Accelerating the 3D Random Walker Image Segmentation Algorithm by Image Graph Reduction and GPU Computing

Jarosław Gocławski, Tomasz Węgliński, and Anna Fabijańska

Lodz University of Technology, Institute of Applied Computer Science,
18/22 Stefanowskiego Str., 90-924 Lodz, Poland
{jgoclaw,tweglinski,an_fab}@kis.p.lodz.pl

Abstract. In this paper the problem of image segmentation using the random walker algorithm was considered. When applied to the segmentation of 3D images the method requires an extreme amount of memory and time resources in order to represent the corresponding enormous image graph and to solve the resulting sparse linear system. Having in mind these limitations the optimization of the random walker approach is proposed. In particular, certain techniques for the graph size reduction and method parallelization are proposed. The results of applying the introduced improvements to the segmentation of 3D CT datasets are presented and discussed. The analysis of results shows that the modified method can be successfully applied to the segmentation of volumetric images and on a single PC provides results in a reasonable time.

1 Introduction

The main challenge of a recent image processing is the fast and efficient segmentation of large image datasets. This can be observed in particular in medical applications where a resolution of three dimensional CT and MRI body scans constantly increases.

Recently, a growing interest is attracted by an interactive graph based image segmentation algorithms such as the random walker [2]. The method has a lot advantages e.g. the ability to obey weak and discontinuous boundaries and the noise robustness. However, it is hardly ever used for segmentation of three dimensional medical data. In such a case, the usage of the method is strongly limited by the enormous size of the graph representing 3D image and the necessity of solving a huge sparse linear system. These hurdles result in the very long computation time and the usage of an extreme amount of memory resources.

Taking into account the above-mentioned limitations, this paper focuses on the problem of optimization of the random walker segmentation approach, both in terms of memory and time usage. In particular, certain techniques for the graph size reduction are proposed. Additionally, the partial parallelization of the algorithm using GPU processing and Compute Unified Device Architecture (CUDA) are introduced [6]. As a result, the method can be successfully applied

R.S. Choraś (ed.), *Image Processing & Communications Challenges 6*,
Advances in Intelligent Systems and Computing 313, DOI: 10.1007/978-3-319-10662-5_6

for the segmentation of volumetric medical images and provides results in a reasonable time.

2 The Idea of Random Walking for Image Segmentation

The Random Walker segmentation method [2] works on an image seen as a weighted undirected graph $G = (V, E)$ with vertices $v \in V$ corresponding to image pixels and edges $e \in E \subseteq V \times V$ spanning the pairs of vertices. Each edge $e_{ij} = (v_i, v_j)$ is assigned a non-negative weight w_{ij} related to the difference of grey level values in CT images. Six edges from any pixel p_i to its nearest neighbours are considered. The set of graph vertices is then divided into the pre-labelled nodes V_L and the unlabelled nodes V_U, where $V_M \cap V_U = \emptyset$ and $V_M \cup V_U = V$. In the Random Walker algorithm vertices from set V_U are labelled according to the probability that the random walker released from each unlabelled vertex will firstly reach a vertex labelled with a label s. Vertices are assigned labels for which the greatest probability x_U^s exists. The problem has an analytical solution in the form of a sparse, symmetric system of linear equations given in Eqn. (1)

$$\mathcal{L}_U \, x_U^s = -B f^s, \quad \mathcal{L} = \begin{bmatrix} \mathcal{L}_M & B \\ B^T & \mathcal{L}_U \end{bmatrix}, \quad \forall i, \sum_s x_i^s = 1, \tag{1}$$

where: $\mathcal{L}$ denotes the Laplacian matrix created from image graph weights, $\mathcal{L}_M$ and $\mathcal{L}_U$ are labelled and unlabelled Laplacian blocks respectively and x_U^s is the vector of probabilities, that a random walking started in each unlabelled pixel p_U will firstly reach the pixel pre-labelled by a label s.

$$f_i^s = \begin{cases} 1 & \text{if } l_i = s \\ 0 & \text{if } l_i \neq s \end{cases}, \tag{2}$$

where l_i is the label of pixel p_i. Finally the image pixel labelling takes the form:

