Efficient model based single and double thresholding for real time recognition

Dror Aiger and Silvio Jamil Ferzoli Guimares

[1] Google Inc.
[2] Universidade Federal de Minas Gerais

Abstract. We present a very fast model based thresholding method that allows real time object recognition suitable for mobile devices. Let I be a gray level image of N pixels in d-dimensional space. Given a binary model m (as a set of connected pixels), we present efficient algorithms for thresholding I by single and double threshold to obtain connected components that are the most similar to m among all possible connected components according to a certain similarity measures. For warm-up we give a very simple single thresholding algorithm with runtime $O(N\alpha(N))$ (for fixed d), $\alpha(N)$ is the very slow growing Ackermann function, that finds a connected component which has the smallest distance to the model where the similarity is some metric on the Hu moment invariants space. Next we use this algorithm to obtain the optimal connected component by double thresholding in $O(KN\alpha(N))$ time where K is the number of gray levels. Last, let $M(\cdot)$ be any additive measure on a set of pixels such that for every two disjoint sets S_1 and S_2, $M(S_1 \cup S_2) = M(S_1) + M(S_2)$. Given some $\varepsilon > 0$ we show how to find all connected components C_i among all connected component (with some limitations that we characterize bellow) that can be obtained by any double threshold, that have $|M(C_i) - M(m)| < \varepsilon$ in practically $O(N \log^2 N + k)$ time independent of the number of gray levels, where k is the number of reported components. We use these algorithms for detecting binary models in images by threshold only, invariant to similarity and rigid transformations.

1 Introduction

Object segmentation aims to extract an object from the background in a given image. This can be viewed also as object detection task if we seek a specific object that is given as a prior. The addition of shape prior information has shown to significantly improve segmentation results and is quite popular [6][16][17]. Variational methods and partial differential equations (PDEs) have been used to analyze, understand and exploit properties of images. These methods commonly designed to solve local optimization problems thus true global optimal solution cannot be found in general. There is an increased interest in graph based segmentation algorithms [2][1] for global energy minimization, and subsequently the addition of prior shape information into their formulations. The graph methods of Felzenszwalb [7] and Schoenemann and Cremers [20] can segment objects under elastic deformations without needing any initialization and

guarantee globally optimal solutions. In [7], non serial dynamic programming is used to find the optimal matching between a deformable template represented by triangulated polygons and the image pixels.

In this paper we present efficient and fast algorithms to find/segment objects in images, which is invariant to transformations of a given geometric model, where a geometric shape model of the object is given. The presented method is by no mean a replacement for the more general yet complex methods described above. It aims to serve in realtime application like in mobile devices or surveillance cameras thanks to its simplicity and speed. The methods are based on simple thresholding that minimizes globally a dissimilarity function between the model and a segmented object. Although this method has limitations and it is not as general as the above methods, it is shown that for many applications it gives a very fast, globally optimal and easy to implement solution. We are based on principles related to the component tree which we briefly survey bellow. It has been used (under several variations) in numerous applications among which we can cite: image filtering and segmentation [9][8][4][12], video segmentation [18], image registration [13][14]. This tree is also fundamental for the efficient computation of the topological watershed introduced by Couprie and Bertrand [4][5].

The component tree concept was first introduced in statistics [10] for classification and clustering. For image processing, the use of this tree in order to represent the "meaningful" information contained in a numerical function can be found in particular, in a paper by Hanusse and Guillataud [9][8], the authors claim that this tree can play a central role in image segmentation. Several authors, such as Breen and Jones [3], Salembier et al. [18] have used some variations of this structure in order to implement efficiently some morphological operators (e.g. connected operators) [19] and granulometries. Various algorithms have been proposed in the literature for computing the component [3][18], the fastest was proposed by Najman and Couprie and has complexity of $O(N\alpha(N))$ where N is the number of pixels in the image [15]. It is based on Tarjańs union-find [21] data structure.

2 Component Trees

Let $\mathbb{A} \subset \mathbb{N}^2$, $\mathbb{A} = \{0, \ldots, \mathrm{w}-1\} \times \{0, \ldots, \mathrm{h}-1\}$, where w and h are two strictly positive integers. A point $x \in \mathbb{A}$ is defined by its two coordinates (u, v). We denote by $\mathcal{F}(\mathbb{A})$ the set composed of all functions from $\mathbb{A}$ to $\mathbb{N}$. Such a function I represents a *gray level image*.

