AUTOMATED EXTRACTION OF LIVER OUTLINES FROM COMPUTED TOMOGRAPHY SCAN IMAGES USING A CUDA-BASED SEGMENTATION METHOD

Yiping Chen, Dong Li, Qing Zhu, Cheng Wang, Jonathan Li

Fujian Key Laboratory of Sensing and Computing for Smart City, School of Information Science and Engineering, Xiamen University, Xiamen 361005, China – (chenyiping, cwang, junli)@xmu.edu.cn

Commission II, WG II/10

KEY WORDS: Live Outline, CT Scan, Object Extraction, Image Segmentation, Hepatic Segmentation, Vascular Extraction,

ABSTRACT:

The traditional fast marching algorithm for segmentation of the liver is suitable for processing on the central processing unit (CPU) platform, however, it is not suitable for implementation on Graphics Processing Unit (GPU). The fuzzy connection algorithm is used to extract the blood vessels in the liver, but there is a calculation error. The refinement algorithm is very time consuming when extracting the target skeleton line from the 3D image. In this paper, the fast-marching algorithm and the thinning algorithm are improved, which can be applied to the GPU computing, The fuzzy algorithm is also improved, and the calculation error of the algorithm is solved, making it more suitable for medical image processing. The computing speed of GPU is far faster than CPU. Medical image processing is one of the earliest applications where the computing performance is improved by GPU. These three segmentation methods, fast marching method, fuzzy connecting method and refinement algorithm are very common in medical image segmentation. Because the increment of medical image data results in the extension of computing time for medical image processing, it is necessary to apply the high parallelism of the GPU to speed up these algorithms. The experiment results demonstrate the feasibility of our accelerating algorithm.

1. INTRODUCTION

1.1 General Instructions

The main work of this paper is to first extract the outline of the liver in the CT image of the abdomen, and then extract the blood vessels in the liver through the fuzzy connection segmentation algorithm. Refining the liver blood vessels and obtaining the skeleton line of the blood vessels finally. These tasks are based on GPU implementations and are compared to the CPU calculations.

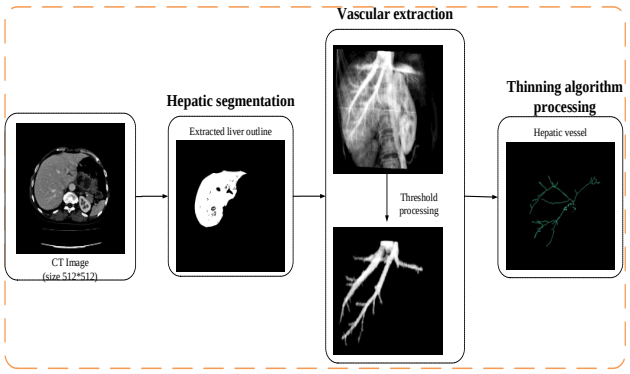

Figure 1. Framework of our method

1.2 Hepatic segmentation

FMM is an effective method of hepatic segmentation (Wan et al. 2000) and achieve a worst case complexity of O(NlogN). It uses the min-heap data structure for adjusting the order of computational step. However, min-heap is very effective data structure on the CPU sequential paltform, which represent a bottleneck that thwarts parallelization. In order to adapt on CUDA SIMD architectures, parallel algorithm extends points processing to thread block processing, and uses Rouy-Tourin algorithm for solving equation in every one thread block. Finally, itachieves more than 3x against the CPU sequential algorithm in 2D hepatic segmentation (Li et al. 2015).

1.3 Vascular extraction

A paralleled CUDA version of kFOE(CUDA-kFOE) was proposed to segment medical images. CUDA-kFOE achieves fast segmentation when processing large image datasets. However, it cannot precisely handle the competition of edge points when update operations happen by multiple threads simultaneously, thus an iterative correction method to improve CUDA-kFOE was proposed. By analyzing all the pathways of marginal voxels affinity and their consequently caused results, a two iteration correction scheme is employed to achieve the accurate calculation. In these two iterations, the resulted marginal voxels from the first iteration are used as the correction input of the second iteration, therefore, the values of affinity are corrected inthe second iteration.

