Published in Image Processing On Line on 2014–05–05.
Submitted on 2013–02–12, accepted on 2013–05–06.
ISSN 2105–1232 © 2014 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2014.71

Interactive Segmentation Based on Component-trees

Benoît Naegel[1], Nicolas Passat [2]

[1] University of Strasbourg
http://icube-miv.unistra.fr/fr/index.php/Benoit_Naegel
b.naegel@unistra.fr
[2] University of Reims
http://crestic.univ-reims.fr/membre/1542-nicolas-passat
nicolas.passat@univ-reims.fr

Abstract

Component-trees associate to a discrete gray-level image a descriptive data structure induced by the inclusion relation between the binary components obtained at successive level-sets. This article presents an interactive segmentation methodology based on component-trees. It consists of the extraction of a subset of the image component-tree, enabling the generation of a binary object which fits at best (with respect to the gray-level structure of the image) a given binary target selected beforehand in the image. Compared to other interactive segmentation methods, the proposed methodology has the following advantages: (i) the segmentation result is only composed of a union of connected components of the level-sets, which ensures that no "false contours" are included; (ii) only one image marker is needed: in particular, there is no need to give a marker for the background (contrary to some other methods); (iii) the method is fast and efficient, leading to a result computed in real-time on common image sizes.

Source Code

The source code and the online demonstration are accessible at the IPOL web page of this article[1].

Keywords: component-tree; mathematical morphology; segmentation.

1 Short Presentation

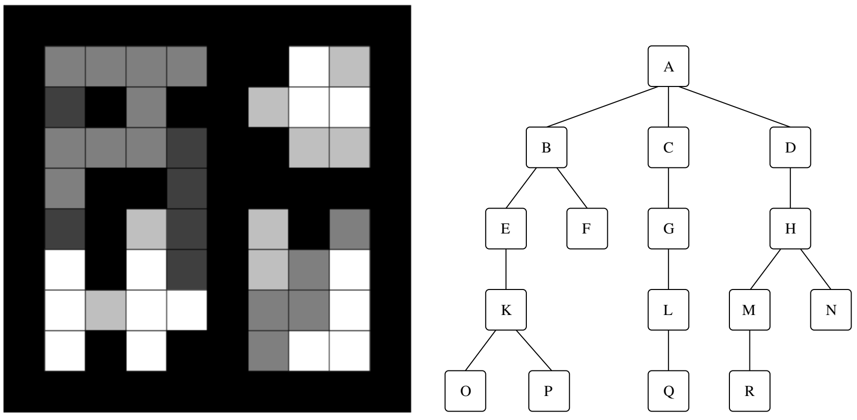
1.1 Component-tree

- A component-tree T associates to a (discrete) gray-level image a descriptive data structure induced by the inclusion relation between the binary components obtained at successive level-sets.

[1]http://dx.doi.org/10.5201/ipol.2014.71

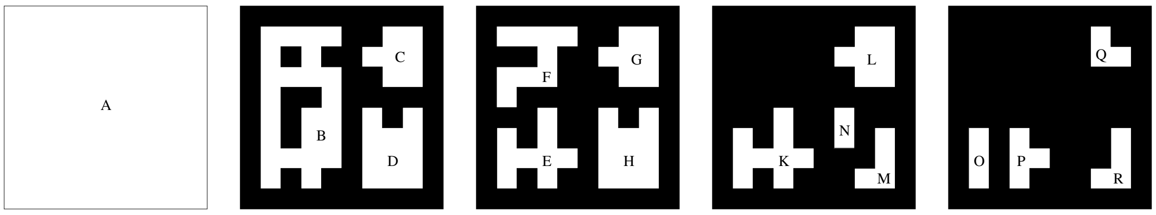
- A level-set of an image $I : E \to V$ at level v is defined as: $X_v(I) = \{p \in E \mid I(p) \geq v\}$.
- A node of the component-tree is a connected component of $X_v(I)$.
- A node N_1 is an ancestor of N_2 if $N_2 \subset N_1$.
- The root of the component-tree is the set E.

