InstantTrace: Fast Parallel Neuron Tracing On GPU

Yuxuan Hou
houyuxuan@zju.edu.cn

Zhong Ren*
renzhong@zju.edu.cn

Qiming Hou
hqm03ster@gmail.com

Yubo Tao
taoyubo@cad.zju.edu.cn

Yankai Jiang
jyk1996ver@zju.edu.cn

Wei Chen
chenvis@zju.edu.cn

State Key Lab of CAD & CG, Zhejiang University
Hangzhou 310058, China

Abstract

Neuron tracing, also known as neuron reconstruction, is an essential step in investigating the morphology of neuronal circuits and mechanisms of the brain. Since the ultra-high throughput of optical microscopy (OM) imaging leads to images of multiple gigabytes or even terabytes, it takes tens of hours for the state-of-the-art technique to generate a neuron reconstruction from a whole mouse brain OM image. We present a novel framework, InstantTrace, that enables parallel neuron tracing on GPUs, reaching a notable speed boost of more than 20× against state-of-the-art methods with comparable reconstruction quality on the BigNeuron dataset. We address two methods to achieve the speed advance. Firstly, the proposed framework takes advantage of the sparse feature and tree structure of the neuron image that serial tracing methods cannot fully utilize. Secondly, all stages of the neuron tracing pipeline, including the initial reconstruction stage that have not been parallelized in the past, are executed on GPU using carefully designed parallel algorithms. Furthermore, to investigate the applicability and robustness of the InstantTrace framework, a test on a whole mouse brain OM Image is conducted, and a preliminary neuron reconstruction of the whole brain is finished within 1 hour on a single GPU. The framework would greatly improve the efficiency of neuron tracing process and allow neuron image experts to get a preliminary reconstruction result instantly before involving manually verification and refinement.

Keywords: ***Neuron tracing, neuron visualization, image processing, GPU acceleration***

1. Introduction

The rapid development of optical microscopy (OM) makes it possible to generate nano-scale neuron images of the brain nervous system. The study of these images is important to the research and treatment of central nervous system (CNS) diseases such as Parkinsons and Alzheimers[12]. New techniques and tools for modeling[23, 7], visualization[3, 22, 15], and analysis[24] of these images are conducted.

Neuron tracing, also known as neuron reconstruction, is an essential step in investigating the morphology of neuronal circuits and mechanisms of the brain. Traced neuron branches save the spatial and topological information of the OM image, making it easier to investigate the connection patterns and topology relations of the brain nervous system. Typically, the reconstruction process consists of (i) an automated stage to deliver a preliminary tracing result, and (ii) a manual stage that the neuron imaging experts verify and correct the tracing result[20, 9, 8, 1].

In recent years, researchers have exerted efforts to develop effective tracing methods. For example, the APP[17] and APP2[26] methods employ fast marching framework to find the minimal distance route from the neuron soma (i.e., the center of the cell). Rivulet[13] and Rivulet2[14] utilize the technique of back-tracing from leaf to soma, lowering the probability of generating over-reconstruction results. FMST[28] combines the fast marching framework with the minimum spanning tree algorithm to optimize the topology of reconstructed neuron circuits. NeuroGPS-Tree[19] adapts the constrained principle curve method to get a neuron forest and then breaks them into separate neuron trees. Tremap[30] utilizes the fast marching method in the 2D projection space and makes reverse mapping back into the original image space to get the final result. SmartTracing[4] in-

vokes a user-provided existing tracing method to generate an initial result and uses an SVM classifier to predict the foreground pixels according to the clue presented by this initial reconstruction.

Most of these conventional automated neuron tracing methods can be divided into three stages: preprocessing, initial reconstruction and refining, as listed in Table 1. First, some necessary transforms are introduced in the preprocessing stage to enhance the image contrast. Then, initial reconstructions considering basic features of the image are performed though they are imperfect (i.e., over-reconstruction or under-reconstruction). Finally, delicate refinements using more advanced techniques and well-designed handcraft features are conducted for high-quality tracing results.

Researchers also developed some deep-learning-based methods for automated neuron reconstruction in recent years. The UltraNPR [29] method utilizes a progressive learning framework that integrates conventional tracing methods and deep segmentation networks for neuron population reconstruction. The DeepBranch [21] method uses multi-scale multi-view convolutional neural networks to precisely detect the neuron branch points. Another research [11] combines 3D U-Net architecture and atrous convolution to tackle the tangled neuron images. The SPE-DNR [5] method uses a two-headed 2D neuron network for tracing and foreground classification, to simulate 3D neuron reconstruction. MP-NRGAN [6] utilizes a generative adversarial network that takes pseudo-labels from conventional tracing methods to make synthetic data and does iterative training from these data. SGSNet [27] is a structure-guided segmentation framework, which is robust when it meets intense background noises.

Due to the rapid development of neuroimaging, the data size generated by the imaging devices poses severe challenges for the neuron tracing algorithms. For example, the data size of a whole mouse brain image will reach multiple gigabytes or even terabytes, whereas the current neuron tracing methods cannot handle such vast data volume efficiently. To our knowledge, none of the automated neuron tracing methods can proceed with neuron reconstruction in the same order of magnitude as data loading time, and the neuron imaging experts have to wait a long time after loading the block before making precise corrections.

In this paper, we propose an efficient parallel neuron tracing framework (InstantTrace) on GPU. First, the proposed parallel framework takes advantage of essential features of the neuron image that current serial tracing algorithms always overlook, resulting in significant speed advance. On the one hand, parallel stream compaction is introduced, and the sparsity of neuron image is fully utilized, while conventional tracing methods take the complete image and suffer from memory limitations. On the other hand, due to the tree structure of the neuron image, the branches leading to different directions are independent to a certain extent. While conventional methods deal with only one neuron segment at a time, the proposed framework conducts parallel tracing and branch pruning among the group of independent segments, improving the efficiency of neuron reconstruction without losing quality.

