Colour perception graph for characters segmentation

Cyrille Berger

Linköping University
Department of Computer and Information Science
581 83 LINKÖPING, SWEDEN
cyrille.berger@liu.se
http://www.ida.liu.se/~cyrbe

Abstract. Characters recognition in natural images is a challenging problem, as it involves segmenting characters of various colours on various background. In this article, we present a method for segmenting images that use a colour perception graph. Our algorithm is inspired by graph cut segmentation techniques and it use an edge detection technique for filtering the graph before the graph-cut as well as merging segments as a final step. We also present both qualitative and quantitative results, which show that our algorithm perform at slightly better and faster to a state of the art algorithm.

1 Introduction

Segmentation of an image has a wide range of application, it is commonly used in image understanding to split up the objects contained in the image. Images can contain different type of objects, like animals, cars or text. Those different types have very different characteristics and require the use of different type of algorithms [1]. For instance, most characters in a text are strokes of a constant width of the same colour, they also have high contrast with the background, however, animals will have very complex shapes, with different colours and textures. The main motivation behind this article is the extraction and detection of characters in natural images, which require an algorithm for extracting sharp and uniform segments in images.

Segmentation by region growing and edges detection A first approach to image segmentation is to use edges between region [2]. An other idea is to consider that image segmentation is about aggregating pixels in group based on their similarity: based on this idea, a region growing algorithm was proposed in [3], however since both approaches have advantages and drawbacks, a combination gives better results [4].

Depending on the selection of seeds, region growing algorithms will give different results, [5] show that the best segmentation algorithm that combine region growing and edge detection relies on using edges to find good seeds [6].

But it would be better to use a method that does not require the selection of seeds at all.

Graph-based techniques for automatic segmentation Images can be perceived as a graph where pixels are nodes and adjacent pixels are connected by an edge [7], this representation of an image allows to apply many of the graph theory algorithms. For instance, the graph cut algorithm is commonly used for solving the supervised segmentation problem [8].

The graph formalism can also used for the unsupervised problem to solve the seed problem. In [9], Felzenszwalb propose to compute a colour distance between each pixels and then to start the region growing process by first connecting pixels which have a small distance, until the sum of all the distance in a region reach a certain threshold. Depending on the choice of the threshold, this algorithm is either too eager or too conservative, making it diffcult to segment characters in images.

Human Colour Perception The most commonly used colour model for images is *RGB*, it is used for sensing, displaying and processing. However it has a poor representation of colour, the *Lab* [10] colour model was designed to model the perception of colour by human, which results in an interesting property, the Euclidean distance between two *Lab* pixels (often noted $\varDelta E^*_{ab}$) is a good representation of the distance between two colours. And it is considered that $\varDelta E^*_{ab}(c_1, c_2) < 20$ means that the human eye cannot make a difference between the two colours c_1 and c_2 which is very convenient, since it allows to define a meaningful threshold.

In [11], Karatzas makes use of that property, he suggested a two steps algorithm, in the first step, pixels are grouped together if the $\varDelta E^*_{ab}$ between a pixel and the average colour of a segment is below the 20 threshold. In a second step, he suggested some heuristic for merging the segments.

Overview of our approach In our approach, colour are modelled using the Lab colour space. The image is represented as a graph of pixels. Pixels are classified on whether they belong to an edge or are interior pixels. Using this classification, edges are removed from the graph. Then the graph is segmented using a modified version of the efficient graph algorithm [9]. After that segmentation, the classification results are used to merge segments together.

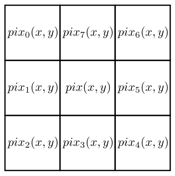


Fig. 1. Notations used for neighbour pixels

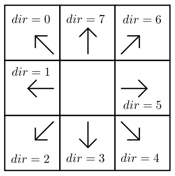


Fig. 2. Direction indices to neighbour pixels.

