Template patch driven image segmentation*

Branislav Mičušík and Allan Hanbury

Pattern Recognition and Image Processing Group, Inst. of Computer Aided Automation, Vienna University of Technology
Favoritenstr. 9/1832, A-1040 Vienna, Austria
{micusik,hanbury}@prip.tuwien.ac.at

Abstract

We present a method that partitions a single image into two layers, requiring that one layer has similar properties in terms of pixel colour variation to a provided template patch. First, the paper provides a new view on defining a similarity function for a pixel with its small neighbourhood to be part of the texture described by the template patch. This results in better description of pixels near the texture boundary. Second, it is shown how the Maximally Stable Extremal Regions (MSERs), originally designed for wide baseline stereo matching, can be used to locally merge pixels having the same intensity and thus reduce the dimension of the graph representing the image. The MSERs help in texture description and yield significant reduction of memory and computation time. Finally the graph is fed into the min.cut/max.flow algorithm to cut the graph into two parts. Performance of the method is presented on some images from the Berkeley database. Finally, restrictions in using the method are discussed.

1 Introduction

Fully automatic image segmentation is well known to be an ill-posed problem since there is neither a clear definition nor an objective measure of a correct segmentation. As it has been shown [10], people do not produce similar segmentations on the same images without a specification of what should be segmented and how detailed the segmentation should be.

It has been pointed out that for doing a semantically correct image segmentation it is essential to take into account a priori information about objects being segmented. Such information is usually provided

i) by the user through specifying background/foreground properties [14, 13, 4, 2], e.g. through marking some pixels in the image, or

ii) by assuming that the object being segmented moves [18, 1, 7], or

iii) by learning of object properties, i.e. models, and their mutual relation and appearance from a manually labeled training database [5, 17, 16]. In this case the image segmentation and object recognition are done simultaneously.

*This work was supported by the Austrian Science Foundation (FWF) under grant SESAME (P17189-N04), and the European Union Network of Excellence MUSCLE (FP6-507752).

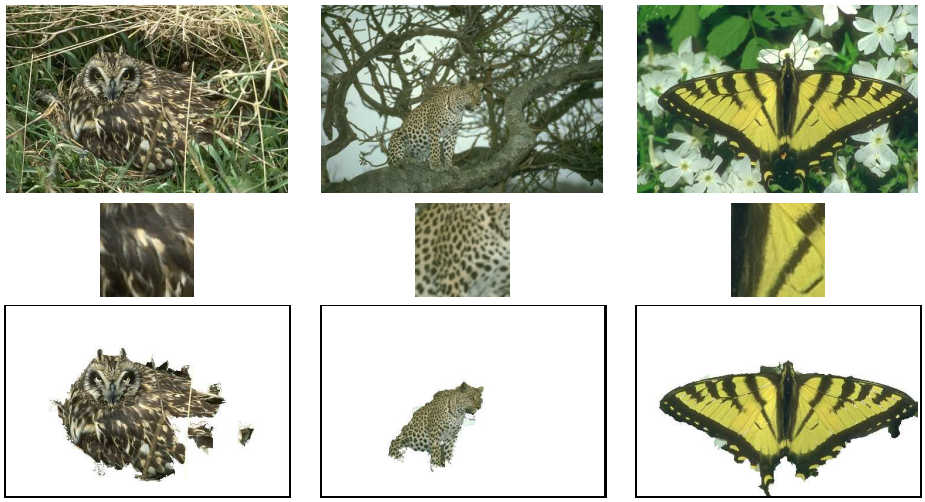
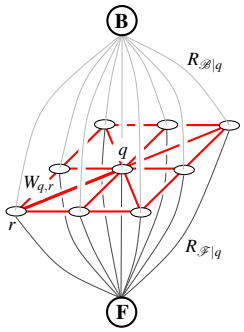


Figure 1: Three examples of segmentation of textured surfaces. Top row: input images. Middle row: the template patches in detail. Bottom row: segmented regions having similar properties to the template patch.