Second, all stages of the conventional neuron tracing pipeline, i.e., preprocessing, initial reconstruction, and refining, are mapped to GPU using carefully designed parallel algorithms. While the preprocessing and refining stages have been successfully implemented on GPU in the past, the initial reconstruction stage in the conventional tracing methods depend on strictly serial data structures such as priority queues [17, 26], which are difficult to parallelize. To tackle this problem, as shown in Figure 1, two more stages are introduced in the proposed InstantTrace framework compared to the conventional tracers, namely seeds generating and topology merging, to parallelize the initial reconstruction stage using a divide-and-conquer strategy. A bunch of candidate seed points are randomly generated in the seeds generating stage, and then traced independently in parallel in the initial reconstruction stage. When the processing is finished among the seeds, the algorithm will get a bundle of disconnected neuron branches, which will finally be connected in the topological merging stage to yield the input for the refining stage.

Different baseline methods can be integrated into the InstantTrace framework to achieve parallel neuron tracing. According to the literature [28, 5], the APP2 method is the fastest among the conventional methods. What's more, the current deep-learning-based methods concentrate on the precision of the reconstruction and will take more than a minute to process a single image block [27, 5], which are slower than APP2. Therefore, we use the APP2 method (InstantTrace-APP2) in our experiments. The performance and reconstruction quality of InstantTrace-APP2 are verified on several challenging datasets.

The major contributions of this work are as follows:

1. We propose an efficient parallel neuron tracing framework (InstantTrace) on GPU. The framework utilizes several essential features of the neuron image that serial tracing methods overlook, achieving significant speed advance. Firstly, the proposed InstantTrace framework takes advantage of the sparse feature of the neuron image and conducts parallel stream compaction, which erases the background pixels and significantly reduces memory usage. Secondly, the framework exploits the independency within the tree structure of the neuron. When a group of segments is located on nonadjacent neuron branches, tracing and refining of these segments are processed in parallel, making the framework notably more efficient than serial methods.

2. We add two more stages, i.e., seed generating and topology merging, to the proposed framework to make the

Method	Preprocessing	Initial Reconstruction	Refining
APP1/APP2 [17, 26]	Greyscale distance transform	Fast marching	Pruning with cover radius
Rivulet/Rivulet2 [13, 14]	Greyscale distance transform	Back-tracing from leaf	Branch cut
FMST [28]	Thresholding	Fast marching	Pruning with spanning tree
NeuroGPS-Tree [19]	Detect soma	Constrained principle curve	Breaking the trees
TReMap [30]	Projection to 2D faces	Fast marching	Reverse mapping
SmartTracing [4]	Together with its base tracer	Together with its base tracer	Removing background

Table 1. The conventional neuron tracing methods and their three stages.

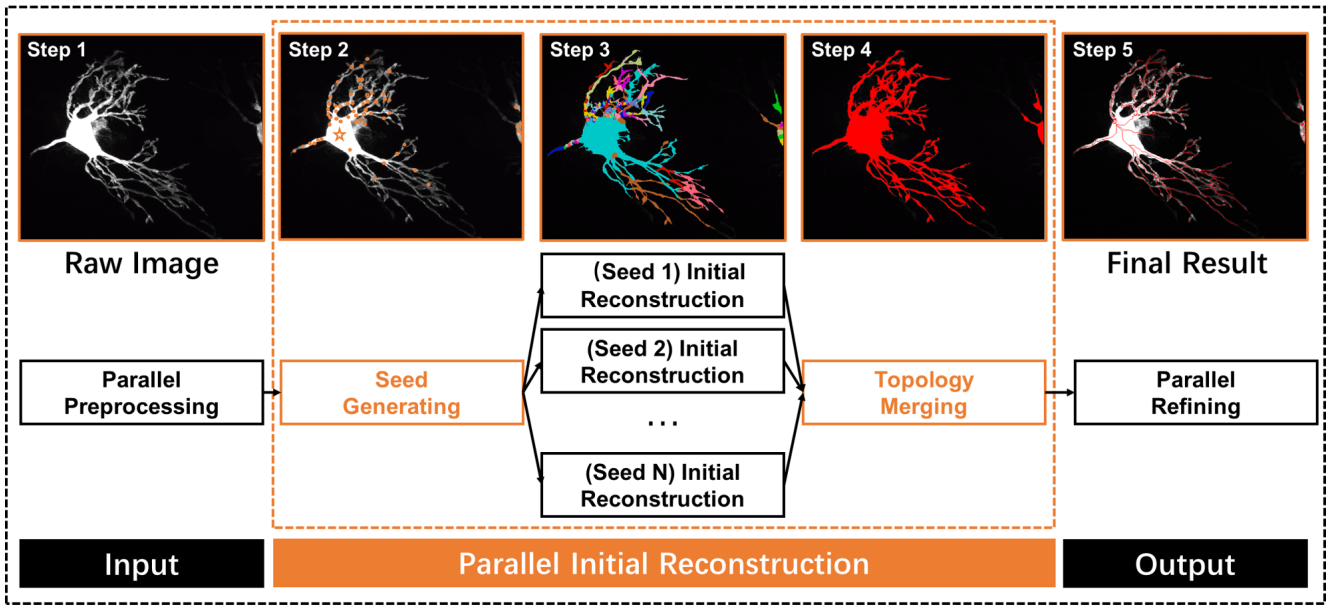

Figure 1. The working pipeline of the InstantTrace framework.

serial initial reconstruction process parallelized. The seeds generating stage generates a bunch of candidate seed points, then tracing is started from different seeds in parallel to generate neuron segments. When all of the tracing processes are finished, the topological merging stage will merge these disconnected neuron segments into a complete neuron tree, and the total workload of initial reconstruction is handled in this divide-and-conquer progress. Solving this problem, all stages of the conventional neuron tracing pipeline, i.e., preprocessing, initial reconstruction and refining, are successfully mapped to GPU using carefully designed parallel algorithms.

3. The speed performance and the reconstruction quality of the proposed parallel tracing framework are verified on the BigNeuron [16] dataset. Moreover, to demonstrate the adaptivity of the InstantTrace framework, a whole mouse brain OM image, split into 25,080 blocks, with a total size of 4 terabytes is introduced. On the BigNeuron dataset, the proposed framework reached a speed advance of 20× compared to the APP2 method, with comparable reconstruction quality. In the whole mouse brain test, the proposed framework made a preliminary neuron reconstruction of the whole brain within 1 hour on a single GPU. This framework significantly enhances the efficiency of automated neuron reconstruction, and is helpful for neuron imaging experts to verify and annotate the neuronal circuits.

