A Surface-based 3D Dendritic Spine Detection Approach from Confocal Microscopy Images

Qing Li and Zhigang Deng*

***Abstract*—Determining the relationship between the dendritic spine morphology and its functional properties is a fundamental challenge in neurobiology research. In particular, how to accurately and automatically analyze meaningful structural information from a large microscopy image dataset is far away from being resolved. As pointed out in existing literature, one remaining challenge in spine detection and segmentation is how to automatically separate touching spines. In this paper, based on various global and local geometric features of the dendrite structure, we propose a novel approach to detect and segment neuronal spines, in particular, a breaking-down and stitching-up algorithm to accurately separate touching spines. Extensive performance comparisons show that our approach is more accurate and robust than two state-of-the-art spine detection and segmentation algorithms.**

***Index Terms*—Spine detection, surface-based segmentation, normalized cut, microscopy images, and touching spines**

I. Introduction

In neurobiology research, 3D neuron reconstruction and dendritic spine identification on a large data set of microscopy images is essential for understanding the relationship between morphology and functions of dendritic spines [21]. Dendrites are the tree-like structures of neuronal cells, and spines are small protrusions on the surface of dendrites. Spines have various visual shapes (e.g., mushroom, thin, and stubby) and can appear or disappear over time. Existing neurobiology literature shows that the morphological changes of spines and the dendritic spine structures are highly correlated with their underlying cognitive functions [21]. Therefore, how to efficiently and accurately detect and extract spines is crucial yet challenging problem.

Existing dendritic spine detection methods can be roughly divided into the following two categories: 2D MIP (maximum intensity projection) image-based algorithms [1] and 3D data-based algorithms [16], [5], [22]. The major drawbacks of the 2D MIP methods are: (1) 3D microscopy images are projected to a 2D plane; a significant amount of information such as spines that are orthogonal to the imaging plane will be inevitably lost. (2) Dendritic structures that overlap along the projection direction are difficult to extract. 3D data based algorithms either use voxel clustering [16] or extract the dendritic skeleton structure of neurons using a medial geodesic function [5]. Some existing commercial software tools (e.g., Imaris) perform semi-automated dendrite and spine detection. However, such a semi-automated process is typically costly, time-consuming, and subject to human bias.

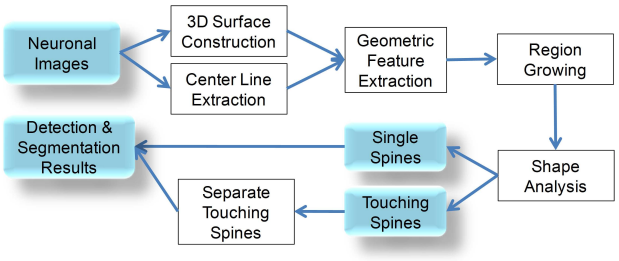

Fig. 1. Pipeline of the proposed spine detection and segmentation approach.

In this paper, we propose a novel 3D surface based dendritic spine detection and segmentation approach (its pipeline is illustrated in Fig. 1). Basically, we first extract the backbones and reconstruct the 3D neuron surface from the noise-reduced neuronal images. Then, by analyzing three geometric features on the 3D surface, spines are separated from the dendrite. After that, based on the spine classification outcomes (i.e., single or touching spines) from our shape analysis module, a normalized cut algorithm [20] is adapted to separate the touching spines in the following two-phase protocol: i) The touching spines are decomposed into small patches, and then ii) the patches are stitched together through maximization of an energy function. Through comprehensive performance comparisons, we show that our approach is more accurate and robust than two chosen state-of-the-art spine detection and segmentation algorithms [13], [16]. As pointed out in existing literature [13], [16], the real challenge of spine detection and segmentation is how to automatically separate touching spines. Our approach provides a novel automated solution to this challenging problem. Also, our method is fully automatic, without requiring any prior training dataset.

II. Related Work

Microscopy Image Processing: Nowadays, microscopy image processing techniques have been widely used in diverse fields such as medicine, biological and cancer research, drug testing, and metallurgy. These techniques aim to enhance, extract, analyze information from digital images acquired by microscope systems [4], [11], [9], [23], [12]. Microscopy images could be 2D, 3D, and in time series. Typical tasks of microscopy image processing include detecting and segmenting boundary of objects [4], [9], classifying objects into

* denotes the corresponding author, and the correspondence Email: *zdeng@cs.uh.edu*. Manuscript initially received on February 9th, 2011, revised on August 1st, 2011, and accepted on August 18th, 2011.

Qing Li and Zhigang Deng are with Computer Science Department, University of Houston, Houston, TX.

multiple categories [23], recovering the motion of moving object in multiple time points [11], [12], etc. Currently, microscopy image processing faces significant challenges due to the following reasons: (1) objects of interest are often touching/overlapping each other or irregularly arranged, with no definite shapes. (2) Illumination variations are not distinctive in thick specimens. Absorption, scattering, and diffraction of the light by structures located above and below the focal plane cause image intensity fall off in deep specimens. (3) The background of microscopy images is usually very noisy.

3D Spine Detection: Previous efforts on dendritic spine detection can be roughly divided into two categories: *classification-based approaches* [16], [13] and *centerline extraction-based approaches* [8], [5], [24], [22]. Classification-based approaches separate points into different groups using a trained classifier. For example, Rodriguez et al. [16] proposed an automated three-dimensional spine detection approach using voxel clustering. Li et al. [13] proposed a 3D surface-based classification algorithm. In their work, 3D neuron surface is reconstructed through the marching cubes algorithm [14]. Each vertex on the surface is classified into vertex on the spines or vertex on the dendrite based on three geometric features. Then, touching spines are separated by adopting surface-based watershed algorithm.