Let $I \in \mathcal{F}(\mathbb{A})$, we define $I_k = \{x \in \mathbb{A}; I(x) \geq k\}$ with $k \in \mathbb{N}$; I_k is called a *cross-section* of I. A connected component of a section I_k is called a *(level k) component* of I. A component of I that has only strictly lower neighbors is called a *(regional) maximum* of I. We define $k_{\min} = \min\{I(x), x \in \mathbb{A}\}$ and $k_{\max} = \max\{I(x), x \in \mathbb{A}\}$, which represent respectively, the minimum and the maximum gray level in the image I.

Definition 1 (Component tree – T) *Let I be an image. Let $\mathcal{C}_k(I)$ be the set of all connected components of level k of I. We define the component tree $T(I)$ as a directed tree such that: (i) the vertices of the component tree are the elements of $T(I)$; (ii) there is an arc from a component $c \in \mathcal{C}_j(I)$ to a component $c' \in \mathcal{C}_k(I)$ if $j = k + 1$ and $c \subseteq c'$.*

3 Model based thresholding and a simple algorithm

3.1 Simple single threshold

Similarly to the process of building the component tree, we suggest a very simple algorithm for finding optimal connected component (and the corresponding threshold). The same idea for filtering connected components based on the component tree was previously suggested by several authors [12, 19, 22]. The optimality is in the sense of a metric on the Hu invariants space [11] (any other shape similarity can be used). We are given a model M that can be described by a set of Hu moment invariants. We seek a connected component over all connected components that can be obtained by a single threshold, which is the most similar to M in this sense. The algorithm is exactly as the one for building the component tree but the tree itself is not created. A description of the algorithm can be viewed in Algorithm 1. The complexity is exactly as building the component tree which is $O(N\alpha(N))$. Being based on moment invariants, the entire algorithm is also invariant to similarity transformation (scale, rotation, translation).

Algorithm 1 Finding a model using threshold

Require: input image $I(u, v)$
Require: model image $M(u, v)$
compute $Hu(M)$, the Hu moments of M
compute all level sets $L(i)$, $i = 0..K$.
create an empty Union-Find structure, U augmented with general moments descriptors
for all i **do**
  for all (x, y) in $L(i)$ **do**
    find and union sets that are connected to (x, y) if needed
    updated the moment descriptors corresponding to the sets that were union, let it be M'
    compute $Hu(M')$
    compute the distance $d(Hu(M), Hu(M'))$
    update the best minimum and corresponding set if needed
  end for
end for
output best i and corresponding set

An example of a result obtained by this algorithm can be viewed in Figure 1. We note that the algorithm is parameter-less. If we allow scale however, we must bound it to some limit otherwise all small components are similar.

Fig. 1. Optimal single threshold based on binary model and Hu invariants. Left - input image, middle - the model, right - the output. The red component is the optimal component while the output image on the right is the binarization by the optimal threshold (containing several other components). Note that for every image I we consider both I and $\bar{I}$.

3.2 First algorithm for double thresholding

Threshold and the connected components created by the component tree algorithm have been used extensively for simple segmentation due to their simplicity and speed. However, in certain situations, an object cannot be distinguished from the background (or other objects) based on a single threshold. For example in Figure 2 the rectangle cannot be segmented from the triangles and the circle, even though it has a unique intensity. The same case in real image is shown in Figure 2 with our result. The use of double thresholding adds more power to the process of segmenting objects. In Figures 3 we show another real image example where an object can be segmented (by threshold) only using double threshold, in complex image with a lot of other objects, clutter and background variations. In terms of functions, single threshold can "segment" local maximal (minimal) regions while double threshold extends to "inflection" regions.

Fig. 2. Left: Double threshold is more powerful - the rectangle cannot be binarized with one threshold. Right: Double threshold is needed.

Using double threshold is much harder from the computational point of view. A naive approach would take every pair $t_1, t_2 \in 0..K-1$, use them to binarize the image and search for optimal resemble to the model in the set of all connected components. It is more efficient to compute the optimal solution by repeating the algorithm from Section 3.1 K times. We sweep t_1 from 1 to $K-1$ and for each iteration we apply the algorithm above where we ignore all pixels bellow t_1.