2. Methods

2.1. Preprocessing

The preprocessing stage takes the original image as input and the enhanced image as output. As for InstantTrace-APP2, the preprocessing stage is composed of stream compaction and grayscale distance transform (GWDT). The stream compaction process reduces memory usage. The grayscale distance transform process increases the intensity in the center of neuron branches and makes the tracing result closer to the neuron skeleton (i.e., the center of the neuron branch).

Stream Compaction. The parallel stream compaction process[2] throws the background pixels and rearranges the foreground pixels according to a specific threshold. In this process, locations of the foreground pixels are logged, and these pixels form a new image array. The size of this array

is significantly small compared to the original image, so the GPU memory usage of the subsequent processes is reduced. A bijection map is generated between the original location and the current index of each foreground pixel so that the spatial relationship of the pixels remains in this compressed form.

Moreover, due to the integrated thresholding process in stream compaction, the background noises are erased, and the global contrast of the image is enhanced. The proposed method takes the same thresholding configuration as serial APP2:

$$t = \mu + 0.5\sigma \tag{1}$$

where t denotes the threshold, μ and σ denote the mean value and standard variance of the original image.

GrayScale Distance Transform. The grayscale distance transform is an advanced preprocessing technique defined in the APP2 program [26] to increase the intensity in the center of neuron branches and decrease the intensity near the boundary. This image enhancement process will make the tracing algorithm go through the branches' center line and make the tracing result more accurate.

The GWDT process is converted into the fast marching problem in the APP2 method. The edge distance between consecutive image pixel vertices is defined as

$$e(x, y) = ||x - y|| \cdot I(y) \tag{2}$$

where x is the current pixel, y is the neighbor, and I(y) denotes the intensity value of y. In the fast marching method, the initial distance value is set as

$$d(x) = \begin{cases} I(x) & if x \in background \\ \infty & if x \notin background \end{cases} \tag{3}$$

The distance values are updated iteratively using the following formula:

$$d(x) = min\{(d(y)+e(y,x)\}, y \in \{neighbors\ of\ x\} \tag{4}$$

After the iteration ends, the intensity-weighted distances to the background are measured for each pixel in the image. The border pixels of the neuron branches are of a low distance value, while the center pixels of the branches will get a relatively high value. Finally, the intensity of the whole image is normalized, and the contrast of the neuron branch skeleton will be enhanced.

In this study, the parallel implementation of GWDT is based on the parallel fast marching method, which will be described in detail in the following section. In this situation, all of the background pixels on the border of neuron branches are added to a startup tracing frontier set, and the frontier will march one pixel towards the neuron center line in each iteration step, as shown in Figure 2. The iteration process tends to end in a few steps because the width of the neuron branches is minimal compared to their length.

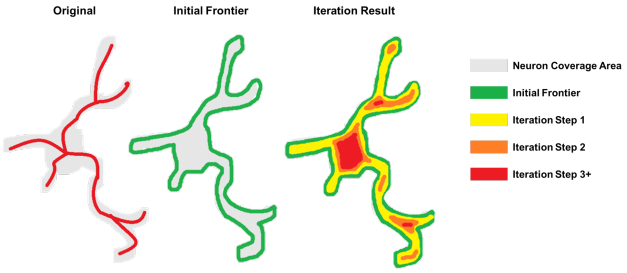

Figure 2. Diagram of the GWDT process. The iteration starts from the border of the neuron, and the frontier marches one pixel per step towards the neuron center. After the iteration, the image is normalized according to the weighted distance from the initial frontier, thus the intensity of neuron center is enhanced.

2.2. Initial Reconstruction

The conventional tracing methods for initial reconstruction such as the fast marching method and constrained principle curve method always have serial nature. For example, the fast marching method in APP2 adopts the priority queue, an extremely-serialized data structure. It is difficult to parallelize this serial data structure, as the race condition may mislead the correct item added into the priority queue, and the degree of parallelism will be low.

In InstantTrace-APP2, other than parallelizing the data structure, we parallelize the algorithm. First, a bunch of seeds is randomly generated, and then the fast marching method is started from every seed in parallel. The optimal item in the priority queue is turned into an optimal frontier set, which vastly enlarges the parallelism in the initial reconstruction stage. After tracing, the results generated by different seeds are merged topologically to make a complete initial reconstruction.

Seed Generating. We demonstrate two principles to generate the seeds. First, since increasing the number of seeds enhances efficiency but brings more memory costs, the number of seeds should compromise parallelism and resource occupancy. Second, the clustered seeds would generate clustered tracing results, which are hard to tell apart in topology. Therefore, the seeds should not interfere with each other.

Simple sampling methods are unable to meet these principles. For example, the naïve random sampling method tends to suffer from the seed cluster problem. Besides, the uniform sampling method is not optimal because the regular grid points do not fit the sparse neuron branch well, and most of the generated points would be invalid, which is a waste of computation resources. In the proposed method, we use the adaptive parallel Poisson disk sampling [25] for generating seeds. The Poisson disk sampling will generate

seeds at a minimum distance r from each other. By setting the minimum distance, the proposed method could adjust the number of seeds according to the memory bound and meet the requirement of parallelism without interference. What's more, the adaptive sampling method could take the greyscale intensity as a guide. More sampling points will fall on the neuron branches, and fewer points will be discarded.

Fast Marching. In InstantTrace-APP2, the fast marching method is parallelized to generate the initial neuron construction. Our GPU implementation is described in Algorithm 1. The algorithm starts from the seed group S generated by the previous stage and adapts the fast marching method iteratively. Within the iteration process, a tracing frontier set F is maintained, other than a single value with max priority in the serial fast marching method. In the beginning, the seed group becomes the initial frontier. The iteration process is composed of two steps, namely, extend and update. In the extend step, every element f in the frontier set is popped out, and its neighbors are visited. The algorithm tries to update the minimum distance from the seeds to the neighbors. Note that a pixel might be updated by more than one frontier element in a run, so this update process is done atomically in a temporary distance array, preventing race conditions. After that, the modified neighbors try to update their minimal distances by comparing them with the distance in the temporary array. If a pixel refreshes its minimal distance, it will be put into the frontier set for the next iteration. The iteration continues until the frontier set is empty, implying that the whole image space is visited.