Centerline extraction-based approaches detect all the possible centerlines of certain objects in the image (e.g., using a curvilinear structure detector [22] or local binary fitting model of level sets [24]) and treat dendritic spines as small protrusions attached to the dendrites. However, these methods often require empirically designed post-processing procedures and limited to processing relatively simple neuron structures. Later, Koh et al. [8] adopt a thinning method to extract centerlines and apply the grassfire propagation technique to assign each dendritic point a distance to the medial axis of the dendritic structure. Since segmentation is achieved by global thresholding and limited geometric information is considered for spine detection, this method may detect pseudo spines. Janoos et al. [5] present a method for dendritic skeleton structure extraction using a curve-skeletons approach based on the medial geodesic function which is defined on the reconstructed isosurfaces.

Geometry-based Segmentation: Many approaches [10], [2], [19] employ simple, interpretable geometric features for 3D mesh segmentation, but they are often limited to either a single generic rule (e.g., concavity, skeleton topology, fitting shape primitives) or a single feature (e.g., shape diameter, curvature tensor, geodesic distances) to partition an inputted mesh. Recently, researchers either match points between meshes based on rigid mesh alignments [3] or simultaneously segment and label parts in 3D meshes using a data-driven scheme [6]. One common limitation of these approaches is that they often require a consistently labeled training set for different types of meshes.

III. Image Acquisition and Preprocessing

In the image acquisition stage, mice brain slices were labeled by ballistic delivery of fluorescent dye DiI (Molecular Probes). Confocal microscope (Zeiss LSM 510) with an oil immersion lens (EC Plan-Neofluar $100\times$, $1.4N.A.$) was used for imaging. DiI was excited using a Helium/Neon $543nm$ laser line. The resolution of the image stack was $0.064\mu m \times 0.064\mu m \times 0.12\mu m$ (with over-sampling). Please refer to [7] for more details.

In the data preprocessing stage, a 3D median filter with a $3 \times 3 \times 3$ kernel size was first applied to the images to remove noise. The median filter is a commonly used nonlinear operator that replaces the original gray level of a pixel by the median of the gray levels of the pixels in a specified neighborhood. As a type of ranking filters, the median filter is based on the statistics derived from rank-ordering a set of elements. It is often useful because it can reduce noise without blurring edges in the image. The noise-reducing effect of the median filter depends on two factors: (1) the spatial extent of its neighborhood and (2) the number of pixels involved in the median calculation.

We chose the median filter since it is able to remove certain noise that cannot be removed by conventional convolution filtering. Then, to correct uneven illumination degradation, a top-hat filter was adopted. After that, fuzzy C-mean clustering [15] was used to cluster the image into 3 clusters: background, weak spines, and dendrite with strong spines. Weak spines and the dendrite represent the neuron. Subsequently, we employed the marching cubes algorithm [14] to reconstruct the 3D surface of the neuron. Then, a low-pass filter and mesh decimation were used to remove noise and reduce the tessellation density. The number of iterations in the low-pass filter and the decimation factor control the smoothness of the resultant 3D neuron surface.

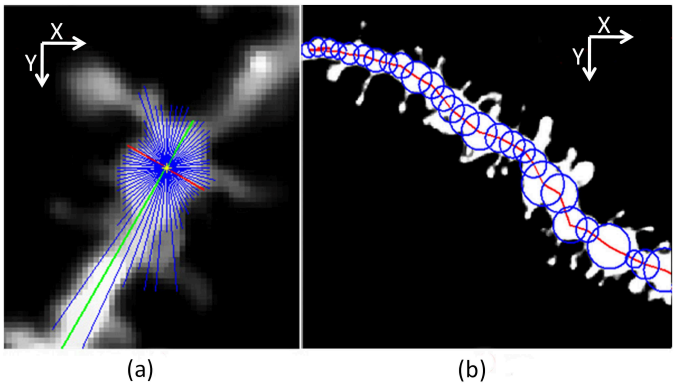

Fig. 2. Illustration of Rayburst sampling. (a) Rayburst sampling in XY plane. Blue lines show the rays casting out in all the directions from the predicted center point. The red line is the ray with the shortest length (diameter) while the green line has the longest length (local orientation). (b) Extracted backbone of the neuron and the estimated radiuses are illustrated in the maximum intensity projection image.

IV. Neuron Backbone Extraction

After the above data preprocessing, the dendrite backbone and the approximated radiuses along the backbone are generated by extending the Rayburst sampling algorithm [17]. The Rayburst sampling core can be 2D or 3D with a variable number of rays that are uniformly spaced within a unit sphere. Compared with 3D Rayburst sampling, 2D Rayburst in the

XY and XZ (or YZ) planes is less computationally expensive, while it reliably yields comparable results [17].

Inspired by this idea, our Rayburst sampling algorithm works as follows: at the beginning, a seed point (an initial center point) inside of the dendrite is selected by users. Then, 2D Rayburst sampling in both the XY and XZ (or YZ) planes are adopted. A threshold ϕ is used to control the maximally allowed intensity difference between the center point of a neuron and its dendritic boundary. The length of the shortest ray in XY plane is the estimated diameter, and the location of the center point is updated as the midpoint of the shortest ray. Then, rays sampled in the XZ (or YZ) plane are used to adjust the Z coordinate of the center point. The next two center points toward both the ends of the dendrite are assumed to follow the local orientation of the current center point: the orientation of the longest ray in XY plane is the approximated local orientation of the dendrite. This procedure repeats until the predicted center point reaches the border of the stack or it goes into the background. If the dendrite contains a branch structure, a user-specified seed point is needed for each branch. In this work, the number of rays and ϕ are experimentally set to 36 and 80, respectively. Fig. 2 illustrates this process and one example result.