Notations and Definitions We will adopt the following notations in this article, $pix(x, y)$ represent the pixel at coordinates (x, y) in the image. $pix_i(x, y)$ is one of the neighbouring pixel of (x, y) (see figure 1), those pixels are ordered so that $pix_{i-1}(x, y)$ and

$pix_{i+1}(x, y)$ are neighbour pixels of $pix(x, y)$ and $pix_i(x, y)$. $pix_{i+4}(x, y)$ is the opposite pixel of $pix_i(x, y)$, for instance, $pix_0(x, y) = pix(x, y - 1)$, then $pix_1(x, y) = pix(x + 1, y - 1)$ and $pix_4(x, y) = pix(x, y + 1)$.

A similar notation is use to indicate direction (as shown on figure 2). For instance, $dir = 1$ correspond to the horizontal direction, from pixel $pix_1(x, y)$ to pixel $pix_5(x, y)$.

We define the $ratio$ function as follows:

$$\text{if } a < b, ratio(a, b) = \frac{a}{b} \text{ otherwise } ratio(a, b) = \frac{b}{a} \quad (1)$$

2 Colour Perception Graph

2.1 Colour Distance Measurements

In the first step we compute a set of colour distance that will be helpful for the classification process, the first one is the *Neighbour Pixels Distances*, it is the colour distance between a given pixel and its neighbour. With $i \in [0, 7]$:

$$ND_i(x, y) = \Delta E^*_{ab}(pix(x, y), pix_i(x, y)) \quad (2)$$

This measurement allows to determine how much the pixel colour is different from its neighbour, a high value would indicate the presence of an edge. In case $ND_i(x, y)$ has a high value and $ND_{i+4}(x, y)$ has a low value, then the pixel is clearly located near the edge of an area of the image. However if both values are high, it can either means that the pixel is part of a smooth edge (see table 1) or is a one pixel wide segment (for instance, a black line on a white background).

pixel index	0	1	2	3	4	5	6	7
pixels value	0	0	0	100	155	255	255	255
segment label	a	a	a	b	c	d	d	d

Table 1. Illustration of a smooth gradient from an area of value 0 to 255. Based uniquely on the Euclidean distance applied in the Lab colour space, this sequence of pixels would be cut in four segments *a*, *b*, *c* and *d*. In reality, the segments *b* and *c* are artefacts caused by sensor noise and lack of sharpness and pixels *3* and *4* should belong to either segment *a* and *d*.

Using the fact that pixels that belongs to a transition will form a gradient, it is possible to use the *Average Colour Distances* to find out if a pixel is part of a smooth edge, with $dir \in [0, 3]$:

$$AD_{dir}(x, y) = \Delta E^*_{ab}\left(pix(x, y), \frac{pix_i(x, y) + pix_{i+4}(x, y)}{2}\right) \quad (3)$$

The idea behind this measurement is that if the pixel (x, y) is part of a smooth edge, then it will be a mix between the colour in the two different area, hence it should be close in colour to the average to the opposite colours.

Finally, one pixel wide segement can be detected using the *Opposite Neighbour Distances*, with $i \in [0, 3]$:

$$OD_i(x, y) = \Delta E^*_{ab}(pix_i(x, y), pix_{i+4}(x, y)) \tag{4}$$

This last measurement is useful to determine if the pixel is part of a 1-thick area, since a small value of $OD_i(x, y)$ combine with high values on $ND_i(x, y)$ indicate that the current pixel is very different from its neighbour, but that the neighbours are very similar to each other.

2.2 Graph structure

Given an image $\mathcal{I}$, the pixels $pix(x, y)$ are the node of the graph. By default, each pixel is connected to its eight neighbour in the image. Each edge of the graph is labelled with the Euclidean distance between the colours of the two neighbour pixels: $ND_i(x, y)$.

3 Sharp Segmentation

The first section describes the edge classification of each direction in a pixel, the second section explains how this classification is used to pre-process the graph, the third section describe the modified version of the Efficient Graph-Based Image Segmentation [9] and the last two sections describe two segments merging algorithms.

