DOI: 10.4149/BLL_2017_093

BIOMEDICAL METHODS

Use of graph algorithms in the processing and analysis of images with focus on the biomedical data

Zdimalova M[1], Roznovjak R[1], Weismann P[2], El Falougy H[2], Kubikova E[2]

Institute of Anatomy, Faculty of Medicine, Comenius University in Bratislava, Bratislava, Slovakia.
hisham.elfalougy@fmed.uniba.sk

ABSTRACT
INTRODUCTION: Image segmentation is a known problem in the field of image processing. A great number of methods based on different approaches to this issue was created. One of these approaches utilizes the findings of the graph theory.
METHODS: Our work focuses on segmentation using shortest paths in a graph. Specifically, we deal with methods of "Intelligent Scissors," which use Dijkstra's algorithm to find the shortest paths.
RESULTS: We created a new software in Microsoft Visual Studio 2013 integrated development environment Visual C++ in the language C++/CLI. We created a format application with a graphical users development environment for system Windows, with using the platform .Net (version 4.5). The program was used for handling and processing the original medical data.
CONCLUSION: The major disadvantage of the method of "Intelligent Scissors" is the computational time length of Dijkstra's algorithm. However, after the implementation of a more efficient priority queue, this problem could be alleviated. The main advantage of this method we see in training that enables to adapt to a particular kind of edge, which we need to segment. The user involvement has a significant influence on the process of segmentation, which enormously aids to achieve high-quality results *(Fig. 7, Ref. 13)*. Text in PDF *www.elis.sk.*
KEY WORDS: image analysis, intelligent scissors, medical data, CT image, MRI scan.

Introduction

Image segmentation

Image segmentation is a classical problem of image processing, which is being studied during the last decade (1). Its main aim is to split the image into disjoint regions formed by pixels; each such region is bordering a meaningful object or area in the image. The regions created are usually referred to as segments.

One of the biggest problems of image segmentation is ambiguous image information, frequently accompanied by noise (2, 3). The more a priori information is available for the process of segmentation the better result can be achieved. Segmentation may be divided into:

1) Complete segmentation – dividing the image into non-overlapping regions, corresponding to specific objects or areas of the image.

2) Partial segmentation – dividing the image into regions, which are homogeneous considering certain characteristics, such as brightness, color, and texture.

[1]Slovak University of Technology in Bratislava, Faculty of Civil Engineering, Department of Mathematics and Descriptive Geometry, and [2]Institute of Anatomy, Faculty of Medicine, Comenius University in Bratislava, Slovakia

Address of correspondence: H. El Falougy, MD, PhD, Institute of Anatomy, Faculty of Medicine, Comenius University in Bratislava, Sasinkova 2, SK-813 72 Bratislava, Slovakia.
Phone: +421.2.59357374

Acknowledgements: The first author acknowledges Vega support by the grant from Grant Agency VEGA 1/0420/15/.

A high number of segmentation algorithms, which approach this problem in various ways, were developed. These methods can be classified into five groups (2, 3, 4, 5, 6).

1) Thresholding
2) The edge-oriented methods
3) Methods based on regions
4) Methods based on a comparison
5) Active contour models

Thresholding

It is the simplest segmentation technique, which is computationally undemanding and rapid. A constant value of brightness called threshold is used for segmentation of objects and background. These methods try to determine the threshold value automatically. The threshold value may be constant for the entire image (global thresholding) or may vary in different parts of the image (local thresholding). Using global thresholding only in specific cases is sufficient. In addition to these two basic types of thresholding, there are many other specifications (3, 4).

The edge-oriented methods

They are based on edge detection using edge detector. These edges mark places where there is a change in the level of gray, color, and texture. The most common problem with this approach, due to noise or incorrect information in the image, is finding the edges in the inappropriate locations or absence of edges in the places where they should be located (3, 4).

Methods based on regions

These methods do not seek border bounding regions, but the regions themselves formed by a set of pixels. These methods are appropriate in cases of noisy pictures when it is hard to find the boundaries between objects. The basic idea of these methods is the division of the image into regions with the greatest possible homogeneity; the homogeneity criteria may be based on the level of gray, color, and texture (3, 4).

