Coarse-to-Fine Foraminifera Image Segmentation through 3D and Deep Features

Qian Ge*, Boxuan Zhong*, Bhargav Kanakiya*, Ritayan Mitra†, Thomas Marchitto† and Edgar Lobaton*
*Department of Electrical and Computer Engineering
North Carolina State University, Raleigh, North Carolina 27695–7911
Email: qge2@ncsu.edu, bzhong2@ncsu.edu, bkanaki@ncsu.edu, edgar.lobaton@ncsu.edu
†Institute of Arctic and Alpine Research
University of Colorado Boulder, Boulder, Colorado 80309–0552
Email: Ritayan.mitra@colorado.edu, thomas.marchitto@colorado.edu

***Abstract*—Foraminifera are single-celled marine organisms, which are usually less than 1 mm in diameter. One of the most common tasks associated with foraminifera is the species identification of thousands of foraminifera contained in rock or ocean sediment samples, which can be a tedious manual procedure. Thus an automatic visual identification system is desirable. Some of the primary criteria for foraminifera species identification come from the characteristics of the shell itself. As such, segmentation of chambers and apertures in foraminifera images would provide powerful features for species identification. Nevertheless, none of the existing image-based, automatic classification approaches make use of segmentation, partly due to the lack of accurate segmentation methods for foraminifera images. In this paper, we propose a learning-based edge detection pipeline, using a coarse-to-fine strategy, to extract the vague edges from foraminifera images for segmentation using a relatively small training set. The experiments demonstrate our approach is able to segment chambers and apertures of foraminifera correctly and has the potential to provide useful features for species identification and other applications such as morphological study of foraminifera shells and foraminifera dataset labeling.**

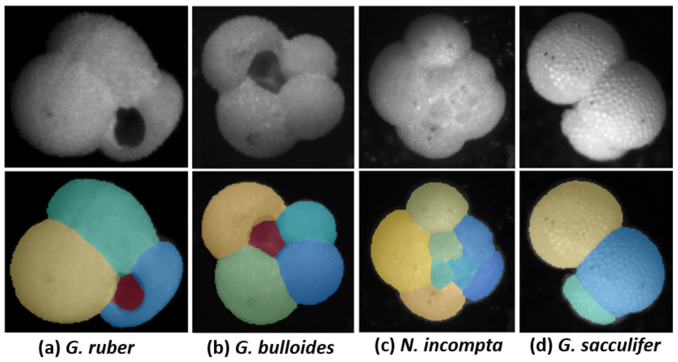

Fig. 1. Sample segmentation results using proposed approach. Top row: Foraminifera images of four different species from our dataset (one of 16 images per sample). Note that some of the boundaries between chambers are hard to see due to the similarity of patterns between adjacent chambers and low image quality. Bottom row: Segmentation result. Apertures are labeled as red and different chambers are labeled as other different colors. The morphology that is captured by the segmentation approach uniquely characterizes these four species; hence, it could be used for classification.

I. Introduction

Foraminifera are single-celled marine organisms that can be identified on the basis of their shells, which are usually less than 1 mm in diameter [1]. Foraminifera have existed since the Cambrian period, and an estimated 10,000 species are still in existence [2], with the vast majority of them living on the seafloor. They are common in many modern and ancient environments, and as such have become invaluable tools for both academic and industrial purposes, such as petroleum exploration [3], biostratigraphy [4], paleoecology [5] and paleobiogeography [6]. One of the most common tasks associated with foraminifera is the identification of species from samples, like rock or ocean sediment samples. As different species live in different environments and at different geologic times, fossil foraminifera species found in samples are usually used for determining environmental or climate conditions in the past and the relative ages of marine rock layers [4]. For petroleum exploration, identities of the foraminifera species from rock samples of oil wells can specify the likelihood and the quality of oil to be found [7].

A sample from the ocean can contain thousands of foraminifera and the identification of species from samples has to be done by students or paid personnel in most of the laboratories. The identification process is tedious and time consuming, which may require weeks or even months of work for a typical study. Therefore, an automatic visual foraminifera species identification system is desirable. Such a system first captures images of foraminifera samples through a microscope. Then visual features are extracted from the images for classification, and finally each foraminifera sample is picked based on the classification result. Since foraminifera shell characteristics such as aperture location and chamber shape and arrangement, are some of the primary criteria for species identification [8], An example of this is shown in Fig. 1. None of the proposed image-based, automatic classification approaches [9], [7] make use of segmentation. This is partly due to the lack of accurate segmentation methods for foraminifera images. Foraminifera segmentation can also assist in morphological studies of foramnifera shells [10], [11], by automatically computing the size and location of chambers and apertures, thus eliminating the need for manual selection and measurement. Furthermore, the same segmentation methods can be extended to the identification of other microfossils such