Algorithm 1 Parallel Fast Marching of InstantTrace-APP2

Require: The seed array, S
1: Build the sweep frontier F, $F = S$
2: Build the distance array D, $D[i] = 0$ if $i \in S$, otherwise $D[i] = \infty$
3: Build the temporary frontier and distance array F', D'
4: **WHILE** F is not empty
5: **FOR** $F_i \in F$ **Parallel**
6: Invoke **CUDA_EXTEND_KERNEL**(F, F', D, D')
7: **FOR** $F'_j \in F'$ **Parallel**
8: Invoke **CUDA_UPDATE_KERNEL**(F', D, D')
9: $F = F'$
10: **END WHILE**

Topology Merging. Saving complete neuron information is vital in neuroscience. However, the parallel fast marching process is started from multiple seeds, and the tracing result tends to be topologically disconnected. To solve the challenge, the InstantTrace framework introduces the topology merging stage, which detects the intersections between the branches extended from different seeds and merges the disconnected parts into a complete neuron, as shown in Figure 3. The intersections are detected from the update information in the parallel fast marching process. For each pixel, if its minimal distance has been updated by more than one seed during the iteration process, it is likely to be located on the border of two different branches and marked as candidates of intersects. If the last and the second last updater of this pixel come from different seeds, it is considered a real intersect.

Algorithm 2 CUDA_EXTEND_KERNEL(F, F', D, D')

Require: The sweep frontier F, the distance array D, the temporary sweep frontier F', and the temporary distance array D'
1: $tid = thread\_index$
2: $index = F[tid]$
3: **FOR** $direction \in (0, 26)$
4: $neighbor = getNeighbor(index, direction)$
5: $delta = getLength(index, neighbor)$
6: $temp = D[index] + delta$
7: **BEGIN ATOMIC**
8: **IF** $(temp < D'[neighbor])$
9: $D'[neighbor] = temp$
10: Put $neighbor$ into F'
11: **END IF**
12: **END ATOMIC**
13: **END FOR**

Algorithm 3 CUDA_UPDATE_KERNEL(F', D, D')

Require: The temporary sweep frontier F', the distance array D, and the temporary distance array D'
1: $tid = thread\_index$
2: $index = F'[tid]$
3: $D[index] = D'[index]$

After the intersect detection, we keep a disjoint set to merge the branches. In the beginning, each branch segment is regarded as an individual cluster, and the radius of seed points is calculated for the merging process. The radius is calculated as:

$$r_i = max_r(\frac{B_{i,r}}{A_{i,r}} < 0.001) \tag{5}$$

$$A_{i,r} = \frac{4}{3}\pi r^3 \tag{6}$$

$$B_{i,r} = \Sigma_j \delta(I(j) = 0), \forall j \in \Omega_r(i) \tag{7}$$

where $A_{i,r}$ denotes the spherical coverage area of the radius, $B_{i,r}$ denotes the number of background pixels in this area, δ denotes the sign function, and Ω_r denotes the spherical neighborhood of radius r. The larger radius means the

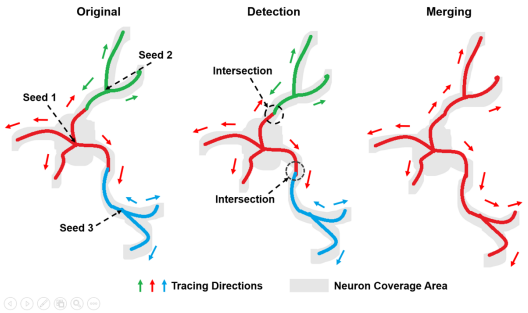

Figure 3. Diagram of neuron segments before and after topology merging. The red branch is generated from the neuron center (seed 1), and the green and blue branches are generated from seeds 2 and 3. The arrowheads mark the tracing direction of the branches. After merging, a complete neuron is traced.

seed has more foreground pixels in the neighborhood and more importance in reconstruction.

When two individual intersected branches are merged, they will be regarded as a merged cluster, and the one with a larger seed radius becomes the leader of the cluster. In this cluster, the seed of the leader is considered the new tracing root, and the tracing directions will be reset from this root. When merging two intersected clusters, the algorithm finds the leaders of them, and the leader with a larger seed radius becomes the new leader of the merged cluster. In this stage, the neuron center has the largest seed radius and tends to be the final root of the complete neuron tree.

2.3. Refining

In the refining stage of InstantTrace-APP2, branch pruning is conducted in parallel. While the serial APP2 program checks branches once a time, a topological sort is introduced in InstantTrace-APP2. In this sorting process, a dependency table is built according to the tree structure in the tracing results, and each child branch depends on the parent branch. In one iteration, the algorithm solves all of the current independent branches in parallel and updates the dependency table (i.e., removing the dependency of checked branches) before the next iteration starts.

The proposed method calculates the overall cover ratio when considering whether to keep or prune a branch segment. Every point i in segment S has a radius r_i. Within the spherical coverage area of the radius, the numbers of deleted pixels d_i are calculated. Then, the overall volume of the radius spheres R and the total number of deleted pixels D are calculated as follows:

$$D = \Sigma_i d_i, \forall i \in S \tag{8}$$

$$R = \Sigma_i (\frac{4}{3}\pi r_i^3), \forall i \in S \tag{9}$$

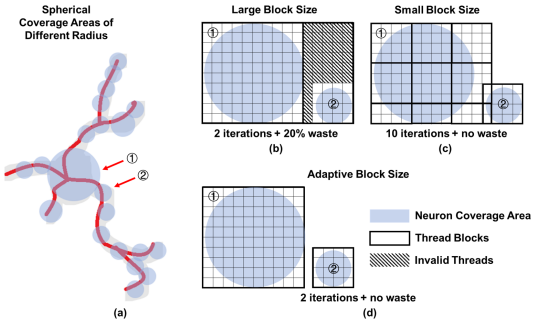

Figure 4. The different choices of parallelism in covering the segment. (a) The different cover radius of points in a segment. (b) Using the large block size in the thread block causes a waste of computation. (c) Using the small block size in the thread block dramatically increases the number of iterations. (d) Using the dynamic parallelism feature, the block size in small and large areas are changed adaptively.