Methods based on a comparison

They are used to locate objects that are similar to the pattern designated by us. The correlation between the object and the pattern is determined by the optimality criterion based on the characteristics of the object. The value of optimality criteria is calculated for each position and rotation of the pattern in the image. If this value exceeds a certain threshold, that location represents the location of the pattern (2, 3, 4).

Active contour models

We place an initial curve around the object that we wish to segment. The curve is then deformed in order to minimize the energy functional. This functional is formed as the sum of internal and external energy. The internal energy is designed by the shape of the curve. A smooth curve without significant changes shall take the form of low value. External energy depends on the image and becomes a low value on the edges where the gradient shall make the greatest value (2, 3, 4).

A further possible division of the segmentation methods constitutes of:

1) Automatic methods

2) Interactive methods

Automatic methods do not require the user interaction during the process of segmentation.

The segmentation by these methods is, therefore, faster and more convenient. However, these methods do not always give the desired results, as in the case of medical data in which objects are ambiguously defined or contain strong noise.

Interactive methods can overcome this problem by involving humans in the process of segmentation. The disadvantage is a long segmentation time and the need for participation of a person in the process of segmentation (1, 2, 3, 4, 5, 6).

Segmentation and graph theory (Fig. 1)

Many of the methods of image segmentation use the findings of the graph theory. In this part of the paper, we describe these methods. However, we need to define at first certain concepts (1, 3, 4).

Let $G = (V, E)$ be a graph, where $V = \{v1, v2, ..., vn\}$ is a set of vertices that can represent a set of image pixels or regions in Euclidean space. Let E be the set of edges that connect adjacent peaks. Each edge $(vi, vj) \in E$ is measured by the weight $w (vi, vj)$, which represents one variable dependent, for example, on the values of pixels that the edge connects. The image can be divided into separate components, being accepted that component A is a connected graph $G = (V', E')$, where $V' \subseteq V$, $E' \subseteq E$ and E' contain only the edge of the vertices V'. Therefore non-empty sets $A1, A2, ..., Ak$ form a division of the graph G if $Ai \cap Aj = \emptyset$ for $i \neq j$, i and $j \in (1, 2, ... ,k)$ and $A1 \cup ... \cup Ak = G$.

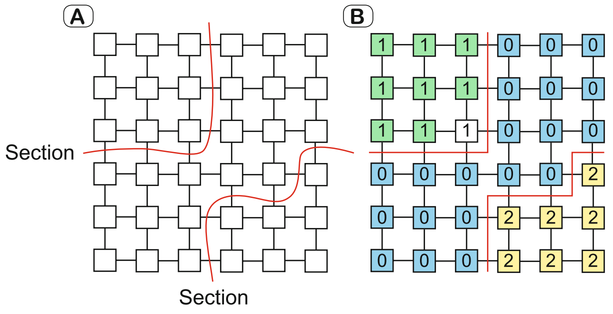


Fig. 1. (A) Illustration of two graph cuts. Cuts divide the image into three segments. (B) The assignment of values from the set L = {0, 1, 2} for the entire peak (pixels). The image again is divided into three segments.

For the components should be valid, that the properties such as brightness value, color, and texture are similar throughout the whole component. Then, the degree of dissimilarity of two components can be calculated as the cut of the graph. This cut passes through the edges in the graph, which divides the graph into two disjoint sets A and B. The graph cut is defined as

$$cut\ (A, B) = \sum_{u \in A, v \in B} w(u, v),$$

where u and v are the vertices of the different components, Fig. 1. Afterwards, the issue of image segmentation can be treated as an optimization problem in which we try to minimize certain criterion. In this case it shall be optimal to divide the graph into two segments, which minimize the graph cut.

Image segmentation is frequently formulated as a problem, in which we assign a value of L sets to elements from the set of sites S. An example could be set $L = \{object, background\}$ and set S formed by the set of pixels of the image. In this way, the image is divided into two segments, and each pixel is clearly assigned to one of the two segments (1, 5, 7).

