Fast 3D Structure Localization in Medical Volumes using CUDA-enabled GPUs

Sharan Vaswani, Rahul Thota, Nagavijayalakshmi Vydyanathan and Amit Kale
Siemens Corporate Research and Technologies
Bangalore, India
Email: {sharan.vaswani, rahul.thota, nagavijayalakshmi.vydyanathan, kale.amit}@siemens.com

***Abstract*—Effective and fast localization of anatomical structures is a crucial first step towards automated analysis of medical volumes. In this paper, we propose an iterative approach for structure localization in medical volumes based on the adaptive bandwidth mean-shift algorithm for object detection (ABMSOD). We extend and tune the ABMSOD algorithm, originally used to detect 2D objects in non-medical images, to localize 3D anatomical structures in medical volumes. For fast localization, we design and develop optimized parallel implementations of the proposed algorithm on multi-cores using OpenMP, and on GPUs using CUDA. We evaluate the quality, performance and scalability of the proposed algorithm on Computed Tomography (CT) volumes for various structures.**

***Index Terms*—localization; medical image processing; GPGPU computing; CUDA**

I. Introduction

Automation of clinical procedures involving analysis of imaging data, such as tissue volume quantification, screening, diagnosis as well as surgical procedures, not only helps to improve patient throughput but also enhances repeatability, safety and quality of patient care. Typically, analysis of medical imaging data includes operations such as image segmentation, registration, feature extraction, recognition and classification. As medical images suffer from inherent noise and low contrast and spatial resolution [1], accurate segmentation, registration or classification is difficult and computationally intensive. For example, several 3D anatomy segmentation and recognition algorithms take several minutes for execution even with GPU acceleration [2], [3], [4]. In addition, these algorithms are highly tuned and specific to an anatomical structure, like the lung or liver. To address the above issues, a generic preprocessing step that localizes any structure can be very useful in improving both speed and accuracy of the above procedures. For example, high precision segmentation of tumors can be accomplished faster by executing complex domain-specific segmentation algorithms on a localized region around the tumour, rather than the entire volume.

Localization can be used to improve the speed and quality of diagnosis for difficult cases. A doctor can scan his past patient data to retrieve a subset of imaging records that comprise of the structure of interest. Domain-specific algorithms can then be run on localized regions in the relevant records to identify similar cases quickly, which the doctor can consult before making critical diagnoses. Localization can also be applied to track anatomical structures in image-guided surgical procedures. Thus localization can play a crucial first step in automated analysis of medical imaging data.

In this paper, we propose an iterative approach for structure localization in medical volumes that is based on the adaptive bandwidth mean-shift algorithm for object detection (ABMSOD) [5]. We extend and tune the ABMSOD algorithm, originally used to detect 2D objects in non-medical images, to localize 3D anatomical structures in medical volumes. The ABMSOD algorithm is an iterative meanshift based object detection algorithm which estimates the position as well as the size and orientation of the target object in the given image. ABMSOD estimates the object position using conventional meanshift and the object scale and orientation using an adapative bandwidth for the meanshift kernel. To enable fast localization of structures, we develop optimized parallel implementations of our localization technique on multi-cores using OpenMP and graphics processors using CUDA.

We evaluate the quality, performance and scalability of our algorithm on Computed Tomography (CT) volumes for the following structures: brain stem, eye and the parotid gland. Our evaluations show that in 40% of the runs, we are able to encapsulate more than 90% of the structure, while in 65% of the runs we are able to capture the structure partially with atleast 50% of coverage. As our proposed technique is generic enough to accomodate any target description, our future work is to experiment with different target descriptors trading off between localization accuracy, speed and flexibility. The GPU-acceleration of the proposed algorithm yields a 97x speedup over the sequential implementation and a 28x speedup over the OpenMP parallel implementation, and scales well with increasing dimensions of the search space.

II. Related Work

Localization and segmentation of anatomies has been a primary focus area of research over the recent years. Localization techniques are typically based on either boundary delineation methods using active shapes, appearances [6], [7] and deformable models [8], [9] or through machine learning techniques [10], [11], [12], [13], [14], [15]. Active shapes and appearance models have been used to perform 2D segmentation in medical images that have a fairly consistent shape and gray level appearance [16], [17], [18]. However, these techniques require extensive apriori knowledge of the structures and intensive training to build robust models, especially for 3D segmentation where the gray levels and the boundary appearances may vary largely [19]. Also, these works focus on specific imaging modalities. Other model-based localization techniques, including deformable models [8], [9] also require extensive training and are structure specific.

