Hybrid parallelization of a seeded region growing segmentation of brain images for a GPU cluster

Anna M. Westhoff*†
*Simulation Lab Neuroscience - Bernstein Facility for Simulation and Database Technology
Institute for Advanced Simulation
Jülich Aachen Research Alliance
Forschungszentrum Jülich
Jülich, Germany
†Postal address: Forschungszentrum Jülich GmbH
Jülich Supercomputing Centre
52425 Jülich, Germany
Email: a.westhoff@fz-juelich.de

***Abstract*—The introduction of novel imaging technologies always carries new challenges regarding the processing of the captured images. Polarized Light Imaging (PLI) is such a new technique. It enables the mapping of single nerve fibers in postmortem human brains in unprecedented detail. Due to the very high resolution at sub-millimeter scale, an immense amount of image data has to be reconstructed three-dimensionally before it can be analyzed. Some of the steps in the reconstruction pipeline require a previous segmentation of the large images. This task of image processing creates black-and-white masks indicating the object and background pixels of the original images. It has turned out that a seeded region growing approach achieves segmentation masks of the desired quality. To be able to process the immense number of images acquired with PLI, the region growing has to be parallelized for a supercomputer. However, the choice of the seeds has to be automated in order to enable a parallel execution. A hybrid parallelization has been applied to the automated seeded region growing to exploit the architecture of a GPU cluster. The hybridity consists of an MPI parallelization and the execution of some well-chosen, data-parallel subtasks on GPUs. This approach achieves a linear speedup behavior so that the runtime can be reduced to a reasonable amount.**

I. Introduction

To understand the function of the human brain, it is necessary to study also its structure. An evolving imaging technique is Polarized Light Imaging [1], [2], [3]. It is applied to sections (slices) of postmortem human brain tissue and allows to analyze the course of nerve fibers between different brain regions at sub-millimeter scale.

Before the PLI data can be analyzed, some image processing steps have to be applied beforehand. The major step is a three-dimensional reconstruction of the stack of sections. An important prerequisite of this task is a prior segmentation of the images identifying the brain and non-brain regions. The main challenge of segmenting the PLI images is their immense number and the size of each image as there are terabytes of data per human brain which makes a parallel approach indispensable.

Since segmentation is a task of image processing which is required by a lot of use cases, a bunch of different approaches already exists such as thresholding, (seeded) region growing, neural networks, level sets, classification based methods or graph cuts. Although some parallelizations of segmentation approaches have been published, e.g. [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], they cannot be adopted without modifications for the PLI data because all algorithms are optimized for the present image characteristics like color mode, contrast or textures. A large variety of algorithm classes is also observable from region growing and level sets to neural networks, graph based algorithms and random walks. Some of them suffer from over- or under-segmentation in these cases where it is a typical problem of the algorithms.

The authors use different parallelization approaches based on shared or distributed memory, some also use GPU(s). Most approaches distribute parts of an image between the calculation units, so the threads or tasks. In the publications [12] and [13], the solver is parallelized instead, i.e. the algorithm as such. Depending on the chosen type of parallelization, the achieved speedups vary from a weak improvement compared to the sequential implementation up to a linear speedup behavior.

Since none of the mentioned approaches can be used for the PLI data without further enhancements and the variety on all levels of yet parallelized algorithms seems to be large, another procedure has been chosen to find an appropriate, parallelizable segmentation algorithm. Existing sequential segmentation tools have been applied to a representative small subset of the PLI images. The one achieving the best results, thus the best black-and-white masks, has then been adopted and parallelized.

This paper focuses on the hybrid parallelization of a segmentation in form of a seeded region growing. It describes the parallelization for the GPU cluster JUDGE (**Ju**elich **D**edicated **G**PU **E**nvironment) hosted by Jülich Supercomputing Centre (JSC), Forschungszentrum Jülich. Since a parallelization does not make sense without a previous automation of the choice of seeds, this aspect is also briefly discussed.

II. Material & Methods

A. Human Brain Data

The human brain tissue measured with PLI is from body donor programs of German universities. The measurement is

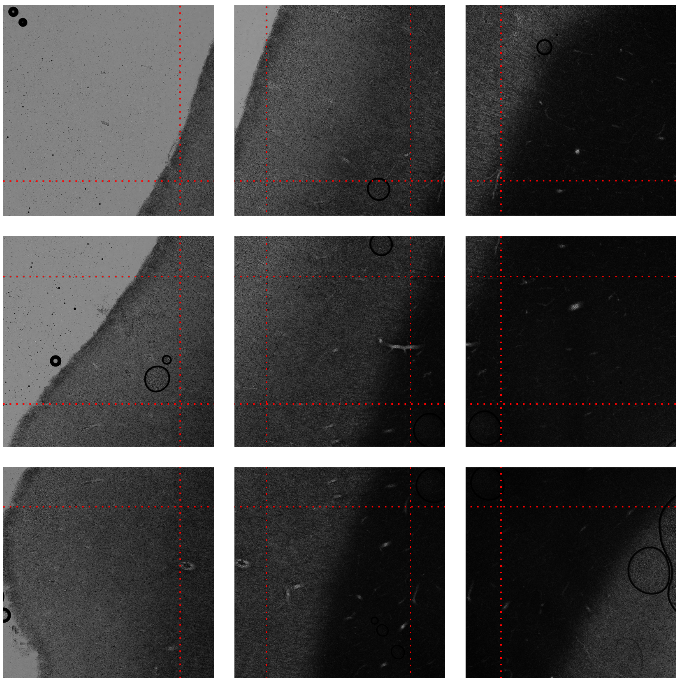


Fig. 1. These PLI image tiles demonstrate the overlapping of neighboring tiles. About 30% of the image information of a tile is also contained in the neighboring ones.

done according to [3] using a Polarizing Microscope developed by Taorad GmbH, Aachen, Germany. This technique takes advantage of the myelin sheaths that envelope the axons of nerve fibers. Myelin exhibits the optical property referred to as birefringence. This reaction to the incident polarized light is used to extract the nerve fiber orientations.