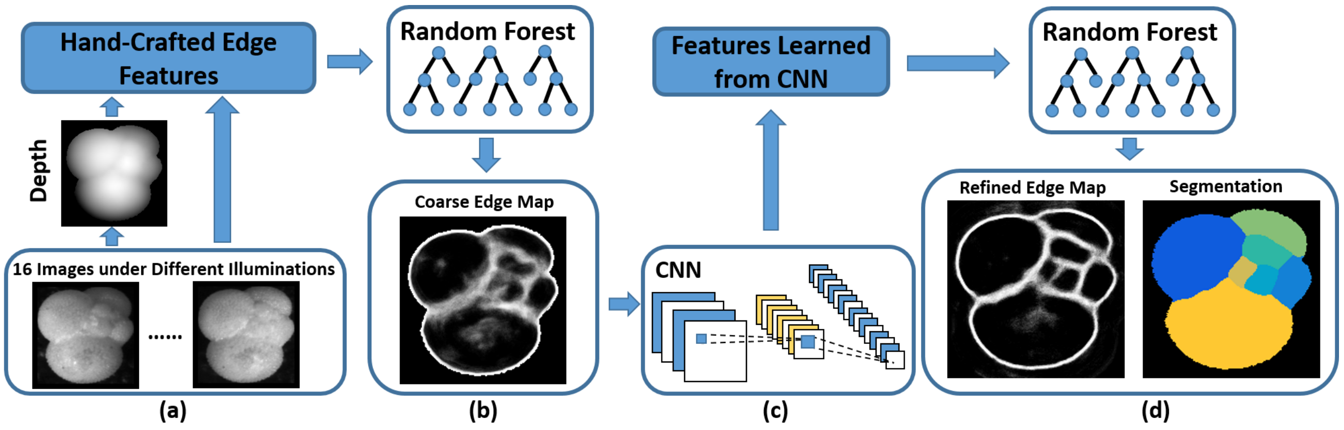

Fig. 2. Pipeline of proposed foraminifera segmentation. (a) The input data of each sample are 16 images under different light source directions. Edge features extracted from 2D images as well as a 3D reconstructed surface computed from the 16 images are properly fused. (b) A random forest is trained to estimate a coarse edge probability map. Brighter color indicates higher probability of edge. (c) A convolutional neural network (CNN) is trained to extract features from the coarse edge probability map. (d) Another random forest is trained to refine the coarse edge map and the final segmentation is obtained by post-processing on the refined edge map.

as diatoms.

In this paper, we propose a learning-based pipeline for foraminifera image segmentation on a foraminifera dataset collected by us. This dataset contains images from six widely used planktonic foraminifera species. Each sample consists of 16 images taken under different light source directions through a microscope with 30x magnification. We adopt a coarse-to-fine strategy to extract the vague edges in the images for foraminifera segmentation using a relatively small training set. A coarse edge probability map is first predicted by properly fused features extracted from original input images and further refined by the features extracted from this coarse predication. Finally the segmentation is obtained by the post-processing on the refined edge map. The entire pipeline used in our experiment is illustrated in Fig. 2 and some sample segmentation results are shown in Fig. 1. The experiments demonstrate our approach is able to segment chambers and apertures of foraminifera correctly and has the potential to provide useful features for species identification.

The remainder of this paper is organized as follows: Section II gives an overview of the related work; Section III introduces the images used for segmentation from our foraminifera dataset; A detailed description of our segmentation pipeline is introduced in section IV; Experiments of our approach as well as the analysis of the results are discussed in section V; Finally, section VI summarizes the paper and discusses the potential applications of foraminifera segmentation.

II. Related Work

Our approach falls into the category of edge-based segmentation. In this section, we briefly overview the state-of-the-art learning-based edge detection algorithms.

Edge detection is one of the crucial low-level operations in image processing and computer vision. Data driven and learning-based methods are desirable in semantic region segmentation, since they are able to treat different types of edges separately. Dollar *et al.* in [12] train a Probabilistic Boosting Tree using a large number of generic edge features at multiple scales, which is able to detect specific object boundaries. Sketch Tokens in [13] groups edge patches into sub-classes based on the edge shape and treat the edge detection as a multi-class classification task. In [14], structured edge detection is proposed to predict labels of multiple pixels simultaneously.

Recently, there has also been work on edge detection based on deep learning as deep learning methods have brought great success to various of computer vision tasks [15], [16]. In [17], features are learned by a convolutional mcRBM to predict boundaries through a deep neural network. In [18], inspired by the Sketch Tokens, edge patches are grouped into sub-classes, and a CNN is trained by sharing positive loss among the edge sub-classes. Bertasius *et al.* in [19] and [20] use object-level cues from pre-trained image classification CNNs to improve the contour detection. Xie and Tu in [21] propose the holistically-nested edge detection by taking advantage of the fully convolutional neural networks and deeply-supervised nets to guide the learning through side responses. Building upon this, Liu and Lew in [22] improve the performance by using relaxed edge labels on lower layers to make the more discriminative higher layers process more false positives.

State-of-the-art edge/boundary detection algorithms, especially deep learning based approaches, have achieved high performance on nature images and other existing datasets. We did not adopt one of the previous works for our dataset because of the following three reasons: (1) Most of the previous works are based on single images and only utilize 2D images features. In our dataset, we have 16 images under different light source directions; however, a single image cannot capture all of the desired edges. Furthermore, fusing those images to only emphasize the edges is nontrivial. Therefore, we make use of the work by Favaro *et al.* [23] to fuse input images by reconstructing a 3D surface for each sample and extracting a 3D shape feature as our additional edge features; (2) Our