Recently, marginal space learning has proved a successful technique in automatic detection of 3D structures in medical images [10], [11]. Here, localization is modeled as a classification problem and structures are identified by performing parameter estimations in a series of marginal spaces with increasing dimensionality rather than scanning the classifer exhaustively over the entire search space. Constrained marginal search exploits correlation amongst the parameters to reduce the search space and quicken convergence. Criminisi et. al. [12] propose a random decision forest based classifier to detect multiple organs in CT images. These works which are based on classification, involve a computationally intensive training phase that requires a large training data set. Pauly, Criminisi and others, proposed an approach to detect the position of multiple objects in MR images using a single fern-based regressor [14], which facilitates faster training. This work introduces features based on MR Dixon channels and is tightly coupled to the MR modality. They use cuboids to localize anatomies. Zhou et. al. [13], [15] describe an ensemble learning based approach for detecting solid objects in CT images. This approach requires lesser number of data sets for the training phase. It detects 3D objects by applying 2D detection techniques across slices in all three dimensions of the image volume, which makes the algorithm computationally intensive (approximately 15 seconds per CT scan), and uses cuboids to localize structures. Apart for the works described above, several researchers have proposed algorithms to localize and segment specific organs by applying domain specific knowledge [10], [20], [21], [22].

All of the above works are either domain specific, computationally intensive or require extensive training. Some works use cuboids for localization which does not tightly capture the orientation and shape of irregular structures. In this paper, we extend the work of Chen et. al. [5] to develop a generic framework that can be used to detect any 3D structure in a medical volume without the need for intensive training or apriori domain specific knowledge. Our algorithm is parallelized on multi-core CPUs and GPUs to enable fast localization.

III. Background

A. Adaptive Bandwidth Mean-shift Algorithm

The Adaptive Bandwidth Meanshift Object Detection (ABMSOD) algorithm is an iterative meanshift based algorithm for 2D object detection in computer vision. The mean shift algorithm is an unsupervised method commonly used in computer vision problems such as filtering, tracking and segmentation. In the mean shift procedure the feature points move toward some significant modes and cluster themselves automatically making it ideal for multi-object detection and localization. The ABMSOD algorithm uses kernel weighted feature histograms (features could be color, texture or haar-like features) to describe the target object and candidate object models. Given a test image and the target kernel-weighted feature histogram, ABMSOD tries to identify the optimum position, scale and orientation of the target in the test image. It follows a two step approach - in the first step, it searches through the whole image to identify rough positions of possible candidate objects. This is done by randomly scattering ellipstical windows all over the image and computing the similarity between the feature histogram of these windows with that of the target using the Bhattacharyya co-efficient, which is defined as

$$\rho(x) = \sum_{u=1}^{M} \sqrt{p_u(x) q_u} \tag{1}$$

where, p and q are the candidate and target histograms respectively, M is the total number of bins in the histogram and x is the position of the candidate window. Only window regions with similarity above a threshold is considered as possible candidate objects. In the second step, for each possible candidate object identified, the optimum position, scale and orientation maximizing the similarity between the target histogram and the candidate histogram, is computed.

Conventional meanshift is used for optimum position estimation. The meanshift procedure assigns weights to the pixels in each elliptical candidate window. These weights depend on the feature value as well as the distance of the pixel from the ellipse centre. The distances are usually weighted by a kernel function like the epanichekov or gaussian kernel [23]. Meanshift finds the best next position of the window in every iteration using the pixel weights. To estimate the optimum axes lengths and orientation, the optimum bandwidth matrix, which encodes the scale and orientation parameters of the candidate ellipse, is computed at each iteration using the equations in [23]. Thus ABMSOD iteratively finds the best position, scale and orientation of the target object in a given image.

B. GPGPU Computing and CUDA

The GPU is a data-parallel computing device consisting of a set of multiprocessing units (SM), each of which is a set of SIMD (single instruction multiple data) processing cores. For example, the Quadro FX 5600 GPU has 16 multiprocessing units, each having 8 SIMD cores, resulting in a total of 128 cores. Each SM has a fixed number of registers and a fast on-chip memory that is shared among its SIMD cores. The different SMs share a slower off-chip device memory. Constant memory and texture memory are read-only regions of the device memory and accesses to these regions are cached. Local and global memory refers to read-write regions of the device memory and its accesses are not cached. The Compute Unified Device Architecture (CUDA) [24] is a C-based programming model from NVIDIA that exposes the parallel capabilities of the NVIDIA GPU for general purpose computing.

In the CUDA context, the GPU is called device, whereas the CPU is called host. Kernel refers to an application that is executed on the GPU. A CUDA kernel is launched on the

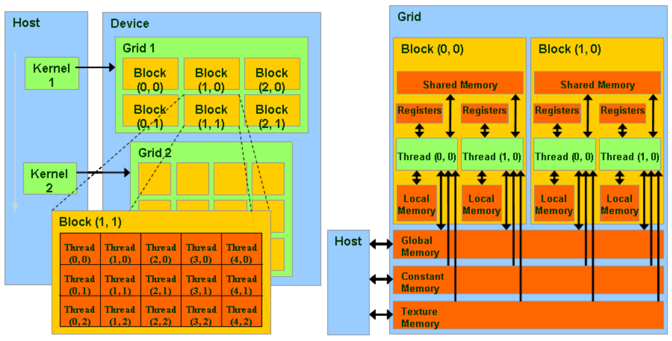


Fig. 1. CUDA Programming and Memory Model. (Courtesy: NVIDIA)

GPU as a grid of thread blocks. A thread block contains a fixed number of threads and can span in one, two or three dimensions. A grid can span in one or two dimensions. Figure 1 shows an example of a kernel launched as a two dimensional 3×2 grid of two dimensional 5×3 thread blocks. Threads are uniquely identified based on their block index and thread index within the block. A thread block is executed on one of the multiprocessors and multiple thread blocks can be run on the same multiprocessor. Consecutive threads of increasing thread indices in a thread block are grouped into what are known as warps which is the smallest unit in which the threads are scheduled and executed on a multiprocessor.