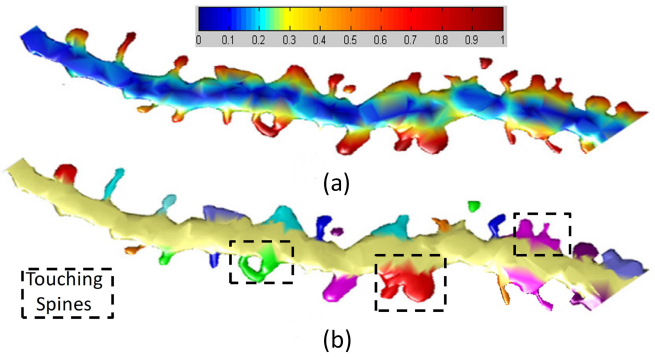

Fig. 3. Visualization of the computed score map after region growing. (a) Visualization of the score map. Each face on the surface has a score from 0 to 1. (b) A threshold ξ is used in the region growing algorithm to separate spines with dendrite. Spines are labeled in different colors. Touching spines are highlighted in black rectangles.

V. Separating Neuronal Spines with Dendrite

In our approach, the following three geometric features are calculated for each 3D face to separate neuronal spines from the dendrite.

1) *Distance to the dendrite backbone:* The distance between a vertex v on the neuron surface and the backbone is defined as *the shortest distance* between v and any point of the backbone. In this way, the distance between each face f and the backbone is defined as the average distance between f's vertices and the backbone. This distance feature is chosen based on the observation that vertices on the spine surface usually have larger distances (to the backbone) than those on the dendrite.

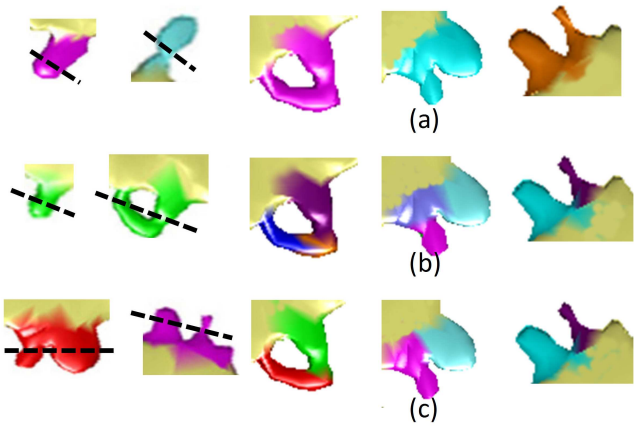

Fig. 4. **Left:** Illustration of our spine shape analysis and categorization principle. When any sampling line is sent from one side of a spine to the other side, if it intersects with at most two faces, then it is a single spine; otherwise, it is a touching spine. **Right:** Visualization of how we separate touching spines: (a) initial touching spines, (b) the results after the touching spines are broken into small patches using a normalized cut algorithm, and (c) the final results after the patches are stitched together to form spines.

2) *Mean curvature on the surface*: The mean curvature of each face on the 3D neuron surface is computed as the average curvature of its surrounding vertices.
3) *Normal variance:* Normal variance of each face f is defined as the average angle between the face normal and the vector that is perpendicular to the backbone and passes each of f's surrounding vertices.

We normalize the above features and add them together to generate a score map. Then, we use a surface-based region growing algorithm to separate spines from the dendrite. In this process, faces whose values in the score map are greater than ξ (a user-specified threshold) are randomly selected as seed points. If the score value of a neighbor of each seed point is larger than ξ and it has not been visited previously, then it will be chosen as a new seed point. This process repeats until all the faces have been visited. In this work, ξ is experimentally set to 0.28. One example result after the region growing is shown in Fig. 3. From this figure, we can see that although most spines are detected and separated from the dendrite, some spines are still touching together. As such, the next step of our approach is to automatically detect and separate touching spines.

Spine shape analysis and categorization: After spines are separated from the dendrite, we perform 3D shape analysis on all the spines in order to automatically categorize them into single spines or touching spines. The core idea of our shape analysis algorithm is illustrated in the left of Fig. 4. Basically, if any sampling line is sent from one side of the spine to the other side, a single spine will have at most two faces intersecting with the sampling line. By contrast, in this case a touching spine will have at least four faces intersecting with the sampling line. Based on this key observation, for each spine segment outputted from the above region growing algorithm, we first randomly select κ sampling vertices on the spine surface. Then, κ sampling lines are sent out from the sample vertices and their directions are in parallel to the

normals of the sampling vertices. In our experiments, we found κ=15 worked sufficiently well. To ensure the sampling vertices are from one side of the spine, they are only chosen from the vertices whose normal variance features are high (Section V). If more than two sampling lines intersect with four or more faces on the spine segment, we categorize the spine segment as a touching spine segment; otherwise, we categorize it as a single spine segment.

Breaking a touching spine into patches: We propose a normalized cuts based algorithm to break a touching spine into patches based on its local geometric features. The normalized cuts is an unsupervised segmentation technique first proposed by Shi and Malik [20]. Generally, the normalized cuts algorithm works on a graph which consists of nodes and edges. A measure of dissimilarity can be established based on distinctive features [20]. The strengths of the edges between nodes are weighted with an exponential factor as shown in Eq. 1.

$$\mathbf{W}_{i,j} = e^{\frac{\mathrm{d}(i,j)}{\sigma_d}} \tag{1}$$

Here $\mathrm{d}(i,j)$ is the measure of dissimilarity between nodes v_i and v_j, and σ_d controls the scale of this measure.