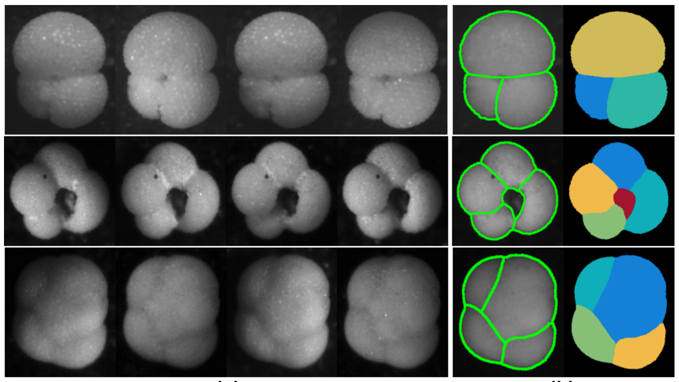

Fig. 3. Sample images and labeling from our dataset. (a) 4 of 16 images of three samples under different lighting conditions. (b) Corresponding manual labeling of samples. Apertures are labeled as red. The unobserved part of the edge between the two lower chambers of the first sample is also labeled, but some possible edges in the largest labeled chamber of the third sample are not included because they are well very visible in any of the 16 images.

images are very different from the nature images. Thus, the edge detection task can hardly benefit from the fine-tuning of the pre-trained image classification networks as the previous deep learning based methods [19], [20], [21], [22]; (3) Unlike the existing well labeled datasets, the labeled data of our dataset is highly limited. Also, due to the vague edges and low image resolution, a relatively complex CNN is required to obtain an acceptable edge predication. Training this CNN from scratch directly on the original images has a high risk of overfitting. Thus, we employ the coarse-to-fine learning strategy to make use of a simpler CNN with less parameters possible.

III. Foraminifera Image Dataset

We are creating a foraminifera dataset used for the study of visual species identification. We put each sample at the center of the field of view of an AmScope SE305R-PZ microscope with 30x magnification and use an LED ring to produce 16 different light source directions for highlighting different geometric features in the sample. The 16 perfectly aligned images of each sample are captured using a 5MP USB camera (AmScope MD500) attached directly to the microscope, which provides an approximate resolution of 450 $\times$ 450 pixels per sample. Fig. 3 (a) shows 4 of 16 images of three samples in our dataset. Till now, a dataset of 1437 foraminifera samples, in which 457 samples are manually segmented, has been collected and can be found on the project website [1]. This dataset includes: *Globigerina bulloides* (*G. bulloides*, 178 samples, 85 labeled), *Globigerinoides ruber* (*G. ruber*, 182 samples, 75 labeled), *Globigerinoides sacculifer* (*G. sacculifer*, 150 samples, 75 labels), *Neogloboquadrina dutertrei* (*N. dutertrei*, 151 samples, 75 labeled), *Neogloboquadrina incompta* (*N. incompta*, 174 samples, 77 labeled), *Neogloboquadrina pachyderma* (*N. pachyderma*, 152 samples, 70 labeled) and species other than those above (450 samples). The images of first six species (987 samples) are used for our study.

[1]https://research.ece.ncsu.edu/aros/foram-identification/

For manual segmentation, the subject has access to all 16 images of each sample and is asked to separate each chamber and aperture along edges, including the unobserved edge parts in images as long as the subject expects the existence of the edge for segmentation purpose with high confidence. However, the edges that are entirely unobserved are not labeled. Also, the regions of apertures are labeled if they are present. Fig. 3 (b) shows the labeling of three samples.

Given the 16 images of a sample, our goal is to segment this sample into several chambers and apertures. Due to the high similarity of the adjacent regions as well as the non-closed edges on the samples as shown in Fig. 3, region-based segmentation is not applicable to our study. Therefore, we adopt an edge-based segmentation methodology. That is, first find edges between chambers and apertures and then perform post-processing on the estimated edges to obtain the final segmentation. Other challenges for edge detection on our dataset are the soft edges observed due to low resolution and loss of focus, the highly limited training set, and the desire of an appropriate fusion approach to efficiently make use of all 16 images. Thus, we apply a coarse-to-fine strategy to obtain the edge detection through a small training set. As samples vary in the actual size, we re-scale the images to make the bounding box of a sample have a resolution of 150 $\times$ 150 pixels before processing the sample further. This produces images at a similar scale and reduce the computational complexity. The black background makes the extraction of the bounding box of each sample straightforward.

IV. Methodology

Let $\Omega \subset \mathbb{R}^2$ be the image domain and I_{N_x} be an image patch centering at x in image I. Given a set of 16 grayscale images of a foraminifera sample $\{I^i : \Omega \to \mathbb{R}, i = 1, \ldots, 16\}$, our goal is to predict the probability of a location $x \in \Omega$ being an edge based on a set of image patches $\{I^i_{N_x}, i = 1, \ldots, 16\}$.

As mentioned before, we apply a coarse-to-fine edge detection strategy containing two stages. First, a coarse edge probability map is learned by a set of properly fused hand-crafted edge features. This stage aims to get a high recall edge probability map, and a relative low precision is acceptable. Then a refined edge map is learned by stacking the prediction of the first stage. This stage aims to remove the noise as well as enhance the weak edges and fill some edge gaps in the coarse probability map to obtain both high recall and precision. Any general classifier can be used at both stages. In our experiment, we use a random forest at first stage, since it can provide efficient and accurate performance in edge detection tasks [13]. For the second stage, both a random forest and a CNN have been tested, where the CNN is used for feature extraction from the coarse prediction.

