Strokes detection for skeletonisation of characters shapes

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. Skeletonisation is a key process in character recognition in natural images. Under the assumption that a character is made of a stroke of uniform colour, with small variation in thickness, the process of recognising characters can be decomposed in the three steps. First the image is segmented, then each segment is transformed into a set of connected strokes (skeletonisation), which are then abstracted in a descriptor that can be used to recognise the character. The main issue with skeletonisation is the sensitivity with noise, and especially, the presence of holes in the masks. In this article, a new method for the extraction of strokes is presented, which address the problem of holes in the mask and does not use any parameters.

1 Introduction

1.1 Detection and recognition of text in natural images

In natural images, texts can be located anywhere, in any orientation, with any kind of distortion and any colours. In order for a computer to automatically read text from natural images, two sets of algorithms are required: the first step is to locate where in the images the text is located [1, 2], while the second step is about recognising the characters. Our focus in this paper is on the second aspect of the problem. The best approach in natural images would be to use a skeleton-based character recognition algorithm [3], the main reason is that from the skeleton of the character, it is possible to obtain a representation that is invariant to rotation and distortion.

For recognising characters with a skeleton-based approach, the first step is to segment the image (using for instance [2]), then each segment mask is transformed into a skeleton [4] then this skeleton can be transformed into features and then matched with existing knowledge [3] of how characters skeleton looks.

1.2 Skeleton

The classical approach for creating a skeleton of a mask is to use a thinning algorithm [4]. A thinning algorithm works by removing border pixels from the

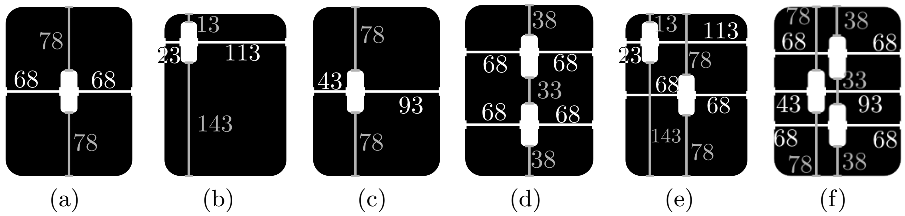


Fig. 1. This figure shows the different type of hole that can happen in a mask. Figure 1(a) shows the letter O, while 1(b) shows the same hole in the corner, making it likely to be due to noise. Figure 1(c) is more tricky, since for some fonts the hole in O maybe not centred, 1(d) shows a 8, 1(e) shows the same number of holes, but the one in the corner is likely to be caused by noise, 1(f) is also tricky since it can either be a 8 or O

mask, until reaching a one-pixel thick set of lines. The most classical thinning algorithm is called Zhang-Suen, the algorithm defines a set of rules which decide whether a pixel should be removed and when no more pixels match the rule the algorithm stop and the skeleton has been found.

But thinning algorithms suffer from several problems, such as generation of spurious line segments or lack of accuracy around strokes crossing. In [5], the masks is decomposed in several blocks and blocks are connected together depending on whether they correspond to a crossing. The detection of where the crossing happen allows for improvement in the resulting skeleton.

More recently, work has been done on algorithm that can create the skeleton without the need for the segmentation. They work by iteratively aligning the image gradient [6, 7]. But those algorithms are very unpractical to use, they tend to be slow and require tuning of many parameters.

However, one of the main issue with skeletonisation algorithm is that they are highly sensitive to noise and especially, one of the biggest problem occurs when there are holes in the mask (see figure 1 for an explanation of the problem and figure 9 for examples). To overcome that problem, it has been suggested in [8] to use morphological operators, however those might not be enough if the hole is to large, also not all holes are noise, many letters have a hole inside the mask (for instance, the letter *o*). It is therefore required to be able to detect whether a hole is noise or is part of the letter.

There is no universal definition of how a good skeleton of a shape should look, since our goal is to use the algorithm for characters recognition, we will qualify of *ideal* a skeleton which provide an outline of the character as it would appear in a font or written by an human.

1.3 Parallel Contour matching for Strokes Detection

In this article, we present an algorithm that is capable of recovering a skeleton that could be considered as *ideal*, even in the presence of noise. Also like in [5], the resulting skeleton do not suffer from spurious line segments and it correctly represent crossing. Lastly, our algorithm output more than just the skeleton, since it also recover the thickness of the stroke in each point, this is why we will refer to the skeleton detected by our algorithm as *stroke*.