3.1 Classification of pixels as edges or non-edges

This algorithm was designed around the idea of using only thresholds that would either have a meaning, like the one used on ΔE, or that would have limited influence on the end results. An other important aspect considered during the design stage of the algorithm was to make sure that region with a width of one pixel would be detected. Also, state of the art approach to edge detection [12] locate edges on a single pixel, while we need to locate the full extent of the edge on the fuzzy area.

Also for each pixels, our algorithm look if there is an edge in the four possible direction: horizontal, vertical and the two diagonal.

This is a multiple step algorithm, first a set of colour distances is computed around each pixels, then those colour distances are used to compute an orientation and then a classification (edge or not edge pixel), using the classification gives the location of the edges in the image.

Classification The goal of the classification process is to determine if there is an edge in the direction dir, the following classes are defined:

- *interior* this means the pixel does not belong to an edge in the direction dir
- *directed edge* this class represents edges for which a direction has been established (from interior of area to an other area) in the direction dir, in other words, it means that $pix_{dir}(x.y)$ and $pix_{dir+4}(x, y)$ belongs to different segment. Only direction $dir = [0, 3]$ are considered, we will call *directed edge* the class when the edge

is going from $pix_{dir}(x,y)$ to $pix_{dir+4}(x,y)$ and *opposite directed edge* when the edge is going from $pix_{dir+4}(x,y)$ to $pix_{dir}(x,y)$.

- *undirected edge* this class represent an edge for which no direction could be found in the direction dir, this is likely to happen because the strength of the edge is too small, or because the pixel is in the middle of smooth edge
- *1-thick* represent a pixel belonging to a 1-thick area, in other words $pix_{dir}(x.y)$, $pix(x.y)$ and $pix_{dir+4}(x,y)$ all belongs to different segment
- *Centred edge* it is a class when a directed edge is coming in opposition from each direction

The following values are used to determine the class of the pixel:

$$sum(x,y) = \sum_{i=0}^{3}(ND_i(x,y) + ND_{i+4}(x,y)) \tag{5}$$

$$avg(x,y) = \frac{sum(x,y)}{2} \tag{6}$$

$$max(x,y) = \max_{i\in[0,3]}(ND_i(x,y), ND_{i+4}(x,y)) \tag{7}$$

$$rat^{dir}_{norm}(x,y) = ratio(ND_{dir}, ND_{dir+4}) \tag{8}$$

$$rat^{dir}_{across}(x,y) = ratio(OD_{dir}, sum(x,y)) \tag{9}$$

$$rat^{dir}_{average}(x,y) = ratio(AD_{dir}, avg(x,y)) \tag{10}$$

Check the conditions in the following order:
We note $C_{dir}(x,y)$ the class of pixel $pix(x,y)$ in the direction dir

1. $max(x,y) < 5$ implies $C_{dir}(x,y)$ is *interior*, this condition implies that the colour distance between the pixel $pix(x,y)$ and its neighbour pixels is small, and therefore all three pixels are likely to belong to the same segment
2. $max(x,y) > 20$ and $rat^{dir}_{across}(x,y) < \frac{1}{3}$ implies $C_{dir}(x,y)$ is *1-thick*. A colour distance above 20, implies that the two colours are perceived as different colour by the human eyes, at the same time, the low value on $rat_{across}(x,y)$ indicates that the pixel $pix(x,y)$ is not part of a gradient between $pix_{dir}(x,y)$ and $pix_{dir+4}(x,y)$.
3. $max(x,y) > 20$ and $rat^{dir}_{norm}(x,y) < \frac{2}{3}$ implies $C_{dir}(x,y)$ is a *directed edge*. A small value for $rat_{norm}(x,y)$ indicates that there is a larger difference between $pix_i(x,y)$ and either of its neighbour pixel $pix_{dir}(x,y)$ or $pix_{dir+4}(x,y)$, meaning that is likely that $pix_i(x,y)$ is part of a segment with either of the neighbour pixel.
4. $max(x,y) > 20$ implies $C_{dir}(x,y)$ is an *undirected edge*. The high value on the color edge indicates that there is an edge, but it is not possible to determine the direction.
5. $rat^{dir}_{average}(x,y) > \frac{3}{4}$ implies *interior*. This check if the colour distance between $pix(x,y)$ and its neighbours are similar, indicating that the colour difference is likely caused by noise.
6. $OD_{dir}(x,y) > 20$ and $rat^{dir}_{norm}(x,y) < \frac{2}{3}$ implies $C_{dir}(x,y)$ is a *directed edge*. The neighbour pixels $pix_{dir}(x,y)$ and $pix_{dir}(x,y)$ are very different which indicate the presence of an edge.
7. $OD_{dir}(x,y) > 20$ and $rat^{dir}_{norm}(x,y) > \frac{2}{3}$ implies $C_{dir}(x,y)$ is an *undirected edge*. Same as previous condition, but in this case it is not possible to be certain about direction.