A. Hand-Crafted Feature Extraction

At this stage, inspired by [12] and [13], we compute several types of edge features from all 16 images per sample to capture

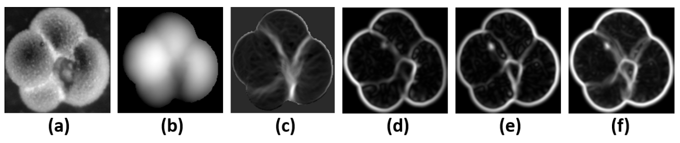

Fig. 4. Selected hand-crafted features of the second sample in Fig. 3. (a) Standard deviation image. (b) Depth map of reconstructed 3D surface. (c) Curvature map. This map captures all the edges of chambers but can hardly capture the aperture edges. (d) and (e) Two ridge maps computed from 2 of 16 images. Part of the aperture boundary is not captured in (d) but captured in (e). (f) Maximum of 16 ridge maps.

the edge cues in various aspects. We employ both generic edge features and a 3D shape feature computed from a reconstructed 3D surface of the sample.

The generic edge features used in our experiment include gradient, difference of Gaussian (DoG), and ridge detector. The gradient magnitudes are computed per image using Gaussian kernels with $\sigma = 0.8$ and 2. In addition, we split the gradients into four directions, $\theta = 0$, $\pi/4$, $\pi/2$ and $3\pi/4$, to compute 4 more gradient maps per image at each scale. Two DoG maps are computed from the difference of three Gaussian blurs with $\sigma = 2.5$, 4 and 6.4 per image. Ridge points are defined as the maximum or minimum points in the main principal curvature of an image function [24]. The ridge detector is adopted due to the presence of some thick edges on samples that resemble ridges or valleys of the image function, which produce double low magnitude lines when applying edge detectors. Because most of the edges have color intensities lower than the chambers, we only use the ridge detector to highlight the valley in the images. The detector has the form

$$L = \frac{1}{2}(F_{xx} + F_{yy} - \sqrt{(F_{xx} - F_{yy})^2 + 4F_{xy}^2}), \qquad (1)$$

where F_{xx}, F_{yy} and F_{xy} are the second derivatives of the image I in the gradient direction x, y and xy, respectively. We compute one ridge map per image using a Gaussian filter with $\sigma = 3.2$. Fig. 4 (d) and (e) show examples of the ridge maps.

As the entire set of edges can hardly be captured in a single image, simply concatenating features of all the input images may introduce noise and is computationally inefficient as well. However, all the edge information in the images can be collected by maximizing the edge cues over the 16 images. Since the images are perfectly aligned, the fused feature maps can be obtained by taking the maximum feature value at each pixel location over the 16 images. We do not utilize quantiles for feature fusion to get rid of some noise occasionally shown in the images because it may remove some weak desirable edges as well, which leads to a lower recall. After the aggregating, the number of feature maps reduces from 208 to 13.

As we can see in Fig. 3, some edges have small variations under different light sources, the standard deviation at each pixel location over 16 images can provide relatively rich edge information (An example of this is shown in Fig. 4 (a)). Therefore we compute a DoG of the standard deviation image using $\sigma = 3.2$ and 5.1 as an additional feature map.

To make use of the 16 images which highlight the different geometric features of the sample, we apply an efficient uncalibrated photometric stereo technique [23] to reconstruct a 3D surface of each sample for better fusion of the input images, followed by a Bilateral filter [25] with $\sigma = 0.1$ to remove the small variations on the chambers. A sample depth map of a reconstructed 3D surface is shown in Fig. 4 (b). The noise introduced by color changes and textures on the samples is removed, since they do not have large changes in depth. Also, edges between chambers are captured well in the depth map. We compute the maximum normal curvature [26] of this reconstructed surface as another feature map. This feature map measures the maximum bending at each point of the surface and is computed by

$$\kappa = H + \sqrt{H^2 - K}, \qquad (2)$$

where H is the mean curvature and K is the Gaussian curvature of the surface. Please refer to [26] for more details. Fig. 4 (c) shows an example curvature map.

In summary, we have 15 maps per sample, consisting of 10 gradient maps, 3 DoG maps, 1 ridge map, and 1 curvature map. Each pixel location is represented by a 3375 dimensional feature vector, obtained by centering a 15×15 patch on that pixel and concatenating the features obtained from each of the 15 feature maps.

B. Coarse Edge Probability Map Prediction

We use a random forest to predict the coarse edge probability map. A random forest is an ensemble of randomly trained decision trees and the output is the average prediction of individual trees [27]. Since boundaries of foraminifera and apertures are comparatively sharper than the edges between chambers, these two types of edges are divided into two classes to increase the training efficiency as well as to distinguish between apertures and chambers for segmentation. Afterwards, the coarse edge probability map is predicted by a 3-class classification random forest.

Fig. 7 (c) and (d) show some coarse edge probability maps estimated at this stage. Though the recall of the observed desired edges is high, due to the outcomes of max pooling during feature extraction, the estimated edges are thick, and edges observed in the texture (but not corresponding to real chamber boundaries) are also detected in the probability map. Besides the reason that some edges on the samples are narrow regions rather than thin lines, Fig. 4 illustrates another reason arises the thick predicted edges. The shade of the edge varies slightly under different light source directions, which introduces small offsets to the edge features over 16 images. Thus, the edge in the fused feature map is a thick region rather than a thin line. Other undesired patterns of the coarse edge maps are the low probabilities of some edges and non-closed boundaries due to the weak edges or the unobserved edges in the images. To obtain higher precision edge maps, a second stage for refinement is necessary.