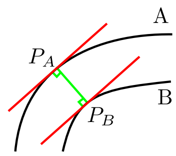


Fig. 2. A stroke is made of two parallel curves A and B, where the perpendicular of the tangent in $P_A \in A$ is perpendicular to the tangent in $P_B \in B$

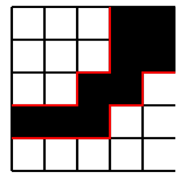


Fig. 3. Contours (in red) are set on the edges between pixels.

We follow the same assumption than in [1], that a stroke used to draw a character has a relatively constant thickness, meaning that the two opposite contours of a stroke are parallels and that the perpendicular of the tangent of a point of one of the contour is also the perpendicular of the tangent on the intersection point of the other contour. Obviously, this assumption is theoretical and in practice the rule is relaxed. But this assumption is the driving force in the design of our algorithm.

2 Strokes Detection

Contours are represented with a four-connected paths and instead of going through the center of a pixel, contours are going through the corners of the pixels of the mask, so that the contours follow the edges between two pixels (see figure 3. This allows to support stroke paths with a thickness of one pixel. Then the algorithm follow those steps:

1. Compute the normal direction, using the euclidean path [9].
2. Find matches between contour points.
3. Group matches into stroke hypothesis.
4. Use a strokes *compatibility graph* to find the most likely solution to the stroke detection problem.
5. Connect strokes.
6. Filter to remove spurious lines.

2.1 Tangent

The normal direction is computed using discrete lines (see figure 4) on euclidean path [9], they offer good accuracy and good performance. Discrete lines are a set of points that satisfy the following inequalities:

$$\mu \leq \alpha x - \beta y < \mu + \alpha + \beta \tag{1}$$

α, β amd μ are called the characteristics of the discrete line and are computed incrementaly around each point. We will note $dll(P)$ the length of the discrete line for the point P, in other words, the distance between the two last points around P, where the equation 1 still hold.

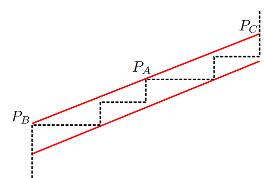


Fig. 4. Discrete line around P_A, going from P_B to P_C ($dll(P_A) = distance(P_B, P_C)$)

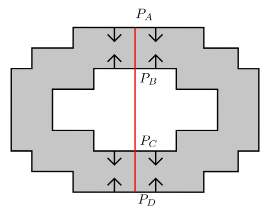


Fig. 5. The point P_A will be matched with P_B and P_C since their normal are in the opposite direction, but P_A is not matched to P_C. P_B is only matched to P_A, P_C to P_D while P_D is matched to P_C and P_A.

2.2 Contour points matching

Once the tangent is computed for each point P, the normal $\mathbf{n(P)}$ on that point can be easily deduced, the normal is oriented such as the vector director point toward the interior of the mask.

For each point P of every contours, the ray $\mathbf{r(P)}$ starting in P and following the normal $\mathbf{n(P)}$ is used to find matching point. A point P' is considered a match of P, if the ray $\mathbf{r(P)}$ intersect a contour on the point P' and $\mathbf{n(P)}.\mathbf{n(P')} < 0$.

Figure 5 shows an example of this point matching process. This part of the process is following the hypothesis we have defined in the introduction, that a stroke is encompassed between two parallels curves. In an ideal situation, $\mathbf{n(P)}.\mathbf{n(P')} = -1$ would be true, but that is rarely the case in reality, meaning that if P' is a match of P, it does not necessary means that P is a match for P'.