In the following text, we describe the basic partitioning methods using the knowledge from the graph theory (1, 6).

Methods based on minimal spanning tree

Frame T is a tree of a graph G, at which is valid $T = (V, E')$, where $E' \subseteq E$. The graph may have several different spanning trees. However, minimal spanning is the frame having the smallest scales on the edges from all the skeletons. The graph can have a number of different spanning trees; however, minimal spanning tree is the one having the smallest weight on the edges from all the spanning trees. To find such a minimal spanning tree, we may use for instance Prim's algorithm, in which iteratively adding to the edges of the tree with the smallest weights. The main concept of these methods lies in the properties of the minimal spanning tree. It is the clustering of pixels with similar values into subgraphs. We obtain these subgraphs if the minimal spanning tree is partitioned in the places where the edges acquire the highest weight.

Methods based on the minimal cuts

In addition to graph cut defined in the equation above, there are alternative definitions of cuts for which we want to minimize value. From the previous statement, the advantage of these methods is apparent, including the ability to use different types of cuts for various applications. The main disadvantage of the cut defined as (3.1) is a preference for finding small components. This problem can be solved by using a normalized cut

$$Ncut(A, B) = \frac{cut(A, B)}{vol(A)} + \frac{cut(A, B)}{vol(B)},$$

where the term $volA = \sum_{v_i \in A, v_j \in V} w(v_i, v_j)$ presents the sum of the weights among all the peaks of the A set and all vertices of the graph. If no edge appears between the vertices, then the weight shall equal zero. For in this waydefined cut it is valid that if the set A or B contains a few vertices, the value of the cut will not be low. Segmenting so defined cut also has its disadvantages, which are creating segments with similar weights on the edges. Another definition of a cut uses the information about the edge and interior of the segment. In particular, this cut is defined as

$$RegionCut\,(A, B) = \frac{cost(P)}{weight(P)},$$

where P is an oriented path. In the graph G starting and ending at the same peak, *Cost (P)* is the length of the *path P*, and *weight (P)* is the sum of the weights of the area bounded by the path P. The main disadvantage of this cut definition is the preference of large objects and the need to segment only objects with closed borders. Besides these cuts, there are several other definitions (1, 3, 4, 7).

Graph cuts based on Markov random fields

Study of psychology has shown that to interpret an image it is necessary to take into account the context of the image information. It means that if we attempt to identify an object, which we do not see entirely, we may not succeed. The image should, therefore, be understood in the visual and spatial context. The theory of Markov random fields allows this approach (1, 2, 7, 8, 9).

Methods based on the shortest paths in graphs

Finding the shortest path between two vertices is a classical problem of the graph theory (for further reading about shortest paths in the graph, see (3, 4). The best-known algorithm that solves this issue is Dijkstra's algorithm that, together with other, will be described more closely later. Segmentation problem, in which we are looking for the best segment boundaries, is reformulated as a problem of finding the shortest path between two vertices. In practical applications, the user interaction is applied, thereby increasing the accuracy of the segmentation. The live-wire method is used, which allows the user to select the first point on the border. Subsequently, the shortest route between the first point and the actual location of the cursor is displayed in real-time. A user task is to select the position of the cursor so that the drawn shortest path makes the best approximated the border part. Subsequently, new shortest paths will be drawn from this point, and this process iterates until the creation of closed boundaries. Formed boundary is represented by a sequence of oriented edges between pixels; each edge is valued. The value is low if the edge is appropriate as a border of the object. To create the shortest path between two points, it must pass through the edges with the lowest values. Hence the portrayed shortest path will cross the border of the object. This approach of segmentation requires the finding of the shortest paths from point to all other points in each step. It is also a major drawback whereas a certain computing time is needed, which increases with the number of pixels in the image. We describe a further segmentation technique called "Intelligent Scissors," based on the principle of seeking the shortest paths in a graph.