$$\forall i \in U, \; l_i = \underset{s}{max}(x_i^s). \tag{3}$$

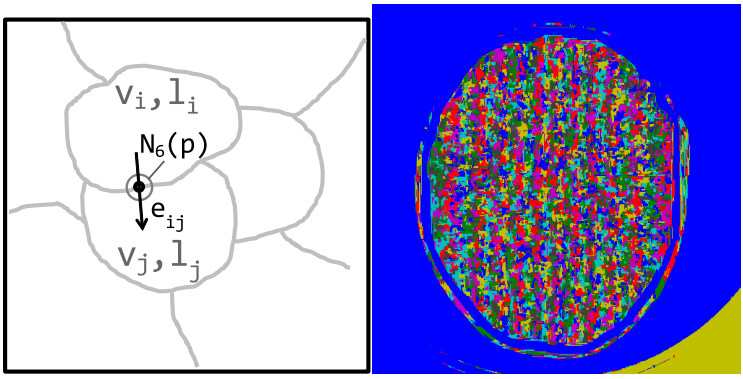
The Laplacian $\mathcal{L}_U$ size is usually huge, approximately equal to the square of the image graph order, assuming that relatively a few pixels can be labelled manually as seeds in the image space. Therefore $\mathcal{L}_U$ is stored in the host memory as a sparse matrix using the coordinate format defined in [3]. As a result, the execution of the Random Walker algorithm is very slow for a typical CT dataset.

3 Reduction of the Image Graph Size

The main idea of the introduced approach is to accelerate the classic Random Walker algorithm by the significant reduction of the number of nodes and edges in the image graph. This can be achieved by applying the idea of super-pixels

i.e. atomic, homogeneous and irregularly shaped regions within an image. The image graph is next built by assigning nodes to super-pixels rather than to single pixels and then by connecting the neighbouring super-pixels by edges. For creation of super-pixels the authors proposed an approach which uses the Minimum Spanning Tree (MST) pyramid algorithm [4].

The idea of super-pixels is explained in Fig. 1a. Super-pixels generated by the MST pyramid algorithm are shown (for an exemplary brain slice) in Fig. 1b.



(a) (b)

Fig. 1. The idea of super-pixels; (a) the illustration of an image graph with irregular regions and edges between region neighbours; p - edge pixel of the image graph vertex v_i; $N_B(p)$ - the neighbourhood of the pixel p; l_i, l_j - the labels of the example regions v_i, v_j; e_{ij} - the edge between v_i and v_j; (b) visualisation of super-pixels provided by MST pyramid algorithm

The introduced method for super-pixel creation in three-dimensional images is outlined in Algorithm 2.. The method creates a series of image region partitions, each of them using the parallel algorithm of MST building formulated by Boruvka [9]. At the k-th pyramid level the smallest weight edges $ME^{(k)}(i)$ are collected around the edge connected components $CC^{(k)}(i)$ in step 3. The edges to be contracted to the components are filtered in step 4 by Felzenschwalb-Huttenlocher comparison predicate [1]. For the purpose of image graph reduction the algorithm is additionally stopped at the pyramid level, when the mean count of super-pixels $CV^{(k)}$ exceeds the assumed limit value CV_{min}.

4 Image Graph Creation Based on Super-Pixels

To perform the Random Walker image segmentation efficiently, the image graph should be built at minimum time cost. Therefore, in the introduced approach the parallel computing is applied in order to determine coordinates of graph

Algorithm 2. MST pyramid segmentation for image graph reduction

Input: $G^{(0)}(CC^{(0)}, E)$ – image graph
Output: $G^{(0)}, G^{(1)}, \ldots, G^{(k)}$ – graphs at each level of the pyramid
1: k=0
▷ $CC^{(k)}(i)$ – i-th region of $G^{(k)}$
2: **repeat**
3: $ME^{(k)} \leftarrow$ the smallest edges around each $CC^{(k)}(i)$
4: $CE^{(k)} \leftarrow ME^{(k)}$ filtered by the Felzenszwalb predicate
5: $CV^{(k)} \leftarrow$ mean size of the regions $CC^{(k)}(i)$
6: $G^{(k+1)} \leftarrow \{G^{(k)} \cup CE^{(k)}\}$
7: $k \leftarrow k + 1$
8: **until** $CV^{(k)} > CV_{lim} \vee G^{(k)} = G^{(k-1)}$

edges and the corresponding edge weights. These are computed with regard to properties of super-pixels.

The procedure of image graph creation with regard to super-pixels is presented in Algorithm 3.. It requires a kernel counting of the predicted edges with the condition shown in step 2. To obtain the edge array E the count texture buffer is exclusively scanned using CUDA *thrust* template library [7].

Selected edges must be sorted by *thrust* function to eliminate possible duplicated items (step 4). After intensity normalisation, edge weights are transformed by exponential weighting function. For millions of super-pixels in CT images the times of data transfer between host an device memories can be approximately neglected.

Algorithm 3. The GPU based algorithm of building the reduced image graph from irregular super-pixels in three dimensions

Input: L – super-pixel label image, I – label grey levels
Output: E – super-pixel edges, W – super-pixel weights

1: copy L, I images to device memory
2: count & thrust scan boundary edges, while $L(p) < L(q),\ q \in N_6(p)$
3: $E \leftarrow$ store edge coordinates
4: thrust sort E, remove item duplications
5: normalise image grey-levels I
6: map exponentialy grey-levels to edge weights
7: copy edges and weights to host memory

5 Experimental Results

This section presents the results of applying the considered graph size reduction method to the Random Walker segmentation of example CT brain datasets. The tests were performed on a single PC computer equipped with a 6-core,

3.5 GHZ *Intel I7* processor and using *Windows 7 64 bit* operating system with 32 GB RAM memory. The program codes were included in C++ MEX files called from *MATLAB 2013* environment. GPU computing was performed using *Nvidia GeForce GTX Titan* graphic card with 6144 MB of device memory.