Figure 1 illustrates an image and its component-tree and Figure 2 gives its successive threshold sets.



Original image I Component-tree of I

Figure 1: An image and its component-tree.



$X_v(0)$ $X_v(1)$ $X_v(2)$ $X_v(3)$ $X_v(4)$

Figure 2: Successive threshold sets of I for v from 0 to 4.

1.1.1 Segmentation Based on Component-tree

Let $\mathcal{K}$ be the set of all connected components of all threshold sets:

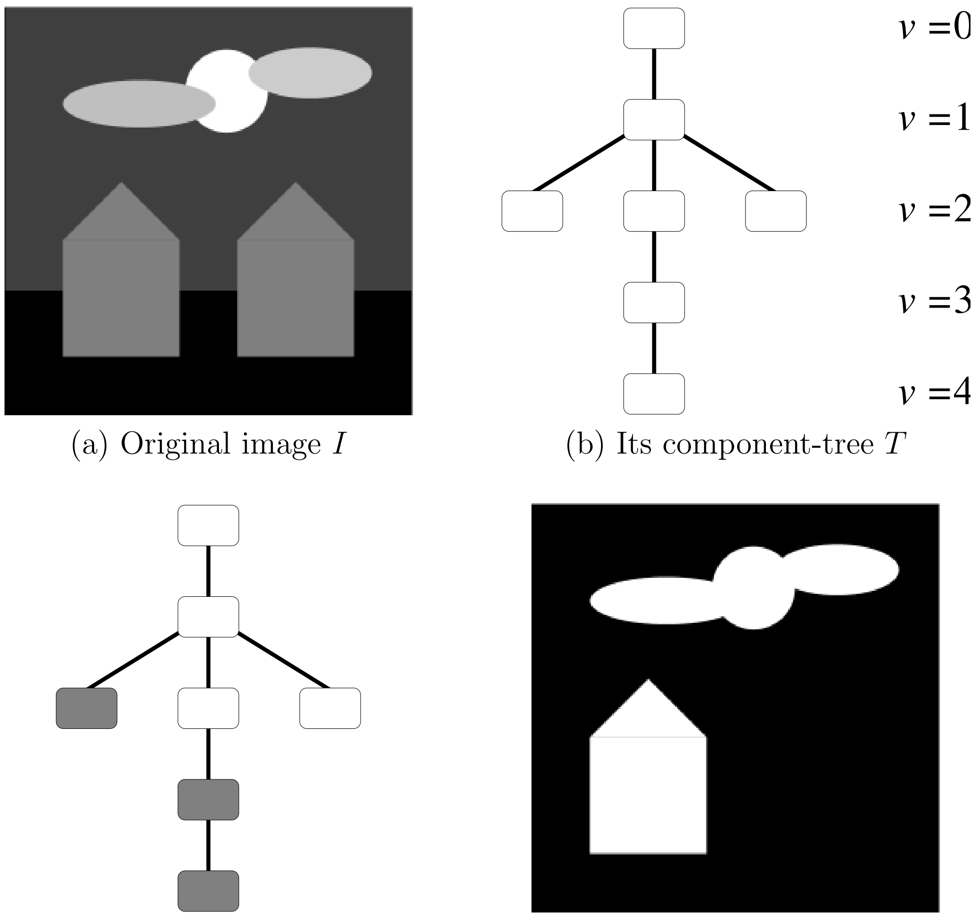
$$\mathcal{K} = \bigcup_{v \in V} \mathcal{C}[X_v(I)],$$

where $\mathcal{C}[X]$ denotes the connected components of X.

A segmentation from the component-tree is performed by selecting a subset $\mathcal{K}' \subseteq \mathcal{K}$ of nodes and computing the associated binary image:

$$S = \bigcup_{N \in \mathcal{K}'} N.$$

Figure 3 illustrates such steps.