Later it will be useful to estimate the quality of the match and it seems natural to use the inner product of the normal, since it would give the highest score when the normal are parallels, hence when P is also a match for P':

$$s_{match}(P, P') = -\mathbf{n(P)}.\mathbf{n(P')} \tag{2}$$

We will note $\mathcal{M}(P)$ the set of matches of the point P

2.3 Stroke hypothesis

In this step, the matches are used to create stroke hypothesis. Given two contours C_0 and C_1, the idea is to simply group together matches that follow the rule: two consecutive points $P_A \in C_0$ and $P_B \in C_0$ are part of the same stroke hypothesis if $\exists P_C \in C_1 \cap \mathcal{M}(P_A)$ and $\exists P_D \in C_1 \cap \mathcal{M}(P_B)$ such as

$$distance(P_C, P_D) < \frac{distance(P_A, P_C)}{2} + dll(P_C) \quad (3)$$

$$distance(P_C, P_D) < \frac{distance(P_B, P_D)}{2} + dll(P_D) \quad (4)$$

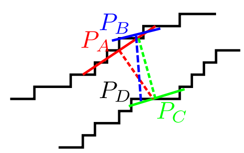


Fig. 6. P_A and P_B are two consecutive points of a contour, P_C is the match for P_A, while P_D is the match for P_B, P_A and P_B belongs to the same stroke hypothesis if P_C to be in the part of the curve covered by P_D's extended line

1. Select a point $P_A \in C_A$ and a match $P_C \in \mathcal{M}(P_A)$, with P_C a point of contour C_C
2. Check if there is a point $P_B \in C_A$ and $P_D \in C_C$ that respect the condition defined in 4
3. Iteratively check if the neighbor points of P_A and P_C respect the condition defined in 4

This process is applied for all contour points and all their matches to form the full set of stroke hypothesises.

2.4 Compatibility graph

As explained in section 2.2, there can be several matches per contour point, which means that a contour point can belong to several different stroke hypothesis. In the final skeleton, a contour point can only be associated to one stroke. Meaning that two stroke hypothesises that have a contour point in common are not compatible, and only one of them can be selected for the final skeleton.

In the *compatibility graph* $\mathcal{G}$, the nodes are a group of hypothesises, and nodes that are compatibles are connected by an edge. Then the problem of finding the groups of stroke hypothesis that are compatible with each other s reduced to finding the list of cliques[1], to solve that problem we can use the Bron-Kerbosch algorithm [10]. However that algorithm is rather expensive, since its execution time is proportional to $(3.14)^{n/3}$ (where n is the number of nodes in the graph). The idea is to reduce the number of nodes, lets consider an interior contour, as we mentioned previously, it is either the result of noise, and therefor the stroke hypothesises connected to that contour should be discarded, or it is part of the character (like in *o*) and then all of them should be considered as part of the solution.

A first step is to generate group of stroke hypothesises. It is reasonable to assume that compatible stroke hypothesises connected to an interior contour can be merged in a single node, as either they are all part of the valid solution, either the contour is noise and all hypothesises should be discarded. Since not all stroke hypothesises on the inside contour are compatible with each other, several groups will be created (see example on figure 7). This is also a clique problem, but it is a smaller one. It helps reducing the number of vertices in the *compatibility graph*, for instance, in figure 7, instead of having five nodes in the *compatibility graph*, the number is reduced to two.

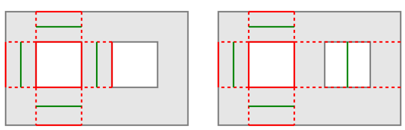


Fig. 7. This figure shows the two possible set of stroke hypothesis for the left interior contour of a *8* shape. The red continuous line shows the border of the stroke hypothesis on the contour, and the green line is the corresponding skeleton.

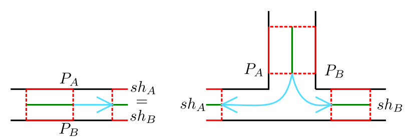


Fig. 8. This figure shows how stroke hypothesises are connected together. In red, it shows the different stroke hypothesises and in blue the connection between two hypothesises. The left figure show the case with only one connection and the right one shows what happen at a crossing.

The second step is to apply the Bron-Kerbosch algorithm [10] to the *compatibility graph.*

The last step is to select the best solution among the multiple solutions to the clique problem. For each solution, the following score is computed:

$$S = \frac{1}{N_{sh}}\left(\sum_{sh} \frac{\sum_{P \in sh} s_{match}(P, P')}{|sh|} \cdot c(sh) \cdot density(sh)\right) \cdot \frac{N}{NT} \tag{5}$$

Where N_{sh} is the number of stroke hypothesises used in the solution. sh is a stroke hypothesis that is part of the solution. $|sh|$ is the number of contour points in the stroke hypothesis. $density(sh)$ is the density of the stroke hypothesis, in

[1] The clique of an undirected graph G is a subgraph of G where all the vertices are connected

other word, the ratio between the number of the strokes that belong to the mask with the one that do not belong (in figure 7, on the right, the three small stroke hypothesises have a density of 1, while the big stroke hypothesis has a density of 0.66). $N = \sum_{sh} |sh|$ is the number of contour points that are used by this solution. NT is the to al number of contour points in the mask.