In our work, we focus on graph algorithms seeking the shortest path in the graph (10, 11, 12). Specifically, the application of Dijkstra's algorithm method called "Intelligent Scissors." At first, we will describe Dijkstra's algorithm and its method.

Dijkstra's algorithm

Dijkstra's algorithm solves the problem of finding the shortest paths in edge weighted and directed graph $G = (V, E)$, whereas its all edge shall have non-negative weights (13). Thus the shortest path is determined from a specific vertex to all other vertices. The algorithm maintains a set of vertices S, whose length of the shortest path has already been determined. The algorithm repeatedly chooses vertex $u \in V\text{-}S$ with a minimum length of the shortest path adds the vertex u to the set S, and relaxes all edges directed from the vertex u. In the following implementation it is considered a priority queue Q of vertices, whose priority is determined by a minimum value $d\,(.)$. This means that its vertices will be listed in increasing order according to the estimated value of the shortest path, whereas at the beginning of the queue is the vertex with the minimally assessed value of the shortest edge.

Dijkstra's algorithm pseudo-code looks like this:

```
Dijkstra [G(V, E), w, s]
1.  INITIALIZE[G(V, E), s]
2.  $S=\emptyset$
3.  $Q=V$
4.  while $Q\neq\emptyset$
5.  u=EXTRACT_MIN(Q)
6.  $S=S\cup\{u\}$
7.  for each vertex $v\in$ neighbors(u)
8.  RELAX[u, v, w]
```

The initialization values $d(.)$ and $\pi(.)$ take place at the beginning of the algorithm. Subsequently, S is adjusted to the empty set, and the priority queue contains vertices V. During the program run time; it is valid that $Q=V\text{-}S$. In the **while** loop, from the priority queue vertex u with the smallest value d is selected $(.)$, which is then placed in the set S. As firstly selected vertex will be always initial vertex s chosen, from which we calculate the shortest paths to other vertices of the graph. In the lines 7 and 8, relaxation takes place for all edges directed from the vertex u. Therefore, if the path to the vertex v through the vertex u reduces the value $d(v)$, we shall update the values $d(v)$ and $\pi(v)$. The algorithm never embeds new values to queue Q after line 3. Thus every vertex is selected from queue Q, and inserted into the set S only once. This indicates that the **while** loop will have exactly $|V|$ iterations. The length of calculation time of the algorithm depends on how the priority queue is implemented. If the queue is implemented as a field, computational complexity of the algorithm will be $O(|V|^2)$. When using a Fibonacci heap as a priority queue, it is possible to achieve computational complexity $O(|E|+|V|log|V|)$.

Methods

Segmentation technique "Intelligent Scissors" (Fig. 2)

Now we describe an interactive segmentation technique called "Intelligent Scissors" (11). It is based on finding the optimal path in the graph, and its main advantage is the use of "live-wire". This tool allows the user to interactively select the portion of the segmented boundaries so that the shortest path from the selected pixel to the current position of the mouse pointer will immediately be displayed to him.

Shortest paths are found for all pixels in the image, and thus the user can choose the shortest path that best describes the boundary of the segmented object. After selecting the shortest path, the program uses the last pixel of that path to find the shortest paths again, and this process continues until the creation of a closed segment. Therefore, the last selected pixel of the shortest path always forms a new starting pixel for the new part of the segmented boundaries. Boundary chosen in the previous steps remains constant.

The pixel is generated automatically using the technique of "boundary cooling," to minimize the need to select a new pixel forming the end of the segmented portion. This technique uses the fact that most of the shortest paths will have several first common pixels. Set of pixels, which is obtained as the intersection of a certain amount of shortest paths, is selected as the new part of the segmented boundaries.

Selecting a new pixel includes another problem. This is the uncertainty, which arises from the mouse handling and the small size of the pixel in the image. This issue is solved by so-called cursor snapping when a new pixel is chosen from a specific area, in which is the largest value of the size of the gradient situated. In addition, the technique of "Intelligent Scissors" includes an effective training. This allows the algorithm to adapt to different types of edges.