To image a post-mortem brain with PLI, it is cut into sections with a thickness of $70\mu m$. A section is imaged 18 times under linearly polarized light whereby the orientation of the polarized light is rotated by ten degree for every shot. In order to reach a high resolution, a section is not captured once for each light configuration but several times. This way a mosaic of image tiles is captured for each section. The tiles have a resolution of $1.6\mu m \times 1.6\mu m$ per pixel and an image size of 2048×2048 pixel. Neighboring tiles are imaged with a large overlapping as it is illustrated in figure 1. The outer edges of a section are only included in one tile but these edges do not show brain tissue. In total, there are at minimum 1500 sections per human brain, each consisting of about 25×30 tiles. This results in at minimum $1500 \cdot 25 \cdot 30 = 1,125,000$ images per human brain that have to be segmented.

B. Basic Seeded Region Growing Algorithm

The seeded region growing algorithm presented in [14] produces good and reasonable masks if applied to PLI images. The quality of this algorithm can also be guessed since it has been re-used in several other cases like [15], [16]. The algorithm is known as the first published region growing using seeds which has a significant effect concerning the quality of the segmentation mask because the number of classes in the mask can be directly controlled. In this way over- and under-segmentation can be avoided.

The algorithm is originally designed for intensity, i.e. gray-value, images. It divides the image into n classes $A_1, \ldots, A_n$ which contain the seeds in the beginning. Afterward, the algorithm assigns all remaining pixels iteratively to one of the $A_i, i \in [1, n]$. In each iteration, one pixel is assigned to one of the sets A_i. Therefore, the set T of all non-labeled pixels which directly border on the already labeled regions is defined as in equation (1). It contains all pixels which have not yet been added to any of the A_i but which are directly adjacent to one of the A_i. $N(x)$ is the set of all direct neighbors of a pixel at the one-dimensional position x.

$$T = \left\{ x \notin \bigcup_{i=1}^{n} A_i \mid N(x) \cap \bigcup_{i=1}^{n} A_i \neq \varnothing \right\} \tag{1}$$

If pixel $x \in T$ borders on exactly one of the A_i, let $i(x)$ be the index for which $N(x) \cap A_{i(x)} \neq \varnothing$. Otherwise, if x is neighbor of two or more of the A_i, $i(x)$ is defined as the index for which $N(x) \cap A_{i(x)} \neq \varnothing$ and a measure $\delta(x)$ is minimized. This measure $\delta(x)$ defines how similar a pixel x is compared to the region it adjoins. It may be defined as follows with $g(x)$ being the gray value of pixel x:

$$\delta(x) = \left| g(x) - \underset{y \in A_{i(x)}}{\text{mean}} \left[g(y) \right] \right| \tag{2}$$

Alternatively, pixels adjacent to two or more of the A_i can be assigned to a new set B containing all boundary pixels. In the end of each iteration, the pixel $z \in T$ with

$$\delta(z) = \min_{x \in T} \{\delta(x)\} \tag{3}$$

is labeled corresponding to $A_{i(z)}$ and appended to this set. The definitions (2) and (3) assure that the regions A_i are as homogeneous as possible and, by the use of $N(x)$, that each of the regions is connected.

C. Automating the Choice of Seeds

The first major challenge in the development process of a segmentation for microscopic images acquired with PLI was the adoption to the immense number of images. The choice of the seeds had to be brought into focus. A definition by hand results in an unacceptable effort because at least one seed for each of the hundreds of thousands of images per brain has to be chosen. Hence, an automation of this step was needed to get away from an interactive processing of the images and thus enable a parallelization. This step has been done based on high level knowledge of the images.

It has been decided to use the so-called transmittance of the 18 different shots of the same brain region for the segmentation because this modality allows a clear distinction between brain tissue and background. Automating the choice of seeds step for the PLI data, it had to be kept in mind that there are tiles showing both brain tissue and background but that some only show one of the two classes as it is visible in figure 2. It is observable that the brain is in principle brighter than the rest of the image. Therefore it was possible to utilize an intensity histogram based choice of seeds. Taking the different image contents into account, it was not reasonable to use separate histograms of the different tiles because they might not contain information about both brain and background

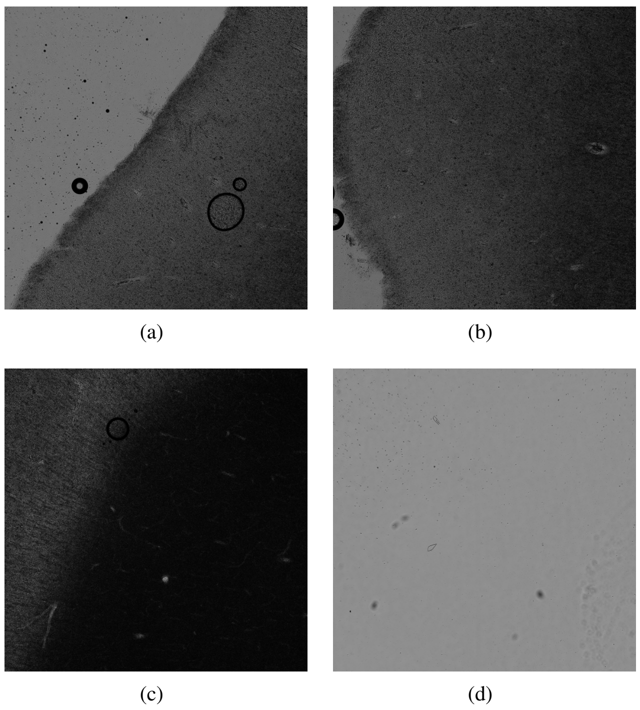


Fig. 2. These four images are representative examples for PLI microscopic tiles. 2(a) and 2(b) show both the dark brain with a large intensity variance on a brighter background. 2(c) contains only brain tissue and 2(d) only background.

intensities. Instead, the joint histogram of all available mosaic tiles of all sections has been calculated.