8. otherwise *interior*

Directed Edges Diffusion Edges will be detected when two pixels have *directed edge* in opposite directions. However, at this point, most of the pixels are labelled with an *undirected edge*, which does not allow to determine the exact location of an edge. It is therefore necessary to determine the direction of each of those edges into directed edge, using a diffusion process.

As long as the image still contains *undirected edges*, for each *undirected edge* at pixel (x, y) in the direction $idir$, we will refer as the *positive neighbour* of (x, y) the pixel (x_p, y_p) located in direction $idir(x, y)$ while the *negative neighbour* (x_n, y_n) is in the direction $-idir(x, y)$.

As part of the directed edges diffusion, we introduce a new possible edge class, that we call *centred edge*, which happen when $pix_{dir}(x, y)$ and $pix_{dir+4}(x, y)$ are *directed edges* of opposite directions and the direction is pointing toward $pix(x, y)$.

The following condition are checked in the given order:

1. if either the *positive neighbour* or *negative neighbour* is *undirected edge* or *interior* and the other one is either *directed edge* or *1-thick* then the pixel (x, y) is relabelled as *directed edge* and its direction is aligned with the *directed edge*
2. if either *positive neighbour* or *negative neighbour* are *directed edge* of the same direction then the pixel is relabelled as a *directed edge*
3. if either *positive neighbour* or *negative neighbour* are *directed edge* of the opposite direction and pointing toward $pix(x, y)$ then the pixel is relabelled as a *centred edge* otherwise $pix(x, y)$ is relabelled as *interior*
4. if either *positive neighbour* or *negative neighbour* are *1-thick* then the pixel is relabelled as a *1-thick*

3.2 Cutting the graph using a connectivity rule

	1t	DE	ODE	I	CE
1-thick (1t)			1		
Directed Edge (DE)	1	1	1	1	
Opposite Directed Edge (ODE)			1		
Interior (I)			1	1	
Centred Edge (CE)					

Table 2. This table show the connectivity rules $\mathcal{C}(pix_1, pix_2)$ (pix_1 is in the row and pix_2 the column, a 1 indicate that the two pixels are connected in the graph, while an empty cell indicates no connection.

Using this connectivity function it is possible to remove graph edges. The graph edge between pixels pix_1 and pix_2 is kept only and only if $\mathcal{C}(pix_1, pix_2) = 1$, otherwise it is discarded. The connectivity function is defined in the table 2.

It is worth to mention that the edge detection algorithm is not perfect, while it always correctly mark edge pixels in at least one direction, it may miss some direction. It also fails to work with a very smooth gradient. This is why it is still necessary to apply the efficient graph segmentation algorithm.