(a) Original image I (b) Its component-tree T

(c) Selected subsets of nodes (in gray) (d) Associated binary image

Figure 3: Illustration of the segmentation process based on component-tree.

1.1.2 Problem to Solve

- Let G be a binary image (for example a manual marker depicted by the user)
- How to select a subset of nodes of T whose union fits "at best" the binary image G?
- This problem can be summarized as a minimization problem, consisting of determining:

$$\widehat{\mathcal{K}} = \arg \min_{\mathcal{K}' \in \mathcal{P}(\mathcal{K})} \{d^\alpha(\bigcup_{N \in \mathcal{K}'} N, G)\}.$$

- Given a parameter $\alpha \in [0,1]$, d^α is a pseudo-distance that takes into account the amount of false-positives/negatives between G and $\bigcup_{N \in \mathcal{K}'} N$. This distance is defined by:

$$d^\alpha(X, Y) = \alpha.|X \setminus Y| + (1-\alpha).|Y \setminus X|.$$

1.1.3 Solution

A solution of this problem can be found by using dynamic programming. Let $\mathcal{F}^\alpha$ and c^α be the functions recursively cross-defined, for all $N \in \mathcal{K}$, by:

$$(\mathcal{F}^\alpha(N), c^\alpha(N)) = \begin{cases} (\{N\}, \alpha.n(N,G)) & \text{if } \alpha.n(N,G) \prec (1-\alpha).ps(N,G) + \sum_{N' \in ch(N)} c^\alpha(N') \\ \\ (\bigcup_{N' \in ch(N)} \mathcal{F}^\alpha(N'), \\ \;(1-\alpha).ps(N,G)+ \\ \;+\sum_{N' \in ch(N)} c^\alpha(N')) & \text{otherwise} \end{cases} .$$

where:

- $\prec \in \{<, \leq\}$;
- $ch(N)$ is the set of children of N;
- $E_N = N \setminus \bigcup_{N' \in ch(N)} N'$ (the points of N which do not belong to the children);
- $ps(N,G) = |E_N \cap G|$ the number of points of N which belong to G and which do not belong to any children of N;
- $n(N,G) = |N \setminus G|$ (the number of points of N which do not belong to G).

The set of nodes $\mathcal{F}^\alpha(E)$ enables to minimize $d^\alpha(., G)$

2 On Line Demo

Outline of the interactive demo

1. Interactive drawing of marker set (painting method) on the original image.
   or
   Upload marker (binary) image of same size than original image.
2. Computation of component-tree (automatic step, no interaction).
3. Interactive selection of the parameter α, which defines the distance between the manual marker set and the segmentation result (binary image).
   or
   Enter an α value (real number between 0.0 and 1.0).
4. Display segmentation result.
5. To refine the segmentation, go to 1.

3 Algorithm

3.1 Description

The algorithm is based on two steps:

1. Component-tree computation of the image to be segmented;
2. Computation of the segmentation result based on a cost minimization.

An interesting consideration is that when using the interactive version of the method (where the α parameter is adjusted in real-time) the first step (component-tree computation) needs only to be performed once.

Algorithm 1: Segmentation Algorithm

```
input  : Image $I$ (to be segmented);
Image $G$ (marker (binary) image);
Parameter $alpha$ (weight parameter for false-positives/negatives)
output: Image $S$ (final (binary) segmentation of $I$);
////////////////////////////////////////
// Step 1: Component-tree computation
////////////////////////////////////////
compute component-tree $T$ ;
for all nodes $N$ of $T$ do
    store $n(N,G)$ ;
    store $ps(N,G)$ ;
////////////////////////////////////////
// Step 2: Cost minimization
////////////////////////////////////////
for $v$ from $I_{max}$ to $I_{min}$ do
    for all $N$ such that gray-level of $N$ is $v$ do
        if $N$ is a leaf then
            if $alpha * n(N,G) < (1 - alpha) * ps(N,G)$ then
                $cost(N) = alpha * n(N,G)$;
                $F(N) = \{N\}$;
            else
                $cost(N) = (1 - alpha) * ps(N,G)$;
                $F(N) = \{\emptyset\}$;
        else
            if $alpha * n(N,G) < (1 - alpha) * ps(N,G) +$ sum of cost(N') for all children $N'$ of $N$ then
                $cost(N) = alpha * n(N,G)$;
                $F(N) = \{N\}$;
            else
                $cost(N) = (1 - alpha) * ps(N,G) +$ sum of cost$(N')$ for all children $N'$ of $N$ ;
                $F(N) = \cup F(N')$ for all children $N'$ of $N$ ;
////////////////////////////////////////
// Step 3: Result computation
////////////////////////////////////////
$S = F(E)$;
```