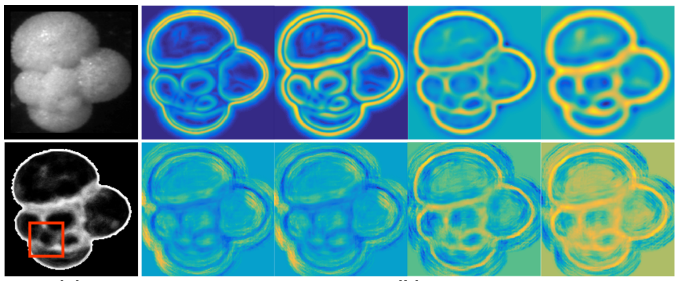

Fig. 5. Hand-crafted features and deep features computed from a coarse edge probability map. (a) Top: One of original sample images. Bottom: Corresponding coarse edge probability map. The unclear edges in the original images get low probability (marked with a red box). There is also some noise due to the texture on the chambers. (b) Top row: Four hand-crafted edge features of the coarse edge map, including two gradient magnitude maps and two DoG maps. Bottom row: Four sample deep feature maps. In the hand-crafted feature maps, especially for the first two, the weak edges of the coarse map have almost the same patterns as the noise on the upper and right chambers; however, in deep feature maps, the weak edges are shown much clearly with patterns far different from the noise, which makes the classifier distinguish weak edges from noise more easily. Best viewed in color.

C. *Refinement of Coarse Edge Probability Map*

At this stage, we further refine the coarse edge probability map to obtain a high precision by thinning the edges, removing noise, enhancing the probability of desired edges and closing as many boundaries as possible. From Fig. 7 (d), we can tell the location of chamber and aperture edges purely based on the noisy coarse edge map. But those edges cannot be extracted automatically with a high accuracy through a simple process, like thresholding. Therefore, we apply a learning-based approach at this stage and only the predicted coarse edge map is used as input. Similar to the previous stage, features are first extracted from the coarse map and each pixel is represented by a small patch centered around it. Then the edge map is refined through another classifier. Two types of features have been tried in our experiment: hand-crafted edge features and deep features learned by a CNN trained for edge map refinement. Both approaches are discussed in detail in this section. We start from the hand-crafted feature extraction.

1) Hand-Crafted Features: Since the edge location in the probability map is much clearer than in the original 16 images, a smaller number of features need to be extracted from the probability map. This reduced feature set includes gradient and DoG with the same parameters as computed from the original images at the first stage. In summery, 12 feature maps are computed and a patch size 15×15 gives a 2700 dimensional feature vector for each pixel location. Because the classification of aperture boundaries at the first stage is accurate enough for aperture region segmentation and all types of edges have similar patterns in the input coarse probability map, pixels are labeled as two classes: edge and non-edge. Next, the refined edge map is obtained by training a classification random forest again. Fig. 7 (e) shows some results of the refined edge map. The edge probability map is much clearer than before and the weak edges are enhanced as well. Also, some non-closed boundaries are closed, because the training data contains some examples of filling edge gaps, which makes the classifier have the ability to hallucinate edges by fusing features in the patch as discussed in [12]. Better boundary completion performance can be achieved through careful selection of training data and use of larger patches.

2) Deep Features: The features we extract for refinement in section IV-C are purely edge detectors. However, there may exist better features to remove noise as well as enhance the low confidence edges in the coarse map. To investigate this, we employ a CNN to learn a set of features automatically from the coarse probability map.

A CNN consists of one or more convolutional layers followed by one or more fully connected layers. It often contains a pooling layer between two convolutional layers to reduce complexity [28]. In our experiment, we use four convolutional layers and two fully connected layers as used in [18], which is sufficient for low-level feature extraction as suggested in [18]. The only difference is that we do not use local response normalization layers (LRN). The input of this CNN is a 32×32 patch of coarse probability map and it is trained to label the center point of each patch as edge or non-edge. Dropout [29] with probability 0.5 is used in the first fully connected layer (FC1) during training to reduce the risk of overfitting.

Similar to [18], we use the output of FC1 (a 128-dimensional feature vector) as our deep features of each pixel and thus we can show the feature maps in the same way as in [18]. The comparison of hand-crafted features and deep features can be found in Fig. 5. Both weak and strong edges in the coarse edge map are more distinguishable from noise in deep feature maps, which makes the noise suppression and edge enhancement much easier through a classifier. We concatenate the features of all the pixels in a 11×11 patch to obtain a 15488-dimensional feature vector to represent the centering pixel. Again, a random forest is trained using the deep features for the edge probability map refinement. Fig. 7 (f) shows some sample results. The refined edge maps are comparable to those refined by hand-crafted features, but are more accurate for some small details and also get more closed boundaries because of the more powerful features.