Without the above mentioned steps it is only possible to partition the image into regions coherent in colour and texture which do not follow any semantic cues [15, 6, 12]. However, as psychophysics experiments have shown [19], at the beginning of the human procedure leading to scene understanding, some pre-segmentation using boundaries and regions coherent in texture and colour is performed. Humans then use a huge object database in their brains to tune the segmentation handling large occlusions, strong shadows and geometric distortions. It implies that a rough pre-segmentation has to be done before running any iterative recognition $\leftrightarrow$ segmentation algorithm.

In this paper we focus on the first group of segmentation methods mentioned above, where the objects/regions of attention are described in advance, in our case by a small representative template patch. The patch is provided by a user or by any salient texture detection method. Our proposed method segments an image into two layers where one is similar to this patch in terms of pixel colour variation. The need for a background template patch is omitted, which is an advantage compared to previous work [14, 4, 2]. By excluding the need for a background model we can use one foreground template patch to search for the same object over many images even though the background changes. However, compared to the most similar previous approaches [14, 3, 2], our method cannot be viewed as a tool for doing an accurate manual segmentation driven by a user. Our method is oriented towards being used in a chain leading to the automatic segmentation / detection of an object described by the template patch.

The main contribution of this paper is twofold. First, we introduce a new view on the similarity measure of an arbitrary pixel with its small neighbourhood to be part of the foreground. This measure performs better on the boundary between the foreground and background regions compared to standard approaches. Currently the measure is based on the colour variation of pixels which may fail in some cases discussed later. However,



edge	cost	region
$\{q,r\}$	$W_{q,r}$	$\{q,r\} \in \mathscr{N}$
$\{q,F\}$	$\lambda R_{\mathscr{F}\|q}$	$\forall q$
$\{q,B\}$	$\lambda R_{\mathscr{B}\|q}$	$\forall q$

Figure 2: Left: Graph representation for a 9 pixel image and a table defining the costs of graph edges. Symbols are explained in the text. The tuning parameter λ was set to 0.1.

what is important is the way in which the measure is formulated. Adding other features can be done in a similar way. Second, we show how Maximally Stable Extremal Regions (MSERs) [11], originally designed for wide baseline stereo, can be used to merge together pixels having the same intensity. This reduces the size of the graph built on the image and contributes to better handling of textures.

The structure of the paper is as follows. First, the problem is formulated in Sec. 2. Then the building of the graph, highlighting the difference to other approaches, is described in Sec. 3. The final algorithm is summarized in Sec. 4 and some results are given in Sec. 5. Finally, discussion of restrictions in the use of the method and the conclusion complete the paper.

2 Problem formulation

Many segmentation methods are based on minimization of the well-known Gibbs [4, 2, 12, 16] or another type of energy [1, 13, 6, 17]. The form of the energy is not so significant. The energy usually consists of variously encoded data and a smoothness term which keeps the segmentation consistent. In the following we use the Gibbs energy to formulate the segmentation problem.

Suppose that the image is a graph $\mathscr{G} = \langle \mathscr{V}, \mathscr{E} \rangle$, where $\mathscr{V}$ is a set of all vertices and $\mathscr{E}$ is a set of all edges connecting adjacent vertices. The Gibbs energy of the segmentation $\mathbf{x}$ is

$$E(\mathbf{x}) = \sum_{i \in \mathscr{V}} E_{data}(x_i, \mathbf{z}_i) + \lambda \sum_{(i,j) \in \mathscr{E}} E_{smooth}(x_i, x_j, \mathbf{z}_i, \mathbf{z}_j), \tag{1}$$

where $\mathbf{x} = (x_0, x_1, \ldots)^\top$ corresponds to a vector with label x_i for each vertex. We concentrate on a bi-layer segmentation where the label x_i is either 0 (background) or 1 (foreground). $\mathbf{z}_i$ corresponds to the measurement in the i-th graph vertex, e.g. to a 3 colour vector. λ is a weighting constant controlling the influence of the smoothness term. The smoothness term describes how strongly neighbourhood pixels are bound together. It is usually set using an Ising prior, or better as a colour gradient [4, 2, 16] or colour+texture gradient [12] between adjacent pixels having different labels. The data term describes how likely the pixel is part of the foreground or the background.