The cover ratio RT is calculated by $RT = D/R$. When RT is larger than a specific threshold (for example, 0.5), this branch segment is regarded as covered by other segments and pruned. Otherwise, this segment is kept, and the coverage area of the segment is deleted in the image.

In the parallel refining process, both coarse-grained and fine-grained parallelism is conducted. Firstly, due to the topological sort, segments on different neuron branches are independent and can be processed concurrently. Secondly, when dealing with one segment, the coverage area of the pixels in the segment is summed up, as shown in Figure 4 (a), and the total workload is divided into parallel thread blocks. As shown in Figure 4 (b, c), using a fixed large block size in the thread block causes waste of computation, while using a fixed small block size will dramatically increase the number of iterations. The proposed method utilizes the CUDA dynamic parallelism feature [10] to make each point in the segment start threads according to its radius adaptively other than using a fixed thread block size, as shown in Figure 4 (d). Therefore, the load balance is kept, and there is no waste of computation. After pruning, the final result of the neuron tracing is reached.

3. Results and discussion

3.1. Reconstruction on the BigNeuron dataset

The BigNeuron dataset includes many neuron images under the optical lens, together with the ground truth of the reconstructions. To validate the working efficiency and quality of the InstantTrace framework, considering the generalization of different species, we pick 50 out of 166 neuron constructions, consisting of chick, frog, zebrafish, fly, mouse, fruitfly, and human neurons. The main objective is

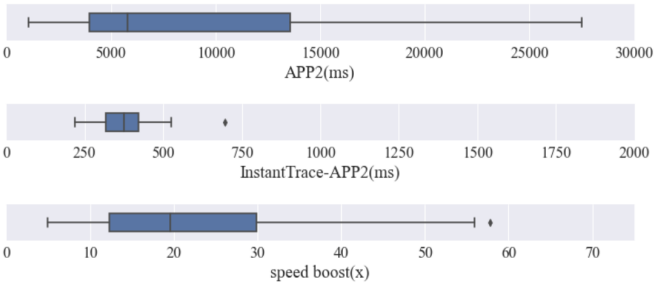

Figure 5. The speed up of the proposed InstantTrace framework against the serial APP2 program on the BigNeuron dataset.

extracting the central neuron cell in sight and erasing the cluttering background. The APP2 and the InstantTrace-APP2 methods are tested in the experiment, and the efficiency and reconstruction quality are measured. The Vaa3d software (https://github.com/Vaa3D) carries the APP2 algorithm implementation, and the proposed method is implemented with the Nvidia CUDA toolkit. The source code of the proposed neuron tracing framework and the example image blocks are available under the repository of (https://github.com/jifaley/InstantTrace).

Speed and Quality Results. We have implemented and tested the InstantTrace framework on an Intel(R) Core(TM) i9-10900KF CPU @ 3.70GHz processor with 64GB RAM, and an NVIDIA GeForce RTX 3090 (24GB) graphics card.

As shown in Figure 5, the proposed method delivers the reconstruction result immediately, while APP2 uses more than 5 seconds to reconstruct the neuron. The InstantTrace-APP2 method gets a speed up of 5-50× and an average speed up of more than 20× in the testing data.

What's more, the reconstruction quality is measured using five indicators defined by the APP2 work: ESA12, ESA21, ESA_MEAN, DSA, and PDS, as listed in Table 2. These indicators are also measured by the neuron distance plugin in the Vaa3d software. As shown in Table 3, the measurement shows the InstantTrace-APP2 method reaches comparable reconstruction quality with the serial APP2 method.

Figure 6 presents 7 examples from the 50 samples to illustrate the reconstruction quality of the two methods. In column (a) and column (b), the InstantTrace-APP2 method tends to trace all of the neuron branches due to the seed generating stage, while the serial APP2 algorithm only traces the neuron in the central location. Note that the ground truth of the BigNeuron dataset only consists of a single neuron, so only the largest tree in the reconstruction forest of InstantTrace-APP2 is used for quality comparison in this test. As for column (c), APP2 and InstantTrace-APP2 fail to erase the high-intensity noise around the soma. For column (d), both methods fail to pick out the neighboring neuron cell. In columns (e) to (g), both methods succeed in making a good reconstruction. In summary, the APP2 and the InstantTrace-APP2 methods show visually comparable results among the whole dataset.

Performance Breakdown. Figure 7 shows the running time composition of each tracing stage on the BigNeuron dataset. Note that the seed generating and topology merging stages are not case sensitive. Firstly, the computational cost of seed generating is only relevant to the density of the sampling grid, which is a fixed parameter. Secondly, since the topology merging stage always ends in less than 20 microseconds, its contribution to the total time cost can be neglected. In contrast, preprocessing, initial reconstruction, and refining stages account for a large portion of the reconstruction time in our framework, and their time cost is positively associated with the complexity of neuron image. What's more, though the GPU warmup cost is around 140ms despite the kind of the neuron, it can be hidden by handling image blocks continually when processing data streams and vast datasets.

Ablation Study. To investigate the importance of the two added steps, seed generating and topology merging, an ablation study is conducted on the BigNeuron dataset, as shown in Table 4. When the seed generating process is not activated, the parallel initial reconstruction stage is degraded into the single seed reconstruction, and the parallelism will get down. Under this situation, the reconstruction branches are grown from the very same seed, and there will be no need for topology merging. When the seed generating is activated, but the merging process is not conducted, the final reconstruction result will be topologically disconnected, which is difficult for neuron imaging experts to utilize. At the same time, the average length of the traced branch will be decreased significantly. In the refining stage, long branches have larger coverage area and more probability of being kept in the final result. Therefore, without the topology merging process, the main branches of the neuron will have fewer advantages over minor neurites to be kept, and the total reconstruction quality will go down. Moreover, due to less dependency between branches, fewer minor neurites will be dropped when a useless branch is pruned, and the pruning efficiency will also degrade.