Finally $c(sh) = 1$ if a stroke hypothesis only belongs to the exterior contour, otherwise:

$$c(sh) = 1 - \frac{max\_thickness - min\_thickness}{max\_thickness} \tag{6}$$

Where $min\_thickness$ and $max\_thickness$ is the minimum and the maximum of the average thickness among the stroke hypothesis that belongs to the same indoor contour. For instance, in figure 7, for the left figure, $c(sh) = 1$ since all stroke hypothesises have the same thickness, but $c(sh) = 0.25$ for the right one. This is a measure of how likely the hole and a given set of stroke hypothesises belonging to that hole are to not be noise.

2.5 Connect strokes

Once the best solution has been selected, the final step is to connect all stroke hypothesises together.

Given a stroke hypothesis sh and two end points $P_A \in C_A$ and $P_B \in C_B$ on the same side of the stroke hypothesis. Point P_A is connected by the contour to a stroke sh_A and the point P_B to a stroke sh_B (see figure 8).

1. if $sh_A = sh_B$, the same stroke hypothesis is found on both contour, then the stroke hypothesis are simply merged together (like on the left part of figure 8)
2. if $sh_A \neq sh_B$, two different hypothesises are found, it means there is a crossing and multiple stroke hypothesises will have to be connected together (like on the right part of figure 8)

2.6 Filtering

To improve the results, a final step is applied, to filter out some of the remaining strokes. We use two criteria for removing a stroke hypothesis:

1. if a stroke hypothesis is fully contained interior an other one, then it is removed
2. if a stroke hypothesis sh_1 is connected to an other one sh_2 with a much bigger thickness, then sh_1 is removed, according to the following equations:

$$3 \cdot thickness(sh_1) < thickness(sh_2) \tag{7}$$
$$length(sh_1) < thickness(sh_2) \tag{8}$$

3 Results

The masks used in this section were created using the segmentation algorithm presented in [2] applied on images taken from the ICDAR 2003 competition data set [11]. It is worth to note that while about 80% of the masks produced by the segmentation do not present any noise problem, we have only selected noisy masks for this section, since noise free masks show similar results with our algorithm as with the state of the art.

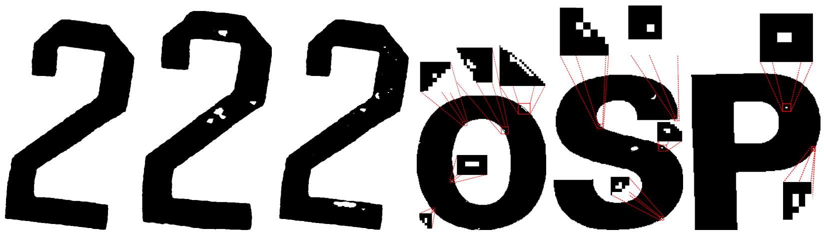
Figure 9 shows the algorithm on three similar masks for the number *2*, as we can see, when using our algorithm the resulting skeleton is similar, no matter whether there is noise or not. While the Zhang-Suen algorithm produce a broken skeleton for the noisy result. An other benefit of our approach is that it avoids the problem of the small line that can appears in corners, like we can see on the bottom right part of the Zhang-Suen skeleton for the left and right masks.

Figure 9 shows results for different characters on medium size masks. Once again, it shows that our algorithm is capable of recovering a quiet good skeleton despite the noise. The figure also show the effect of post processing the skeleton to remove small masks. However, one can notice a smell artefact on the skeleton for the *S* mask (centre figure of 9), on the top extremity of the letter, there is a small segments that goes back up. This artefact is caused by the use of rather simplistic rules for filtering and for finding incompatibilities, more complex rules would be able to remove the artefact, but would require higher computational power.