The proposed approach was tested on 10 CT brain datasets. They consisted of $115 - 336$ slices, 512×512 pixels each. Several example slices from these datasets are shown in Fig. 2. The corresponding results of image segmentation are marked with grey colour. Additionally, under each slice 3D visualization of the segmentation result is given.

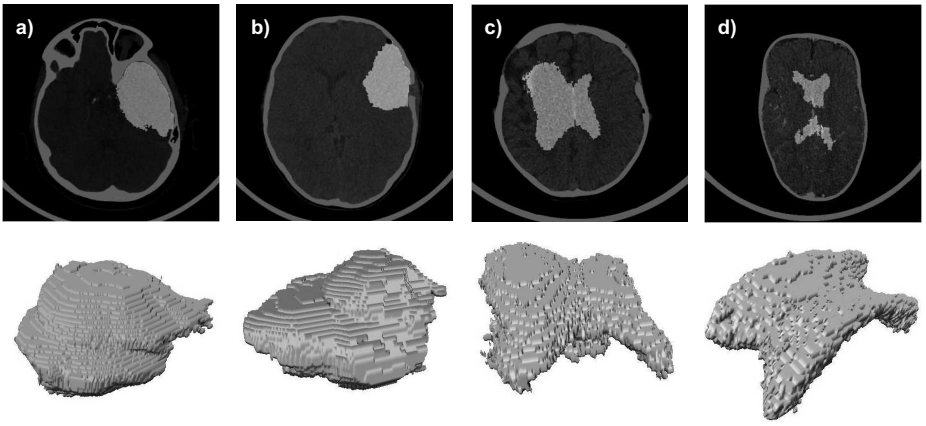


Fig. 2. Example slices of brain CT images with different lesions detected by the proposed Random Walker algorithm and the corresponding 3D visualization of the segmentation result. The diseased regions are exposed as grey regions.

The level of graph size reduction and resource usage by the introduced approach are summarized in Table 1. In particular, the case ID is given in the first column. The corresponding number of slices is given in the second column. This is followed by the time of method execution, which includes: time of image division into super-pixels (series: *Pyramid*), time of graph creation (series: *Graph*) and the total time of method execution including Random Walker segmentation (series: *Total*) given in columns number 3, 4 and 6 respectively. Two last columns consider memory usage by GPU (series: *Device*) and host RAM memory (series: *Host*).

The results presented in Table 1 show that the proposed approach significantly reduces the size of image graph. In particular, the size of the graph build upon super-pixels is on average equal to 1.25% of the original graph (i.e. the one build based upon single pixels). The graph size reduction is then on average $98,75\%$. The total time of algorithm execution vary from $17s$ to $65s$. This time is acceptable, especially taking into account that the running time of the original method is extremely long and sometimes it is impossible to run the original Random Walker on a single PC due to lack of host memory resources.

Table 1. Execution time and memory workload for the accelerated Random Walker segmentation algorithm. A common size of CT image slices: $512 \times 512 px$, Graph Reduct. - the percentage of graph size reduction

Case ID [-]	Num Slices [-]	Graph Reduct. [%]	Time Pyramid [sec.]	Time Graph [sec.]	Time Total [sec.]	Memory Device [MB]	Memory Host [MB]
1	115	98.51	15.65	0.28	20.28	1320	2172
2	336	98.89	46.64	0.48	64.65	2284	5880
3	172	98.72	22.27	0.33	31.50	1320	3376
4	200	98.80	26.61	0.36	39.77	1469	4053
5	110	98.91	14.23	0.27	17.35	1320	1870
6	207	98.60	27.60	0.36	44.00	1497	4474
7	200	98.80	27.04	0.36	40.13	1471	4133
8	204	98.63	27.12	0.37	41.16	1320	4315
9	156	98.90	20.35	0.32	26.38	1320	2700
10	267	98.78	33.95	0.41	53.35	1735	5362

Division of an image into super-pixels (by creation of MST pyramid) is an important step which takes on average $\approx 70\%$ of the total execution time. However, this step ensures a considerable reduction of the number of graph vertices and the resulting graph size.

In the proposed approach the Eqn. (1) was solved by calling the built-in $MATLAB$ solver for symmetric and sparse Laplacian matrix.