Independent of any image data (such as grey value), the image elements are considered to have a fuzzy adjacency relationship (Eapen et al. 2014), wherein the fuzzy adjacency instance of the fuzzy relationship can be expressed as the following formula:

$$\mu_\omega(a,b) = \begin{cases} \frac{1}{1+k_1\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2}}, & if\ \sum_{i=1}^{n}|a_i-b_i| \le \mathrm{n} \\ 0, & otherwise \end{cases} \quad (1)$$

where k_1 = a non-negative constant
a, b = any two points on the n-dimensional Euclidean space

The fuzzy affinity of a fuzzy relationship ξ is expressed as:

$$\mu_{\xi}(a,b)=\frac{\mu_{\omega}(a,b)}{1+k_2|f(a)+f(b)|} \quad (2)$$

where k_2 = a non-negative constant, $f(a)$, $f(b)$ is equal to the gray values of point a and point b.

1.4 Vascular skeleton line extraction

Thinning algorithm is a frequently used method for extracting object skeletons in 3D image. However, sequential thinning algorithm is a very time consuming computing process in CPU platform. By using powerful computation of Graphic Process Unit (GPU) to parallelize the 3D thinning algorithm is very necessary. There are two major methods to image thinning: a) decision trees and b) kernel-based filters. Because of more suitable for CUDA architectures, a12-subiteration image thinning algorithm which is kernel-based has been implemented for hepatic vessel thinning. The speedup achieved with the parallel algorithm for GPU achieves 11.6x, 12.0x and 13.8x against the CPU single-process version.

Eventually, this paper adopt a 3D thinning algorithm based on 12-sub-direction iteration (Zhang et al. 2017). The algorithm compares the simple points by using the mask method, which is suitable for computing in the parallel architecture of the GPU. The binarized blood vessel image obtained in the second step is refined to achieve a skeleton line of the blood vessel (Chowriappa et al. 2014).

2. RELATED WORK

In terms of image registration, (Eklund A et al. 2012) proposed CUDA uses volumetric registration with inseparable 3D filtering. (Yuen et al. 2008)proposed a real-time ultrasonic volume dynamic compensation system based on GPU, which can be used to complete mitral valve repair in heartbeat. In terms of image segmentation, (Rumpf et al. 2001) first used the GPU to accelerate the level set method to segment the 2-D brain image. (Erdt et al. 2008) combined the characteristics of GPU to accelerate the region growing algorithm in parallel, which is applied in liver vessel segmentation. (Pan et al. 2008)implemented a CUDA-based replacement fast region growth algorithm to segment different human tissues and organs. (Strzodka et al. 2003) implemented an object detection and gesture recognition algorithm using Hough transform on the GPU. In terms of image denoising, (Schwarzkopf et al. 2003) achieved the effect of 3D image denoising based on nonlinear anisotropic diffusion filtering by CUDA acceleration. (Xu and Pattanaik. 2005) based on bilateral filtering to achieve Monte Carlo method on the GPU to achieve noise reduction.(Liu et al. 2018) implemented an improved 3D U-net architecture, which achieves a more precise segmentation effect, (Alirr et al. 2018)introduced a fully automated liver tumor segmentation method in the contrast-enhanced CT dataset. (Lebre et al, 2018) proposes an automatic three-dimensional skeleton segmentation method based on bones. (Matthies P et al, 2019) verified an interactive software tool that provides semi-automatic segmentation of co-registered image data. (Bal E et al, 2018) present an automatic method for liver segmentation (CT) portal vein phase scan liver segmentation and fibrosis classification. This method can perform liver volume segmentation and fibrosis grade in abdominal CT images. A multi-distribution level set method is proposed (Pan Q et al, 2018) to overcome the insufficient segmentation and over-segmentation in liver tumor segmentation.