The image is represented by a graph, where each pixel of the image accounts for a vertex of the graph. All edges of the graph are valued and oriented by weight, with eight edges heading from each vertex to the neighboring vertex. The only exceptions are the vertices on the edges of the image. Figure 2 illustrates the vertices and edges of such graph. Preview of the tool "Live-wire" in image segmentation is shown in Figure 2.

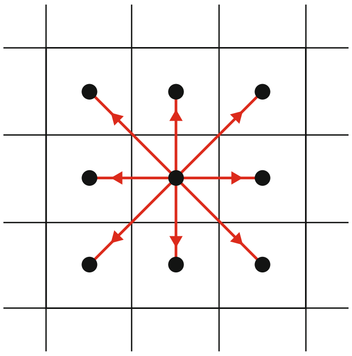


Fig. 2. Illustration of pixels and oriented edges in a graph representing an image.

The calculation of the weight values on the edges

If p and q are two adjacent pixels in the image, then $l(p, q)$ is the weight value of the directed edge from the pixel p to q [8]. The function $l(p, q)$ is a weighted sum of the functions of image characteristics, which are:

f_Z: Laplacian zero-crossing
f_G: Gradient size (magnitude)
f_D: Gradient direction
f_P: Pixel value
f_I: "Inside" value of the pixel
f_O: "Outside" value of the pixel

Thus the resulting function $l(p, q)$ has the following form:

$l(p, q) = \omega_Z * f_Z(q) + \omega_G * f_G(q) + \omega_D * f_D(p, q) + \omega_P * f_P(q) + \omega_I * f_I(q) + \omega_0 * f_0(q),$

where ω is the weight of each respective function. All these features have the range of the interval $\langle 0, 1\rangle$. Empirically, it was determined that weights $\omega_Z = 0.3$, $\omega_G = 0.3$, $\omega_D = 0.1$, $\omega_P = 0.1$, $\omega_I = 0.1$ and $\omega_O = 0.1$are giving good results for most images. If necessary, these weights can be modified, but the sum shall be equal to 1.The functions f_Z, f_G and f_D are termed static features. It means that they are fixed and do not use any a priori information about the image. Thus, their functional coast is fixed. On the contrary, the functions f_G, f_P, f_I, and f_O are designated as dynamic.

Results and discussion

Implementation, new approach, and optimization (Fig. 3)

Our aim was to implement the method of "Intelligent Scissors." We created new software in Microsoft Visual Studio 2013 integrated development environment Visual C++ in the language C++/CLI. We created a format application with graphical users development environment (Windows Forms Applications) for system Windows, with using the platform .Net (version 4.5).

The graph is often used in the memory of the computer as the matrices of adjacencies (2, 3, 4). For representation of the image of the graph we created classes for a graph. It contains information about the height and weight of the picture and one dimension - field, which size is equal to the number of pixels. Every vertex

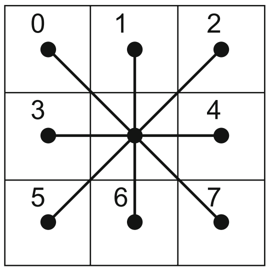
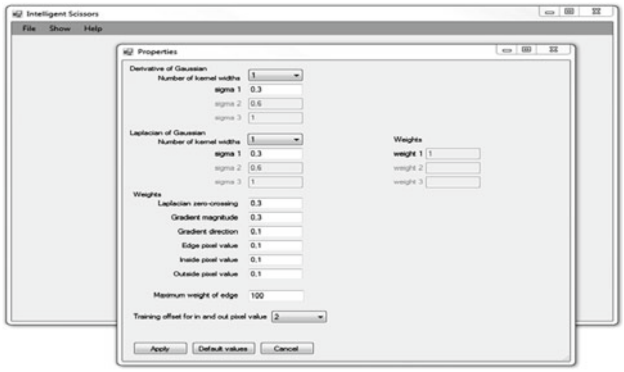