If we have a geometry $\mathbf{G}$ with N elements, and we assume a bipartition of $\mathbf{G}$ into A and B, where $\mathbf{A} \cup \mathbf{B} = \mathbf{G}, \mathbf{A} \cap \mathbf{B} = \emptyset$, and $\mathbf{W} \in \mathbb{R}^{N \times N}$ is a symmetric matrix with

$$\mathbf{W}(i,j) = w_{ij} \tag{2}$$

Then,

$$\mathrm{cut}(\mathbf{A}, \mathbf{B}) = \sum_{u \in \mathbf{A}, v \in \mathbf{B}} \mathbf{W}(u,v) \tag{3}$$

A normalized cut, $\mathrm{Ncut}(\mathbf{A}, \mathbf{B})$, is defined as:

$$\mathrm{Ncut}(\mathbf{A}, \mathbf{B}) = \frac{\mathrm{cut}(\mathbf{A}, \mathbf{B})}{\mathrm{assoc}(\mathbf{A}, \mathbf{G})} + \frac{\mathrm{cut}(\mathbf{A}, \mathbf{B})}{\mathrm{assoc}(\mathbf{B}, \mathbf{G})} \tag{4}$$

where

$$\mathrm{assoc}(\mathbf{A}, \mathbf{G}) = \sum_{u \in \mathbf{A}, p \in \mathbf{G}} \mathbf{W}(u,p) \tag{5}$$

and $\mathrm{assoc}(\mathbf{B}, \mathbf{G})$ is defined in a similar way. In order to maximize the total dissimilarity between $\mathbf{A}$ and $\mathbf{B}$, the total similarity within $\mathbf{A}$, and the total similarity within $\mathbf{B}$, Shi and Malik [20] formulate and solve the following generalized eigen-vector problem:

$$\mathbf{W}\mathbf{y} = (1 - \lambda)\mathbf{D}\mathbf{y} \tag{6}$$

They apply a threshold to the eigen-vector with the second largest eigen-value to yield the optimal bipartition. In Eq. 6, $\mathbf{D}$ is a diagonal matrix such that

$$\mathbf{D}(i,j) = \sum_{j=0}^{N} \mathbf{W}(i,j) \tag{7}$$

and y is a group membership indication vector. A detailed description of how to solve Eq. 6 can be found in [20].

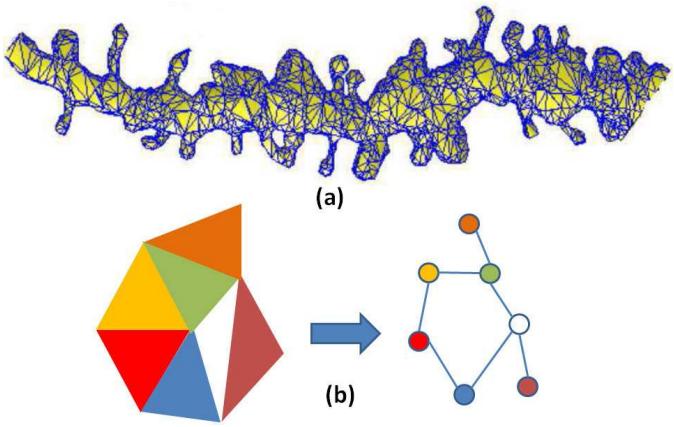

Fig. 5. Illustration of mapping a 3D neuron surface to a graph: (a) an example of the reconstructed 3D neuron geometry (i.e. triangular mesh). (b) The process of mapping a triangular mesh to a graph.

In our algorithm, each triangular face in the reconstructed neuron geometry (i.e. triangular mesh) is considered as a node in a graph. In the graph, nodes are connected by edges to their neighbors. Since the neuron surface is represented as a triangular mesh; each face has exactly three face neighbors. How we map a triangle mesh to a graph is illustrated in Fig. 5. The dissimilarity between two connected faces f_i and f_j is defined as follows:

$$\mathbf{W}_{f_i,f_j} = e^{\mathrm{dihedral}(f_i,f_j)} \tag{8}$$

Here the function *dihedral* calculates the dihedral angle between the two faces. Let u_i and u_j be the vectors which are orthogonal to faces f_i and f_j, respectively. Then, the dihedral angle between the two faces, f_i and f_j, is computed as follows:

$$\mathrm{dihedral}(f_i, f_j) = \arccos\left(\frac{|u_i \cdot u_j|}{|u_i||u_j|}\right) \tag{9}$$

Then, we solve Eq. 6 for eigen-vectors with the smallest eigen-values and use the eigen-vector with the second smallest eigen-value to bipartition the graph into two patches (subgraphs). We repeat this process on the partitioned patches (graphs) until the number of patches reaches a user-specified maximum. At the end, the algorithm returns a group of eigen-vectors. Each eigen-vector represents a patch, and every element of the eigen-vector can be mapped to one face on the surface: if the element equals to 1, it means its corresponding face belongs to the patch. The (b) panel in the right of Fig. 4 shows some example results after we break touching spines into patches based on local geometric features.

Stitching spine patches: In this step, we introduce a novel spine stitching algorithm to group some patches together to form a new spine based on high-level geometric features. We first examine the boundary of each patch as follows: if one patch is connected with another patch, we add them as a pair in a candidate list. Whether two patches p_i and p_j should be stitched together is primarily based on two high level geometric metrics (illustrated in Fig. 6). The first geometric metric is the projected distance between the centroids of the two connecting patches. The second geometric metric is the

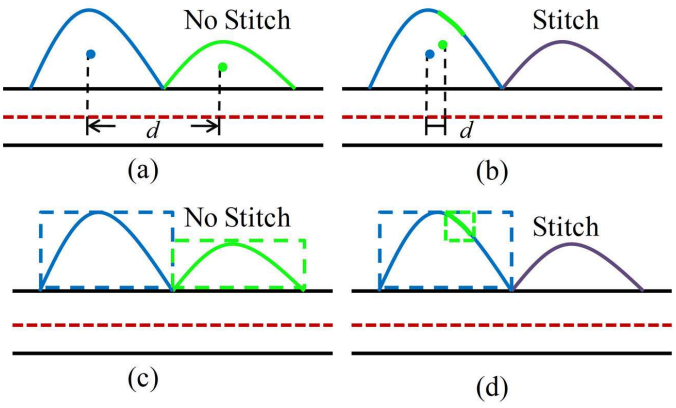

Fig. 6. Illustration of our two high-level geometric features/metrics. The blue and green curves represent two connecting patches. The red dash line represents the backbone.