Fig. 3. The right car can be binarized with one threshold ($t = 185$) while the left car needs a double threshold ($t_1 = 172, t_2 = 224$) to be well distinguished from the background

The set of components obtained (again without actually building the tree) are the components that can be obtained by all pairs (t_1, t_2), $t_1 \leq t_2 < K$. Thus we apply the single threshold algorithm K times spending $O(N\alpha(N))$ time in each step. Maintaining the best component so far results in the optimal component that can be obtained by any double threshold. For Hu invariants, it seems that this is the best possible since Hu invariants are not additive. The runtime of this algorithm depends on the number of gray levels and it can be considered nearly linear only with fixed K. Moreover, the number of times we apply the single threshold is always K. It thus takes $O(KN\alpha(N))$ for any input and K is typically 256 which means a quite slow algorithm. The advantage is that we can maintain invariance under similarity transformation.

We show that having any additive measure (to be defined bellow) on sets of pixels, we can get efficient algorithm which has runtime independent of K and nearly linear in N. Our algorithm is limited in some sense (described bellow), thus cannot detect *any* possible component but we can identify and characterize exactly for which components it works and as we will show, it gives good results in practical images. The main idea is to base on the inclusion of the component tree and to apply binary search over levels on the component tree without computing the tree for every level.

4 Efficient double thresholding using additive measure

In this section we propose efficient algorithm using *any* additive measure on sets of pixels based on binary search over differences between levels in the component tree. A measure $M(\cdot)$ is additive if for every disjoint sets, S_1, S_2, $M(S_1 \cup S_2) = M(S_1) + M(S_2)$. A specific example is *the area*. While additive measure cannot be used for invariance under similarity transformation (since the size can change) it can be used as a first filter for rigid invariant model detection. General moments are additive but central moments are not thus we cannot use (maintain) them to measure similarity between components. However we can use for example the area (first moment) to first filter out all connected components of area close to the area of the model (since rigid transformations preserve the area). Then we can use e.g. moments to find the one that is most similar to our model. In this section we analyze the conditions that allow us to use this

efficient algorithm and characterize the limitations of this algorithm. We show that even though it has some limitations, its efficiency and effectiveness in real images and its simplicity makes it a good candidate for model based segmentation. The algorithm is output sensitive in the sense that its complexity depends on the number of extracted components. While this number can be $O(KN)$ in the worst case, it is much smaller in practical cases which makes the algorithm much faster than the algorithm described in Section 3.2

A double threshold (t_1, t_2) partitions an image I into three kinds of regions: regions that have all their pixels in the range $[t_1, t_2]$ denoted by $C_{t_1 \leq g \leq t_2}$, regions that have all their pixels bellow t_1 denoted by $C_{g<t_1}$ and regions that have all their pixels above t_2 denoted by $C_{g>t_2}$ (g represents gray value). We are interested in the first kind. In particular, for a given model m with (additive) measure $M(m)$, we wish to extract all connected components of type $C_{t_1 \leq g \leq t_2}$ possibly with a pair (t_1, t_2) that realizes each one) that have a (additive) measure close to $M(m)$. Let $C = \{C_1, C_2, .., C_k\}$ be the set of all connected components of the first type for some (t_1, t_2). We can identify four different cases of connectivities among the components of every subset of them. Figure 4 shows all four (for simplicity we show just two components).

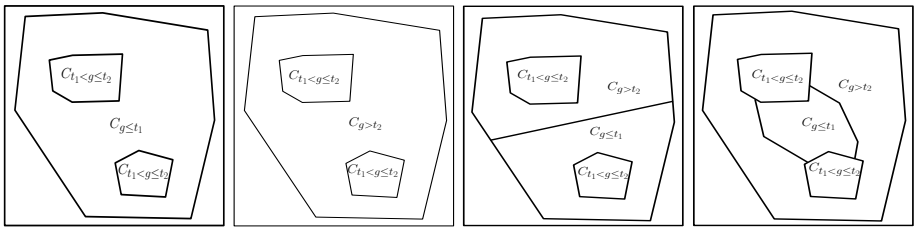

Fig. 4. All kind of connectivities between two connected components of a given double threshold (t_1, t_2) (the figures are not the most general ones for simplicity. There can be of coarse more than two components).