IV. 3D Structure Localization Algorithm

To detect 3D structures in medical volumes, we extend the ABMSOD algorithm (described in Section III-A), which was originally proposed to detect 2D objects in the computer vision domain. We extend the algorithm to perform 3D localizations and adapt it for medical image processing. As a first step to the extension, we derive the expression for the 3D bandwidth matrix(H), which encodes the scale and orientation parameters of the candidate ellipsoid. There are 9 independent parameters of the candidate ellipsoid - the 3D coordinates of the center of the ellipsoid, the lengths of the 3 axes - a,b,c and the orientation of the ellipsoid about the X, Y and Z axes - α, β, and γ. The H matrix, therefore, is given by

$$H = AA^T; A = R(\alpha, \beta, \gamma) \times D(a, b, c) \tag{2}$$

Here, R denotes the rotation matrix and D denotes the diagonal matrix. R is defined as

$$R = R_x(\alpha) \times R_y(\beta) \times R_z(\gamma) \tag{3}$$

where R_x, R_y and R_z represent the rotation matrices about the X, Y and Z axes respectively. D is given by

$$D = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \tag{4}$$

At every iteration, we scale the parameters of the ellipsoid by a factor σ following the intuition of Ning et. al. [25] to search in a region slightly bigger than that obtained from the previous iteration so as to capture more of the local context around the search window. With this scaling, the equation of the ellipsoid is given by

$$S = \{s|(x-s)^T H^{-1}(x-s)\} \leq \sigma^3 \tag{5}$$

Here, S is the set of all points that lies within the ellipsoid, centered at x, with a bandwidth matrix of H. The lengths of the three axes of the ellipsoid are scaled by a factor σ.

The feature values of the points in S are used for the histogram calculation. The candidate histogram for an ellipsoid described by the bandwidth matrix H and centred at the point x is given by

$$p_u(x) = C_H \sum_{s \in S_0} |H|^{-1/2} K((x-s)^T H^{-1}(x-s))\delta[b(s) - u] \tag{6}$$

where $u = 1, 2...M$ denotes the histogram bins, $b(s)$ denotes the bin number to which the feature of pixel s lies, C_H is the normalization constant, and δ is the Kronecker delta function. Any feature that can be represented by a histogram such as the pixel intensity, histogram of gradients or local binary patterns, can be used. The target histogram q is constructed using the ground truth data of the structure to be localized. It represented in a similar way as the candidate histogram. We also measure the center of the structure to be localized and the scale and orientation parameters of the closest fitting ellipsoid that encloses the structure in the training volumes and use this to initialize the search space. In our experiments, we used one volume for each structure for training.

The search space is defined by a bounded region around the expected center of the structure (as computed through the training procedure described above). The dimensions of the search space should be large enough to account for the maximum variance in the structure positions. Our algorithm works by scattering random points within the search space. Each of these points marks the center of a candidate ellipsoid. The axes lengths of the candidate ellipsoids are intialized using the values obtained through training along with a certain variance. All orientations are initialized to zero. This constitutes the intial H matrix. To fix the value of σ, the algorithm is run on the train volume using different values of σ and the value which gives the tightest enclosing ellipsoid is chosen.

As mentioned in Section III-A, the ABMSOD algorithm uses meanshift for position estimation.At each iteration of meanshift, the next position for the search window depends on the coordinates and weights assigned to the pixels in the ellipse. It is given by:

$$\Delta \mathbf{x} = \frac{\sum_{s \in S} G_H(x-s) w(s) \mathbf{s}}{\sum_{s \in S} G_H(x-s) w(s)} \tag{7}$$

where K is the kernel function used to weigh the pixel contributions to the histogram and $w(s)$ is the weight assigned to each pixel s in the search window
We use the Gaussian function as our kernel function, i.e $K(x) = c \exp \frac{-x}{2}$ The function $w(s)$ is defined as $w(s) = \sum_1^M \sqrt{q_u/p_u(x)}\delta[b(s) - u]$ and $G_H(x) = -K'_H(x)$