The user has to define a rough threshold differentiating between brain and background intensities based on the joint intensity histogram. The effort of this manual step is independent of the number of images to be processed, so setting one value per brain. So the segmentation has been split into two independent tools, the first one calculates the joint intensity histogram, the second one performs the region growing as follows. In between, the manual choice of the threshold takes place.

The user-defined threshold divides the intensities into two intervals. For both intervals within the histogram, the median intensity $q_{0.5}$ and the quantiles q_α and $q_{1-\alpha}$ have been calculated. Based on this information, a measure m_{cand} has been calculated that defines how well suited a pixel is to be used as a seed for the respective class. (x, y) denotes the two-dimensional coordinates of a pixel and $g\,(x, y)$ the intensity of this pixel.

$$m_{cand}(x,y) = \begin{cases} \frac{g(x,y)-q_{0.5}}{q_{0.5}-q_\alpha}, & g(x,y) \leq q_{0.5} \\ \frac{q_{0.5}-g(x,y)}{q_{1-\alpha}-q_{0.5}}, & g(x,y) > q_{0.5} \end{cases} \tag{4}$$

$$= \max\left(\frac{g(x,y)-q_{0.5}}{q_{0.5}-q_\alpha}, \frac{q_{0.5}-g(x,y)}{q_{1-\alpha}-q_{0.5}}\right) \tag{5}$$

This measure can be computed independently for each pixel and is stored in a measure image. With this definition, every pixel is a candidate seed given that $m_{cand}(x, y) \leq 1$. The candidates with the smallest values are the comparably best seeds. Pixel with intensities similar to the threshold are not seeds for any of the regions because the intensity ranges of brain and background overlap.

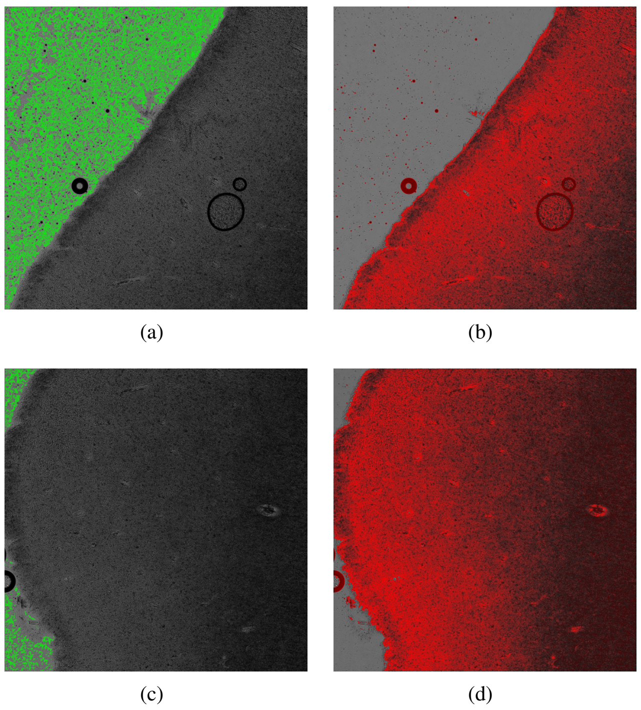


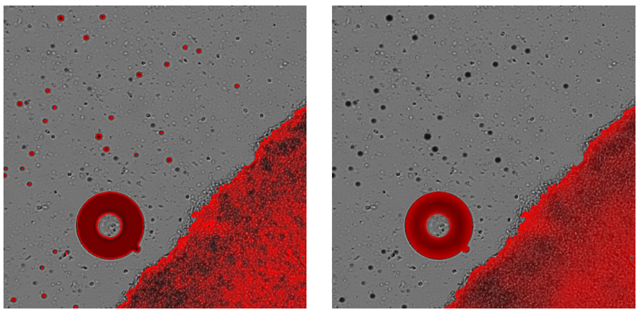
Fig. 3. These images demonstrate the measure m_{cand} for two of the four examples. The original images are visible in the background. The measure values are illustrated as overlays; the background seeds are green, brain seeds are red. Green resp. red pixels correspond to measure value 0, i.e. reliable candidate seeds. For measure values in $]0, 1]$, a gradient from green resp. red to transparent is used. All pixels with measure values larger than 1 have a transparent color in the overlay.

In fact, there are two measure images, one based on the brain and one based on the background intensity interval. If candidate seeds would be chosen using these measures m_{cand}, also noise pixels were seed candidates. Image noise denotes pixels with a variation in intensity or color compared to the neighboring ones that bears no reference to the captured object and which mostly appears as isolated mis-colored pixels. Therefore, also spatial information has been included in the definition of the measure δ in form of a linear smoothing filter.

The calculation of the initial measure and the following smoothing operation could be combined to a single discrete convolution, i.e. a linear image filter. The final seed measure images m_{final} have been computed using convolution equation (6).

$$m_{final}(x,y) = w\,(x,y) * m_{cand}\,(x,y)$$
$$= \sum_{i=-m}^{m} \sum_{k=-n}^{n} w\,(i,k) \cdot m_{cand}\,(x+i, y+k) \tag{6}$$

It turned out that a weighted averaging smoothing filter w works best for the PLI tiles, if for a central value p all other entries have the same value $\frac{1-p}{(2m+1)\cdot(2n+1)-1}$. The central weight has been chosen to $p = 0.1$ and $m = n = 4$. Figure 3 illustrates the final measure images m_{final} for brain and



(a) (b)

Fig. 4. These images demonstrate the elimination of small isles of brain seeds. The side effect of this step is a smoothing of the remaining measure values.

background seeds of the example tiles showing both brain and background.