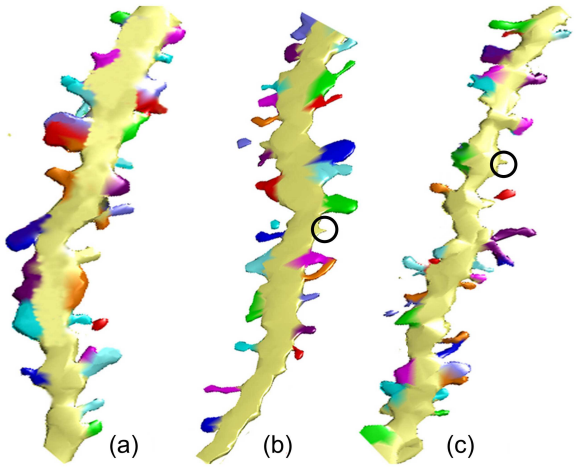

Fig. 7. 3D surface-based spine detection and segmentation results of three example images. Segmented spines are labeled in different colors. Two mis-detected tiny spines are highlighted in two black circles.

intersecting volume ratio of the bounding boxes of the two connecting patches. The two geometric metrics are designed to guide the algorithm to only stitch patches on the same spine, avoiding stitching patches that belong to different spines. We combine the two metrics in the following way:

$$\eta(p_i, p_j) = \frac{\text{vol}\left(p_i \cap p_j\right)}{\min\left(\text{vol}\left(p_i\right), \text{vol}\left(p_j\right)\right)} - \text{dis}_{\text{proj}}(p_i, p_j) \quad (10)$$

In Eq. 10, the first item calculates the intersecting volume ratio of the bounding boxes of the two connecting patches, and the second item is the distance between the projected centroids of the two connecting patches.

If a patch is associated with only one other patch based on the above constructed candidate list, we can simply apply a threshold γ to $\eta(p_i, p_j)$ in Eq. 10 to determine whether the two patches should be stitched. However, more complicated cases often exist such as each patch is associated with more than one other patches. Therefore, in this work we stitch spine patches through maximization of a global possibility.

Specifically, in our algorithm, a 0-1 integer programming is employed to determine the multi-association cases. Let $P = \{p_i | i = 1, 2, ...n\}$ denote n patches associated each other and need to be stitched properly. For a patch p_i, we assume m_i other patches are identified as stitching candidates $M_i = \{(p_i, p_j) | j = 1, 2, ..., m_i\}$. Therefore, the total number of candidate stitching pairs is $N = \sum_{i=1}^{n} m_i$. These candidate pairs are $M = \bigcup_{i=1}^{n} M_i$.

The optimal matching strategy is to determine the optimal solution $\mathbf{z}^* = \{0/1\}^N$ that maximizes an energy objective function (Eq. 11) while maintaining a constraint (Eq. 12): each patch can be associated with at most one patch at a time. The objective function is formulated as follows.

$$\begin{aligned} \mathbf{z}^* &= \arg\max_{\mathbf{z} \in \{0,1\}^N} (\text{f}(\mathbf{z})) \\ \text{f}(\mathbf{z}) &= \sum_{l=1}^{N} \left[\mathbf{z}(l) \times \eta(p_{i_l}, p_{j_l})\right] \end{aligned} \quad (11)$$

Here $\mathbf{z}$ is a $N \times 1$ vector to denote the decisions of the N possible associations: $\mathbf{z}(l) = 1$ indicates the lth possible association is chosen; while $\mathbf{z}(l) = 0$ indicates the lth possible association is discarded. (i_l, j_l) indicates the patch labels in the lth possible association. The one-to-one association constraint can be formulated as follows:

$$\mathbf{A}\mathbf{z} \leq \mathbf{b} \quad (12)$$

Here $\mathbf{A}$ is a $n \times N$ matrix: its rows indicate all the patches and its columns correspond to the decisions of the N possible association pairs; $\mathbf{b}$ is a $n \times 1$ vector with all 1s, which means that each patch can be associated with at most one other patch at a time. For the N possible associations $M = \{(p_{i_l}, p_{j_l}) | l = 1, 2, ..., N\}$,

$$\mathbf{A}(k, l) = \begin{cases} 1; k = i_l \\ 1; k = j_l \\ 0; \text{otherwise} \end{cases}, 0 < l \leq N \quad (13)$$

The above optimization problem can be solved through 0-1 integer programming. In this work, we use branch-and-bound (LPBB) based linear programming algorithm [18] to solve it. The optimization process of LPBB is to build a searching tree by repeatedly discretizing (0 or 1) the variables (branching) and pruning the tree branches based on the optimal value of this node (bounding) computed by linear programming. Note that this optimization process is only applied to groups in which more than one patches are associated in multiple stitching pairs. For simple one-to-one cases, a threshold γ is used to determine whether the two patches should be stitched (described above). Several patch stitching results are shown in (c) panel in the right of Fig. 4. After the patch stitching, we combine the separated touching spines with single spines to obtain the final spine detection and segmentation results, as shown in Fig. 7.