The comparison of the introduced approach (series: *Fast RW*) with the original Random Walker approach (series: *Raw RW*) is given in Table 2. The comparison considers both the usage of time and memory resources as well as the image segmentation accuracy. Because it was impossible to run the original Random Walker approach on the whole input CT datasets, the comparison was performed using the data subset of 10 consecutive slices from each dataset.

Results presented in Table 2 clearly show, that the execution time of the accelerated algorithm is on average 30 times shorter than in the case of the original Random Walker approach. Additionally, the host memory usage by the new approach exceeds merely 5.4% of the original method.

The F-score accuracy measure [5] listed in the last column of Table 2 fits in the range $0.87 \div 0.98$ (with an average value equal 0.93) indicating good compatibility between segmentation results provided by both regarded segmentation methods.

The accelerated method still uses the standard random walking parameter β applied to proper mapping of graph edge weights. In the described tests β was equal to 600. The new method also introduces an additional parameter $CV^{(k)}$ which limits regions size and stops the MST pyramid growth. The latter parameter was set experimentally to $CV^{(k)} = 1000\, px$. However, it should be underlined, that the algorithm behaviour is insensitive to small changes of this parameter.

Table 2. Resource usage and relative accuracy for the original and the accelerated Random Walker segmentation algorithm. CT image size: $512 \times 512 \times 10px$, F-score: accuracy measure,Graph Reduct.: the percentage of graph size reduction

Case ID [-]	Graph Reduct. [%]	Raw RW Time [sec.]	Raw RW Memory [MB]	Fast RW Time [sec.]	Fast RW Memory [MB]	F-score [-]
1	98.45	62.55	12916	2.04	725	0.98
2	98.37	71.70	13694	2.05	706	0.96
3	98.55	66.35	13471	1.96	719	0.94
4	98.43	65.40	13197	2.03	680	0.96
5	98.68	69.16	13884	2.06	733	0.94
6	98.43	74.32	13439	2.24	731	0.93
7	98.43	69.32	13608	1.98	735	0.97
8	98.32	71.29	13423	2.06	736	0.97
9	98.64	74.83	13940	1.95	742	0.92
10	98.95	67.96	13555	1.99	733	0.87

6 Conclusions

This paper considered the problem of acceleration and optimization of the Random Walker segmentation algorithm. The advantages of this method, like robustness to the weak or vanishing edges, predispose it to the segmentation of medical images. However, due to the very long execution time and the extreme memory workload for medical datasets (containing billions of pixels), the method cannot be directly applied for image processing.

Improvements proposed in this paper eliminate the main hurdles of the Random Walker segmentation approach. This goal is obtained by both: applying the idea of super-pixels for image graph creation and using GPU computing. The proposed algorithm reduces an image neighbourhood graph about 80 times for full size images and accelerates the segmentation above 30 times for the blocks of 10 slices. To make the full size comparison appropriate computer grid will be required. In the future also other pre-segmentations like parallelised region growing will be tested to improve the method acceleration.

The proposed improvements enable applying the method to segmentation of large three dimensional CT images using a single PC. Additionally, it provides results in a reasonable time. As a result, the number of possible applications of the Random Walker approach is increased.

Acknowledgements. This research was funded by the Ministry of Science and Higher Education of Poland from founds for science in years 2013-2015 in a framework of Iuventus Plus Programme (project no. IP 2012 011272).

References

1. Felzenschwalb, P., Huttenlocher, F., Efficient, D.P.: graph-based image segmentation. International Journal of Computer Vision 59, 1–25 (2004)
2. Grady, L.: Random Walks for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(11), 1–17 (2006)
3. Intel Sparse matrix storage formats (2013), http://software.intel.com/sites/products/documentation/hpc/mkl/mklman/GUID-9FCEB1C4-670D-4738-81D2-F378013412B0.htm
4. Ion, A., Kropatsch, W.G., van Haxhimusa, Y.: Considerations Regarding the Minimum Spanning Tree Pyramid Segmentation Method. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR&SPR 2006. LNCS, vol. 4109, pp. 182–190. Springer, Heidelberg (2006)
5. Labatut, V., Cherifi, H.: Accuracy Measures for the Comparison of Classifiers (2012), http://arxiv.org/ftp/arxiv/papers/1207/1207.3790.pdf
6. Nvidia Corporation Cuda Toolkit Documentation (2013), http://docs.nvidia.com/cuda/index.html
7. Nvidia Corporation Thrust Quick Start Guide (2013), http://docs.nvidia.com/cuda/pdf/Thrust_Quick_Start_Guide.pdf
8. Pratt, W.K.: Digital Image Processing, 4th edn. John Wiley & Sons, Inc., Los Altos (2007)
9. Vineet, V., Harish, P., Suryakant, P.: Fast Minimum Spanning Tree for Large Graphs on the GPU. In: Proceedings of the Conference on High Performance Graphics, pp. 167–171 (2009)