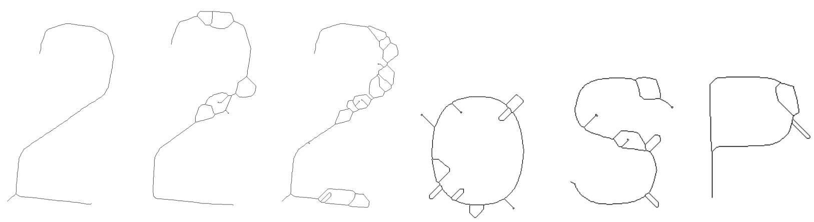
Figure 10 shows that the algorithm is not only capable of finding the skeleton for large masks but that it still works for small ones.

The results of figure 9 and 10 shows that post filtering of strokes brings significant improvements to the results of our algorithm. And it raises the question of whether it would be possible to apply similar filtering techniques to the Zhang-Suen algorithm, however, one of the reason that the filtering works nicely with our algorithm is that even before the filtering process, the ideal skeleton is still visible, while the output of Zhang-Suen shows that around the noisy part of the mask, the ideal skeleton has been damaged, this is especially visible for the right *2* mask of figure 9. An other aspect to take into consideration is that the output of our algorithm is a skeleton with an associated thickness, which makes it straight forward to define rules on whether part of the skeleton is likely to be noise, while the output of Zhang-Suen is a one pixel thick skeleton.

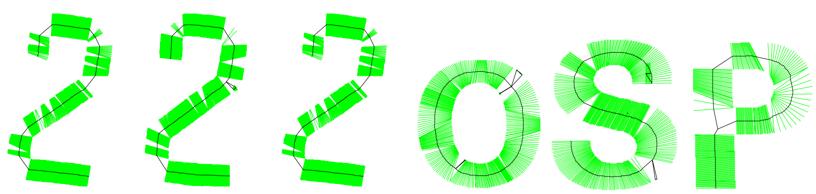
Table 1 show the execution time for both algorithms, it appears that on larger mask, the algorithm presented in this article is more than three times faster, while on smaller mask Zhang-Suen is about two times faster. This can be easily explain by the nature of the Zhang-Suen algorithm, it is a thinning algorithm that works by iteratively removing pixels from the mask, until only an one pixel line is left, a larger masks has two effects, the number of pixels to process is larger, but also, the number of iterations increase. On the other hand, our algorithm complexity is mostly dependent on the number of contour pixels, which increase at a slower rate, on a noiseless mask. However, on noisy masks, the noise has the effect of increasing the number of contour pixels, this can have



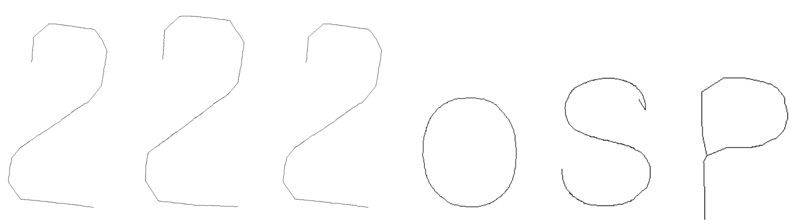
(a) Masks



(b) Zhang-Suen

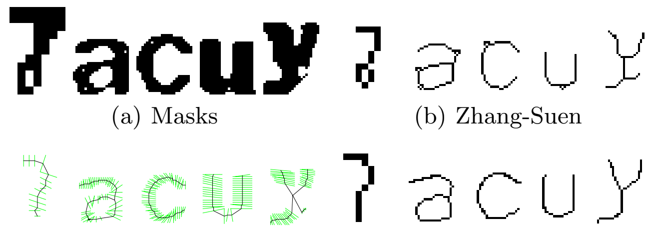


(c) Our algorithm unfiltered



(d) Our algorithm

Fig. 9. This figure shows the result from Zhang-Suen and our algorithm on masks representing the number 2, the two figures on the right are results from segmentation, while the figure on the left was created by manually filling the holes in the mask from the right mask (the size is approximately 400 pixels by 600 pixels). The green segment represents the thickness as detected by our algorithm. On some of the masks, the red squares indicates area that are zoomed in to show the problematic holes.