3.3 Efficient graph segmentation using a colour perception metric

In [9], Felzenszwalb suggested to group the nodes in segments, such that for all segment S

$$\sum_{pix(x,y) \in S pix_i(x,y) \in S} \Delta E^*_{ab}(pix(x,y), pix_i(x,y)) < \mathcal{T} \tag{11}$$

Instead, we suggest to ensure that:

$$\forall pix(x,y) \in S \Delta E^*_{ab}(pix(x,y), colour(S)) < \mathcal{T} \tag{12}$$

Where $colour(S)$ is the average colour over all the pixels in the segment:

$$colour(S) = \frac{1}{|S|} \cdot \sum_{pix(x,y) \in S} pix(x,y) \tag{13}$$

In order to guarantee that similar pixels are grouped first, the list of graph edges is sorted from smallest to largest and then for each graph edge, between pixel $pix(x,y)$ and $pix(x',y')$, if $pix(x',y') \in S$ then the condition 13 is checked between $pix(x,y)$ and the average colour of S and if that condition holds, then the pixel $pix(x,y)$ is added to the segment S. The process is repeated until all graph edges have been considered.

3.4 Fusing segment using edge information

After the initial segmentation, it is possible to improve the segmentation by merging segments that follow certain properties [13]. A graph of segments is generated where nodes in the graph are segments and two nodes of the graph are connected if the nodes have adjacent pixels. Also, for each segment S, $|S_{int}|$ is the number of interior pixels (ie pixels that are only connected to pixels that belongs to segment S), while $|S_{ext}|$ is the number of exterior pixels (ie pixels which are connected to at least one pixel that does not belong to S).

For each edge between segment S_i and S_j of the graph, the following values are computed:

1. $n_{ce}(i,j)$ number of connections with *centred edges*
2. $n_{cp}(i,j)$ number of connected pixels
3. $n_{ncp}(i,j)$ number of non connected pixels

Pixels are considered connected following the connectivity rule of section 3.2.

Segments are sorted in size and we first try to merge smaller segments.

Given a segment S_i and $\mathcal{N}_i$ is the set of neighbour of S_i, if there is a connected segment $S_j \in \mathcal{N}_i$ that respect the following conditions:

1. $n_{ce}(i,j) > 2.0 * n_{ncp}(i,j)$
2. $\forall S_k \in \mathcal{N}_i / \Delta E^*_{ab}(avg_{colour}(S_i), avg_{colour}(S_j)) < \Delta E^*_{ab}(avg_{colour}(S_i), avg_{colour}(S_k))$
3. At least S_i or S_j have less interior pixels than exterior pixels

Where $avg_{colour}(S)$ is the average colour of the segment S. If such a segment S_j exists, then S_i and S_j are merged.

Once all the segments have been tested for the previously mentioned conditions, the following conditions are checked:

1. $n_{cp}(i,j) > 2.0 * n_{ncp}(i,j)$
2. $\forall S_k \in \mathcal{N}_i / \Delta E^*_{ab}((S_i, S_j) < \Delta E^*_{ab}((S_i, S_k)$
3. At least S_i or S_j have less interior pixels than exterior pixels

And once again, if such a segment S_j exists, then S_i and S_j are merged.

3.5 Fuzzy merging

In [11], Karatzas present a fuzzy merging algorithm that is applied after a very simplistic and conservative segmentation algorithm. In reality, its fuzzy merging algorithm can also be applied as a last step for our algorithm. The main idea behind his merging algorithm is to merge segment that have similar colour and that are strongly connected to each other.

4 Results and Evaluation

We are now presenting qualitative as well as quantitative results of our algorithm (called CPGS) compared to the Human Color Perception (HCP [11]). We also show result with the fuzzy filtering enabled and disabled on the CPGS algorithm.

4.1 Qualitative results

We have included several results of the use of the segmentation algorithm. First when the algorithm is applied to a full image (see figure 3) and also when applied only around words (see figures 4).

The result of HCP and CPGS without fuzzy do looks roughly equivalent to the eye and neither algorithm seems to perform better than the other, sometimes HCP gives better results, like on the “2” of figure 3, or CPGS without fuzzy on the “R” and “c” of figure 4.

More interestingly is that the results show that for some of the images, the fuzzy merging step is too eager when applied to the result of our algorithm (CPGS), as can be seen on figures 3, where letters are completely disappearing from the segmentation result. At the same time, it can improve the quality of the results and reduce the amount of noise, as can be seen on the left images of figure 3 or by comparing the result of CPGS with and without fuzzy in figure 4, looking around the letters “Y”, “a”, “r” and “m” show clearly that some of the small segments are correctly integrated.