3. PROPOSED METHOD

3.1 Description

The main work of this paper is to extract the liver contour from the abdominal CT image and finally get the vascular skeleton line. These tasks are all based on GPU implementation. It consists mainly of three main steps:

1. Using the improved FMM algorithm to extract the liver contour from the abdominal CT image.
2. Extraction of blood vessels in the liver by fuzzy connection segmentation algorithm.
3. Extracting the skeleton line of blood vessels from the blood vessels of the liver by a refinement algorithm.

3.2 Hepatic segmentation

The Fast Marching method proposed by (Sethian et al, 2003) is a very effective numerical method for solving the boundary value problem of the equation function. It is limited by the upper limit of the single core frequency of the CPU, and the solution speed is not very fast. The iterative method for solving the equation of the equation is proposed by Rouy and Tourin. Due to the simplicity and parallelism of the Rouy-Tourin iterative algorithm, it is very suitable for running under the CUDA architecture, and each thread can correspond to one pixel. However, due to the operation of the architecture, each thread can be iterated for one pixel. Due to the parallel disorder of threads and thread blocks in CUDA programs, it is difficult to control the iterative calculation at the point of no value, which leads to the phenomenon of inefficiency. In view of the above defects, this paper combines the narrow band idea of FMM algorithm and applies it to the control of CUDA thread block, which solves the problem of non-computing thread block repeated loading participation operation.

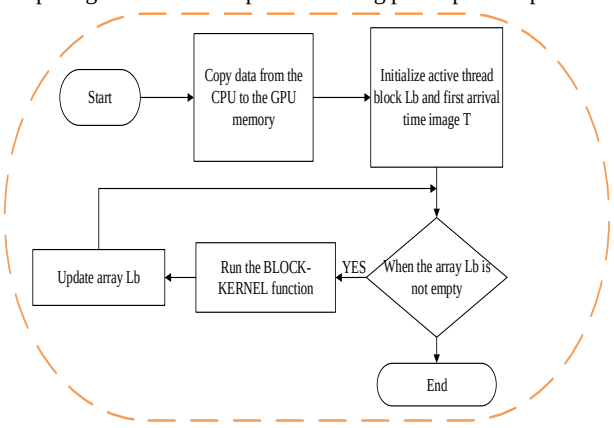

Figure 2. Algorithm flow based on thread block iteration

3.2.1 Algorithm Description: The algorithm consists of two main steps, including initialization and update. In the initialization, the initialization time value $T_{i,j} = 0$ of the velocity image seed point coordinates is first set, and the remaining points are set to a very large value. The next step is to add the block where the seed point is located to the active block array L_b, initialize the iterative kernel function and load it into the CUDA device. After the active thread block iterates enough times, use the protocol operation to determine the thread block diffusion direction. Repeat this step until the thread The block array L_b is empty. The block diagram of the algorithm is shown in FIG. 2.

3.3 Vascular skeleton line extraction

Fuzzy connection segmentation was first proposed by (Udupa et

al, 1996). The algorithm divides the target and background by comparing the connectivity of the seed point with the target area and the background area. The fuzzy connected image segmentation algorithm is abbreviated as kFOE, which uses the Dijkstra algorithm of the dynamic programming method to find the best path from the target point o to each pixel.

The CUDA-kFOE algorithm proposed by (Zhuge et al, 2009) has parallelized fuzzy connections, which is divided into two steps:

1. Affinity calculation. Affinity calculation is a calculation between voxel pairs (c, d) and stores the calculated affinity $\mu_k(c, d)$ in GPU-specific memory.
2. Update fuzzy connections. The fuzzy connection algorithm for voxels is essentially a single-source Dijkstra algorithm. In devices with CUDA computing power greater than 1.1, atomic operations are used to solve multi-threaded access. The use of atomic operations solves the problem of multi-threaded access to the same address conflict, and basically achieves its parallelization.

The CUDA-kFOE algorithm has two shortcomings: First, calculating the ambiguity in step 1 and saving it in the GPU memory requires a lot of extra space, which is difficult to perform in a graphics card with limited memory capacity; Second, Mutual interference at the edges between the blocks causes errors in the calculations at the edges of the blocks.