In this figure, it is observable that some of the dirt particles in the background region of the image tiles are erroneously labeled as seeds for the brain region. Since the dirt particles result in isles of seeds that are much smaller than any brain region, they could be erased out of m_{final}:

For all brain seeds, the number N of brain seeds within a square neighbor region with a size of $(\text{radius} \cdot 2 + 1)^2$ has been determined. Furthermore, the sum S of the inverted measure values of these neighboring seeds has been computed. ($\mathbb{1}$ denotes the indicator function.)

$$\begin{aligned} S(x,y) = \sum_{i,k=-\text{radius}}^{\text{radius}} &\{(1.0 - m_{final}(x+i, y+k)) \\ &\cdot \mathbb{1}_{\{x \leq 1\}}(m_{final}(x+i, y+k))\} \end{aligned} \tag{7}$$

Using N and S as in equation (8), seeds are rated higher, i.e. they are more reliable, if they have other comparably good seeds in their surroundings. Otherwise, they are downgraded as in case of the seeds isles caused by the dirt particles in the background. The effect of the elimination of small brain seed isles is illustrated in figure 4.

$$\hat{m}_{final}(x,y) = \begin{cases} m_1, & \frac{N}{(\text{radius}\cdot 2+1)^2} \geq \text{threshold} \\ m_2, & \frac{N}{(\text{radius}\cdot 2+1)^2} < \text{threshold} \end{cases}, \tag{8}$$

$$m_1 = 1.0 - \frac{S(x,y)}{(\text{radius} \cdot 2 + 1)^2},$$

$$m_2 = 1.0 + (1.0 - \text{threshold}) \cdot \frac{S(x,y)}{(\text{radius} \cdot 2 + 1)^2}$$

A deeper explanation and analysis of the automated choice of seeds can be found in the Master's thesis [17].

D. Parallelization for a GPU cluster

As it has been mentioned above, the enormous number of PLI microscopic tiles can only be handled if the segmentation

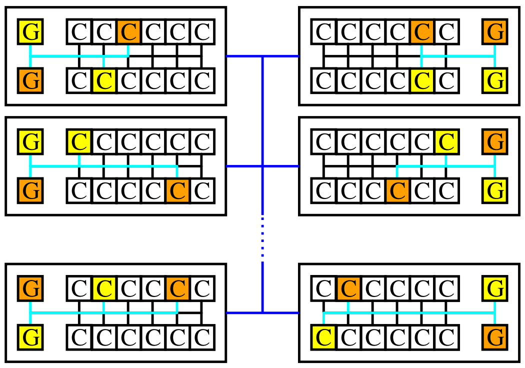


Fig. 5. Schematic illustration of the architecture of JUDGE. Each node consists of two NVIDIA GPUs and two 6-core Intel Xeon processors. For the present application, two pairs of CPUs and GPUs (marked in yellow and orange) are used per node. This one-to-one assignment is used because there are some parts which are not ported to the GPUs but which cause a significant load on the CPUs. So the most suitable assignment is one GPU to one CPU. The CPUs may communicate using the network (blue lines). The remaining ten CPUs stay idle and may be used by other applications.

software is parallelized. JSC hosts among other supercomputers the GPU cluster JUDGE. Each of its 206 compute nodes consists of two Intel Xeon X5650 (Westmere) six-core processors and two NVIDIA GPUs as illustrated in figure 5. 54 of the nodes contain Tesla M2050 (Fermi) GPUs, the others Tesla M2070 (Fermi). These two GPUs only differ regarding the available memory which is with 3GB respectively 6GB sufficient for the present application. There are 96GB of main memory per node available. The network is an InfiniBand QDR HBA.

To exploit this architecture, the parallelization has been done on two levels. A multi-core approach using distributed memory has been combined with a transfer of some algorithmic steps on the GPUs. The two levels can be used in combination or separately so that the software can be ported to supercomputers with a different architecture without much effort.

1) Multi-core Approach: Since neighboring tiles of a section are captured with a sufficiently large overlapping, it is possible to process all tiles independently of the others because all required information is already present in the respective tile.

To port the seeded region growing segmentation to a multi-core platform, the tiles of a section have been cyclically distributed between all available processes. This distribution has been used for the computation of the joint histogram as well as for the seeded region growing. In case of the joint histogram, every process first calculates the joint histogram of the tiles assigned to it. Thus, each process has got a part of the global joint histogram that covers the whole intensity range but only a part of the tiles. Afterward, these distributed histograms are collected by a master process using the Message Passing Interface (MPI) and combined to the global joint histogram using the `MPI_Reduce` method with `MPI_SUM` as the reduction operation.

The seeded region growing of a tile can be done completely independent of the other tiles so that no MPI communication is required. This is possible because the overlapping region of

TABLE I. PROPORTION OF RUNTIME REQUIRED BY THE DIFFERENT STEPS OF THE SEGMENTATION.