An optimal segmentation is obtained as a minimum of the energy $E(\mathbf{x})$ over all possible $\mathbf{x}$, i.e. $E^* = \text{argmin}_{\mathbf{x}} E(\mathbf{x})$. The minimization problem is equivalent to searching for the minimum cut or maximum flow in the graph [3], see Fig. 2, splitting the graph into two parts. There exists an efficient algorithm for finding the min.cut/max.flow [4] which is also used in our proposed method.

Usually the vertices of the graph correspond to pixels in the image and edges connect adjacent pixels in some local neighbourhood. However, in our case one to one correspondence between pixels and graph vertices is violated as pixels in each MSER [11] are assigned just to one vertex, described later.

It has been pointed out that what mostly matters in segmentation methods is the form (model) of *the data term* and not really the strategy for obtaining the segmentation, i.e. whether the segmentation is found by searching for the min. cut in the graph [3], minimum of some energies [1, 17], or optimal Geodetic Active Region [13].

3 Constructing the graph

Fig. 2 shows the graph used here which is split by the min. cut/max. flow algorithm to obtain the final segmentation [4]. In this section we explain the difference in construction of the graph and setting the data term compared to standard approaches.

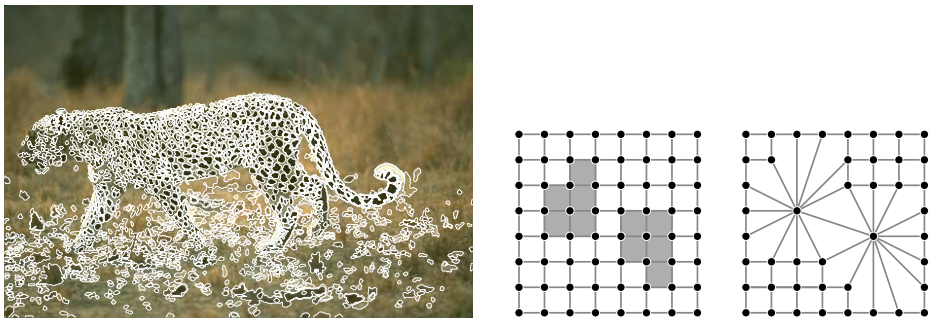
3.1 Graph contraction

Assigning each image pixel to a different node in a graph leads to a large graph which is very impractical for two reasons: i) the large memory consumption, ii) longer computation time of operations done on the graph, e.g. min. cut. Standard approaches reduce the size of the image to keep the graph dimension within reasonable bounds. However, this the operation destroys details in the image, specially texture structures.

We take a different approach. We do not reduce the size of the image but we merge pixels having similar colours. As we need edges in the image to be preserved we found MSERs [11] to be suitable technique for this purpose. The MSERs are the regions which are stable across multiple thresholds during thresholding of the image from minimum to maximum intensity and vice versa. The edges in the image are usually the boundaries of the regions and therefore are not lost by merging the pixels into regions. Using MSERs has an advantage compared to doing, e.g., watershed on the input image [8], since we compress only places in the image where it is possible. Where there are no regions we leave pixels alone and assign individual nodes to them in the graph.

Fig. 3 shows an example of detected MSERs in the leopard image and depicts the principle of graph node reduction. All pixels lying in one MSER are merged to one node in the graph, preserving all edges except for edges inside the MSER. The edges inside the MSERs would create self-loops at the contracted node. In our approach we neglect the self-loops.

Producing MSERs, i.e. image thresholding, preserves the relation $R_t \subseteq R_{t+1}$, where R_t, resp. R_{t+1} is the set of all pixels inside a region at level t, resp. $(t+1)$ of the thresholding. This is an important property which is used in situations where a larger region contains smaller one. In such situations pixels in the smaller region are assigned to an own node in the graph and the pixels lying in the larger region but *not* in the smaller one are assigned to another own node.