3.2. Reconstruction of the whole mouse brain

To investigate the applicability and robustness of the parallel InstantTrace framework, another test in a whole mouse brain OM Image is conducted. The image size is 25000 × 15000 × 5000, and a total size of 4 terabytes. The sampling ratio in the x, y and z axis is 1:1:0.16.

To handle the whole brain reconstruction, the image is sliced into 25,080 blocks, each with a block size of 512 × 512 × 512, and a padding of 2 in x, y and z axis. The proposed processing pipeline handles the blocks in stream. When a block is being processed and reconstructed, the next

Metric	Description	Range
ESA12	Entire Structure Average distance from neuron 1 to 2	$[0, +\infty)$
ESA21	Entire Structure Average distance from neuron 2 to 1	$[0, +\infty)$
ESA_mean	Average of ESA12 and ESA21	$[0, +\infty)$
DSA	Different Structure Average distance	$[0, +\infty)$
PDS	Percent of Different Structure	$[0, 1]$

Table 2. The metric of the tracing quality of neuron reconstruction.

	ESA12	ESA21	ESA_mean	DSA	PDS
APP2	**4.18**	7.54	5.86	10.08	**0.45**
InstantTrace-APP2	4.50	**6.78**	**5.64**	**9.21**	0.52

Table 3. The average reconstruction quality of the BigNeuron dataset. The bold font represents the method perform better.

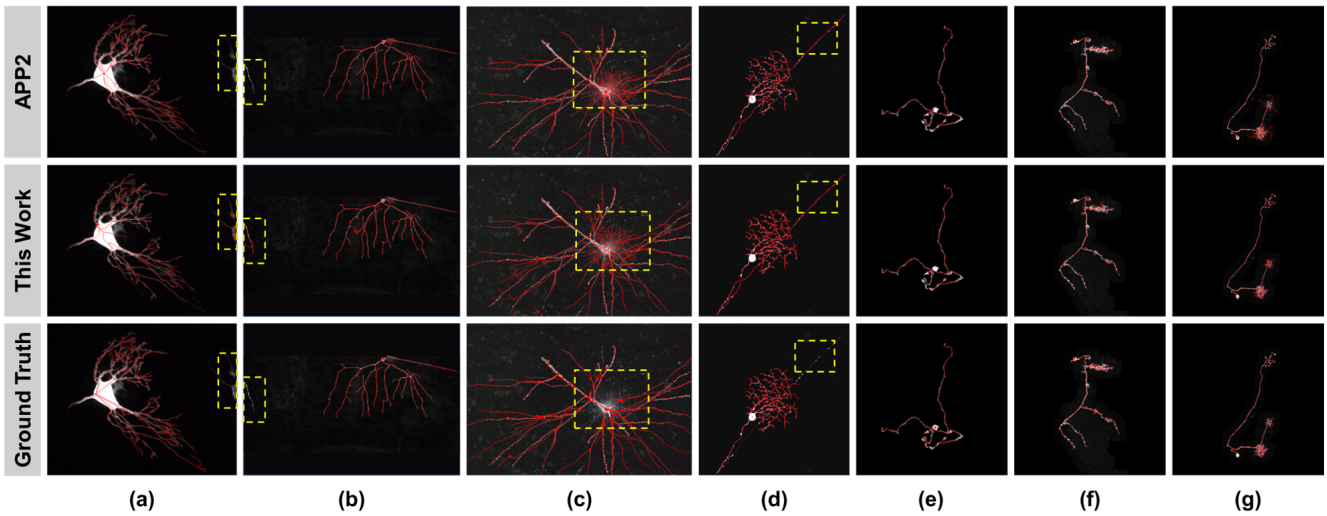

Figure 6. The visual comparison of reconstruction quality of APP2 and InstantTrace-APP2 on the BigNeuron dataset. The three rows are the tracing result made by APP2, InstantTrace-APP2, and the reconstruction ground truth. Column (a) and column (b) show the difference between the two methods brought by the seed generation. The rest of the columns show the similarity in the tracing results of the two methods (c-d: both fail, e-g: both succeed).

	Time(ms)	ESA12	ESA21	ESA_mean	DSA	PDS
APP2	10049.88	4.18	7.54	5.86	10.08	**0.45**
None	399.25	6.25	8.50	7.41	11.97	0.47
Seed	381.50	**3.21**	9.78	6.18	9.72	0.48
Seed + Merge	**377.98**	4.50	**6.78**	**5.64**	**9.21**	0.52

Table 4. The ablation study result on the BigNeuron dataset. The Seed + Merge row denotes the complete InstantTrace-APP2 pipeline. The None row denotes the process without both seed generating and topology merging. The Seed row denotes the process with seed generating but without topology merging. The APP2 row is the serial APP2 method as the control group. Note that there is no Merge group because the topology merging process is dependent on multi seeds.

block can be pre-loaded to save the scheduling cost, and the GPU warmup proceeds only once in the whole reconstruction progress. Note that neurons may exist across multiple blocks under the whole brain reconstruction situation. Therefore, we follow the solution of UltraTracer [18] to fuse the neurites in adjacent blocks. When the tracing in one block is finished, the intersecting points of the reconstructed neuron and the border of the image block are logged. When handling the next block, the intersecting points of its processed neighbors are set as extra tracing seeds so that the neurons can be traced across the image blocks continually.

The overview of the whole mouse brain reconstruction of

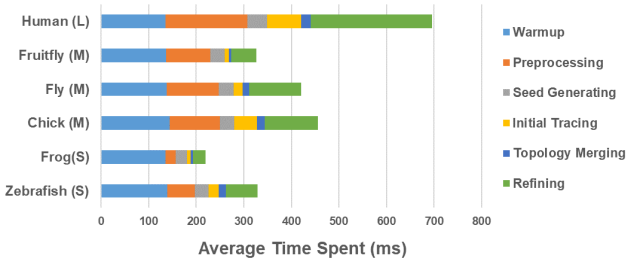

Figure 7. Breakdown of the reconstruction time on the BigNeuron dataset. Each row corresponds to a single kind of neuron image in the dataset. The S, M, and L markers denote small ($<$ 50 MB), medium (50-150 MB), and large ($>$ 150 MB) image sizes among the dataset.