4.2 Characters recognition

The end goal would be to be able to recognize characters, so we were interested in testing how our algorithm performed when used with an Optical Characters Recognition algorithm. For those tests, we used the “Word recognition” dataset from the ICDAR

2003 Competition [14] and the NEOCR dataset [15], using HCP or CPGS, the image was segmented and then for each segment the Tesseract OCR engine [16] was used to recognize the character.

The results are shown in table 3. They show that CPGS without fuzzy gives slightly and faster result than HCP.

The reason why the NEOCR dataset shows much slower results is because images have a size of 4000x3000 while ICDAR have images in the size of 1000x1000. Also, the NEOCR dataset is much more challenging, which explain the lower recall rate.

Conclusion

We have presented an algorithm for segmenting text characters in natural images that is both fast and gives good results. We have presented a comparison with a state of the art algorithm [11]. We have also tested whether we could further improve the results of our algorithm, by using the fuzzy filtering of [11] and unfortunately, while on some images it does seem to bring improvement in the segmentation, it also ruin it for many images.

Further work would involve integrating the algorithm in a full process of automated character recognition in natural images, which would involve improvement to the detection of character location as well as character recognition. There was two main motivation behind our work, the first one is that we considered that the principle behind the algorithm in [11] were sound and that the use of the colour distance was a good solution for segmenting characters, however the algorithm was too slow when applied to natural images and we wanted a faster algorithm, which we have managed to achieve. However we also wanted to generate segment of better quality, less sensitive to noise and especially with fewer holes inside and we have not achieve that. It is possible that further improvement can be applied to the segmentation algorithm to also solve that problem. Otherwise, future work could involve improving the character recognition algorithms, so that this sensitivity to noise is less of a problem.

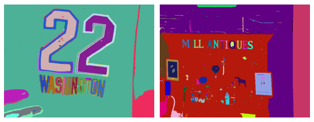
	ICDAR 2003		NEOCR	
	recall	time (ms)	recall	time (ms)
CPGS (No Fuzzy)	0.72	84	0.19	12151
CPGS (with fuzzy)	0.54	92	0.14	544937
HCP	0.70	145	0.17	307407

Table 3. Characters recognition on ICDAR 2003 Words (4874 images) and NEOCR dataset (660 images)

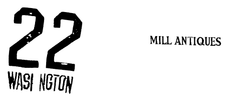
References

1. Ho, P.G.P., ed.: Image Segmentation. InTech (2011)

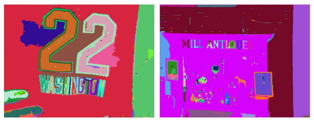
(a) Original image



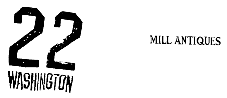
(b) HCP Segmentation (2171 ms, 390ms)



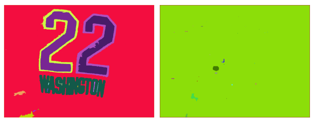
(c) HCP character masks



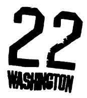
(d) CPGS (No Fuzzy) Segmentation (1877 ms, 258ms)



(e) CPGS (No Fuzzy) character masks

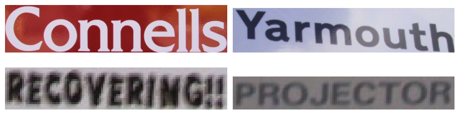


(f) CPGS (with fuzzy) Segmentation (5804 ms, 371ms)

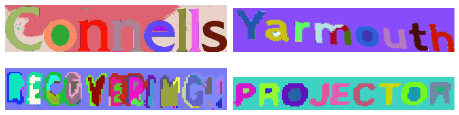


(g) CPGS (with fuzzy) character masks

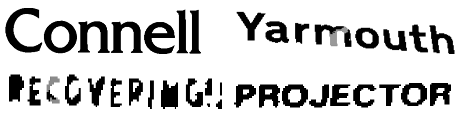
Fig. 3. Full images segmentation and character masks