ABMSOD derives the expression for the optimum 2D H matrix for estimating the scale and orientation at each iteration. We derive the optimum H matrix for any N dimensional super-ellipsoid. It is shown by Chen et. al. [23] that maxmization of the Bhattacharyya coefficient is equivalent to the maximization of the following function

$$f(x, H) = |H|^{-1/2} \sum K((x-s)^T H^{-1}(x-s))w(s) \tag{8}$$

To derive the optimum N-dimensional H matrix, we fix the position x of the ellipsoid in Equation 8. Replacing K by the gaussian kernel equation, we get

$$f(H) = c|H|^{-1/2} \sum \exp -(x-s)^T H^{-1} \frac{(x-s)}{2} w(s) \tag{9}$$

By taking logarithm on both sides and by using Jensen's inequality, we have

$$ln(f(H)) \geq \sum ln(c) + \frac{1}{2} ln(H^{-1}) + \frac{((x-s)^T H^{-1}(x-s))w(s)}{2} \tag{10}$$

Let L denote the RHS of the above equation. Differentiating L with respect to H^{-1} and setting it to zero, we derive

$$H = \frac{\sum_{s \in S}(x-s)(x-s)^T w(s)}{\sum_{s \in S} w(s)} \tag{11}$$

Equation 11 is used to update the H matrix in every iteration.

The algorithm is terminated after a fixed maximum number of iteration or when the Bhattacharyya coefficient saturates. This iterative procedure is executed for each of the initial points of the search space. Thus, each of these points converges to a local maxima giving the optimum position and ellipsoid parameters in the neighbourhood of the initial point. The ellipsoid with the maximum Bhattacharyya coefficient localizes the structure most accurately and is the output of the algorithm. Sufficient number of points is necessary to ensure convergence to the correct solution.

A. Parallel 3D Localization on Multi-cores

The iterative search for the optimum position, shape and orientation for each initial random point is independent of each other. Hence, for a 'n' core processor, we spawn 'n' threads, where each thread simultaneously performs the iterative search procedure for an initial random point. Since the computations for each point are independent, there is no need for sharing memory among the threads. Each thread has local memory for the meanshift algorithm specific calculations. We thus use the OpenMP 'parallel for' construct to distribute the points amongst the threads. As each search path can take different times for convergence, the 'schedule dynamic' clause is used to dynamically load balance the distribution of points amongst the threads. This results in a speed-up of 3.5x for the parallel implementation on the CPU.

B. Parallel 3D Localization on GPUs

We design and implement the parallel variant of the localization technique on GPUs using the CUDA programming model [24]. As explained in Section III-B, CUDA exposes two levels of parallelism - a coarser level using thread blocks and a finer level using threads within a thread block. The independent exploration of different search paths originating from each initial random point is distributed amongst thread blocks. In addition, using the finer level of parallelism offered by threads within a thread block, we further parallelize operations within each search iteration. We make the threads in a thread block handle computations for a subset of the voxels from a cube which completely encloses the ellipsoid. Computations such as the application of the kernel function, construction of the candidate histogram, weight assignment to the voxels, H matrix computation etc. are all done in parallel where each thread is responsible for a set of voxels. Summation of values across threads is performed through parallel reduction.

Algorithm 1 depicts the pseudo-code for the GPU accelerated 3D localization algorithm. To reduce the synchronization operations among threads during histogram computation, we allow each thread to construct a local histogram of the voxels handled by that thread. These histograms are stored in shared memory for fast access. After all local histograms are constructed, the histograms are bin wise aggregated by the threads in parallel to form the global candidate histogram. The CT volume is stored in a 3D texture and the access to the volume is ensured to be in a way that maximizes spatial locality and efficiently utilizes the texture cache. Constant variables like σ and the target histogram are stored in constant memory to utilize the constant cache. Data shared by threads within a block like the H matrix, local histograms etc. are stored in shared memory for fast retrieval through the broadcast mechanism supported by CUDA. Enough number of threads are launched to keep all the cores of each streaming multiprocessor (SM) busy. We try and maximize the occupancy for each SM. The occupancy is however limited by the amount of shared memory and the number of registers available per SM. We also ensure that there is no register spilling. The kernels are designed to minimize the warp divergence and maximize the Instructions per cycle (IPC). In addition all global memory accesses (loading the initial ellipsoid parameters and storing back the final optimum parameters for each ellipsoid) are coalesced. This gives us a speed-up of 97x for the GPU implementation of the algorithm.

V. Performance Evaluation

The proposed structure localization algorithm was evaluated on a system with an Intel(R)Xeon(R) X5450 quad-core processor clocked at 3 GHz and with 3.25 GB RAM. The system included a NVIDIA Tesla C2050 GPU as a PCI-express device. This GPU has 14 multi-processors each having 32 cuda cores, resulting in a total of 448 cuda cores. The cores are clocked at 1.15 GHz. Each multi-processor has 48 KB of shared memory and 32 K registers. The GPU device has 3 GB of device memory. The Tesla C2050 GPU has a compute capability version of 2.0 and cuda toolkit and sdk version 4.0 was used to develop and execute CUDA kernels on the device.