In order to solve the problem of insufficient ambiguity storage space, this paper uses the lookup table method to optimize the problem of insufficient GPU memory space. In the process of determining the global fuzzy relationship, a local fuzzy relationship is needed. Therefore, many identical pairs of pixels are repeatedly calculated in the calculation process. In addition, the object studied in this paper is mainly the blood vessel segmentation inside the liver and the gray scale of the CT image. Values are often distributed between $100 - 230$. The above analysis results in constructing a 256×256 lookup table to store all the connection strengths $\mu_k(c, d)$ of the pixels between (c, d) before calculating the global fuzzy scene, where $c \in [0, 255], d \in [0, 255]$, Then, in the process of calculating the fuzzy scene, the affinity value of any pixel pair (c, d) can be directly taken out from the table, which not only shortens the calculation time, but also solves the problem that the GPU has limited video memory.

In order to solve the error problem, it is necessary to analyze the source of the error. Considering that the starting point of the algorithm is a single seed point, and the fuzzy scene is calculated in the breadth-first order, the calculation process can be regarded as the generation process of the tree structure rooted at the seed point. If the affinity value that needs to be propagated outward is indicated in a, then when a is passed to the edge of the block, the problem of competition is caused by the need to propagate the affinity value across the block. Since the seed point propagates outward in the tree during the propagation process, the a between the block and the block does not appear to be a loop, otherwise it contradicts the tree propagation.

From the above analysis, it can be concluded that in theory, three times correction iterations are needed to completely eliminate the influence of the thread block edge point competition. However, after experimentation, two times iterations are enough to eliminate the error caused by the transfer between such threads. Therefore, the algorithm in this paper combines the two steps in CUDA-kFOE into one step, and also reduces the time for data exchange between the host and the device in the iterative operation. The flow of the improved CUDA-kFOE is shown in FIG. 3.

3.4 Vascular skeleton line extraction

The binary refinement algorithm is usually used to find the center line or skeleton line of the target in the image. The main central idea is to continually iteratively etch the surface of the target until the target has only the skeleton line. In order to ensure the center position of the skeleton line and the connectivity of the target body, the corrosion must be performed symmetrically. In order to prevent voids in the target body, the corrosion operation must be very cautious. At present, the two main image refinement methods are: 1) based on decision tree method (Lee et al, 1994) , 2) based on refinement kernel filtering (Jonker et al, 2000). This section proposes a refined kernel filtering based algorithm for CUDA for the second method.

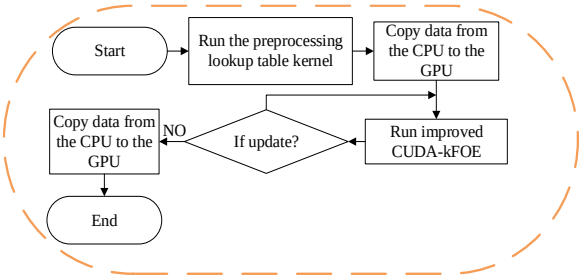

Figure 3. Improved CUDA-kFOE algorithm flow

Most 3D refinement algorithms are considered to be "parallel" refinement algorithms, but "parallelization" does not specify execution on the CPU or GPU. When all of these voxels satisfy a deletion condition at the same time, in the single iteration of the refinement algorithm, the order of some voxel deletions will affect each other, so that these "parallel" ways of deleting voxels will cause the 3D model topology (Kuba et al, 1998). Therefore, it is very important to use some different strategies (Kong et al, 1995) to avoid this kind of out-of-synchronization:

1. 1. k-direction refinement algorithm: Each global iteration is divided into several sub-iterations and the template of the template is changed. Usually the $3 \times 3 \times 3$ neighbor position of a voxel will determine the result of the template.
2. Sub-domain-based refinement algorithm: This method is different from 1). The method consists in dividing the 3D model into several sub-domains, and removing some voxels contained in these sub-fields during each sub-iteration.

It is easy to create a binarized kernel function that is responsible for determining voxels by each thread. As this step is executed, the entire program model enters the main loop, and the next step will be the core step of the algorithm, which is the loop that handles the sub-iterations.