(a) (b) (c)

Figure 3: Graph contraction. (a) The leopard image with MSERs superimposed. (b) An illustration of an image cutout in standard regular grid representation. Two gray polygons correspond to the MSERs. (c) The contracted graph from (b) taking into account MSERs and 4-point connectivity.

3.2 Smoothness term

We use 4-point connectivity in the illustration shown in Fig. 3, however, more general neighbourhoods can be used. In our implementation we compute the distance transform on each region and take neighbours within a pre-defined distance controlling the size of the neighbourhood. The distance 1 corresponds to a 4-point neighbourhood, 1.5 to an 8-point neighbourhood, the distances larger than 2 correspond to non-planar graphs. We have used 1.5.

Generally, larger neighbourhoods contribute to better stability of the segmentation in terms of smoothness, however, it is more memory and computationally demanding. In our case, a large neighbourhood is not required, since the MSERs cover pixels in a larger area. Thus in essence a larger area than the 8-point neighbourhood used for creating graph edges is captured. Moreover, pre-merging of pixels with the same intensities into MSERs reduces redundant edges. The most important edges for segmentation are those which connect pixels with different colours. Such edges are preserved by the contraction step.

The smoothness term between two adjacent nodes q, r is computed using the Euclidean distance in the CIELab colour space similarly to [3] as

$$W_{q,r} = \exp(-\gamma \|\mathbf{z}_q - \mathbf{z}_r\|^2), \tag{2}$$

where γ is a tuning parameter (0.5 in our implementation). $\mathbf{z}_q$ is the colour vector at node q. When the node represents a region, $\mathbf{z}_q$ is the mean colour of pixels assigned to the region q.

The problem arising after graph contraction is that one node representing an MSER can have many more edges than a node corresponding to a single pixel, see Fig. 3(c). It may cause the sum of weights of all edges going from the node to be too large to be cut out from pixels in the surroundings even though it would correspond to another label. We solved this problem by normalizing the weights on edges. We divide all edge weights going from one node to sum to 1, i.e. $\sum_{r \in \mathcal{N}} W_{q,r} = 1$. By this we obtain an *oriented graph* which indirectly takes into account the number of neighbours. The normalization has led

to better performance as experiments have shown.

3.3 Data term

The form and setting of the data term is the most important for segmentation consistency. Usually for the description of the background and the foreground region either parametric (Gaussian Mixture Model (GMM) on the colour histogram [2]) or non-parametric (whole colour histograms [3, 12]) models both built from pixels marked as the background and the foreground, is employed. In this work we avoid the use of histograms for describing the background or the foreground and propose a new strategy.

We assume that a query patch containing colour and structure of interest is provided. Typically, in similar approaches, the feature vectors are created at some interest points in a query texture and an input image. Then feature vectors are compared crosswise to find potential matches. Depending on the number of established matches it is decided whether the query texture occurs in the input image or not.

The problem of standard approaches is that the feature vectors, e.g. well known SIFTs [9] or many others, are computed in a small neighbourhood around the pixel. It means that for pixels near the boundary of a texture the feature vector is affected by the background pixels. This may lead to poor matching or inappropriate setting of the probability that the particular pixel belongs to the texture described by the query patch. Moreover, we need evaluate the measure on each pixel and not just at pixels of interest. To cope with this we design the following strategy.

The penalty of the pixel being foreground is defined as follows

$$\begin{aligned} R_{\mathscr{F}|q} &= \exp(-\alpha f(\mathbf{z}_q, T)), \\ R_{\mathscr{B}|q} &= 1 - R_{\mathscr{F}|q}, \end{aligned} \tag{3}$$

where the T represents a provided template patch, f refers to a similarity function, and α is a tuning parameter discussed later.