Fig. 3. Every vertex keeps the information about the color of the picture and eight-elements field of edges into adjacent vertices.



```
Derivative of Gaussian
      Number of kernel widths   1
                      sigma 1   0.3
                      sigma 2   0.6
                      sigma 3   1
Laplacian of Gaussian
      Number of kernel widths   1                 Weights
                      sigma 1   0.3               weight 1  1
                      sigma 2   0.6               weight 2
                      sigma 3   1                 weight 3
Weights
      Laplacian zero-crossing   0.3
           Gradient magnitude   0.3
           Gradient direction   0.1
             Edge pixel value   0,1
           Inside pixel value   0,1
          Outside pixel value   0,1

       Maximum weight of edge   100

Training offset for in and out pixel value   2

Apply   Default values   Cancel
```

Fig. 4. The window in the program with settings.

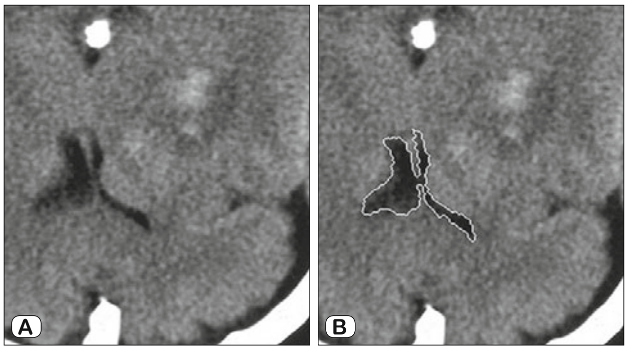


Fig. 5. CT image of the head without pathological changes. A) The original image B) The image after applying our segmentation, which is based on Intelligent Scissors, on the lateral ventricle.

keeps the information about the color of the picture and eight-elements field of edges into adjacent vertices. Indeces of edges in this field are shown in Figure 3.

The object of the class has like data for identification of the number of the entering vertex and also the weight on this edge. For the counting the responsible values of the functions f_Z and f_G the user has to choose three values for the standard deviation σ, which determines the size of preimage of the image. The values of the functions f_D and f_Z are counted just in the beginning of the segmentation after loading the image. The values of other functions are necessary to count always again.

The class Intelligent Scissors contains data and methods, which specify its running. As an example, we can mention the value set the weights w_Z, w_G, w_D, w_P, w_I and w_O, or a field of the pixel, which form the boundary of the segmentation. By active training, we use by creating histogram decreasing function of the weight *w(i)*, which returns higher weights for pixels, which are at the beginning of training set. This function is defined as:

$$\mathrm{w(i)} = -\left(\frac{\mathrm{i}}{2\,.\,(t_S + 1)}\right) + 1,$$

where t_s is the number of pixels in entering set and $i = 0, 1, \dots, t_s - 1$. It is a linear function, for which it holds $w(0) = 0$ a $w(t_s - 1) = 0{,}5$. We created for searching the shortest path created individually class. At the end of segmentation, it is necessary to close the boundary of segmentation by choosing small sizes of pixels. The user has a possibility to choose the size of the surrounding area, which is defined for touching the cursor. It means, if the initial pixel is located in the surroundings of the last chosen pixel in the image, the segmented edge will be automatically closed. After creating closed segment the user can to remove the rest of the segmented region. For this reason, it was necessary to line seeds to fill in the region ("scan – line seed fill"). In this way we were able to detect the set of all pixels of the image located in the segmented part.

User interface (Figs 4, 5, 6, 7)

The user interface is very intuitively and easy to follow. The first set is uploading the image. Supported formats are JPG, PNG, and BMP. The image is uploaded by clicking the mouse on offer *File→Open*. Consequently, we can choose and select from the standard window for opening files. After uploading the file, the window with settings will appear (Fig. 4).