3.2 Component-tree Computation

This step can be accomplished by using Salembier's algorithm [4] based on a recursive flooding of the image or Najman's one [2], based on Tarjan's union-find paradigm. In the case of 8 bits image ($V = [0 \dots 255]$) Salembier's algorithm is twice as fast than Najman's algorithm. The attributes $ps(N,G)$ and $n(N,G)$, associated to each node, can be computed incrementally during the construction of the tree.

3.3 Short Complexity Analysis

The algorithmic complexity of step 1 depends on the chosen algorithm. Salembier's algorithm has a worst-case time complexity of $\mathcal{O}(|E|.|V|)$ whereas Najman's one is quasi-linear if the sorting step is implemented in a linear fashion[2]. The algorithmic complexity of steps 2 and 3 is $\mathcal{O}(\max\{|\mathcal{K}|, |E|\})$, linear with respect to the number of nodes of the tree or the size of the image.

4 Implementation

4.1 Command Line Implementation

A command line implementation of the method in C++ working with PGM gray-level images is available in the IPOL web page of this article (ctseg.tgz)[3]. The program `ctseg` takes three arguments:

1. The source image (PGM file, 8 bits);
2. The marker image (PGM file, 8 bits). This image must have the same size as the source image. All non-zero pixels belong to the marker.
3. The alpha parameter (float number between 0 and 1).

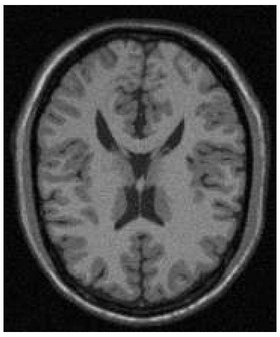
To convert a gray-level image from any "standard" format in PGM you can use ImageMagick:

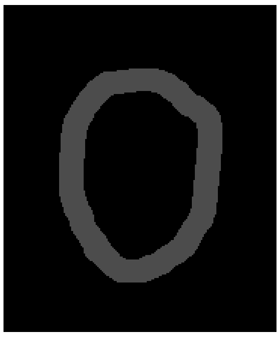
```
convert image.png image.pgm
```

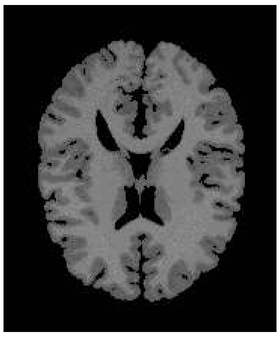
To test the program, go in the `test/Brainweb` folder and type:

```
../../ctseg bw1.pgm bw_mark1.pgm 0.2
```

which provides the image `result.pgm`:


`bw1.pgm`


`bw_mark1.pgm`


`result.pgm`

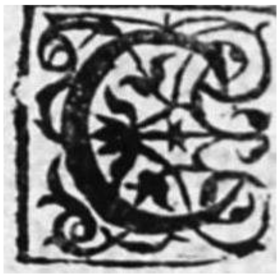
Optionally the flag `negate` can be added as a fourth argument in order to take the negative of the original image, in order to segment dark objects on white background. For example, go to folder `test/dropcaps` and type:

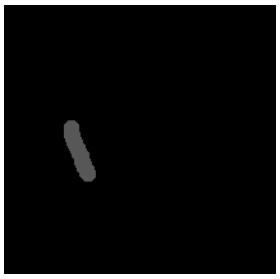
```
../../ctseg C2.pgm C2_mark.pgm 0.01 negate
```

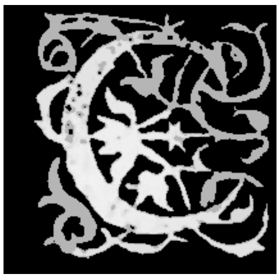
which provides the image `result.pgm`:

[2] For example using counting sort when the weights are small integers [2].