InstantTrace-APP2 and two detailed views of areas with different reconstruction difficulties are shown in figure 8. Each detailed view is a combination of 24 image blocks. As the figure shows, the neurons long enough to cross many blocks are also correctly traced, which is essential in the whole brain level neuron reconstruction. The tracing result is of no significant defect and can become a preliminary tracing result before manual intervention from neuron imaging experts.

What's more, the proposed parallel InstantTrace framework runs very fast, as shown in Table 5. In this mouse brain data, during the processing of the 25,080 blocks, 8,725 blocks include foreground pixels, and the rest blocks only consist of background pixels. Despite the cost of loading blocks, the average process time of non-empty blocks is 263 ms, and the empty blocks cost 67 ms per block. The total time cost of whole-brain construction is 3389 seconds, and the processing speed is 1.20 GB/s, drastically reducing the preliminary reconstruction time cost. In contrast, if we directly apply the APP2 method, the process cost is 11.2s per non-empty block, and the processing speed is 38.84 MB/s, which will cost more than 30 hours to finish the whole brain reconstruction.

4. Conclusion

This paper introduced InstantTrace, a parallel neuron tracing framework on GPU. The proposed framework utilizes the sparse feature of the OM neuron images and the independent nature of the neuron branches, making the parallel upgrade onto the conventional neuron tracing pipeline. While the initial reconstruction stage in the conventional neuron tracing methods is difficult to parallelize due to its serial nature, the proposed framework adds two more stages, namely seed generating and topological merging, to form a multi-seed parallel tracing process.

This framework can be applied to the conventional methods of automated neuron tracing and reach comparable tracing quality with a remarkable speed up. In experiments, the APP2 method is applied to the InstantTrace framework (InstantTrace-APP2) due to its effectiveness and stability. Experiments on the BigNeuron dataset show that the combined InstantTrace-APP2 method gets comparable reconstruction quality and a speed advance of more than 20× against the serial APP2 method. Moreover, a challenging task of whole mouse brain neuron tracing is conducted, and the proposed method generates the whole brain neuron reconstruction within 1 hour of processing time using a single GPU. In conclusion, the proposed framework has the potential to become a base tracer of whole-brain construction processes, generate a preliminary reconstruction of good quality instantly, and promote neuron circuit research in the future.

References

[1] A. K. Ai-Awami, J. Beyer, D. Haehn, N. Kasthuri, J. W. Lichtman, H. Pfister, and M. Hadwiger. Neuroblocks–visual tracking of segmentation and proofreading for large connectomics projects. *IEEE transactions on visualization and computer graphics*, 22(1):738–746, 2015. 1

[2] D. Bakunas-Milanowski, V. Rego, J. Sang, and C. Yu. Efficient algorithms for stream compaction on gpus. *International Journal of Networking and Computing*, 7(2):208–226, 2017. 3

[3] S. Boorboor, S. Jadhav, M. Ananth, D. Talmage, L. Role, and A. Kaufman. Visualization of neuronal structures in wide-field microscopy brain images. *IEEE transactions on visualization and computer graphics*, 25(1):1018–1028, 2018. 1

[4] H. Chen, H. Xiao, T. Liu, and H. Peng. Smarttracing: self-learning-based neuron reconstruction. *Brain informatics*, 2(3):135–144, 2015. 1, 3

[5] W. Chen, M. Liu, H. Du, M. Radojević, Y. Wang, and E. Meijering. Deep-learning-based automated neuron reconstruction from 3d microscopy images using synthetic training images. *IEEE Transactions on Medical Imaging*, 41(5):1031–1042, 2021. 2

[6] X. Chen, C. Zhang, J. Zhao, Z. Xiong, Z.-J. Zha, and F. Wu. Weakly supervised neuron reconstruction from optical microscopy images with morphological priors. *IEEE Transactions on Medical Imaging*, 40(11):3205–3216, 2021. 2

[7] P. Ghahremani, S. Boorboor, P. Mirhosseini, C. Gudisagar, M. Ananth, D. Talmage, L. W. Role, and A. E. Kaufman. Neuroconstruct: 3d reconstruction and visualization of neurites in optical microscopy brain images. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 1

[8] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. W. Lichtman, and H. Pfister. Design and evaluation of interactive proofreading tools for connectomics. *IEEE transactions on visualization and computer graphics*, 20(12):2466–2475, 2014. 1

[9] M. Halavi, K. A. Hamilton, R. Parekh, and G. Ascoli. Digital reconstructions of neuronal morphology: three decades of research trends. *Frontiers in neuroscience*, 6:49, 2012. 1

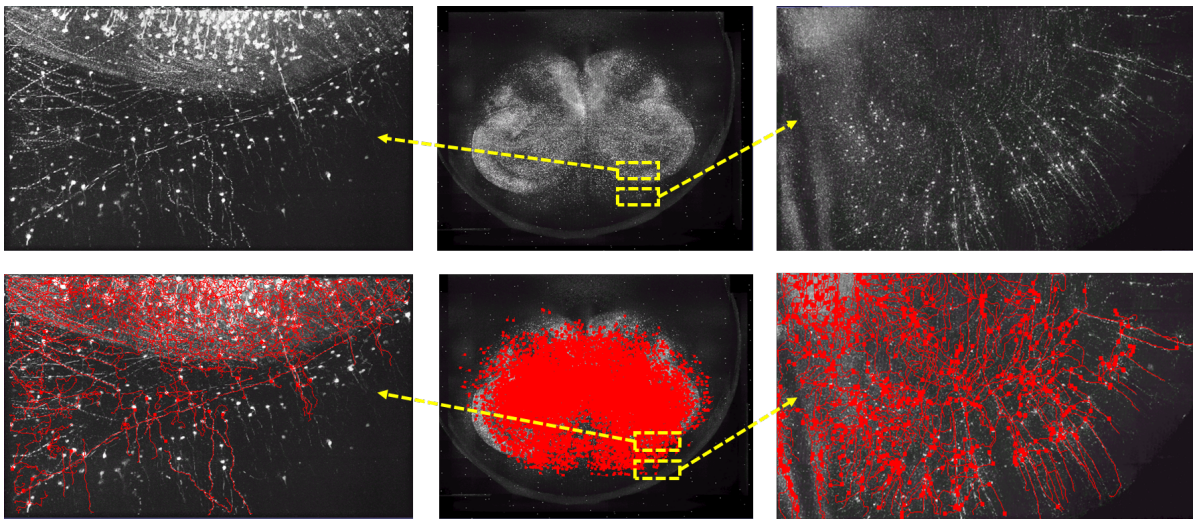

Figure 8. The reconstruction result of a whole mouse brain OM image with InstantTrace-APP2. The middle column is the overview of the raw image and the reconstruction results. The left and the right column are two different brain areas, as well as their corresponding reconstruction details. The area of the right column is clear for tracing methods to handle, while the left area is fuzzy and with a vast brightness gap, which is challenging.