All decimal numbers are necessary to submit with a comma. The first quaternion represents standard deviations of the number convolution masks. The user can settle the value of standard

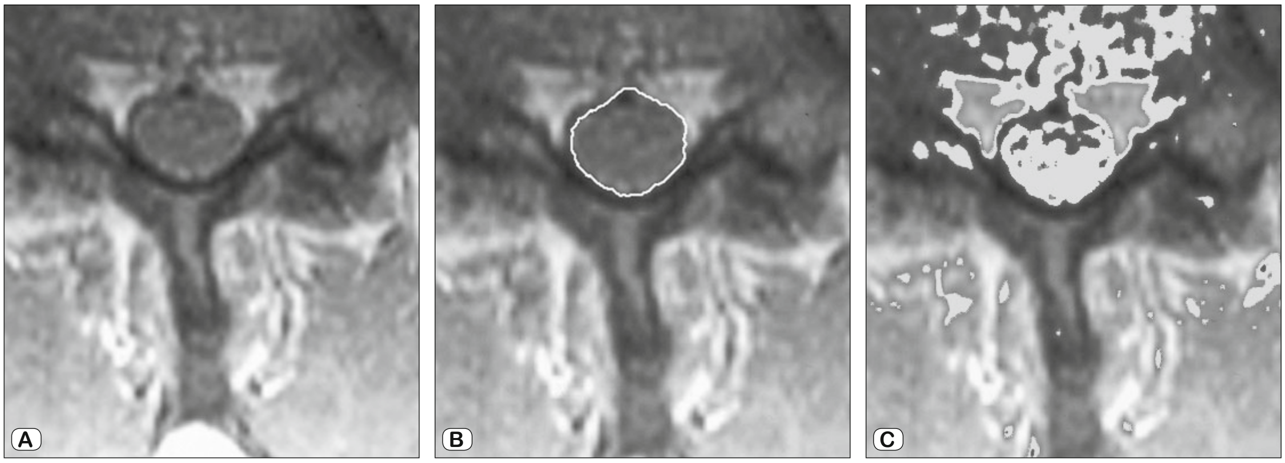


Fig. 6. MRI scan of lumbosacral spine with paramedian disc (L4/L5) herniation on the left side. A) The original image, B) The image after applying our segmentation, which is based on Intelligent Scissors, on the dural sac. C) Unsuccessful application of thresholding method on the dural sac.

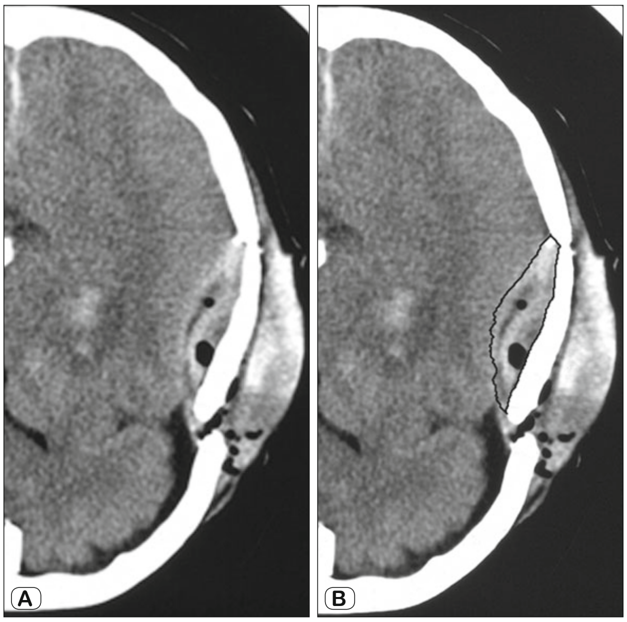


Fig. 7. CT image of the head with subarachnoid hemorrhage. A) The original image, B) The image after applying our segmentation, which is based on Intelligent Scissors, on the subarachnoidal hemorrhage.

deviations for counting values of the function of the size of the gradient f_G. The next quaternion represents the same, just for the function f_Z. On the right-hand size is the setting of the corresponding weights for the convolution center. Hex of weight represents the size of the gradient of the responsible functions of the weights of the edges in the graph. As the last one possibility for setting, there is the setting of the maximal integrated value of the weight in the edge. All these setting are available just by uploading the image. After the confirmation by the button *Apply* the uploaded image is pictured in the main menu. Other types of settings are available through the menu *File→Properties*. They are available the whole time during the segmentation.