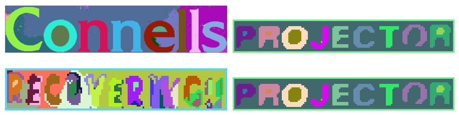
(b) Original image



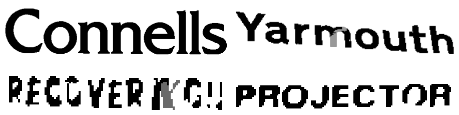
(d) HCP Segmentation (379 ms, 35 ms, 11ms, 34ms)



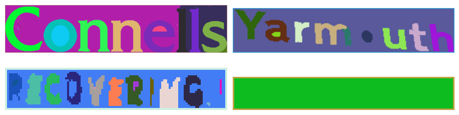
(f) HCP character masks



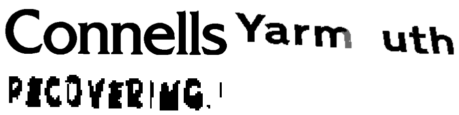
(h) CPGS (No Fuzzy) Segmentation (243 ms, 11ms, 4ms, 31ms)



(j) CPGS (No Fuzzy) character masks



(l) CPGS (with fuzzy) Segmentation (597 ms, 12ms, 4ms, 33ms)



(n) CPGS (with fuzzy) character masks

Fig. 4. Connells, yarmouth and recovering: segmentations and character masks

2. Perkins, W.A.: Area segmentation of images using edge points. IEEE Transactions on Pattern Analysis and Machine Intelligence **2** (1980) 8–15
3. Haralick, R.M., Shapiro, L.G.: Survey: Image segmentation techniques. Computer Vision, Graphics and Image Processing **29** (1985) 100–132
4. Pavlidis, T., Liow, Y.T.: Integrating region growing and edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 225 –233
5. Freixenet, J., Muñoz, X., Raba, D., Martí, J., Cufí, X.: Yet another survey on image segmentation: Region and boundary information integration. In: European Conference on Computer Vision. (2002) 408–422
6. Bonnin, P., Blanc-Talon, J., Hayot, J., Zavidovique, B.: A new edge point/ region cooperative segmentation deduced from a 3d scene reconstruction application. In: SPIE: Applications of Digital Image Processing. (1990) 579–591
7. Lézoray, O., Grady, L., eds.: Image Processing and Analysis with Graphs: Theory and Practice. CRC Press (2012)
8. Delong, A., Osokin, A., Isack, H.N., Boykov, Y.: Fast approximate energy minimization with label costs. International Journal of Computer Vision **96** (2012) 1–27
9. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision **59** (2004)
10. : Cie 1976 l*a*b* colour space standard. International Commission on Illumination (1976)
11. Karatzas, D., Antonacopoulos, A.: Colour text segmentation in web images based on human perception. Image and Vision Computing **25** (2007) 564–577
12. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **8** (1986) 679–698
13. Gagalowicz, A., Monga, O.: A new approach for image segmentation,. In: International Conference on Pattern Recognition. (1986) 227–248
14. et al, S.L.: Icdar 2003 robust reading competitions: Entries, results and future directions. International Journal on Document Analysis and Recognition **7** (2005) 105–122
15. Nagy, R., Dicker, A., Meyer-Wegener, K.: Neocr: A configurable dataset for natural image text recognition. In: Camera-Based Document Analysis and Recognition Workshop at the International Conference on Document Analysis and Recognition. (2011) 53–58
16. Smith, R.: An overview of the tesseract ocr engine. In: International Conference on Document Analysis and Recognition. (2007) 629–633

Acknowledgments

This work is partially supported by the Swedish Research Council (VR) Linnaeus Center CADICS, the ELLIIT network organization for Information and Com- munication Technology, and the Swedish Foundation for Strategic Research (CUAS Project).