We implemented three variants of the structure localization algorithm - a sequential version in C, a OpenMP-based parallel version that utilizes the 4 cores of the Xeon X5450 processor,

Algorithm 1 GPU accelerated 3D localization algorithm

```
1: Initialize randomly the postions $x$ of N ellipsoids within a fixed search space centred about the structure location in the target
2: Initialize randomly the scales with a certain variance about the scales of the target structure
3: Initialize all orientations to zero and construct the bandwidth matrix according to Equations 2 3 4
4: for each candidate ellipsoid $S_c$ centered at $x$ with a bandwidth matrix $H$ do    ▷ Perform the iterative search for the optimum position, scale and orientation for each candidate ellipsoid
5:     $iter\_cnt \leftarrow 1$
6:     $bhatcf \leftarrow 0$    ▷ Initialize Bhattacharyya coefficient
7:     $max\_bhatcf \leftarrow 0$    ▷ Maximum Bhattacharaya coefficient across all iterations
8:     $delta\_bhatcf \leftarrow THRESHOLD$
9:     $x_{opt} \leftarrow x$
10:    $H_{opt} \leftarrow H$
11:    while ($delta\_bhatcf \leq THRESHOLD$ and $iter\_cnt < MAX\_ITERATIONS$) do    ▷ Termination criteria for the meanshift search
12:        for all CUDA threads $t \in$ CUDA thread block $B_c$ do    ▷ One CUDA thread block is launched for each candidate ellipsoid
13:            $V(t) \leftarrow$ set of voxels handled by thread $t$
14:            for each voxel $v \in V(t)$ that satisfies Equation 5 do    ▷ for every voxel inside the candidate ellipsoid
15:                $LCH_t(b_v) \leftarrow LCH_t(b_v) + c\exp\frac{-d_v}{2}$    ▷ $LCH$ is the partial candidate histogram local to each thread, $b_v$ is the bin index for voxel $v$ and $d_v$ is the distance of voxel $v$ from the center of the ellipsoid
16:            $GCH_c \leftarrow$ global candidate histogram    ▷ all threads do a parallel reduction to aggregate the local histograms and form the global histogram
17:            for each voxel $v \in V(t)$ that satisfies Equation 5 do    ▷ for every voxel inside the candidate ellipsoid
18:                $w_v \leftarrow \sqrt{\frac{GCH_c(b_v)}{TH(b_v)}}$    ▷ A weight is computed for each voxel as the ratio of the bin heights of the candidate histogram and the target histogram. $TH$ refers to the target histogram
19:                $\delta_{xv} \leftarrow G_H(v) w_v s_v$    ▷ $\delta_{xv}$ denotes the contribution of the voxel $v$ to the change in the position of the ellipsoid and is computed according to Equation 7
20:            $x_{new} \leftarrow \frac{\sum_{\forall v} \delta_{xv}}{\sum_{\forall v} G_H(v) w_v}$    ▷ $x_{new}$ is the new position of candidate ellipsoid and all threads do a parallel reduction to compute the summations required in this step
21:            for each voxel $v \in V(t)$ that satisfies Equation 5 do    ▷ for every voxel inside the candidate ellipsoid
                   Repeat Steps 15 to Step 18 for the candidate ellipsoid $S_c$ centered at $x_{new}$
22:                $\delta_{Hv} \leftarrow (x_{new} - s_v)(x_{new} - s_v)^T w_v$    ▷ $\delta_{Hv}$ denotes contribution of a voxel to the H matrix
23:            $H_{new} \leftarrow \frac{\sum_{\forall v} \delta_{Hv}}{\sum_{\forall v} w_v}$    ▷ $H_{new}$ is the optimum H matrix at the new position of the candidate ellipsoid and all threads do a parallel reduction to compute the summations required in this step
24:            $bhatcf \leftarrow \sqrt{(GCH_c)(TH)}$
25:            $delta\_bhatcf \leftarrow max\_bhatcf - bhatcf$
26:            if $bhatcf > max\_bhatcf$ then
27:                $max\_bhatcf \leftarrow bhatcf$
28:                $x_{opt} \leftarrow x_{new}$
29:                $H_{opt} \leftarrow H_{new}$
30:            $iter\_cnt \leftarrow inter\_cnt + 1$
31: Among the N final candidate ellipsoids, output $x_{opt}$ and $H_{opt}$ corresponding to the ellipsoid with the maximum Bhattacharyya coefficient.
```

and a CUDA-based GPU accelerated version. We evaluate the quality and accuracy of the proposed algorithm as well as the performance and scalability for the three variants described above.

The algorithm was tested on 3 structures in CT volumes - brain stem, eye and the parotid gland. We tested on 17 CT volumes for the brain stem localization, 19 volumes for the right eye and 9 volumes for the left parotid gland. The experiments were conducted with N = 400 initial candidate ellipsoids, a maximum of 30 iterations and a termination threshold value of 0.02. All values reported are averages of three runs. Figure 2 shows the ground truth for these structures and the corresponding localized region as detected by our algorithm for one volume. Though the algorithm performs 3D localization, for easy visualization, we depict the 2D central slices of the volume. The ground truth is shown after contrast enhancement to enable easy viewing of the structure. However, as can be seen in the images showing the localization results, the contrast resolution of medical images is low, which makes the task of accurate localization very difficult.