For a given (t_1, t_2), we call any $C_i \in C$, 1-connected if for every other $C_j \in C$, any path from C_i to C_j is not fully contained in a $C_{g>t_2}$ component. C_i is called 2-connected if for every other $C_j \in C$, any path from C_i to C_j is not fully contained in a $C_{g<t_1}$ component. If there are component in C that are connected to C_i by $C_{g<t_1}$ **and** there are others (or the same) that are connected to it by $C_{g>t_2}$ components, we call C_i *doubly-connected*. For example, if C_1 is connected to C_2 by $C_{g<t_1}$ component and is connected to C_3 by $C_{g>t_2}$ component, then C_1 is *doubly-connected*. In Figure 4(d) the two $C_{t_1 \leq g \leq t_2}$ components are both *doubly-connected*.

For any given (t_1, t_2), our algorithm identifies subsets (not all of them) of C that have together (their union) a measure that is close to a given measure. In particular, it identifies every **single** component provided that it is not *doubly–connected*. For *doubly-connected* components, our algorithm identifies only their

union with some other components. As we show in our experimental results, in most practical cases, most of the components are not *doubly-connected* thus detected as single components by our algorithm.

To describe our efficient double thresholding algorithm we will need two lemmas. The first lemma states that computing the subtracted connected component between a node in a component tree and all its sons of a specific level can be done efficiently. The second lemma insures that all connected components that are not *doubly-connected* and can be obtained by double thresholding will be actually found by our algorithm.

Lemma 1. *Given a component tree T with n nodes, a node N and a level L (given by an order list of all nodes in it) we denote by $T(N)$ the subtree of T that is rooted from N. The leftmost node l in $T(N)$ in the level L (if exists) can be found in $O(\log(C))$ time where C is the number of nodes of T in level L.*

Proof: The set of nodes in $T(N)$ is an interval in the ordered set of nodes in T when these are taken as the index of a (e.g.) left turn post order traversal of T. For a given node k in L we can find in constant time whether it is in $T(N)$. All the nodes in L that are to the left of l have smaller index in $T(N)$ and all nodes to the right of the rightmost have larger index. We can thus apply binary search on L to find l.

Lemma 2. *Let I be an image and $\bar{I}$ its complement. Let $T(I)$ be the component tree of I and $T(\bar{I})$ the component tree of $\bar{I}$. Any connected component that is not doubly-connected and can be obtained by double threshold on I, must be one of these:*

1. *A node in $T(I)$*
2. *A node in $T(\bar{I})$*
3. *The difference between a node either in $T(I)$ or in $T(\bar{I})$ and the union of all its sons in the tree in some level.*

Proof: A general configuration of a component that is not *doubly-connected* (corresponding to a given double threshold) can be described by Figure 5 for double threshold (t_1, t_2). If all neighbors components C_i have values above t_2 or all have values bellow t_1 than C is a local (regional) maximum of either I or $\bar{I}$ and thus is a node in one of the corresponding component trees. Otherwise without limiting the generality, assume that all the pixels in $C_{a_1}, C_{a_2}, .., C_{a_k}$ among the set of all C_i have values bellow t_1. Then, the union of $C, C_{a_1}, C_{a_2}, .., C_{a_k}$ is a node n in $T(I)$ and we have $C = n \setminus (C_{a_1} \cup C_{a_2} \cup, .., \cup C_{a_k})$ as required. The same is true for pixels above t_2 with $T(\bar{I})$. Note that some other component of type $C_{t_1 \leq g \leq t_2}$ can be connected to one of the C_{a_i} but since C is not *doubly-connected* it can be connected only to one kind of the neighbors of C thus allowing the subtraction above.

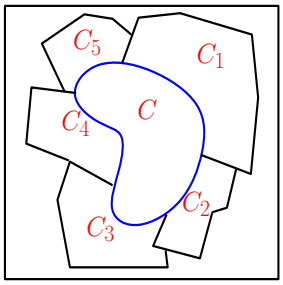

Fig. 5. Double threshold: the connected component C is obtained by thresholding I with a double threshold (t_1, t_2) and has all C_i as its neighbors. The pixels in every C_i have either all values bellow t_1 or all values above t_2. All pixels in C have values above or equal t_1 and equal or bellow t_2.