[3] `http://dx.doi.org/10.5201/ipol.2014.71`


`C2.pgm`


`C2_mark.pgm`


`result.pgm`

4.2 Implementation Details

Component-tree computation In our implementation, we use Salembier's algorithm to compute the component-tree. Therefore an implementation of the algorithm is provided in the source code.

Processing the nodes The first loop (line 11-12) ensures that all nodes are processed after all their descendants. In our implementation, this is done by performing a preliminary breadth-first scan of the tree and storing all the nodes in an array. This array is then processed in reverse order.

Node data structure A node is implemented as a data structure having, along other attributes, a pointer on its father and the set of its childs.

5 Examples

We present some results obtained with our segmentation method in two different contexts: drop caps segmentation and medical imaging.

5.1 Drop Caps Segmentation

In this example drop caps obtained from digitization of archive documents are composed of bright objects on dark background. Figure 4 illustrates some segmentation results obtained for various markers and α values.

5.2 Medical Images Segmentation

Our segmentation method has been compared with a graph-cuts based method, leading to similar (and slightly better) results. In our experiments, we use an interactive segmentation tool providing a graph-cuts method [1]. We use the following parameters: no smoothing, $\alpha = 1$, $\sigma = 1$, $\lambda = 0$, histogram quantization=1. The algorithms are evaluated in terms of quality with respect to the time of interaction. The segmentation quality is computed in terms of proximity to ground truth image, based on two distances: the point-to-set distance and the κ index defined as: $\kappa = \frac{2TP}{2TP+FP+FN}$ where TP, FP, FN define respectively the amounts of true positives, false positives and false negatives with respect to the ground-truth images. See [3] for further details.

Figures 5 and 6 provide a performance comparison between our interactive segmentation method and the graph-cuts one. Both algorithms have been used in a similar context by three experts during two minutes in order to obtain the best possible result.