Figure 3 shows the extent to which appropriate scales and orientations are captured. We see that our approach, which computes the orientation and scale parameters in tandem in continous space rather than in discrete space, is able to adapt to the scale and orientations of the brainstem structure quite smoothly. The localized region in the XY plane (see Figure 2(b)) is almost a circle while in XZ and YZ planes, (see Figure 3), the scale and orientations of the plotted ellipse adapts to the orientation and shape of the structure.

We measure the quality of the localization achieved using two metrics: coverage ratio (CR) and tightness ratio (TR). Coverage ratio is a measure of the extent of coverage of the structure of interest within the localized region and is defined as the percentage volume of the structure that is encapsulated within the localized region, i.e $CR = \frac{(S \cap L)}{S}$ where S denotes the structure of interest and L denotes the localized region. The tightness ratio (TR), is a measure of how tightly the structure is localized and is defined as the percentage volume of the localized region that is composed of the structure under consideration, i.e $TR = \frac{(S \cap L)}{L}$.

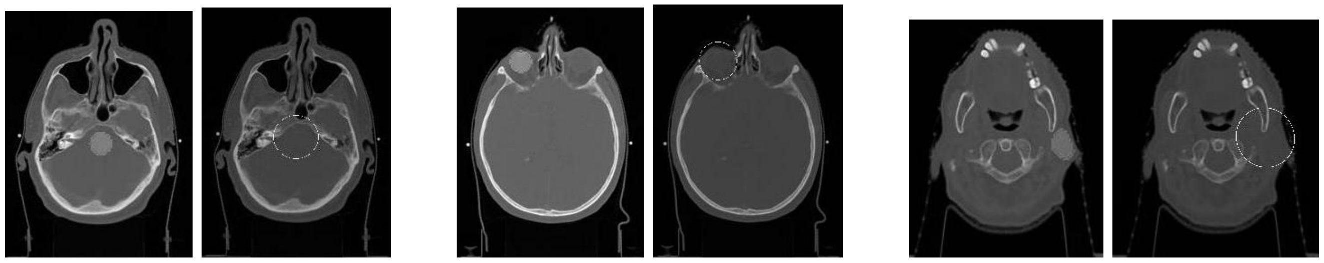

(a) Brain Stem (b) Right Eye (c) Left Parotid

Fig. 2. Contrast-enhanced ground truth (left) and localized regions (right) (central slice along the XY plane) for brain stem, right eye, and the left parotid gland

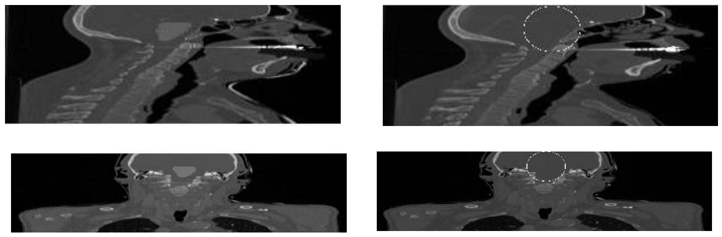


Fig. 3. Contrast-enhanced ground truth and localized regions (central slice along the XZ plane (top) and the YZ plane (bottom) for the brain stem

Structure	Seq	Multi-core accel	GPU accel
Brain Stem	545.8	151.9	5.5
Right Eye	269.3	73.9	2.9
Left Parotid	1070.9	302.7	9.39

TABLE I
AVERAGE LOCALIZATION TIME IN SECONDS FOR THE SEQUENTIAL, MULTI-CORE ACCLERATED AND THE GPU ACCELERATED VARIANTS.

Figure 4(a) shows the plot of the coverage ratio. In general, the quality of localizations is better for the brain stem and the right eye (brain stem and the right eye are localized with atleast 50% coverage in about two-thirds of the volumes). This is because, we use a intensity histogram based approach for localization and the contrast ratios are higher for the brain stem and the right eye as compared to the parotid gland. In 40% of the runs, we are able to encapsulate more than 90% of the structure, while in 65% of the runs we are able to capture the structure partially with atleast 50% of coverage. A coverage ratio of zero implies an incorrect localization, which accounts for 16% of the runs. Figure 4(b) plots the tightness ratio for the four structures. On an average about 20% of the localized region is made up of the structure being localized. Since, we use a simplistic intensity histogram as our target description function, shifts in intensity profiles across volumes as well as the low contrast ratio inherent in medical images, impacts the quality of localization. One approach to address this is to plug-in better descriptor functions (feature-based and domain specific), while trading off between the computational intensity of the training and detection and the required accuracy levels.

Table I shows the average localization times. We see that the multi-core version yields an average speedup of about 3.5x over the sequential variant on the 4 cores, while the GPU accelerated version yields an average speedup of about 97x over the sequential variant. Thus, leveraging the GPUs as massively parallel computing devices yields two orders of magnitude improvements in localization speeds.