Step of the Algorithm	**Proportion of Runtime**
Calculation of the measure image m_{final}	48.46%
Elimination of the seed isles out of m_{final}	48.21%
Labeling of the seeds in the masks	0.11%
Seeded region growing	3.21%

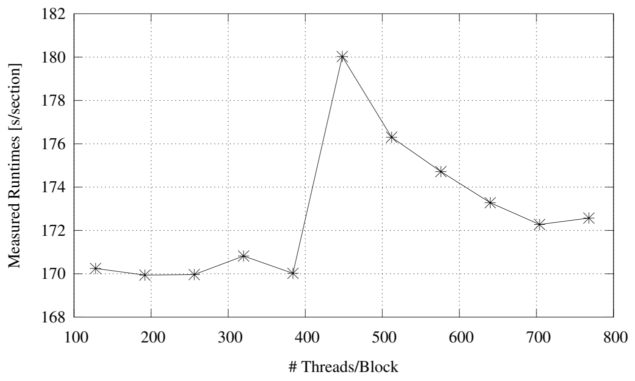


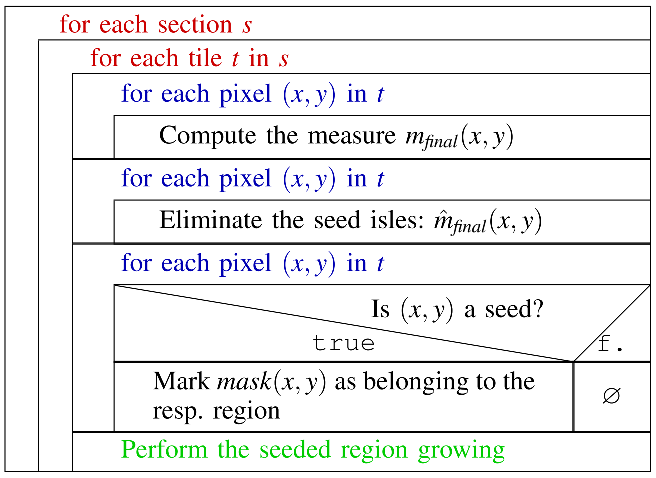
Fig. 6. The runtime of all GPU-parallelized steps of the seeded region growing executing the tool with thread block sizes from 128 to 768 in steps of 64. The shortest runtime is reached using block sizes of 192, 256 and 384 threads. The jump from 384 to 448 threads can be explained by analyzing the achieved occupancy of the device. For up to 384 threads per block, four blocks can be executed simultaneously on the same multiprocessor of the GPU. For 448 and more threads, only three blocks are possible. This reduces the achieved occupancy of the device drastically, i.e. the runtime is increased. For larger numbers of threads per block, the enlarging of the block size fills the multiprocessor again with threads so that the occupancy grows.

two neighboring tiles is much larger than all used neighborhood sizes. Furthermore, the seeded region growing algorithm is stable in respect of the choice of seeds, i.e. a moderate change of the seeds does not result in a different segmentation mask.

2) GPU Approach: For the second level of parallelism, the required runtime per section of the different region growing steps has been analyzed in a sequential execution. The measured proportions of runtime of the different main steps on the total runtime are listed in table I. The two steps required to obtain the seeds for the region growing consume together 96.67% of the total runtime, i.e. the calculation of m_{final} and the elimination of seed isles. Therefore, it has been worthwhile to revise these steps. The runtime of the calculation of the joint intensity histogram does not have to be considered with respect to a GPU parallelization because it consumes only 0.6% of the runtime per section compared to the in table I presented joint runtime of seed determination and region growing.

An analysis of the algorithms of these two steps revealed that both are data parallel operations, i.e. the operations can be computed independently for each pixel. This can be directly extracted from the definitions (6) and (8) which refer only to a single pixel (x, y). Nevertheless, information of a defined amount of neighboring pixels is required to compute these measures.

GPUs are processors that can exploit data parallelism since they have thousands of parallel processing units designed to execute the same instruction on thousands of pixels in the same processor cycle. So they are fitting to the two steps of the



for each section s
for each tile t in s
for each pixel (x, y) in t
Compute the measure $m_{final}(x, y)$
for each pixel (x, y) in t
Eliminate the seed isles: $\hat{m}_{final}(x, y)$
for each pixel (x, y) in t
Is (x, y) a seed?
true
f.
Mark $mask(x, y)$ as belonging to the resp. region
$\varnothing$
Perform the seeded region growing

Fig. 7. This figure demonstrates how the two levels of the hybrid parallelization of the seeded region growing work together. The red loops are parallelized using the multi-core approach with MPI, the blue ones are executed on the GPUs using CUDA.

algorithm mentioned above. Thus, for each of the available CPUs, an additional GPU has been used so that a one-to-one assignment has been achieved as it is illustrated in figure 5 by the yellow and orange colored processing units.

For a re-implementation of these two steps on a GPU, the framework CUDA, version 4.1, has been used. Since the pixels of the images can be processed independently, a one-to-one assignment of pixels to CUDA threads has been applied. It has been decided to use one-dimensional thread blocks in form of rows because this shape exploits the data locality of the operations best: The images are stored in C order, so row-wise. For the processing of two neighboring pixels, the required other pixels are nearly the same which implies this one-dimensional shape for a thread block.

In order to find out the optimal size of the thread blocks, the region growing has been executed with thread block sizes varying from 128 to 768 in steps of 64 threads per block. The results are illustrated in figure 6. It figured out that the optimal thread block size is at 256 threads per block and thus an amount of 16384 thread blocks for the standard tile size of PLI.

3) Hybrid Parallelization: Figure 7 demonstrates how the two levels of the hybrid parallelization are combined. The seeded region growing contains outer loops iterating over all available image tiles, i.e. over all sections and all tiles within a section. These loops marked with the red color have been parallelized in the multi-core approach. Using the section number and the tile coordinates within the section, a consecutive index for every tile has been computed. Iterating over this index *idx* using P processes, a process P_i gets every Pth tile.