The MSERs are detected in the template patch and, as for the input image, pixels inside the MSERs are merged together. Let N_T denote the number of all nodes in the template patch. At each i-th node we build a feature vector composed of three sub-features, i.e. $\mathbf{t}_i$, $\mathbf{t}_i^{min}$, $\mathbf{t}_i^{max}$. In our case the vectors are the 3-element colour vectors expressed in CIELab colour space. $\mathbf{t}_i$ corresponds to the colour of the pixel (or mean of the colours of pixels inside a region), $\mathbf{t}_i^{min} = \text{argmin}_{\mathbf{t}_j, j \in \mathscr{C}_i} \|\mathbf{t}_j - \mathbf{t}_i\|^2$, and $\mathbf{t}_i^{max} = \text{argmax}_{\mathbf{t}_j, j \in \mathscr{C}_i} \|\mathbf{t}_j - \mathbf{t}_i\|^2$. $\mathscr{C}_i$ is some local neighbourhood around i-th node. We take a circle with some pre-defined radius (8 pixels in our case) around the i-th point or region. We can pre-compute the feature vectors for the whole template patch. We then cluster feature vectors which are close enough in terms of Euclidean distance in the colour space. For an arbitrary point $\mathbf{z}_q$ we can formulate the similarity function of the point being foreground as the following

$$f(\mathbf{z}_q, T) = \min_{1<i<N'_T} \left(\beta \|\mathbf{t}_i - \mathbf{z}_q\|^2 + \min_{j \in \mathscr{C}_q} \|\mathbf{t}_i^{min} - \mathbf{z}_j\|^2 + \min_{j \in \mathscr{C}_q} \|\mathbf{t}_i^{max} - \mathbf{z}_j\|^2 \right) / c_i, \tag{4}$$

where c_i is the number of vectors belonging to the same cluster, N'_T is number of clusters, and β is a constant controlling the influence of similarity between nodes (we use 20, obtained by experiment). We solve this minimization problem by brute force at each graph node q.

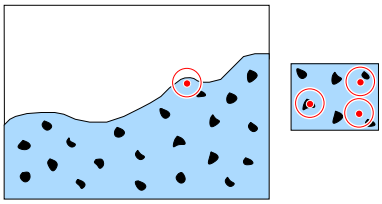


Figure 4: Texture description. Left: Texture with an example of circle $\mathscr{C}$ at a point near the texture boundary. Right: A texture patch with three most representative points.

Eq. (4) says that we look in area $\mathscr{C}$ around the given pixel/region in the input image for the points being closest to min/max points stored in a template patch feature vector. By this strategy, part of the circle in the input image may come from the background region and even then, on the remaining part of the circle, min/max colours closest to one of the template feature vectors can still be found, see Fig. 4. Such a strategy guarantees better performance than to compute features at each node in the input image and then compare them crosswise to template features as most techniques do.

4 Algorithm

We shortly summarize all the steps leading to the final image segmentation:

1. Find MSERs[1] in the input image, create an oriented graph respecting weights $W_{q,r}$ given by Eq. (2) and perform further normalization so that the sum of outgoing edges is 1.

2. Find MSERs in the template patch and pre-compute feature vectors $\mathbf{t}_i$, $\mathbf{t}_i^{min}$, $\mathbf{t}_i^{max}$ at each node. Cluster the feature vectors to obtain the most representative ones with weights c_i.

3. Process all nodes in the input image graph by searching for the closest template feature vector, i.e. evaluate the similarity function $f(.)$ in Eq. (4). and set the data term as the penalties $R_{\mathscr{F}|q}$, $R_{\mathscr{B}|q}$ in Eq. (3).

4. Find the final segmentation by searching for min. cut/max. flow in the constructed graph by the algorithm [4].

5 Experiments

We performed tests on images in the Berkeley dataset[2], see some results in Fig. 5. Even though the formulation of the data term is simple it returns reasonable results. Usually the textures are composed of colour changes contributing to good performance of the proposed data term. The situation in which the method fails is when different textures with different structure but the same colours are in the image, see Fig. 5 (last row). This

[1] http://www.robots.ox.ac.uk/∼vgg/research/affine/

[2] http://www.cs.berkeley.edu/projects/vision/grouping/segbench

directly follows from the definition of the similarity function in Eq. (4). To solve this problem, features other than simple colour changes would also have to be taken into account, see discussion in Sec. 6.