The first step of the segmentation is to choose the initial point. The user starts creating boundaries from this point. The user by moving the mouse and clicking on the mouse selects a new region, which will create a new part of the segmented boundary. After every click of the mouse, the user chooses a new point of the border, from which the shortest path in the graph is created. After closing the segmentation boundary, the image is divided into two regions. The user selects a region, which he considers as a final segment by clicking in the corresponding region. The user has a possibility to save the coordinates of pixels creating the boundary of the segmentation (*File→Save→Coordinates*), or the selected part of the image in format PNG by the menu *File→Save→Segment*.

The user can any time during the segmentation check the values of the weights on the edges by the menu *Show→Graph weights*. The user can close the application anytime by choosing the offer *File→Close*. Basic information about the program is in *Help→About*.

The program was used for handling and processing the original medical data. The images come from different parts of our medical faculty. In this section of the paper, we are applying our implemented segmentation, which is based on "Intelligent Scissors," on the biomedical data. Figures 5 – 7 illustrate the segmented results.

Conclusion

The method of "Intelligent Scissors" is like any other method, has its advantages and disadvantages. The major disadvantage is the computational time length of Dijkstra's algorithm. However, after the implementation of a more efficient priority queue, this problem could be alleviated. Another disadvantage is the need to know how the method works, or at least to know what the effect of altering specific parameters is. The main advantage of this approach, we see in training that enables the method to adapt to a particular kind of edge, which we need to segment. Another advantage is the number of settings that allow the user to adjust the method to a wide variety of different images. The user involvement has a significant influence on the process of segmentation, which enormously aids to achieve high-quality results. The greatest challenge and a great potential for the future, we see in the processing of biomedical images, where we need to seek and to distinguish local and global segmentation. This is of great use in the processing of biomedical data.

References

1. Bolovinou A, Pratikakis I, Perantonis S. Bag of spatio-visual words for context inference in scene classification. Pattern Recognition 2013; 46 (3): 1039–1053.

2. Cheng HD, Jiang XH, Sun Y, Wang Y. Color image segmentation: advances and prospects. Pattern Recognition 2001; 34 (12): 2259–2281.

3. Sonka M, Hlavac V, Boyle R. Image processing, analysis, and machine vision. Stamford: Cengage Learning, 2014: 1–930.

4. Peng B, Zhang L, Zhang D. A survey of graph theoretical approaches to image segmentation. Pattern Recognition 2013; 46 (3): 1020–1038.

5. Wang H, Zhang H, Ray N. Adaptive shape prior in graph cut image segmentation. Pattern Recognition 2013; 46 (5): 1409–1414.

6. Zhang YJ. A survey on evaluation methods for image segmentation. Pattern Recognition 1996; 29 (8): 1335–1346.

7. Tavakoli V, Amini AA. A survey of shaped-based registration and segmentation techniques for cardiac images. Computer Vision and Image Understanding 2013; 117 (9): 966–989.

8. Zhang D, Islam MM, Lu G. A review on automatic image annotation techniques. Pattern Recognition 2012; 45 (1): 346–362

9. Sridevi M, Mala C. A survey on monochrome image segmentation methods. Procedia Technology 2016; 6: 548–555.

10. Liu Y, Zhang D, Lu G, Ma WY. A survey of content-based image retrieval with high-level semantics. Pattern Recognition 2007; 40 (1): 262–282.

11. Magzhan K, Jani HM. A review and evaluations of shortest path algorithms. Internat J Sci Technol Res 2013; 2 (6). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.307.5792&rep=rep1&type=pdf.

12. Mortensen EN, Barrett WA. Interactive segmentation with intelligent scissors. Graphical models and image processing 1998; 60 (5): 349–384.

13. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. Cambridge: The MIT Press, 2009: 1–1312.

Received March 28, 2017.
Accepted May 2, 2017.