	Processing time per non-empty block	Total time cost of 25,080 blocks	Average process speed throughput
APP2	11.20 s	$>$ 30 h	38.84 MB/s
InstantTrace-APP2	0.26 s	3389 s	1.20 GB/s

Table 5. The running time of APP2 and InstantTrace-APP2 for whole mouse brain reconstruction.

[10] S. Jones. Introduction to dynamic parallelism. In *GPU Technology Conference Presentation S*, volume 338, page 2012, 2012. 6

[11] Q. Li and L. Shen. 3d neuron reconstruction in tangled neuronal image with deep networks. *IEEE transactions on medical imaging*, 39(2):425–435, 2019. 2

[12] M. Liu, W. Chen, C. Wang, and H. Peng. A multiscale ray-shooting model for termination detection of tree-like structures in biomedical images. *IEEE Transactions on Medical Imaging*, 38(8):1923–1934, 2019. 1

[13] S. Liu, D. Zhang, S. Liu, D. Feng, H. Peng, and W. Cai. Rivulet: 3d neuron morphology tracing with iterative back-tracking. *Neuroinformatics*, 14(4):387–401, 2016. 1, 3

[14] S. Liu, D. Zhang, Y. Song, H. Peng, and W. Cai. Automated 3-d neuron tracing with precise branch erasing and confidence controlled back tracking. *IEEE transactions on medical imaging*, 37(11):2441–2452, 2018. 1, 3

[15] T. McDonald, W. Usher, N. Morrical, A. Gyulassy, S. Petruzza, F. Federer, A. Angelucci, and V. Pascucci. Improving the usability of virtual reality neuron tracing with topological elements. *IEEE transactions on visualization and computer graphics*, 27(2):744–754, 2020. 1

[16] H. Peng, M. Hawrylycz, J. Roskams, S. Hill, N. Spruston, E. Meijering, and G. A. Ascoli. Bigneuron: large-scale 3d neuron reconstruction from optical microscopy images. *Neuron*, 87(2):252–256, 2015. 3

[17] H. Peng, F. Long, and G. Myers. Automatic 3d neuron tracing using all-path pruning. *Bioinformatics*, 27(13):i239–i247, 2011. 1, 2, 3

[18] H. Peng, Z. Zhou, E. Meijering, T. Zhao, G. A. Ascoli, and M. Hawrylycz. Automatic tracing of ultra-volumes of neuronal images. *Nature methods*, 14(4):332–333, 2017. 8

[19] T. Quan, H. Zhou, J. Li, S. Li, A. Li, Y. Li, X. Lv, Q. Luo, H. Gong, and S. Zeng. Neurogps-tree: automatic reconstruction of large-scale neuronal populations with dense neurites. *Nature methods*, 13(1):51–54, 2016. 1, 3

[20] K. Svoboda. The past, present, and future of single neuron reconstruction. *Neuroinformatics*, 9(2):97–98, 2011. 1

[21] Y. Tan, M. Liu, W. Chen, X. Wang, H. Peng, and Y. Wang. Deepbranch: Deep neural networks for branch point detection in biomedical images. *IEEE transactions on medical imaging*, 39(4):1195–1205, 2019. 2

[22] W. Usher, P. Klacansky, F. Federer, P.-T. Bremer, A. Knoll, J. Yarch, A. Angelucci, and V. Pascucci. A virtual reality visualization tool for neuron tracing. *IEEE transactions on visualization and computer graphics*, 24(1):994–1003, 2017. 1

[23] S. Wang, J. Wu, M. Wei, and X. Ma. Robust curve skeleton extraction for vascular structures. *Graphical Models*, 74(4):109–120, 2012. 1

[24] Y. Wang, Q. Li, L. Liu, Z. Zhou, Z. Ruan, L. Kong, Y. Li, Y. Wang, N. Zhong, R. Chai, et al. Teravr empowers precise

reconstruction of complete 3-d neuronal morphology in the whole brain. *Nature communications*, 10(1):1–9, 2019. 1

[25] L.-Y. Wei. Parallel poisson disk sampling. *Acm Transactions On Graphics (tog)*, 27(3):1–9, 2008. 4

[26] H. Xiao and H. Peng. App2: automatic tracing of 3d neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree. *Bioinformatics*, 29(11):1448–1454, 2013. 1, 2, 3, 4

[27] B. Yang, M. Liu, Y. Wang, K. Zhang, and E. Meijering. Structure-guided segmentation for 3d neuron reconstruction. *IEEE Transactions on Medical Imaging*, 41(4):903–914, 2021. 2

[28] J. Yang, M. Hao, X. Liu, Z. Wan, N. Zhong, and H. Peng. Fmst: an automatic neuron tracing method based on fast marching and minimum spanning tree. *Neuroinformatics*, 17(2):185–196, 2019. 1, 2, 3

[29] J. Zhao, X. Chen, Z. Xiong, D. Liu, J. Zeng, C. Xie, Y. Zhang, Z.-J. Zha, G. Bi, and F. Wu. Neuronal population reconstruction from ultra-scale optical microscopy images via progressive learning. *IEEE Transactions on Medical Imaging*, 39(12):4034–4046, 2020. 2

[30] Z. Zhou, X. Liu, B. Long, and H. Peng. Tremap: automatic 3d neuron reconstruction based on tracing, reverse mapping and assembling of 2d projections. *Neuroinformatics*, 14(1):41–50, 2016. 1, 3