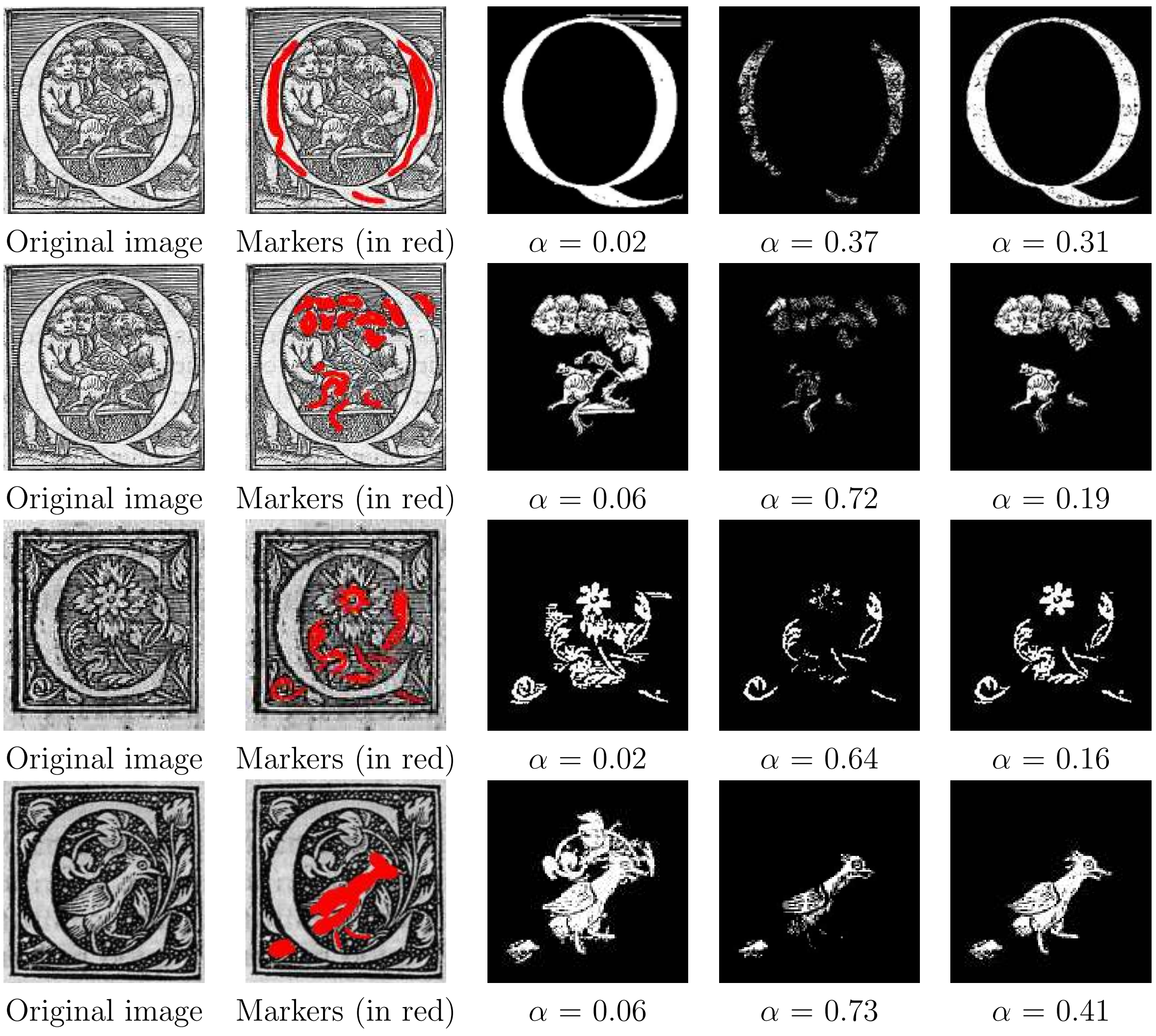


Figure 4: Drop caps segmentation

Image Credits

All images and contours have been created given by the authors.

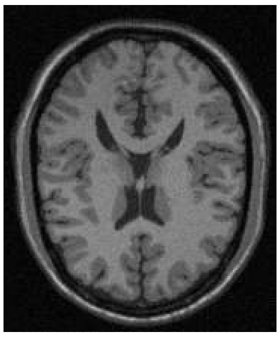
References

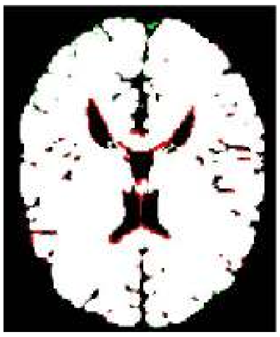
[1] K. McGuinness and N.E. O'Connor, *A comparative evaluation of interactive segmentation algorithms*, Pattern Recognition, 43 (2010), pp. 434–444.

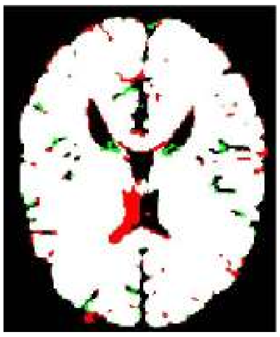
[2] L. Najman and M. Couprie, *Building the component tree in quasi-linear time*, IEEE Transactions on Image Processing, 15 (2006), pp. 3531–3539.