Inside these outer loops iterating over all indexes, the seeded region growing is performed tile by tile. This segmentation starts with three blue marked loops iterating over all pixels of the respective tile. Inside these loops, the final measure m_{final} is computed, the brain seed isles are eliminated and the obtained seeds are marked in the segmentation mask. Since all three operations are data-parallel, they have been ported to the

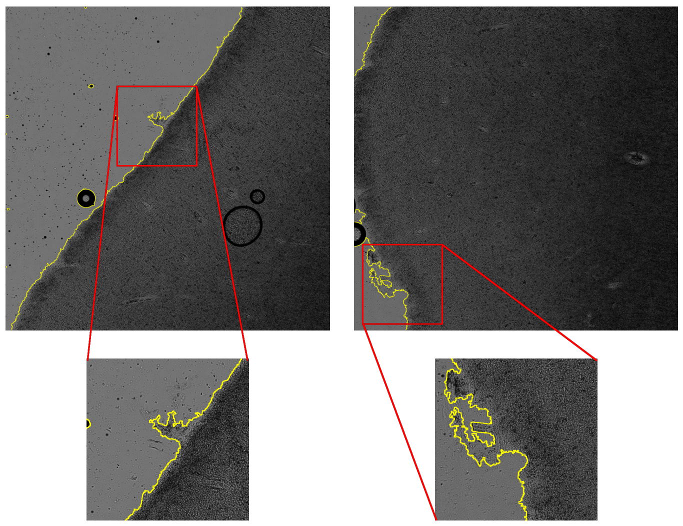


Fig. 8. The segmentation result is illustrated as edges between the classes within the masks. The detail images demonstrate that the edges follow the real outer edges of the brain tissue. The extracted regions are connected and do not contain holes.

GPU(s) using CUDA.

This schematic picture of the algorithm structure demonstrates not only how the two levels of the hybrid parallelization can be used in combination, but also that they can be omitted independent of each other. It has to be noted that the only non-parallelized part of the algorithm is the green marked line which is the seeded region growing.

III. Results

A. Segmentation Masks

The first important step to evaluate the results of the developed region growing is to have a closer look at the segmentation masks. If the masks are considered separately from the original images, it is difficult to see if the extracted mask regions fit the actual regions. A good alternative is to display the original image together with the edges between the mask regions drawn in as thin lines as it has been done for figure 8.

In this illustration, it is visible that the brain tissue is identified very well by the presented enhancement of the seeded region growing. The segmentation masks present the desired quality which is only possible because the developed automated choice of seeds works correctly. In particular, no seed for one of the regions is placed into the other region. The dirt particles in the background are ignored during the seed determination. The black rings visible in figure 8 are air bubbles which occur during the preparation of the brain tissue. They are always added to the brain region since they may occur in both the brain and the background region of the image and this assignment is important for other steps in the reconstruction pipeline.

B. Runtime and Speedup

To evaluate the hybrid parallelization of the seeded region growing, the runtime respectively speedup behavior of both

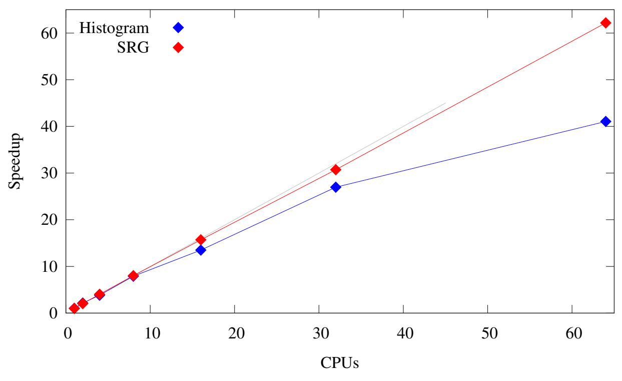


Fig. 9. These speedup curves per section have been measured on JUDGE with up to 64 processes. The blue curve belongs to the calculation of the joint intensity histogram of all image tiles. It flattens out as a result of the collection of the distributed computed histograms. The red curve shows the speedup behavior of the region growing part of the segmentation. Since there is no inter-process communication needed, an optimal, linear speedup is reached.

TABLE II. Proportion of runtime required by the different steps of the segmentation with and without the GPU parallelization.

Step of the Algorithm	**Proportion of Runtime**	
	sequential	***GPU parallel***
Calculation of the measure image m_{final}	48.46%	9.87%
Elimination of the seed isles out of m_{final}	48.21%	5.58%
Labeling of the seeds in the masks	0.11%	3.31%
Seeded region growing	3.21%	81.24%

levels has to be analyzed. Figure 9 shows the speedup curves of the multi-core approach. In particular, there is a speedup curve of the calculation of the joint histogram and another one of the region growing itself. These two parts of the segmentation tool are interrupted by the user input of the intensity threshold.

The speedup curve of the calculation of the joint intensity histogram is linear for small numbers of processes but flattens out for larger ones. This is justifiable by the increasing effort for the inter-process communication to collect the distributed histograms. In case of the seeded region growing, an optimal and thus linear speedup is reached since no communication is needed.

To evaluate the effect of the GPU parallelization approach, the runtime required by the different steps of the algorithm has once again been measured as it is visible in table II. The summed up proportions of runtime of the seed determination steps is reduced from 96.67% to 15.45% using the CUDA version instead of the sequential one. This results in an increase of the proportion of the (not modified) region growing from 3.21% to 81.24% so that the runtime behavior is dominated by the effort for the actual growing process. This is also the reason for the one-to-one assignment of GPUs and CPUs. Figure 10 illustrates the influence of the CUDA parallelization on the runtime behavior of the segmentation.

We have executed the segmentation using on the one hand up to 64 CPUs and on the other hand up to 64 pairs of CPUs and GPUs. The total runtime per section of the region growing has been measured. It is observable that the trend of the runtime curve is for both cases the same. The additional use of a GPU per process results in a shift down of the runtime curve, i.e. the region growing is accelerated by a factor of about 20.