As we all know, communication between CPU and GPU requires a lot of computing resources (bandwidth), so the basic principle of GPGPU programming is to put most of the computing information into GPU memory, so that a lot of data can be avoided through PCI-E. Time dissipation due to information transfer on the CPU and GPU side. Similarly, in order to reduce the time taken by the GPU to launch the calling kernel, the three sub-iteration steps obtained from the above analysis are merged into two kernels for processing. In these 2 kernels, each thread is responsible for determining whether a voxel is a boundary point,

and if it is a boundary point, it generates rotation and mapping of its 3 × 3 × 3 neighbors in a specific direction, and then matches 14 templates (only You need to match any template successfully), and finally delete all the simple points.

It is noted that the main purpose of shared memory in the GPU is to reduce the amount of data that is repeatedly accessed in global memory and copy the data into shared memory to reduce the extra data access overhead. Because in the kernel function, if global memory is used as the intermediate storage station, the global memory is accessed once when the boundary point neighbor is read, then the neighbor index is converted and written to the global memory once, and finally the converted neighbor needs to be read. And repeatedly read and write global memory, such as matching with the template, which will inevitably lead to additional data access overhead.

Another noteworthy thing is that in order to avoid data inconsistency, a double cache strategy will be adopted to ensure data consistency. If only one cache is used, it may cause unpredictable results when the neighbor point determined by thread B between thread A and black dot p contains this point p. For this reason, two pieces of memory will be requested in global memory, where cache A is used to store 3D images and cache B is used to store its copy results. In the kernel function, the voxel and neighbor information is read from cache A, and the template is parsed and tried, and finally the simple point of relative should be deleted in cache B. In the next sub-iteration, the two caches can be exchanged.

4. RESULTS AND DISCUSSION

In this experiment, three CT datasets were subjected respectively to liver extraction to obtain a liver portion. Figs. 4 are the original three sets of CT images, and Fig. 5 is the three sets of liver contours extracted by the CPU and GPU respectively. As can be seen from the figures, there is almost no difference in the outline of the liver extracted by the CPU and GPU. Fig. 6 shows a comparison diagram of the three sets of data after the fuzzy connection processing of Fig. 6 is performed before and after the threshold processing. Fig. 7 is a binarized image obtained by reducing the blur connection of Fig. 6.

4.1 Hepatic segmentation

In terms of data, semi-automated liver contour segmentation was performed using multiple sets of abdominal CT slices in different regions. The image size was 512×512.

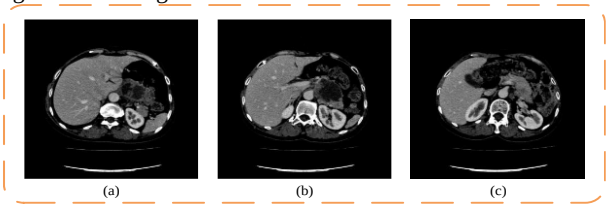

Figure 4. Three CT datasets: (a) I, (b) II, (c) III.

As shown in FIG. 4, the liver contours of the three CT data are arranged from the largest to the smallest, but from the acceleration ratio in the table 1, the acceleration performance of the GPU decreases as the area decreases. This is because the information exchange with the CPU during the GPU algorithm operation process, this part of the time as the scale of the operation decreases, the proportion of the total time will become larger and larger, so there is the speedup ratio of GPU and CPU. Increase as the liver contour increases.

As shown in Figs. 5, (a), (c), and (e) are the results of calculation by the CPU, and (b), (d), and (f) are the results of calculation using the GPU. It can be seen from the figure that there is no difference in the outline of the liver, but there are subtle differences in internal details. Because the GPU is limited to double-precision floating-point operations, so it takes a single-precision floating-point operation and is different from the CPU instruction set in rounding.