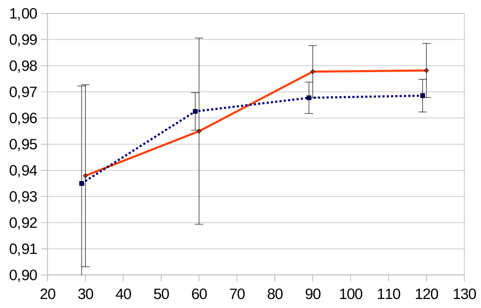
[3] N. Passat, B. Naegel, F. Rousseau, M. Koob, and J.-L. Dietemann, *Interactive segmentation based on component-trees*, Pattern Recognition, 44 (2011), pp. 2539–2554.

[4] P. Salembier, A. Oliveras, and L. Garrido, *Anti-extensive connected operators for image and sequence processing*, IEEE Transactions on Image Processing, 7 (1998), pp. 555–570.


(a)


(b)


(c)


(d) Time of interaction (x-axis, in seconds) vs. κ index.

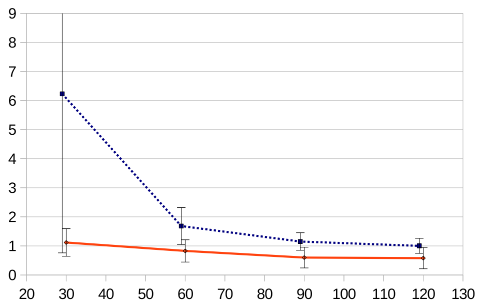
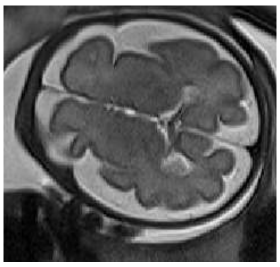
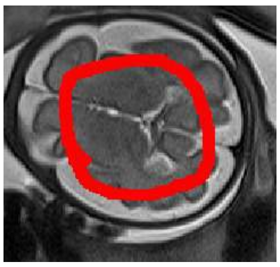
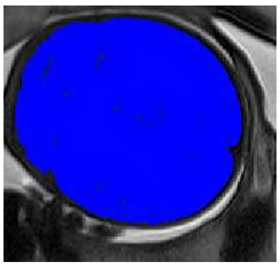
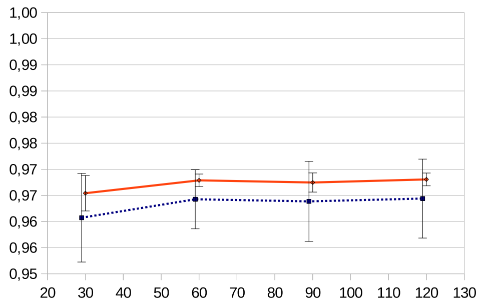

(e) Time of interaction (x-axis, in seconds) vs. point-to-set distance.

Figure 5: **Brainweb images (simulated normal brain MRI data)**. (a) Original image. (b) Segmentation using the proposed method. (c) Segmentation using graph-cuts. False-positives are in red, false-negatives in green, and true-positives in white. (d-e) κ index and point-to-set distance between segmented image and ground-truth for our method (in blue) and method based on graph-cuts (in red).


(a)


(b)


(c)


(d) Time of interaction (x-axis, in seconds) vs. κ index.

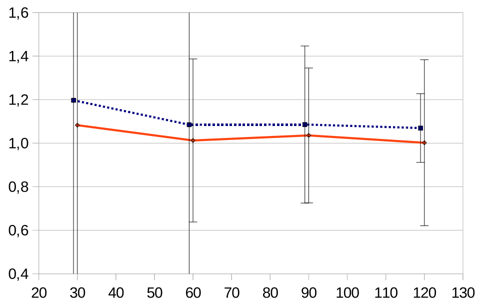

(e) Time of interaction (x-axis, in seconds) vs. point-to-set distance.

Figure 6: **In vivo foetal brain MRI images**. (a) Original image. (b) Manual marker (in red). (c) Segmentation result (in blue). (d-e) κ index and point-to-set distance between segmented image and ground-truth for our method (in blue) and method based on graph-cuts (in red).