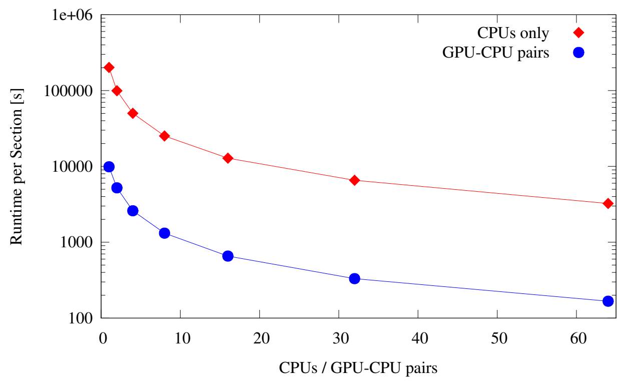


Fig. 10. This diagram compares the scaling behavior of the multi-core approach with and without the additional use of the GPU parallelization, i.e. up to 64 processes respectively 64 pairs of CPUs and GPUs are used. It is observable that the trend of the runtime per section is the same, the use of CUDA results in a shift down of the original curve. The segmentation is accelerated by a factor of 20 due to the additional use of a GPU per process.

IV. Discussion

This paper focuses on the development and hybrid parallelization of a segmentation for PLI image tiles. The additionally presented automation of the choice of seeds was necessary because of the immense number of images to be processed in case of PLI and to make a parallel execution possible. The achieved level of automation is sufficiently high since only a single value, i.e. the threshold within the intensity histogram, has to be defined manually to segment an entire human brain. The developed algorithm works well for PLI but is no solution for every possible use case because it is adopted to the characteristics of the present data. Of course, it is possible to define an analog similarity measure δ using different image characteristics for other use cases. In this sense, the developed automated choice of seeds is generally applicable if the image characteristics are well analyzed and the definition of the measure is adopted to them.

The presented algorithm provides good results for PLI as demonstrated in figure 8 given that the images are sufficiently homogeneous. The variances of the section thickness and of the illumination must not be too high since both would result in a distortion of the joint intensity histogram. The extracted edges between the brain and non-brain regions provide an exactness which is adequate for the processing in the subsequent reconstruction steps.

Also the parallelization of the region growing segmentation is worth discussing. Two comparably simple levels of parallelism have been applied to the seeded region growing. The tiles are distributed among the available processes on the first level. For every process on a CPU, an additional GPU is used to compute the data-parallel parts of the algorithm as a second level of parallelism. Both levels can be used separately or in combination. The use of one level does not influence the scaling behavior of the other one, i.e. the additional use of GPUs does not weaken the linear speedup of the multi-core approach. The other way round, distributing the tiles to different CPU-GPU pairs instead of using only one does not have any influence on the acceleration achieved by the use of the GPU. This independence of the two levels of the hybrid parallelization allows an easy porting from the GPU cluster JUDGE to another supercomputer or cluster. It is also possible to use the tool for small parts of a brain on a normal workstation. Nevertheless, it is important to adjust parameters like the thread block size to the respective available hardware.

The linear scaling behavior of the multi-core approach allows to process a given amount of data as fast as required given that enough processes are available in parallel. Let us assume that it would take about 295 days to process a whole human brain using the sequential variant of the algorithm. The additional use of a GPU reduces the runtime to 15 days. In combination with the multi-core implementation using 64 pairs of CPUs and GPUs, it is even reduced to 5.6 hours. This short example demonstrates how well the two parts of the hybrid parallelization complement each other. The multi-core level copes with the immense number of images, the GPU level addresses the large size of the images consisting of about 4,000,000 pixels per tile.

The optimal linear speedup of the multi-core approach is only achieved if an equal load distribution between the processes is provided. This does not only mean that every process has a work package containing the same number of images but also that the average runtime per tile and core is equally distributed. Images that show only one of the regions, i.e. brain tissue or background, are almost entirely covered with seeds. Since all seeds are directly applied to one of the regions in the mask, the number of remaining pixels for the region growing is much smaller compared to “mixed” images. So these one-region images are segmented much faster than mixed ones. If some processes obtain mainly the one-region images and other mainly mixed ones, the first group of processes will be finished much earlier than the second resulting in long idle parts of runtime. Therefore, it is indispensable to find a clever assignment of image tiles to processes to minimize the idle parts of runtime for every brain to be analyzed.

Other parallelizations of segmentations achieve a smaller acceleration per GPU than the presented factor 20, e.g. in [6] an acceleration by a factor of 4.9 to 6.8 depending on the GPU has been reached. In [7], a multithreaded OpenMP parallelization has been described resulting in a speedup of 2.6 for 4 processes, so an efficiency of 0.65. [5] has reached an almost linear speedup like our presented approach.

Since the growing process consumes more than 80% of the total runtime in the GPU parallelized variant, it should be reconsidered to parallelize this step of the algorithm on a third, additional level. Possible approaches might base on a division of each tile in sub-tiles as it is done in [7]-[11]. These sub-tiles may then be processed either using again a multi-core variant, so distributed memory, or a multi-threaded version with shared memory. Nevertheless, this step always includes an additional communication or synchronization effort so that the perfect linear speedup achieved by the presented multi-core approach will not be maintained. Thus, it would have to be evaluated if the runtime gain due to the new level of parallelism is compensated by the induced communication or synchronization effort.

Another approach for a parallelization of the region growing would be a GPU parallel version of this step since this is the only one which is still computed on the CPU(s). Indeed, this step bases on an intrinsically sequential algorithm. There have been trials for a GPU parallel region growing like [6] but the achieved speedups are not as high as what can be reached in case of other algorithms.

All in all, a further parallelization of the growing process has to be kept in mind but it has to be weighed out if the additional effort is reasonable since with the already implemented two levels of parallelization, the required amount of runtime can be reduced to the desired range given that the needed infrastructure is available. In addition, the time needed to section and image a whole brain is much larger than the few hours or days required to segment the images, i.e. the segmentation is faster than the images are available for access. From this point of view, another enhancement of the parallelization is not required for the PLI data.