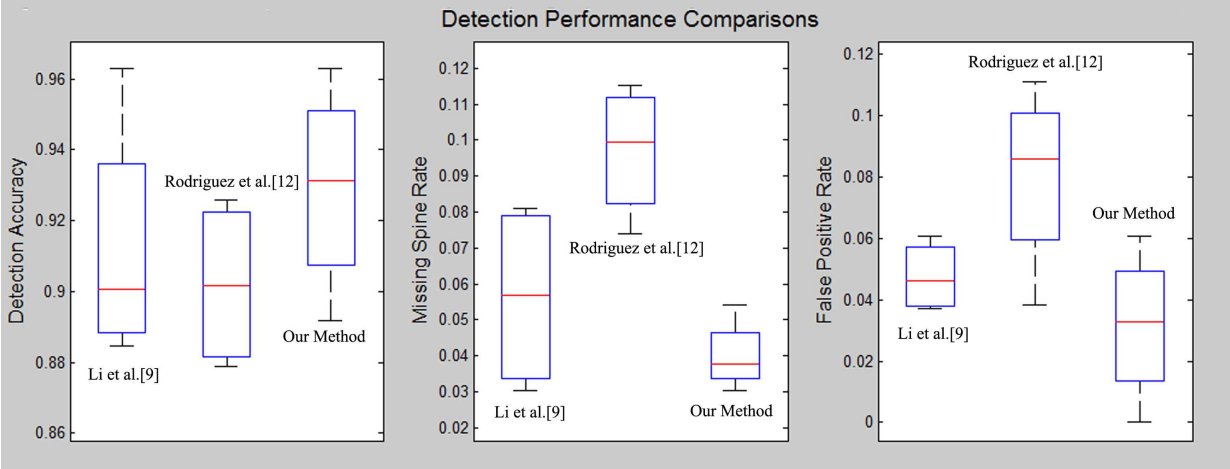

Fig. 8. Detection performance comparisons between our approach and the two chosen spine detection algorithms [13], [16]. The Y axis of the three panels (from left to right) is "detection accuracy", "missing spine rate", and "false positive rate", respectively.

VI. Experimental Results

To evaluate the performance of our approach, we had tested our algorithm on 6 neuronal image datasets. The spine detection and segmentation results of three images are shown in Fig. 7, where spines automatically segmented by our algorithm are labeled in different colors. As shown in this figure, our approach can robustly and automatically segment majority of spines including touching spines, except a tiny number of sporadic mis-detections.

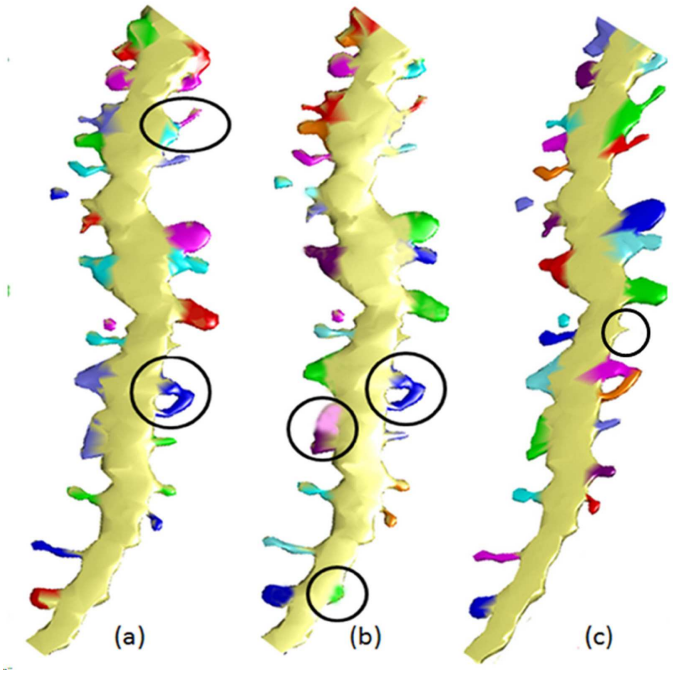

Fig. 9. The visual result comparison of our algorithm with two other spine detection and segmentation algorithms [13], [16] on a neuronal image. (a) shows the result from [13]. (b) shows the result from [16]. (c) shows the result from our algorithm. In all the results, spines are labeled in different colors. The black circles indicate missing, spurious, false positive spines.

To validate the segmentation performance of our approach, we asked a biologist to manually detect and segment spines in the test images. Then, we compared the detection performance of our algorithm with two state-of-the-art, neuronal spine detection algorithms [13], [16] and the ground truth. As shown in Fig. 8, our approach achieved a higher spine detection accuracy, a smaller spine missing rate, and a lower false positive rate than the two chosen spine tracking algorithms [13], [16]. However, in Fig. 7, we also can see that our approach still missed the detection of a few tiny spines, though the same tiny spines were also mis-detected by the two other algorithms [13], [16].

Note that this work can be regarded as a comprehensive improvement/extension of [13]. The major difference between this work and [13] is how touching spines are automatically detected and separated. In [13], we simply adopt watershed algorithm to separate touching spines. The limitation of the watershed algorithm is that it is very hard to control the water level, which will easily lead to over-segmentation results. In this work, we first accurately categorize spines into single spines and touch spines. Then, we introduce a robust break-down and stitching-up algorithm to separate touching spines into single spines.

Quantitative comparison and analysis results are shown in Fig. 8 and the visual comparison is shown in Fig. 9. Note that in Fig. 9 (b), the main reason why occluded spines cannot be detected by [16] could be that the resolution along z direction of neuron images is not fine enough. It is not related to whether 3D median filter or other image processing procedure is applied or not, in our opinions.

As mentioned in the introduction section, spines have been categorized in multiple shapes. The accuracy of the segmentation algorithm will affect the classification results. Therefore, we compared the segmentation performance of our method with the ground truth, as well as methods proposed in [13] and [16]. For each boundary vertex v_i in our segmented spine, we first calculated its closest boundary vertex in the ground truth. Then, the segmentation error for one spine is defined as:

$$SegError = \sum_{i=1}^{N} \frac{\mathrm{dis}(v_i, \mathrm{closest}(v_i))}{\sum_{k=1}^{N} \mathrm{dis}(v_k, v_{k+1})} \quad (14)$$

Here N is the number of boundary vertices of the spine, and $\sum_{k=1}^{N} \mathrm{dis}(v_k, v_{k+1})$ calculates the length of the spine boundary. The comparison results are shown in Fig. 10.