The final segmentation is very sensitive to the parameter α in Eq. (3) due to poor formulation of $R_{\mathscr{B}|q}$ in Eq. (3). The reason is that we have no information about the background. We set $\alpha = \frac{\langle f(\mathbf{z},T)\rangle}{10}$ where $\langle \ldots \rangle$ is the expectation over all edges joining F (foreground) node. Sometimes better segmentation can be obtained by slightly changing this value. Fig. 5 shows both automatically obtained and manually tuned segmentations. However, it is an open issue to obtain the correct value of α fully automatically.

The method is relatively slow since many simple comparisons are required (it would be possible to run it on GPU instead of CPU). Our current implementation takes 40-60s on 321×481 pixel images on a Pentium 4@2.8GHz using a not very optimized Matlab/C implementation.

6 Discussion

The are two major issues of the proposed strategy which are the topics of our current ongoing research: affine and illumination/colour invariance. We use the same circle for computing feature vectors in the template patch and the same large circle for searching for closest min/max values, Eq. (4). This may be done in a more clever way, e.g. by trying more circles or changing circles to ellipses to get affine invariance. However, it would increase computation time, which is already too high. Some trade-off has to be accepted.

To increase illumination/colour invariance, features other than colour differences have to be taken into account, e.g. gradient magnitude/orientation or scale-space searching at the pixel/region neighbourhood.

7 Conclusion

We have presented a method for a detection / segmentation of the regions in the input image defined by a provided template patch. We have introduced the use of MSERs in graph construction to decrease the dimensionality of the graph and to get a better representation of the image. We have designed a new data term used in a minimization process leading to a final segmentation. The main novelty is in the formulation of the data term enabling the suppression of the influence of the background on segmentations of points near the foreground / background boundary. The method still requires further features other than colour similarity. Even though the features are relatively simple the method performs well on most of the images.

References

[1] N. Apostoloff and A. Fitzgibbon. Bayesian video matting using learnt image priors. In *Proc. CVPR*, pages I:407–414, 2004.

[2] A. Blake, C. Rother, M. Brown, P. Perez, and P. S. Torr. Interactive image segmentation using an adaptive GMMRF model. In *Proc. ECCV*, pages I: 428–441, 2004.

[3] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proc. ICCV*, pages 105–112, 2001.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.

[5] P. Carbonetto, N. de Freitas, and K. Barnard. A statistical model for general contextual object recognition. In *Proc. ECCV*, pages 350–362, 2004.

[6] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In *Proc. CVPR*, pages II: 54–61, 2003.

[7] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Probabilistic fusion of stereo with color and contrast for bi-layer segmentation. *PAMI*, 2006. to appear.

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

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[10] D. Martin, Ch. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, pages 416–425, 2001.

[11] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. BMVC*, pages I: 384–393, 2002.

[12] B. Mičušík and A. Hanbury. Automatic image segmentation by positioning a seed. In *Proc. ECCV*, pages II: 468–480, 2006.

[13] N. Paragios and R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *IJCV*, 46(3):223–247, 2002.

[14] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proc. CVPR*, pages I:18–25, 2000.

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

[16] J. Shotton, J. Winn, C. Rother, and A. Criminisi. *TextonBoost:* Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proc. ECCV*, pages I: 1–15, 2006.

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

[18] Y. Wexler, A. Fitzgibbon, and A. Zisserman. Bayesian estimation of layers from multiple images. In *Proc. ECCV*, pages III: 487–501, 2002.

[19] S. Wolfson and M. Landy. Examining edge- and region-based texture analysis mechanisms. *Vision Research*, 38(3):439–446, 1998.