Fig. 6 explains how the edge detection improves by using the coarse-to-fine strategy. At the first stage, only a set of small patches is used to predict each pixel location. Though the prediction is noisy, the edge information is clearer than the original input. Every time we go to the next stage, we can efficiently gather the information from larger patches in original images by taking small patches in the prediction of this stage and obtaining an even clearer predication. If deep features are employed, each pixel in the final predication is based on a set of 57×57 patches in the original images. This is similar to the image processing through a CNN, which indicates the possibility of using an end-to-end learning process through a CNN. However, instead of learning features from original images directly, we use hand-crafted features at the first stage followed by a relatively simple CNN. This procedure largely reduces the risk of overfitting due to the

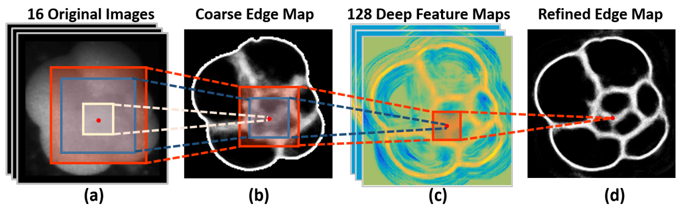

Fig. 6. Data flow of our coarse-to-fine edge detection using deep features. Paths with different colors illustrate computation patch of a pixel location at different stages. (a) Input 16 sample images. (b) Each pixel in the coarse edge map is predicted by a set of 15×15 patches in original images. (c) Each pixel in the deep feature maps is computed by a 32×32 patches in the coarse edge map and thus depending on a set of 46×46 patches in the original images. (d) Each pixel location in the refined edge map is predicted by a set of 11×11 patches in deep feature maps and then depending on a 43×43 patch in the coarse edge map. Thus each pixel in the final refined edge map is predicted based on the information gathering from a set of 57×57 patches from the original images.

highly limited training data.

D. *Post-Processing*

To further close the boundaries, a closing operation using a disk with radius of 2 is applied to the refined edge probability map after thresholding. Afterwards, the final segmentation is obtained by growing the regions separated by those edges based on distance. If aperture edges are detected in the coarse probability map, then a region in the image that is overlapping with the aperture edge is labeled as an aperture given that its average color intensity (over all 16 images) is considered dark enough. We check this by comparing the color against a two group k-means clustering performed using the training data. The clustering using the training data is repeated 5 times in order to gain robustness, and a test region is labeled as aperture only if falls in the appropriate cluster more than 50% of the time (i.e., 3 times in this case). Fig. 7 (g) shows some final segmentation results. The edge thinning by region growing removes some non-closed boundaries but it also gets rid of the small branches that could be produced through skeletonization.

V. Experiment

In this section, we evaluate our foraminifera edge detection and segmentation qualitatively and quantitatively.

A. *Implementation Details*

We randomly choose 100 samples from the 457 labeled samples as our training samples: 25 are used to train the first stage, and 75 for the second stage. That leaves 357 labeled and 530 unlabeled samples for testing. The training edges are thickened by 6 pixel at the first stage and by 4 pixels at the second stage.

For the first stage, 62.5k non-edge patches (2.5k per sample), 37k chamber edge patches (all chamber edges in 25 samples) and 50k other edge patches (2k per sample) are randomly sampled as the training set. At the second stage, the random forest using hand-crafted features is trained by randomly sampling 96k positive patches (all edges in 75 samples) and 150k negative patches (2k per sample) from the 75 samples. To train the deep features, the patches from the 75 samples are divided into a set of 17k validation data points (7k positive and 10k negative) from 5 samples and a set of 229k training data points (89k positive and 140k negative) from 70 samples. Then a random forest using the deep features is trained by randomly sampling 37.5K positive patches (500 per samples) and 60k (800 per sample) negative patches from the 75 samples.

The random forests parameters used in our experiment are all the same. Each random forest is trained until all the leaf nodes only contain one class using a collection of 150 trees, except for the one trained using deep features which consists of 50 trees to reduce the time consumption. The CNN for learning deep feature is trained using 200 epochs with batch size 5000. The learning rate for the first 100 epochs is 0.01 and set to 0.001 for the remaining 100 epochs.

The random forests are implemented in MATLAB and the CNN is implemented in Python using the *Tensorflow* framework [30]. The experimental environment is CPU i7 with 64GB RAM and NVIDIA TITAN GPU. For a sample image with resolution 166×166, computing one coarse edge map needs 24 seconds. Refinement through hand-crafted features requires 18 seconds and through deep features requires about 100 seconds (1 minutes for feature extraction and 40 seconds for edge patch classification). Note that these steps can be largely optimized through implementation of a fully convolutional network and parallel computing.

B. *Qualitative and Quantitative Evaluation*

Results of each stage of our approach are shown in Fig. 7. Coarse edge probability maps (RF) obtained at the first stage are able to collect most of the edge clues from the input image set, but some edges are thick or detected with low confidence, which causes some non-closed boundaries. However, the aperture boundaries and chamber edges are separated well at this stage (RF label). At the second stage, hand-crafted features (RF+RF(HF)) and deep features (RF+RF(DF)) give similar performance; however, deep features provide higher confidence for weak edges, more accurate details and more closed boundaries. Also, as shown in the last row of Fig. 7, if the edges in the input images are too weak to get captured at the first stage, they will not be detected by the refinement either. Another observation is that the machine segmentation can be better than manually labeling in some cases. Edges unclear to humans are located by efficiently collecting information from a neighborhood of each pixel.