The algorithm Based on the above lemmas we can now describe our main algorithm. We are given an image I with N pixels and K gray levels and a model C given as a set of connected pixels (a connected component). We assume an additive measure $M(\cdot)$. We also have some accepted value on M, $\varepsilon > 0$. We wish to obtain from I all connected components C_i that can be obtained by double threshold on I and have $|M(C_i) - M(C)| < \varepsilon$. We limit ourself to those that are not *doubly-connected* and bellow we assume only these components.

We first observe that for a double threshold (t_1, t_2), $t_1, t_2 \in K$, the set of connected components obtained by thresholding I by (t_1, t_2) are all connected components that are the subtraction of the union of all nodes in level t_1 from the union of all nodes in level t_2. This is thanks to the inclusion we have in the component tree. It can be seen that the entire set can be obtained by subtracting from every node in level t_2 the union of all its sons in level t_1. We show bellow, that given any additive measure we can do it efficiently by binary search.

Let n be a node in a component tree and let U be the union of all its sons in some level. We use the notation n also to describe the connected component in node n. $n \setminus U$ can be a set of multiple connected components of coarse. Lemma 2 tells us that in the case that the subtraction has more than one connected component, all of them are nodes in one of the component trees, $T(I)$ or $C(\bar{I})$, or can be obtained as a single component by subtracting levels in $T(\bar{I})$. It follows that if we search the two trees and their nodes before, we can assume that we only have to consider the cases where the subtraction contains only one connected component.

We first construct the two component trees, $T(I)$ and $T(\bar{I})$ by the algorithm in [15]. The two trees can be computed in time $O(N\alpha(N))$ and their size (the number of nodes) is $O(N)$. A component tree is *compressed* if for the tree described in Section 2 in each path from the root to the leafs we replace every consecutive set of similar nodes into one. We note that the component tree in its uncompressed version as described in Section 2 has size $O(N)$ only for fixed K and it can be of size $\Omega(KN)$ in the worst case. Our algorithm uses the uncom-

press version so in the worst case it can take $\Omega(KN)$ time but in contrast to the algorithm in Section 3.2, this is rarely the case. In fact, only images that contain $\Omega(N)$ number of very high gradient has this complexity. Moreover, we believe that an algorithm that has worst case runtime $O(N \log^2 N + k)$ can be found (k is the number of reported components) but we leave this for future research. As we mentioned before this is output/input sensitive algorithm and in the worst case we can have $k = \Omega(KN)$.

We first search these trees by computing all $O(N)$ measures of the nodes in linear time, and we report in $O(N+k_1)$ all k_1 connected components within these nodes that agree with the above condition. Lemma 1 insures that all other needed connected components (not *doubly-connected*) must be in the set of (singles) connected components that are the subtraction of a node in a tree (one of the two) and the union of all its sons in a specific level.

Let n be a node in a component tree T and let $T(n)$ the sub tree of T rooted from n and let l be some level in $T(n)$. According to Lemma 1, in $O(\log(N))$ time we can compute the leftmost and rightmost nodes in $T(n)$ in level l. Let $a_{l_1}..a_{l_t}$ be all connected component in level l. Thanks to inclusion relations in T, all numbers $M(n) - M(a_{l_1} \cup a_{l_2} \cup ...a_{l_t})$ are sorted, thus using binary search we can find all needed k connected components by first finding one (by binary search) and then tracing its neighbor levels until we obtain all k connected components in the given range (s.t. $|M(C_i) - M(C)| < \varepsilon$). Practically we can stop at the first one if we need only a representative among the set of components that are included in each other. The time thus for a single node n is $O(\log^2(N) + k_n)$. In each tree we have $O(N)$ nodes thus we have overall in both trees $O(N \log^2(N) + k)$ time for $k = k_1 + .. + k_N$ reported connected components.

