist then bestCdist = minCdist, store ij
        end
    end
    if bestCdist has changed then
        merge clusters ij
    else
        no possible merges
    end
end
```

Algorithm 1. The constrained Hierarchical Clustering Algorithm

4 Experimental Results

4.1 Experimental Setup

For each of the experiments detailed below, we make use of a subset of images from the MSRC data set [13]. We use first 15 entries in the livestock, tree, building, cow and aeroplane subsets of the data set for this test, resulting a total data set of 75 images. All images are segmented using Mori's method [5] to generate approximately 1000 oversegments for each image, and we train a separate model for each image using between 25 and 50 positive and negative training pairs for each image, which corresponds to between 1 and 3% of all neighboring oversegment pairs in the image. The number of training pairs used depended on the complexity of the image. Void areas from the data set are always treated as uninteresting. Some images from the first livestock category were removed from these calculations, as they contained only uninteresting areas. The same set of training pairs is used in each of the experiments presented.

4.2 Focussing Performance

Primarily, we are interested in determining how effective our merging algorithm is at successfully reducing the number of segments in uninteresting regions while leaving a significant number of segments in areas of interest. To do this, we have defined one class of object as the object of interest, as defined in Table 1 and we determine the number of segments present within these objects and within uninteresting objects both before and after applying our method. As we are interested in the amount of merging that takes place in these areas, we have presented the difference between the average number of partitions in these areas as a percentage of the initial number of oversegments in the same area.

Table 1. The partition focussing performance of our approach. We show the average reduction in the number of partitions within interesting and uninteresting objects after applying our method under several resolutions of the loop condition for five different image classes. Method A prioritises merging, Method B prioritises not merging and HC is a Hierarchical approach to Method B.

	Interesting Object (Average% Reduction)			Uninteresting Object (Average% Reduction)		
Interesting Object	Method A	Method B	HC	Method A	Method B	HC
Livestock	33.8%	32.0%	30.5%	96.7%	95.7%	95.4%
Trees	16.0%	15.2%	14.0%	95.5%	94.3%	93.9%
Buildings	18.0%	17.0%	16.2%	94.7%	93.8%	93.4%
Aeroplanes	27.7%	25.6%	23.7%	95.7%	94.7%	93.6%
Cows	28.9%	27.0%	25.3%	96.2%	94.7%	94.3%
Overall	24.0%	22.5%	21.1%	95.7%	94.6%	94.0%

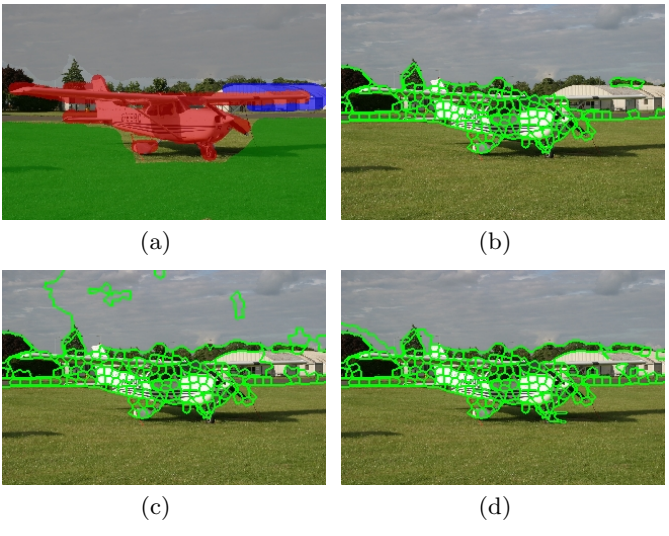


Fig. 2. (a) Input image with ground truth label overlay. (b) Partitioning generated from Method A. (c) Partitioning generated from method B. (d) Partitioning generated from Hierarchical Clustering. The green lines indicate partition boundaries. As can be seen, the partitioning in (b) has allowed excessive merging to the right of screen, combining trees and building with sky. This has been fixed in (c), but with the introduction of unwanted partitions. (d) produces a higher-quality partitioning than (c).

As can be seen from these results, we show a significantly higher reduction in uninteresting areas than in interesting ones under all translation methods. These results demonstrate the effectiveness of our approach at producing a focussed partitioning of an input image given some user information.

In addition to observing the quality of the partition focusing behaviour, we also investigated the amount of segmentation error associated with each of the partitions generated. To calculate this, we first find the partitions that contain

more than one class of object from the ground truth, and then we find the number of pixels that would be classified as a different class if these partitions were to be classified according to the predominant label. As expected, we find that there is some increase in the segmentation error after merging, ranging between less than 0.5% and 7% of the image. The loop resolution method has a large impact on this error, which shows that there is a tradeoff between reducing the number of partitions placed over uninteresting areas and reducing the segmentation error. Figure 2 shows a sample initial image, ground truth label and the image partitions produced by allowing merges, disallowing merges, and with various hierarchical clustering schemes.

Our results are slightly skewed due to disparities between the human labeled ground truth information and the true location of the boundary between objects, which typically results in a larger reduction of the number of partitions in interesting areas being reported than is actually observed. This affects both the focussing effect and the segmentation error calculations and is particularly prevalent in the livestock and aeroplanes subsets, as the interesting region in ground truth is noticeably larger than the actual interesting object. Additionally, when more than one uninteresting area was present in the image, our method reported less reduction in these areas as it retains separation between these objects.

The data presented above also allows us to compare the effectiveness of each of the methods that we have used to resolve ambiguities in the instructions between nodes. We are also able to see the effect of the loop resolution method when translating from the edge graph to image labels. Method A, which favors merges, produces slightly fewer partitions in regions of interest, but significantly fewer in uninteresting regions than Method B, which favors keeping objects separate. The hierarchical clustering approach mimics the behaviour of Method B, but provide an obvious improvement in partitioning performance.

5 Conclusions

We have presented in this paper a framework for mimicking the top-down 'spotlight focussing' aspect of the human visual system in digital images, together with a new way of representing an image using a Markov Random Field to incorporate side information. Under this framework, we demonstrate that we are able to produce a high-quality focussed partitioning of an input image given only a relatively small amount of side information.

References

1. Weichselgartner, E., Sperling, G.: Dynamics of automatic and controlled visual attention. Science 238, 778–780 (1987)
2. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. Int. J. Comput. Vision 59(2), 167–181 (2004)
3. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 888–905 (2000)

4. Nock, R., Nielsen, F.: Semi-supervised statistical region refinement for color image segmentation 38(6), 835–846 (2005)
5. Mori, G.: Guiding model search using segmentation. In: ICCV 2005, vol. 2, pp. 1417–1423 (October 2005)
6. Moore, A., Prince, S., Warrell, J., Mohammed, U., Jones, G.: Superpixel lattices. In: CVPR 2008, June 2008, pp. 1–8 (2008)
7. Meyer, F., Beucher, S.: Morphological segmentation 1(1), 21–46 (September 1990)
8. Varma, M., Zisserman, A.: Classifying images of materials: Achieving viewpoint and illumination independence. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 255–271. Springer, Heidelberg (2002)
9. Li, S.Z.: Markov random field modeling in image analysis. Springer, New York (2001)
10. Rosvall, M., Trusina, A., Minnhagen, P., Sneppen, K.: Networks and cities: An information perspective. Physical Review Letters 94 (January 2005)
11. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. 26(9), 1124–1137 (2004)
12. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001) Software available at, `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
13. Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 1–15. Springer, Heidelberg (2006)