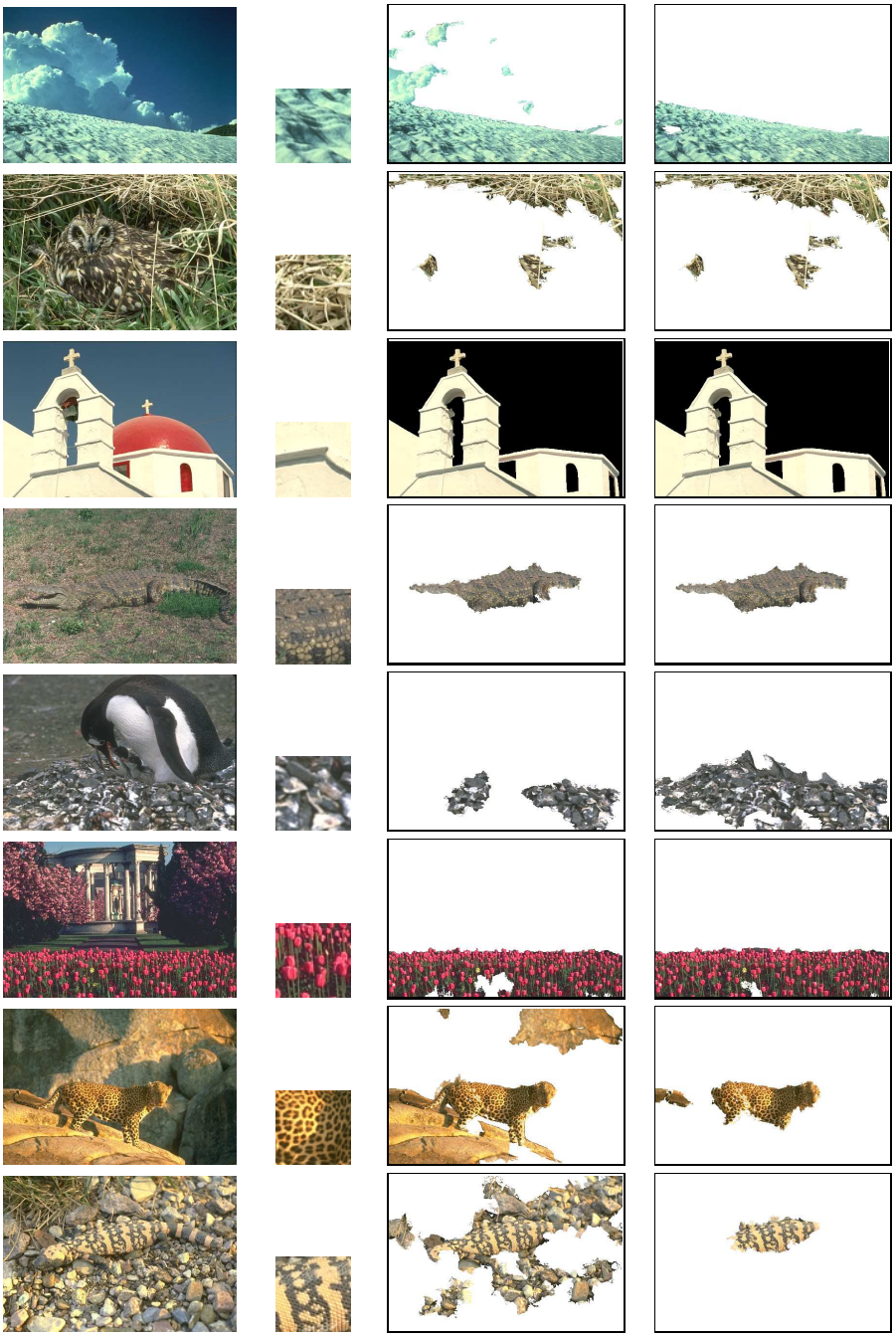


Figure 5: Results. 1st column: Input colour images. 2nd column: Provided template patches. 3rd column: Automatic segmentation results. 4th column: Manually tuned segmentations through parameter α in Eq. (3).