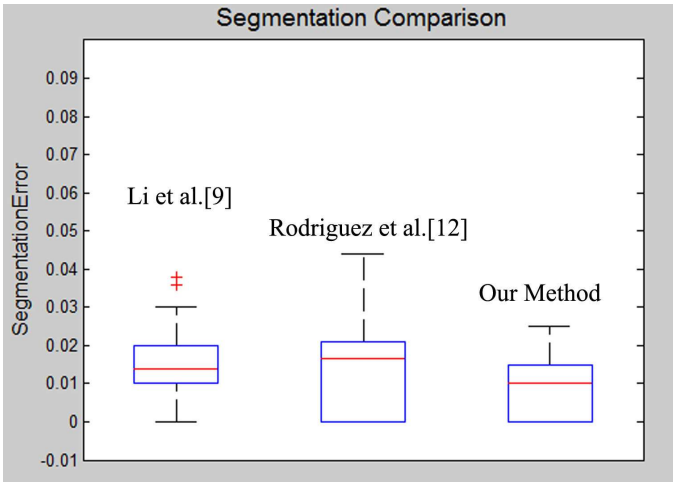

Fig. 10. Segmentation performance comparisons between our approach and the two chosen algorithms [13], [16].

As clearly shown in Fig. 10, our algorithm achieved a lower segmentation error. Rodriguez et al.'s method [16] uses voxel clustering for segmentation and the vertex distance to the backbone is used as the only criterion for segmentation. Its segmentation performance was less consistent with the ground truth. Li et al.'s method [13] had a competitive segmentation performance in single spine cases as our algorithm. However, our approach had a better performance in segmenting touching spines than [13].

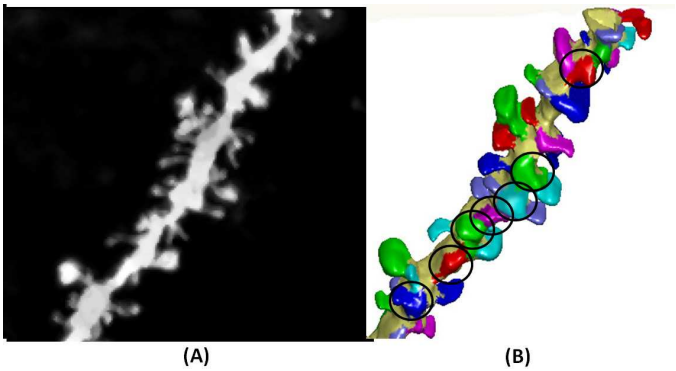

Fig. 11. Illustration of occluded 3D spine detections. (A) shows the XY-Plane MIP (Maximum Intensity Projection) of one 3D neuron image. After this projection, spines along the Z direction are occluded and cannot be detected. (B) The detection result of our algorithm that is able to handle the occlusion issue. Spines that are occluded in MIP but observed in the 3D surface are highlighted in black circles.

One limitation of 2D MIP (Maximum Intensity Projection) based spine detection algorithm is that spines along the Z direction are occluded by other dendritic structures and thus cannot be detected. Fig. 11 illustrates our 3D surface-based spine detection approach is able to handle such occlusion issues. Fig. 11 (A) shows the XY-Plane MIP (Maximum Intensity Projection) of one 3D neuron image. After this projection, spines along the Z direction are occluded and cannot be detected. Fig. 11 (B) shows the results by our algorithm that can detect spines along Z direction. Note that in most 3D microscopy images, compared with the resolution in X and Y direction, the resolution in Z direction is still far from sufficient; substantial information along Z direction is lost at the image acquisition stage.

We implemented our algorithm using Matlab. The average computing time for detecting spines from a neuron image is about 30 seconds using our method on a computer with a 2.20GHz Intel Core 2 Duo CPU and 3G Memory. It is noteworthy that we plan to make our self-contained code of this work publicly available to other researchers in the community upon request.

VII. Discussion and Conclusions

This paper introduces a novel 3D surface based dendritic spine detection and segmentation approach. To validate the performance of our algorithm, we compared the results by our approach with the manually labeled ground-truth as well as the results by two state-of-the-art spine detection and segmentation algorithms [16], [13]. Our comparison results show that our proposed approach is more accurate and robust than the two chosen algorithms.

It is noteworthy that although our approach can achieve a high spine detection and segmentation accuracy, it still mis-detected a few tiny spines (Fig. 7). This in part comes from our region growing algorithm that only uses a single global threshold to separate spines from the dendrite. In the future, to further improve the accuracy of our algorithm, we plan to investigate new algorithms to adaptively learn this parameter based on the local geometric features.

Since the test dataset we were able to acquire only encloses straight neurons, not curvy neurons, we were not able to test our algorithm on curvy neuron datasets. Considering curvy neurons typically have a higher complexity than straight ones, directly applying this current work to curvy neurons may not work well; however, we believe a proper extension and adaptation of this work will help to solve the problem. Another limitation of the current work is that it does not handle single spines with complex shapes. For single spines with complex shapes (e.g., multi-headed spines), more sophisticated shape analysis algorithms need be developed. We believe that with a sufficient spine shape training dataset, accurately distinguishing complex shape spines (e.g., multi-headed spines) from touching spines can be achieved.

Acknowledgments

This research is supported in part by Texas NHARP 003652-0058-2007 and NSF IIS-0914965. We also would like to thank Stephen TC Wong at the Methodist Hospital Research Institute for insightful discussion and help, and Matthew

Baron, Merilee A. Teylan, and Yong Kim at the Rockefeller University for providing the test image data. Any opinions, findings,and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] J. Cheng, X. Zhou, E. Miller, R. M. Witt, J. Zhu, B. L. Sabatini, and S. T. Wong. A novel computational approach for automatic dendrite spines detection in two-photon laser scan microscopy. *Journal of Neuroscience Methods*, 165(1):122–134, 2007.