V. Conclusion

We presented a seeded region growing segmentation specialized for images of the human brain acquired with the technique of Polarized Light Imaging. The choice of seed pixels has been automated so that only a single point of manual interaction is required in order to process an entire human brain, i.e. a threshold within the joint intensity histogram of all images has to be set. The tool has been parallelized for the GPU cluster JUDGE achieving a linear speedup due to a multi-core approach, and another acceleration by a factor of 20 using GPUs in addition to the CPUs. Both levels of parallelism can be used in combination or as alternatives, depending on the available hardware. The segmentation masks are sufficiently accurate for our purposes and the required runtime is fast enough.

Acknowledgement

Partially funded by the Helmholtz Association through the Helmholtz Portfolio Theme "Supercomputing and Modeling for the Human Brain".

References

[1] M. Axer, K. Amunts, D. Gräßel, C. Palm, J. Dammers, H. Axer, U. Pietrzyk, and K. Zilles, "A novel approach to the human connectome: ultra-high resolution mapping of fiber tracts in the brain," *NeuroImage*, vol. 54, pp. 1091 – 1101, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S105381191001178X#

[2] C. Palm, M. Axer, D. Gräßel, J. Dammers, J. Lindemeyer, K. Zilles, U. Pietrzyk, and K. Amunts, "Towards ultra-high resolution fibre tract mapping of the human brain - registration of polarised light images and reorientation of fibre vectors," *Frontiers in Human Neuroscience*, vol. 4, pp. 1–16, 2010. [Online]. Available: http://www.frontiersin.org/human_neuroscience/10.3389/neuro.09.009.2010/abstract

[3] M. Axer, D. Gräßel, M. Kleiner, J. Dammers, T. Dickscheid, J. Reckfort, T. Hütz, B. Eiben, U. Pietrzyk, K. Zilles, and K. Amunts, "High-resolution fiber tract reconstruction in the human brain by means of three-dimensional polarized light imaging (3D-PLI)," *Frontiers in Neuroinformatics*, vol. 5, no. 34, 2011. [Online]. Available: http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2011.00034/abstract

[4] C.-L. Huang, "Parallel image segmentation using modified hopfield model," *Pattern Recognition Letters*, vol. 13, no. 5, pp. 345–353, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016786559290032U

[5] A. Khotanzad and A. Bouarfa, "Image segmentation by a parallel, non-parametric histogram based clustering algorithm," *Pattern Recognition*, vol. 23, no. 9, pp. 961–973, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/003132039090105T

[6] P. N. Happ, R. Q. Feitosa, C. Bentes, and R. Farias, "A parallel image segmentation algorithm on gpus," in *Proceedings of the 4th GEOBIA*, may 2012, pp. 580–585.

[7] P. N. Happ, R. S. Ferreira, C. Bentes, G. A. O. P. Costa, and R. Q. Feitosa, "Multiresolution segmentation: a parallel approach for high resolution image segmentation in multicore architectures," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVIII-4/C7, pp. 28–32, 2005.

[8] D. A. Bader, J. Jájá, D. Harwood, and L. S. Davis, "Parallel algorithms for image enhancement and segmentation by region growing, with an experimental study," *The Journal of Supercomputing*, vol. 10, no. 2, pp. 141–168, 1996. [Online]. Available: http://dx.doi.org/10.1007/BF00130707

[9] A. Hagan and Y. Zhao, "Parallel 3D Image Segmentation of Large Data Sets on a GPU Cluster," in *Advances in Visual Computing*, ser. Lecture Notes in Computer Science, G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnação, C. Silva, and D. Coming, Eds. Springer Berlin Heidelberg, 2009, vol. 5876, pp. 960–969. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10520-3_92

[10] Y. Zhuge, Y. Cao, J. K. Udupa, and R. W. Miller, "Parallel fuzzy connected image segmentation on GPU," *Medical Physics*, vol. 38, no. 7, pp. 4365–4371, 2011. [Online]. Available: http://link.aip.org/link/?MPH/38/4365/1

[11] A. N. Moga and M. Gabbouj, "Parallel Marker-Based Image Segmentation with Watershed Transformation," *Journal of Parallel and Distributed Computing*, vol. 51, no. 1, pp. 27 – 45, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731598914484

[12] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, "Random Walks for Interactive Organ Segmentation in Two and Three Dimensions: Implementation and Validation," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2005*, ser. Lecture Notes in Computer Science, J. Duncan and G. Gerig, Eds. Springer Berlin Heidelberg, 2005, vol. 3750, pp. 773–780. [Online]. Available: http://dx.doi.org/10.1007/11566489_95

[13] J. Wassenberg, W. Middelmann, and P. Sanders, "An Efficient Parallel Algorithm for Graph-Based Image Segmentation," in *Computer Analysis of Images and Patterns*, ser. Lecture Notes in Computer Science, X. Jiang and N. Petkov, Eds. Springer Berlin Heidelberg, 2009, vol. 5702, pp. 1003–1010. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03767-2_122

[14] R. Adams and L. Bischof, "Seeded Region Growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.

[15] O. Gómez, J. A. González, and E. F. Morales, "Image Segmentation Using Automatic Seeded Region Growing and Instance-Based Learning," in *Progress in Pattern Recognition, Image Analysis and Applications*, ser. Lecture Notes in Computer Science, L. Rueda, D. Mery, and J. Kittler, Eds. Springer Berlin Heidelberg, 2007, vol. 4756, pp. 192–201. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-76725-1_21

[16] I. Sanders, "Seeded region growing using multiple seed points," in *16th Annual Pattern Recognition Association of South Africa Annual Symposium*. PRASA, 2005, pp. 177–182.

[17] A. Westhoff, "GPU-accelerated Segmentation of high-resolution Human Brain Images acquired with Polarized Light Imaging," Jülich, 2013, FH Aachen-Jülich, Masterarbeit, 2013. [Online]. Available: http://juser.fz-juelich.de/record/138037