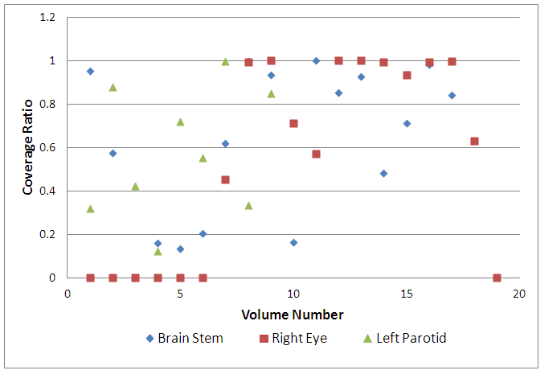
To evaluate the scalability of the proposed technique, we varied the search space and measured the quality as well as performance. Figure 5 plots the localization times as well as the coverage ratio for brain stem for one volume as we increase the dimensions of the search space (we increase the x, y, z dimensions of the search space as well as the number of search points keeping the number of search points density a constant). As expected we see that the quality improves as we increase the search space. The GPU accelerated variant scales extremely well as compared to the other two. Thus, leveraging the graphics processors as massively parallel computing devices yields large orders of improvements in both localization speed as well as scalability.

VI. CONCLUSION AND FUTURE WORK

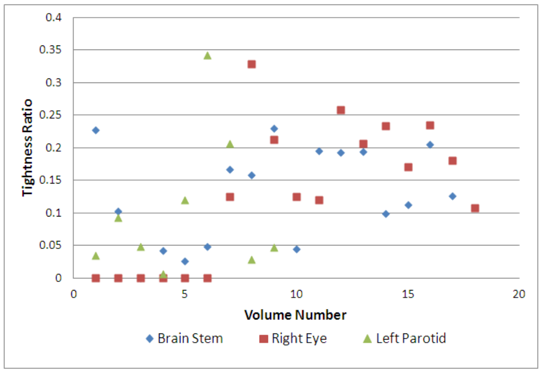
This paper proposes an iterative approach for structure localization in medical volumes that is based on the adaptive bandwidth mean-shift algorithm for object detection (ABMSOD). The ABMSOD algorithm, originally used to detect 2D objects in non-medical images, is extended and tuned to localize 3D anatomical structures in medical volumes. To enable fast localization, optimized parallel implementations are developed on multi-cores using OpenMP and GPUs using CUDA. Our evaluations with three structures (brain stem, eye and parotid) in CT volumes shows that our algorithm is able to localize structures with reasonable accuracy in many cases. However, we noted that the algorithm is sensitive to intensity changes across volumes and has a poor performance when the intensity difference between the train and test volumes is high. To address this issue, we plan to further investigate the algorithm with different target descriptor functions like local binary patterns (LBP), histogram of gradients or haar features. Features like LBP are intensity and rotation invariant and hence should be able to adapt to large differences in intensity between train and test volumes.

REFERENCES

[1] J. T. Bushberg, J. A. Seibert, E. M. Leidholdt, and J. M. Boone, *The Essential Physics of Medical Imaging (2nd Edition)*, 2nd ed. Lippincott Williams & Wilkins, Dec. 2001.



(a)



(b)

Fig. 4. Scatter plot of (a) coverage ratio and (b) tightness ratio

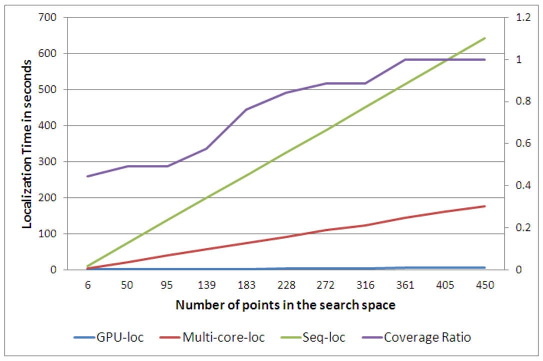


Fig. 5. Scalability of the localization against increasing dimensions of the search space

[2] X. Zhou, T. Hayashi, T. Hara, H. Fujita, R. Yokoyama, T. Kiryu, and H. Hoshi, “Automatic segmentation and recognition of anatomical lung structures from high-resolution chest ct images,” *Journal of Computerized Medical Imaging and Graphics*, vol. 30, no. 5, pp. 299–313, 2006.

[3] B. C. Lucas, M. Kazhdan, and R. H. Taylor, “Multi-object geodesic active contours (mogac): A parallel sparse-field algorithm for image segmentation,” Johns Hopkins University Department of Computer Science, Tech. Rep., Feb 2012.

[4] J.-C. D, F. de Manuel L, P. J, T. JM, R. E, D. M, and S. A. L.-C. MJ., “Optimal multiresolution 3d level-set method for liver segmentation incorporating local curvature constraints,” in *Intl. Conf. of the IEEE Engineering in Medicine and Biology Society*, 2011, pp. 19–22.

[5] X. Chen, H. Huang, H. Zheng, and C. Li, “Adaptive bandwidth mean shift object detection,” in *IEEE Conf. on Robotics, Automation and Mechatronics (RAM)*, 2008, pp. 210–215.

[6] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, “Active shape modelstheir training and application,” *Comput. Vis. Image Underst.*, vol. 61, no. 1, pp. 38–59, Jan. 1995.