5 Applications

Our method has essentially the same functionality as any object segmentation with shape prior or model based object detection. It is therefore useful for the same applications. Indeed, using threshold only is somewhat limited compared to more general schemes, however it offers a much simpler, faster yet globally optimal method that can be practical in many application as we show in this section. The use of double threshold instead of the traditional single threshold provides a much wider spectrum of applications. Generally speaking, our method is applicable whenever there is an object in image (or 3D volume) which is relatively homogeneous and its shape model is given. Thanks to the double threshold the object can be connected to either brighter or darker regions. The fact that our second double threshold method does not depend on the number of colors or gray levels makes it also suitable for color images. The complexity only depends on the number of pixels (voxels in 3D) nearly linearly. The idea can be easily extended to 3D as it is solely based of the component tree computation.

We note that our method is not limited to any specific similarity measure (here we use Hu moment invariants). Once we have a similarity measure between two shapes we can plug it in our method. In particular non rigid measures can be

used for articulated objects. Bellow we present some results for medical imaging, gesture recognition, image retrieval and text analysis. Our method can be used in a p reprocess stage to extract the most important edge features with respect to a given model, thus it can be also viewed as a model based edge detection. Next, other recognition schemes or further similarity measures can be used to verify the exact detection.

Medical Imaging Medical imaging applications commonly need to segment or detect organs in 2D or 3D data where a general shape of the organ is given. For example, the knee can be easily processed as shown in Figure 6, left, to segment the bones being invariant to rotation and limited scale.

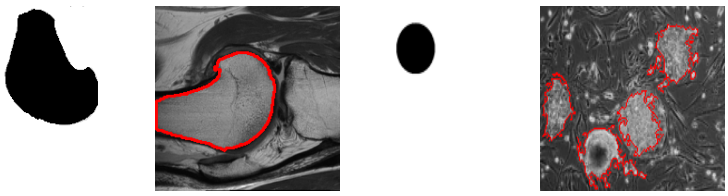

Fig. 6. Medical Imaging: Organ segmentation (left) and cells segmentation (right)

Another application in medical imaging is detecting objects (like cells) in images. In Figure 6, right, we show that our method is applicable for finding multiple instances of an object by reporting all connected components that are similar enough to the model (again, using any plugged in similarity measure).

Gesture Recognition This application aims in detecting specific gestures in images. Searching for the best model among a set of shapes and measuring its similarity to on object in the image can be applied by our method and used for recognition. An example is shown in Figure 7.

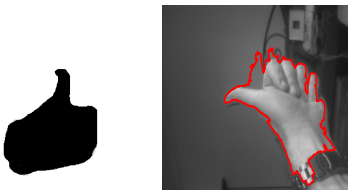

Fig. 7. Gesture detection - left: model, right: result

Image Retrieval In Figures 8,9,10 we demonstrate that based on a very general template, we can relatively robustly search for similar objects in a data set of

images. This can be used as a preprocessing stage in image retrieval system based on a general shape template. It is important to note that some false objects can be also detected (especially if the shape itself is not that unique) and they have to be pruned by a subsequence more careful analysis (e.g. using colors and texture).

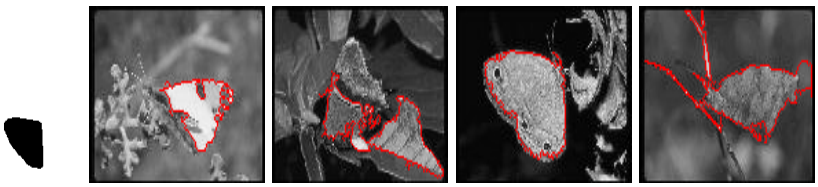

Fig. 8. Object retrieval by shape - left: model, right: result of some images retrieved with a candidate object.

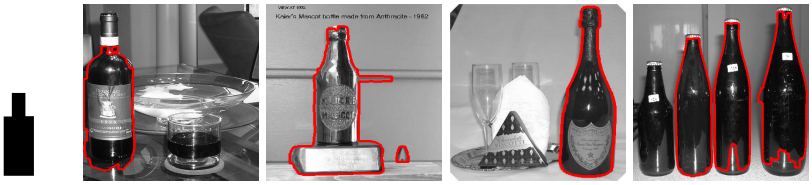

Fig. 9. Bottle retrieval by shape - left: model, right: result of some images retrieved with a candidate object. The left bottle is out of scale limitation in this example.

Fig. 10. Car retrieval by shape - left: the model.