To quantitatively demonstrate the improvement of the second stage and evaluate the performance, we use three edge detection evaluation metrics: the best F-measure for a fixed threshold (ODS), the average F-measure of the best threshold for each image (OIS) and the average precision over all threshold (AP), as well as three segmentation metrics: the best weighted covering score (W), un-weighted covering score (Un-W) [31] and recall of regions (Recall) for fixed thresholds. The edge recall is defined as the percentage of the labeling

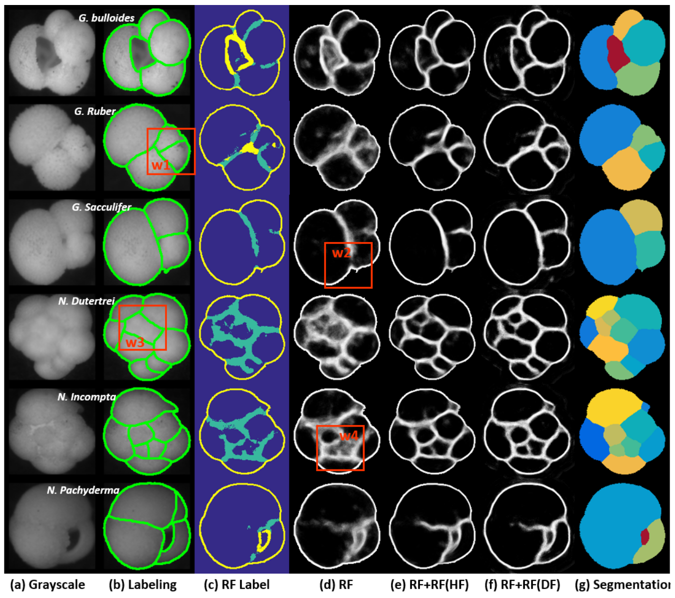

Fig. 7. Sample results of different species. Sample images (a), manual labeling (b), edge labeling after first stage (c), edge probabilities from various approaches (d)-(f), and final segmentation using deep features (g) are shown. For the first stage (c), edges between chambers are labeled in green and other edges are yellow. Aperture is labeled as red in segmentation (g). Window `W1` shows an example in which correct machine labeling and human labeling can have some offsets. Window `W2` shows how deep features can close non-closed boundaries in the refined map. Window `W3` illustrates an example in which machine labeling can be better than human labeling. Window `W4` shows how the refined map (stage 2) can enhance weak edges as well as remove noise from the coarse map. The last row shows a failed example due to edges that are too vague to be captured in the coarse map.

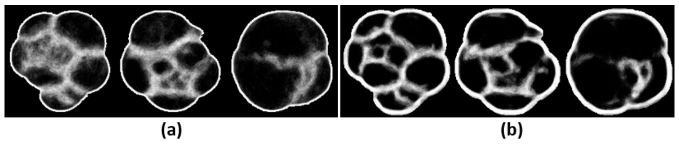

Fig. 8. Additional results. (a) Using patch size 57×57 for random forest. (b) Using deep features computed by a CNN from a 16-channel image.

edges covered by the thresholding estimated edge map. The edge precision is defined as the percentage of the estimated edge pixels covering the labeling edges. The labeling edges are thickened to 8 pixels wide for precision computation since the actual edges between chambers can be narrow regions, which generate thick estimated edges and small offsets between human labeling and the correctly estimated edges. For segmentation evaluation, weighted covering score is the average covering score weighted by the area of each region in human labeling. We also use un-weighted covering score because for species identification purpose, every chamber and aperture have equal importance. To evaluate how well the chambers and apertures are detected, we define the recall of regions for each sample as the percentage of the labeling regions detected in the estimated segmentation, where each region in the human labeling is regarded as detected if there exists an overlapping estimated region with covering ratio greater than 0.5.

Table I presents the scores at each stage. The precision

TABLE I
EDGE DETECTION RESULTS USING DIFFERENT FEATURES AND CLASSIFIERS

	Edge			**Region**		
	ODS	OIS	AP	W	Un-W	Recall
RF	.841	.853	.698	.779	.645	.695
RF+RF(HC)	.867	.875	**.775**	**.825**	.715	.779
RF+RF(DF)	**.872**	**.879**	.771	.821	**.721**	**.786**
RF(2D)	.842	.852	.691	.757	.616	.659
RF(100)	.841	.853	.650	.803	.678	.753
RF+RF(HC43)	.868	.877	.759	.821	.713	.764

is relatively low compared with other metrics even with the thickened labeling because of the thick estimated edges and the inaccurate of human labeling as shown in Fig. 7. But we do observe the improvement of precision after the refinement. Besides this, we can see a large performance gain from the first stage (RF) to the second stage (RF+RF(HC) and RF+RF(DF)), especially for region scores. The closing of boundaries only brings small changes in the sense of edge, but can highly increase the segmentation accuracy. The increase of average precision also suggest the thin edges in refined edge maps. Deep features (RF+RF(DF)) refinement is slightly better than hand-crafted features (RF+RF(HC)). The higher un-weighted covering and region recall scores indicate that deep features are more capable in detecting small regions, most of which are apertures. Thus, deep features are preferred for the applications in which aperture detection is crucial. Otherwise, hand-crafted features, which are good enough to correctly segment most of the chambers, can be used for computational efficiency.

We also report scores of the first stage without 3D features (RF(2D)), the decrease in the region scores of (RF) demonstrates 3D features can enhance some weak edges to close more boundaries. To be comparable with deep features (RF+RF(DF)), which gather information from 43×43 patches in the coarse map, we report the scores of refinement with hand-crafted features using 43×43 patch (RF+RF(HC43)). The similar scores with 15×15 patch size indicate that the better performance in small details cannot be achieved by simply increasing the patch size. To show that the improvement of the second stage is not purely because of the increasing of the training set, the scores of the first stage with all 100 training samples (RF(100)) are reported. It hardly improves the edge detection performance, but improves the segmentation by providing higher detection confidence for some weak edges. Still, it cannot beat the performance of using a second stage for refinement.