Table1. GPU vs. CPU time for 3 different CT datasets

CT data	Seed point coordinates	Stop time (*ms*)	CPU-FMM (*ms*)	GPU (*ms*)	Speedup ratio
1	(129,21)	600	15.2	3.2	4.75
2	(123,245)	450	12.7	3.3	3.85
3	(122,254)	300	7.3	2.9	2.52

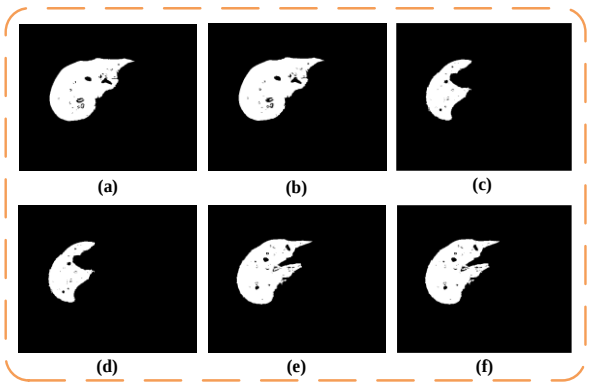

Figure 5. GPU vs. CPU segmentation results

4.2 Vascular extraction

After experimental comparison, as shown in Tables 2 - 4, after experimental comparison, the experimental data after iterative correction is almost identical to the fuzzy connection result of the serial version of the CPU. Although the original CUDA-kFOE algorithm with a slightly iterative correction is slightly behind in speed, it still can improve the speed of the original CPU algorithm. After the fuzzy scene is calculated, a threshold is required to be binarized to produce the final processing result. The experimental results are shown in Fig. 6, where (a), (c), and (e) are the results of no threshold processing, and (b), (d), and (f) are the processing results with a threshold of 0.5.

4.3 Vascular skeleton line extraction

As show in FIG. 7, the test data part of this step is the binary image obtained by the fuzzy connection refinement in the previous step, and the second seed point of the first group and the third seed point of the second group in the fuzzy connection segmentation result are extracted as a test. The experimental data uses the unsigned char data type, in which the voxel value of the blood vessel is 255 and the voxel value of the background point is 0. Comparing the GPU parallel computing time with the time efficiency of the CPU serial operation, the acceleration effect of about 10 times is obtained, which enables real-time display refinement. The experimental results are shown in Table 5.

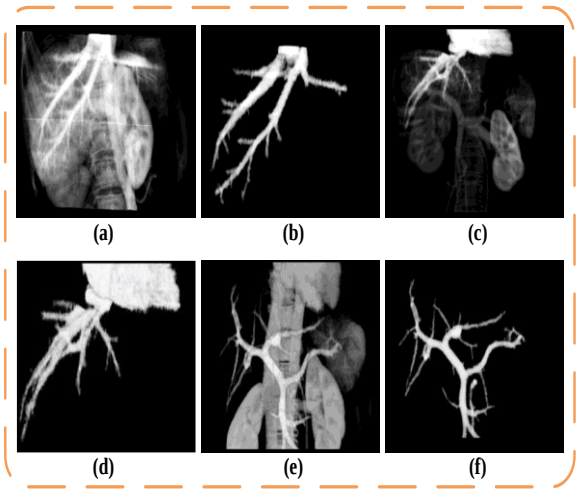

Figure 6. Test results obtained using three datasets

4.4 Conclusion

In this work, we propose an improvement of the FMM algorithm, the CUDA-based fuzzy connected algorithm, and the CUDA-based 12-subiterative thinning algorithm, so that it can run better under the GPU framework than CPU. Firstly, the FMM algorithm is improved to be more suitable for running in the GPU In this work, we propose an improvement of the FMM algorithm, the CUDA-based fuzzy connected algorithm, and the CUDA-based 12-subiterative thinning algorithm, so that it can run better under the GPU framework than CPU. Firstly, the FMM algorithm is improved to be more suitable for running in the GPU and used for liver segmentation. Then the CUDA-based connected algorithm is improved to improve the accuracy while the run time is not greatly extended. Finally, Through the CUDA-based 12-subiterative thinning algorithm, the skeleton line of the target body can be extracted from the 3D image, and compared with the time efficiency of the serial operation of the CPU, an acceleration effect of about 10 times is obtained.