Text analysis Text and document processing commonly deals with binarizing images prior to their processing by OCR or hand writing recognition. Our double threshold algorithm can be viewed as a powerful adaptive threshold with shape similarity criteria. We can use several models for letters to detect or segment letters in images with very hard conditions like the historical document in Figure 11. In some specific cases (even though it is not very general) the fine thresholding in our method can be used to segment hand writing letters that is almost completely connected to others like in Figure 11, right. Due to false detection, our method can only be used in this application as a p reprocess.

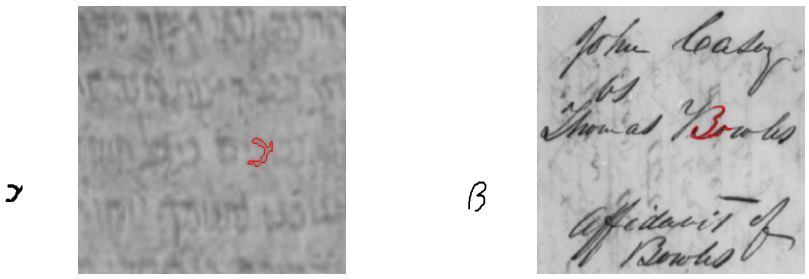

Fig. 11. Text binarization and letter segmentation in hard documents

6 More experimental results

We implemented all our algorithms in C++ on a standard single CPU computer under windows XP. In this section we show results concerning two issues: correctness of the algorithms and computation time. The results show that although the double thresholding algorithm has some limitations (characterized in the paper), it works very well in practical cases. With respect to computation time we show that our binary search algorithm for double thresholding is much faster than the algorithm described in Section 3.2. In the set of images in Figures 12 we applied our algorithm for double thresholding from Section 4 on various kinds of inputs. For every image, a binary model is given.

To demonstrate the speed of our second algorithm we compared (Figure 13) its runtime to the runtime of our first double threshold algorithm (Section 3.2). The first is more general but is slow (always $O(KN)$). The second is somewhat limited but much faster (practically nearly linear in N). We also show the number of output components that we obtain for the algorithm is Section 4.

7 Limitations

Being simple, fast and invariant to similarity transformations, our method can be used in many applications. However, it is important to make clear that there are obvious limitations in using threshold in general and even in using double thresholding. The object in the image should be relatively homogeneous in order to be contained between the two threshold values.

8 Conclusions

We have investigated the idea of optimal single and double thresholding based on a model as a simple and fast alternative to more complex (thought more general) methods for object segmentation with priors, object recognition and edge detection for recognition. The idea was shown first for single threshold for warm-up and then two algorithms for double threshold were presented and the

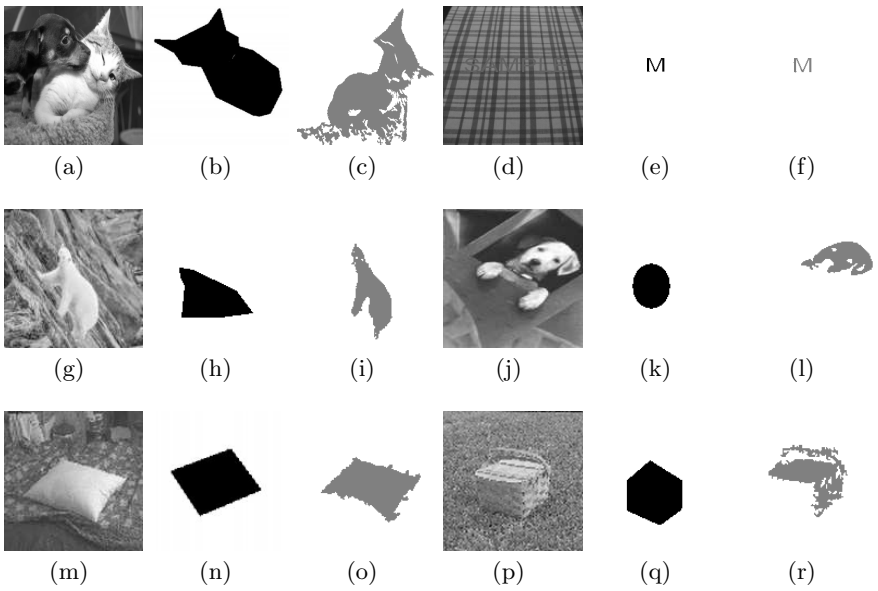

Fig. 12. Several inputs and their model based optimal double threshold. Left: image, middle: model, right: result. The result shows only the most similar component (see the text).

trade-offs were discussed. An important property of our algorithms is that they have no parameters excluding the allowed size change of the object.