We have also tried other two settings with only one stage using 100 training sample images: using patch size 57×57 for a random forest to compare with deep features, and extracting deep features directly from a 16-channel image formed from the original images using a CNN with the same architecture as used in our experiment. Results of these two settings are shown in Fig. 8. The outputs do not have the same quality as the two stage refinement process, which is able to gather information more efficiently and requires a less complex CNN.

VI. Conclusion

In this paper, we propose a coarse-to-fine edge detection approach on our foraminifera dataset. We use a two-stage strategy to achieve accurate edge detection and segmentation performance using a relatively small training data through additional 3D features and features learned from a CNN. The experiments demonstrate that the machine segmentation is able to correctly label chambers and apertures on samples. There are several applications for the segmentation results. Sample images of specific species can be selected by searching among the segmentation results. For example, one can get images of *G. bulloides* by picking the samples with one aperture and four chambers in the segmentation results. Also, by representing regions as nodes in a graph and connecting the nodes of adjacent regions, we can construct a graph to represent the structure of each sample. General structures of different species can be learned from a training set and species can be identified through graph matching. Additionally, the actual size of chambers and apertures can be computed to help with the morphological study of the shells.

Acknowledgment

We thank Svapnil Ankolkar, Mehdi Garcia Cornell, Bryant Delgado and Sadaf Iqbal for the help on image capturing and Jason Farsaie for him help on image labeling. This work was supported by the National Science Foundation under award OCE-1637039.

References

[1] B. K. B. K. Sen Gupta, *Modern foraminifera*. Dordrecht ; Boston : Kluwer Academic Publishers, 1999, includes bibliographical references (p. 299-351) and indexes.

[2] K. Vickerman, "The diversity and ecological significance of protozoa," *Biodiversity & Conservation*, vol. 1, no. 4, pp. 334–341, 1992.

[3] J. M. Bernhard and B. K. Sen Gupta, *Foraminifera of oxygen-depleted environments*. Dordrecht: Springer Netherlands, 2003, pp. 201–216.

[4] J. A. Cushman and A. C. Ellisor, "The foraminiferal fauna of the anahuac formation," *Journal of Paleontology*, vol. 19, no. 6, pp. 545–572, 1945.

[5] W. A. Berggren, "Ecology and palaeoecology of benthic foraminifera." *The Journal of Protozoology*, vol. 39, no. 4, pp. 537–537, 1992.

[6] W. A. BERGGREN, "A cenozoic time-scale some implications for regional geology and paleobiogeography," *Lethaia*, vol. 5, no. 2, pp. 195–215, 1972.

[7] S. Liu, M. Thonnat, and M. Berthod, "Automatic classification of planktonic foraminifera by a knowledge-based system," in *Proceedings of the Tenth Conference on Artificial Intelligence for Applications*, Mar 1994, pp. 358–364.

[8] J. Kennett and M. Srinivasan, *Neogene planktonic foraminifera: a phylogenetic atlas*. Hutchinson Ross, 1983.

[9] R. Marmo and S. Amodio, *A Neural Network for Classification of Chambers Arrangement in Foraminifera*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 271–278.

[10] B. H. Corliss, "Morphology and microhabitat preferences of benthic foraminifera from the northwest atlantic ocean," *Marine micropaleontology*, vol. 17, no. 3-4, pp. 195–236, 1991.

[11] E. Boltovskoy, D. B. Scott, and F. Medioli, "Morphological variations of benthic foraminiferal tests in response to changes in ecological parameters: a review," *Journal of Paleontology*, vol. 65, no. 02, pp. 175–185, 1991.

[12] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 1964–1971.

[13] J. J. Lim, C. L. Zitnick, and P. Dollr, "Sketch tokens: A learned mid-level representation for contour and object detection," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, June 2013, pp. 3158–3165.

[14] P. Dollr and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1558–1570, Aug 2015.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2012, pp. 1097–1105.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 580–587.

[17] J. J. Kivinen, C. K. I. Williams, and N. Heess, "Visual boundary prediction: A deep neural prediction network and quality dissection," in *AISTATS*, 2014.

[18] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3982–3991.

[19] G. Bertasius, J. Shi, and L. Torresani, "Deepedge: A multi-scale bifurcated deep network for top-down contour detection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 4380–4389.

[20] ——, "High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 504–512.

[21] S. Xie and Z. Tu, "Holistically-nested edge detection," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1395–1403.

[22] Y. Liu and M. S. Lew, "Learning relaxed deep supervision for better edge detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 231–240.

[23] P. Favaro and T. Papadhimitri, "A closed-form solution to uncalibrated photometric stereo via diffuse maxima," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 821–828.

[24] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 117–156, Nov 1998.

[25] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Jan 1998, pp. 839–846.

[26] R. Goldman, "Curvature formulas for implicit curves and surfaces," *Comput. Aided Geom. Des.*, vol. 22, no. 7, pp. 632–658, Oct. 2005.

[27] A. Criminisi, E. Konukoglu, and J. Shotton, "Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning," Tech. Rep., October 2011. [Online]. Available: https://www.microsoft.com/en-us/research/publication/decision-forests-for-classification-regression-density-estimation-manifold-learning-and-semi-supervised-learning/

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[31] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, May 2011.