Table 2. CT dataset I

Seed point	Serial algorithm runtime /s	CUDA-kFOE algorithm			Ours algorithm		
		operation hours /s	Speedup ratio	Error points	operation hours /s	Speedup ratio	Error points
(197, 258, 123)	281.6	63.8	4.41	762	72.3	3.89	0
(163, 187, 86)	283.8	60.6	4.68	616	71.0	4.00	0
(196, 188, 110)	247.9	56.2	4.41	956	66.0	3.76	0

Seed point	Serial algorithm runtime /s	CUDA-kFOE algorithm			Ours algorithm		
		operation hours /s	Speedup ratio	Error points	operation hours /s	Speedup ratio	Error points
(183, 279, 76)	288.8	63.9	4.52	3560	74.8	3.86	2
(167, 257, 107)	267.4	63.5	4.21	1147	74.0	3.61	0
(158, 247, 148)	230.5	**61.9**	**3.72**	**526**	**70.8**	**3.26**	**0**

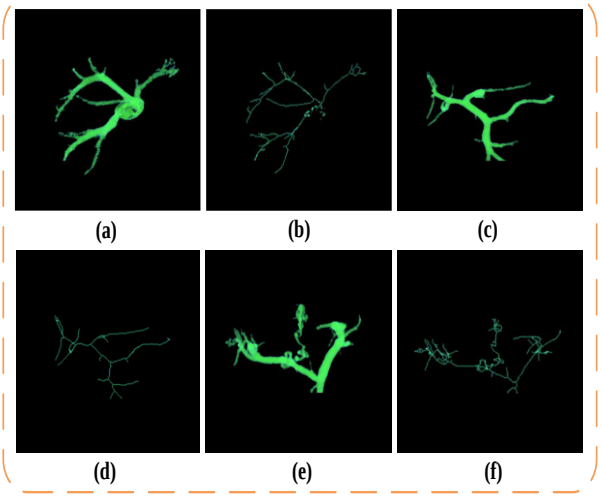

Figure 7. Comparison of results before and after refinement of three datasets

Table 3. CT dataset II

Table 4. CT dataset III

Seed point	Serial algorithm runtime /s	CUDA-kFOE algorithm			Ours algorithm		
		operation hours /s	Speedup ratio	Error points	operation hours /s	Speed up ratio	Error points
(189, 244, 180)	245.5	73.3	3.47	722	81.8	3.11	0.7
(144, 239, 147)	340.1	60.6	5.61	1147	84.2	4.04	0.7
(107, 223, 117)	386.7	73.4	5.27	526	86.8	4.46	0

Table 5. CPU vs. GPU refinement time

Data size	CPU serial time /s	GPU serial time /s	Speedup ratio
512×512×155	27.5	2.30	12.0
512×512×175	22.7	1.64	13.8
512×512×135	13.3	1.15	11.6

5. ACKNOWLEDGEMENT

This work was supported by National Natural Science Foundation of China (Grant No. 61601392 and No.41871380).

6. REFERENCES

Chowriappa A, Seo Y, Salunke S, et al. 3-D vascular skeleton extraction and decomposition. *IEEE Journal of Biomedical and Health Informatics*, 18(1), pp. 139-147, 2014.

Eapen M, Korah R, Geetha G. Computerized liver segmentation from CT Images using Probabilistic Level Set Approach. *Arabian Journal for Science and Engineering*, 41(3), pp. 921-934, 2016.

Rouy E. and Tourin A, A viscosity solutions approach to shape-from-shading. *SIAM Journal of Numerical Analysis*, 29:867–884, 1992.

Li G, Chen X, Shi F, et al. Automatic liver segmentation based on shape constraints and deformable graph cut in CT images. *IEEE Transactions on Image Processing*, 24(12), pp. 5315-5329, 2015.

Wan S Y, Kiraly A P, Ritman E L, et al. Extraction of the hepatic vasculature in rats using 3-D micro-CT images. *IEEE Transactions on Medical Imaging*, 19(9), pp. 964, 2000.

Zhang Q, Fan Y, Wan J, et al. An efficient and clinical-oriented 3D liver segmentation method. *IEEE Access*, 5, pp.18737-18744, 2017.

Eklund A, Andersson M, Knutsson H. fMRI analysis on the GPU—possibilities and challenges. *Computer methods and programs in biomedicine*, pp.105(2): 145, 2012.