[2] A. Golovinskiy and T. Funkhouser. Randomized cuts for 3D mesh analysis. *ACM Trans. Graph.*, 27:145:1–12, 2008.

[3] A. Golovinskiy and T. Funkhouser. Technical section: Consistent segmentation of 3d models. *Comput. Graph.*, 33:262–269, 2009.

[4] Y. Hiraoka, T. Shimi, and T. Haraguchi. Multispectral imaging fluorescence microscopy for living cells. *Cell Structure and Function*, 27(5):367–374, 2002.

[5] F. Janoos, K. Mosaliganti, X. Xu, R. Machiraju, K. Huang, and S. T. Wong. Robust 3D reconstruction and identification of dendritic spines from optical microscopy imaging. *Medical Image Analysis*, 13(1):167–179, 2009.

[6] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D mesh segmentation and labeling. In *ACM SIGGRAPH 2010 papers*, pages 102:1–12, 2010.

[7] Y. Kim, M. A. Teylan, M. Baron, A. Sands, A. C. Nairn, and P. Greengard. Methylphenidate-induced dendritic spine formation and deltafosb expression in nucleus accumbens. *Proc. Natl. Acad. Sci.*, 106(8):2915–2920, 2009.

[8] I. Y. Y. Koh, W. B. Lindquist, K. Zito, E. A. Nimchinsky, and K. Svoboda. An image analysis algorithm for the fine structure of neuronal dendrites. *Neural Comput.*, 14:1283–1310, 2002.

[9] A. Kuijper and B. Heise. An automatic cell segmentation method for differential interference contrast microscopy. In *19th International Conference on Pattern Recognition (ICPR)*, pages 8–11, 2008.

[10] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Fast mesh segmentation using random walks. In *Proc. of 2008 ACM Symp. on Solid and physical modeling*, pages 183–191, 2008.

[11] F. Li, X. Zhou, J. Ma, and S. T. Wong. Multiple nuclei tracking using integer programming for quantitative cancer cell cycle analysis. *IEEE Trans. on Medical Imaging*, 29(1):96–105, 2010.

[12] Q. Li, Z. Deng, Y. Zhang, X. Zhou, U. V. Nagerl, and S. T. Wong. A global spatial similarity optimization scheme to track large numbers of dendritic spines in time-lapse confocal microscopy. *IEEE Trans. on Medical Imaging*, 30(3):632–641, 2011.

[13] Q. Li, X. Zhou, Z. Deng, M. Baron, M. A. Teylan, Y. Kim, and S. T. C. Wong. A novel surface-based geometric approach for 3D dendritic spine detection from multi-photon excitation microscopy images. In *ISBI'09*, pages 1255–1258, 2009.

[14] G. M. Nielson and B. Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *IEEE Visualization'91*, pages 83–91, 1991.

[15] T. D. Pham, D. I. Crane, T. H. Tran, and T. H. Nguyen. Extraction of fluorescent cell puncta by adaptive fuzzy segmentation. *Bioinformatics*, 20:2189–2196, 2004.

[16] A. Rodriguez, D. B. Ehlenberger, D. L. Dickstein, P. R. Hof, and S. L. Wearne. Automated three-dimensional detection and shape classification of dendritic spines from fluorescence microscopy images. *PLoS One*, 3(4):e1997, 2008.

[17] A. Rodriguez, D. B. Ehlenberger, P. R. Hof, and S. L. Wearne. Rayburst sampling, an algorithm for automated three-dimensional shape analysis from laser scanning microscopy images. *Nature Protocols*, 1:2152–2161, 2006.

[18] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[19] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259, 2008.

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on PAMI*, 22(8):888–905, 2000.

[21] R. Yuste and T. Bonhoeffer. Morphological changes in dendritic spines associated with long-term synaptic plasticity. *Annual Reviews in Neuroscience*, 24:1071–1089, 2001.

[22] Y. Zhang, K. Chen, M. Baron, M. A. Teylan, Y. Kim, Z. Song, P. Greengard, and S. T. Wong. A neurocomputational method for fully automated 3D dendritic spine detection and segmentation of medium-sized spiny neurons. *Neuroimage*, 50:1472–1484, 2010.

[23] Y. Zhang, X. Zhou, R. M. Witt, B. L. Sabatini, D. Adjeroh, and S. T. Wong. Dendritic spine detection using curvilinear structure detector and LDA classifier. *NeuronImage*, 36:346–360, 2007.

[24] W. Zhou, H. Li, and X. Zhou. 3D neuron dendritic spine detection and dendrite reconstruction. *International Journal of Computer Aided Engineering and Technology*, 1:516–531, 2009.

Qing Li is a Ph.D. student in the Department of Computer Science, University of Houston, Houston, TX. Her research interests include computer graphics, computer animation, and medical imaging. She had received his B.S. in Computer Science from the Beijing University of Chemical Technology, Beijing, China, in 2006.

Zhigang Deng is currently an Assistant Professor of Computer Science at the University of Houston (UH) and the Founding Director of the UH Computer Graphics and Interactive Media (CGIM) Lab. His research interests include computer graphics, computer animation, virtual human modeling and animation, human computer interaction, visual-haptic interfacing, and GPU-based computing. He completed his Ph.D. in Computer Science at the Department of Computer Science at the University of Southern California (Los Angeles, CA) in 2006. Prior that, he also received B.S. degree in Mathematics from Xiamen University (China), and M.S. in Computer Science from Peking University (China). Over the years, he has worked at the Founder Research and Development Center (China) and AT&T Shannon Research Lab. He is a senior member of IEEE and a member of ACM and ACM SIGGRAPH.