[7] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active appearance models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 681–685, 2001.

[8] S. Frantz, K. Rohr, and H. S. Stiehl, “Localization of 3d anatomical point landmarks in 3d tomographic images using deformable models,” in *Intl. Conf. on Medical Image Computing and Computer-Assisted Intervention*, ser. MICCAI ’00, 2000, pp. 492–501.

[9] S. Wrz and K. Rohr, “Localization of anatomical point landmarks in 3d medical images by fitting 3d parametric intensity models.” *Medical Image Analysis*, vol. 10, no. 1, pp. 41–58, 2006.

[10] Y. Zheng, A. Barbu, B. Georgescu, M. Scheuering, and D. Comaniciu, “Four-chamber heart modeling and automatic segmentation for 3-d cardiac ct volumes using marginal space learning and steerable features,” *IEEE Trans. Med. Imaging*, vol. 27, no. 11, pp. 1668–1681, 2008.

[11] Y. Zheng, B. Georgescu, and D. Comaniciu, “Marginal space learning for efficient detection of 2d/3d anatomical structures in medical images,” in *Proceedings of the 21st International Conference on Information Processing in Medical Imaging*, ser. IPMI ’09, 2009, pp. 411–422.

[12] A. Criminisi, J. Shotton, and S. Bucciarelli, “Decision forests with long-range spatial context for organ localization in ct volumes,” in *Proceedings of the MICCAI workshop on Probabilistic Models for Medical Image Analysis (MICCAI-PMMIA)*, 2009.

[13] “Automated localization of solid organs in 3d ct images: A majority voting algorithm based on ensemble learning,” Poster at the International Workshop on Machine learning in Medical imaging (in conjunction with MICCAI 2010) http://miccai-mlmi.uchicago.edu/past_wkshp/2010/pdf/paper01.pdf.

[14] O. Pauly, B. Glocker, A. Criminisi, D. Mateus, A. M. Möller, S. Nekolla, and N. Navab, “Fast multiple organ detection and localization in whole-body mr dixon sequences,” in *Intl. conf. on Medical image computing and computer-assisted intervention - Volume Part III*, 2011, pp. 239–247.

[15] X. Zhou, A. Watanabe, X. Zhou, T. Hara, R. Yokoyama, M. Kanematsu, and H. Fujita, “Automatic organ segmentation on torso ct images by using content-based image retrieval,” in *Medical Imaging 2012: Image Processing: Proceedings of Society of Photo-Optical Instrumentation Engineers (SPIE)*, vol. 8314, feb 2012.

[16] M. de Bruijne, B. van Ginneken, W. Niessen, J. Maintz, and M. Viergever, “Active shape model based segmentation of abdominal aortic aneurysms in CTA images,” in *Medical Imaging 2002: Image Processing: (SPIE)*, vol. 4684, 2002, pp. 463–474.

[17] N. Boukala, E. Favier, B. Laget, and P. Radeva, “Active appearance model-based segmentation of hip radiographs,” in *In Proceedings of the IEEE International Conference on Industrial Technology (ICIT)*, 2004.

[18] N. Boukala, E. Favier, and B. Laget, “Active appearance model-based segmentation of hip radiographs,” in *In Proceedings of Society of Photo-Optical Instrumentation Engineers (SPIE)*, vol. 5747, 2005, p. 443.

[19] M. de Bruijne, B. van Ginneken, M. A. Viergever, and W. J. Niessen, “Adapting active shape models for 3d segmentation of tubular structures in medical images,” in *Information Processing in Medical Imaging*, 2003, pp. 136–147.

[20] O. Ecabert, J. Peters, H. Schramm, C. Lorenz, J. von Berg, M. J. Walker, M. Vembar, M. E. Olszewski, K. Subramanyan, G. Lavi, and J. Weese, “Automatic model-based segmentation of the heart in ct images.” *IEEE Transactions on Medical Imaging*, vol. 27, no. 9, pp. 1189–1201, 2008.

[21] H. Ling, S. K. Zhou, Y. Zheng, B. Georgescu, M. Sühling, and D. Comaniciu, “Hierarchical, learning-based automatic liver segmentation,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[22] J. Yao, S. D. O. Connor, and R. M. Summers, “Automated spinal column extraction and partitioning,” in *IEEE Intl. Symp. on Biomed. Imaging: From Nano to Macro, USA, April 2006*, 2006, pp. 390–393.

[23] X. Chen, H. Huang, H. Zheng, and C. Li, “Adaptive bandwidth mean shift object tracking,” in *IEEE Conf. on Robotics, Automation and Mechatronics (RAM)*, 2008, pp. 1011–1017.

[24] “Nvidia cuda c programming guide,” http://developer.download.nvidia.com/compute/cuda/32/toolkit/docs/CUDA.

[25] J. Ning, L. Zhang, D. Zhang, and C. Wu, “Scale and orientation adaptive mean shift tracking,” *Institution of Engineering and Technology (IET) Computer Vision*, vol. 6, no. 1, pp. 52–61, 2012.