Yuen S G, Kesner S B, Vasilyev N V, et al. 3*D* ultrasound-guided motion compensation system for beating heart mitral valve repair. *Medical Image Computing and Computer-Assisted Intervention–MICCAI. Springer Berlin Heidelberg*, pp.711-719, 2008.

Zhuge Y, Cao Y, Miller R W. GPU accelerated fuzzy connected image segmentation by using CUDA, *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. IEEE, pp. 6341-6344,2009

Rumpf M, Strzodka R. Level set segmentation in graphics hardware, *Image Processing, 2001. Proceedings. 2001 International Conference on. IEEE*, pp. 3: 1103-1106, 2001.

Erdt M, Raspe M, Suehling M. Automatic hepatic vessel segmentation using graphics hardware. *Medical imaging and augmented reality. Springer Berlin Heidelberg*, pp. 403-412, 2008.

Pan L, Gu L, Xu J. *Implementation of medical image segmentation in CUDA, Information Technology and Applications in Biomedicine, 2008. ITAB 2008. International Conference on. IEEE*, pp.82-85,2008.

Strzodka R, Ihrke I, Magnor M. A graphics hardware implementation of the generalized Hough transform for fast object recognition, scale, and 3D pose detection, Image Analysis and Processing, *2003. Proceedings. 12th International Conference on. IEEE*, pp. 188-193, 2003.

Schwarzkopf A, Kalbe T, Bajaj C, et al. Volumetric nonlinear anisotropic diffusion on GPUs, *Scale Space and Variational Methods in Computer Vision. Springer Berlin Heidelberg*, pp. 62-73,2011.

Xu R, Pattanaik S. Non-iterative, robust Monte Carlo noise reduction. *IEEE computer graphics and applications*, pp. 25(2): 31-35,2005

Sethian J A. Level set methods and fast marching methods. *Journal of Computing and Information Technology*, pp. 11(1): 1-2,2003.

Udupa J K, Samarasekera S. Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation. *Graphical models and image processing*, pp. 58(3): 246-261,1996.

Lee T C, Kashyap R L, Chu C N. Building skeleton models via 3-D medial surface axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, pp. 56(6): 462-478,1994.

Jonker P P. Morphological operations on 3D and 4D images: From shape primitive detection to skeletonization, *Discrete Geometry for Computer Imagery. Springer Berlin Heidelberg*, pp. 371-391,2000.

Pal ágyi K, Kuba A. A 3D 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recognition Letters*, pp. 19(7): 613-627,1998.

Liu C, Cui D, Shi D, et al. Automatic Liver Segmentation in CT Volumes with Improved 3D U-net, *Proceedings of the 2nd International Symposium on Image Computing and Digital Medicine*. ACM, pp. 78-82,2018.

Alirr O I, Rahni A A A, Golkar E. An automated liver tumour segmentation from abdominal CT scans for hepatic surgical planning. *International journal of computer assisted radiology and surgery*, pp. 13(8): 1169-1176,2018.

Lebre M A, Vacavant A, Grand-Brochier M, et al. Automatic 3-D Skeleton-Based Segmentation of Liver Vessels from MRI and CT for Couinaud Representation, *2018 25th IEEE International Conference on Image Processing (ICIP). IEEE*, pp. 3523-3527,2018

Matthies P, Wuestemann J, Pinto F A, et al. First Validation of Semi-automatic Liver Segmentation Algorithm, *World Congress on Medical Physics and Biomedical Engineering 2018. Springer, Singapore*, pp. 279-282,2019.

Bal E, Klang E, Amitai M, et al. Automatic liver volume segmentation and fibrosis classification, *Medical Imaging 2018: Computer-Aided Diagnosis. International Society for Optics and Photonics*, pp. 10575: 1057506, 2018.

Pan Q, Zhang L, Xia L, et al. Liver tumor segmentation based on level set, Tenth International Conference on Digital Image Processing (ICDIP 2018). *International Society for Optics and Photonics*, pp. 10806: 108062N,2018.