References

1. Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222-1239, Nov. 2001.
2. Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *Intl J. Computer Vision*, 70(2):109-131, Nov. 2006.
3. E.J. Breen, R. Jones, "Attribute Openings, Thinnings and Granulometries", *Computer Vision and Image Understanding*, Vol. 64, No. 3, pp. 377-389, 1996.
4. M. Couprie and G.Bertrand. Topological grayscale watershed transform. *In SPIE Vision Geom.V Proc.*,volume 3168,pp. 136-146,1997.
5. M. Couprie, L.Najman and G.Bertrand. Quasi-linear algorithms for the topological watershed. *JMIV,22(2-3):231-249,2005.*
6. *D. Cremers, S. J. Osher, and S. Soatto. Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. Intl J. Computer Vision, 69(3):335-351, Sept. 2006.*
7. *F. Felzenszwalb. Representation and detection of deformable shapes. IEEE Trans. Pattern Anal. Mach. Intell., 27(2):208-220, Feb. 2005*
8. *P.Guillataud, Contribution l'analyse dendroniques des images. PhD thesis, Universit'e de Bordeaux I,1992.*

Figure	Alg. 1 (3.2)	Alg. 2 (4)	Number of CC
12 a	12928	281	110
12 d	10240	250	456
12 g	25728	481	19
12 j	8960	170	57
12 m	2560	80	186
12 p	2560	70	207

Fig. 13. The runtime of our two double threshold algorithms (miliseconds).

9. *P. Hanusse, P. Guillataud, Sémantique des images par analyse dendronique, 8th Conf. Reconnaissance des Formes et Intelligence Artificielle, Vol. 2, pp. 577-588, AFCET Ed., Lyon, 1992.*
10. *J.A. Hartigan. Statistical theory in clustering. Journal of Classification, 2:63-76,1985.*
11. *M-K. Hu, Visual pattern recognition by moment invariants, IRE Trans. on Information Theory, pp. 179-187, 1962.*
12. *R. Jones. Component trees for image filtering and segmentation. In NSIP97,1997.*
13. *J. Mattes, M. Richard and J.Demongeot. Tree representation for image matching and object recognition. In LNCS:1568, pp. 298-309,1999.*
14. *P. Monasse. Morphological representation of digital images and application to registration. PhD thesis,Paris-Dauphine Univ.,June 2000.*
15. *Laurent Najman and Michel Couprie, Building the Component Tree in Quasi-Linear Time, IEEE Transactions on Image Processing, 15(11), pp. 3531-3539, 2006.*
16. *T. Riklin-Raviv, N. Kiryati, and N. Sochen. Prior-based segmentation and shape registration in the presence of perspective distortion. Intl J. Computer Vision, 72(3):309-328, May 2007.*
17. *M. Rousson and N. Paragios. Shape priors for level set representations. In ECCV, pp. 78-92, 2002.*
18. *P. Salembier, A. Oliveras, L. Garrido, Antiextensive Connected Operators for Image and Sequence Processing, IEEE Trans. on Image Processing, Vol. 7, No. 4, pp. 555-570, 1998.*
19. *P.Salembier and J.Serra. Flat zones filtering,connected operators and filter by reconstruction. IEEETr.on Im.Proc.,3(8):1153-1160,1995.*
20. *T. Schoenemann and D. Cremers. Globally optimal image segmentation with an elastic shape prior. In ICCV, 2007.*
21. *R.E. Tarjan. Efficiency of a good but not linear set union algorithm. Journal of the ACM,22:215-225,1975*
22. *Erik R. Urbach and Jos B.T.M. Roerdink and Michael H.F. Wilkinson. Connected Shape-Size Pattern Spectra for Rotation and Scale-Invariant Classification of Gray-Scale Images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29,pp 272-285, 2007.*