(a) Masks (b) Zhang-Suen

(c) Our algorithm (unfiltered) (d) Our algorithm

Fig. 10. Small size masks (approximately 20 pixels by 20 pixels). The green segment represents the thickness as detected by our algorithm.

	2 (left)	2 (middle)	2 (right)	O	S
Zhang-Suen	109.4 (3.1)	117.7 (5.2)	106.4 (3.6)	32.0 (1.41)	27.9 (4.6)
Our algorithm	31.8 (3.8)	37.1 (5.0)	85.6 (3.4)	8.6 (0.78)	8.9 (0.38)
Size	387x630	380x643	387x630	211x219	186x223
	P	a	c	u	y
Zhang-Suen	24.7 (3.6)	0.20 (0.04)	0.16 (0.05)	0.19 (0.05)	0.21 (0.03)
Our algorithm	4.4 (0.8)	0.49 (0.04)	0.29 (0.08)	0.27 (0.05)	0.31 (0.04)
Size	173x210	24x24	23x25	24x24	24x31

Table 1. Comparison of computational time between our algorithm and Zhang-Suen, time is expressed in millisecond, it is averaged between several run and the number in parenthesis is the standard deviation. The computer used for the experiment has an Intel I7 3.4GHz CPU.

dramatic effects on performance, especially since the noise increase the number of interior contours and therefor the complexity of the *compatibility graph*, and the algorithm used to find the clique [10] computational time is proportional to $(3.14)^{n/3}$, where n is the number of nodes in the graph. This explain why the right *2* mask from 9 requires twice as much time than the other two variants of the mask.

4 Conclusion

We have presented a parameter-free algorithm that allow to extract strokes information from masks, and that is robust to noise while being reasonably fast to compute. The main strength of our algorithms is to handle holes that can appear as the result of a segmentation of a character in an image.

There are many possible improvements to the algorithms, which could hopefully improve both performance and quality. It is our belief that improvements should be mainly aiming at reducing the work load of the Bron-Kerbosch algorithm, one idea would be to pre-filter the interior contours, it seems likely that holes that are not centred interior a mask are more likely to be caused by noise than to be part of a letter, similarly a stroke with a very low density is unlikely to be selected as part of the best solution and could be removed from the graph.

An other area that needs improvement is the contour matching and extension process, it is extremely sensitive to noise, meaning that it tends to break, which has the effect of not only increasing the workload of the Bron-Kerbosch algorithm but also to lower the quality and to increase the chances that a noisy solution would be selected at the expense of the *ideal* one.

Since the algorithm presented in this paper has many similarity with stroke-width transform algorithm [1], it should also be possible to use this stroke detection algorithm to detect where the text is located in natural images. It would be possible to segment the image, then to transform each resulting segment mask into its stroke representation, filter out characters that have a too high variation in thickness and try to recognise the remaining characters.

References

1. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: Conference on Vision and Pattern Recognition. (2010) 2963–2970
2. Karatzas, D., Antonacopoulos, A.: Colour text segmentation in web images based on human perception. Image and Vision Computing **25** (2007) 564–577
3. Cheriet, M., Kharma, N., Liu, C.L., Suen, C.Y.: Character Recognition Systems. Wiley Publishing Inc (2007)
4. Parker, J.: Algorithms for Image Processing and Computer Vision. Wiley Publishing Inc (2010)
5. Fan, K.C., Chen, D.F., Wen, M.G.: Skeletonization of binary images with nonuniform width via block decomposition and contour vector matching. Pattern Recognition **31** (1998) 823–838
6. LeBourgeois, F., Emptoz, H.: Skeletonization by gradient regularization and diffusion. International Conference on Document Analysis and Recognition **2** (2007) 1118–1122
7. Yu, Z., Bajaj, C.: A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion. In: Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1. (2004)
8. Ashok, M., SreeDevi, J., Bai, M.R.: An approach to noisy image skeletonization using morphological methods. International Journal of Scientific and Engineering Research **3** (2012)
9. Braquelaire, J.P., Vialard, A.: Euclidean paths: A new representation of boundary of discrete regions. Graphical Models and Image Processing **61** (1999) 16–43
10. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM **16** (1973) 575–577
11. Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R.: Icdar 2003 robust reading competitions. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2. ICDAR '03, Washington, DC, USA, IEEE Computer Society (2003) 682–

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).