Deep Neural Network for 3D Surface Segmentation based on Contour Tree Hierarchy

Wenchong He * Arpan Man Sainju * Zhe Jiang * Da Yan †

Abstract

Given a 3D surface defined by an elevation function on a 2D grid as well as non-spatial features observed at each pixel, the problem of surface segmentation aims to classify pixels into contiguous classes based on both non-spatial features and surface topology. The problem has important applications in hydrology, planetary science, and biochemistry but is uniquely challenging for several reasons. First, the spatial extent of class segments follows surface contours in the topological space, regardless of their spatial shapes and directions. Second, the topological structure exists in multiple spatial scales based on different surface resolutions. Existing widely successful deep learning models for image segmentation are often not applicable due to their reliance on convolution and pooling operations to learn regular structural patterns on a grid. In contrast, we propose to represent surface topological structure by a contour tree skeleton, which is a polytree capturing the evolution of surface contours at different elevation levels. We further design a graph neural network based on the contour tree hierarchy to model surface topological structure at different spatial scales. Experimental evaluations based on real-world hydrological datasets show that our model outperforms several baseline methods in classification accuracy.

1 Introduction

Given a 3D surface defined by an elevation function over a 2D grid as well as non-spatial features observed at each pixel, the problem of surface segmentation aims to classify pixels into segment classes based on non-spatial features and surface topology. For example, in hydrology, scientists are interested in mapping the surface water extent from remote sensing imagery based on not only spectral features of pixels but also the geographic terrains [1, 2, 3, 4]. The problem is unique from traditional image segmentation in that the spatial extent of class segments follow a topological structure based on surface contour patterns. For example, flood extent boundary spreads over a geographic terrain with equal elevation contours.

Societal applications: Topological surface segmentation is important in many applications such as flood extent mapping on a topographic surface in hydrology, crater detection in planetary surface, molecular surface segmentation in bio-science [5], plant branching structure analysis in phenology. In hydrology, the topological surface is a digital elevation map, the non-spatial features can be optical remote sensing imagery imposed on the surface, and the target output map can be water surface extent [6, 7, 8, 9, 10]. Mapping water surface extent plays an important role in national water forecasting and disaster response. In astronomy, scientists are interested in detecting craters on a 3D planetary surface based on the topological structures. Such information can reveal details on how a planet evolves over time. In bio-science, people are interested in segmenting protein surface based on the topology to identify rigid components and to understand the function of cavities and protrusions in protein-protein interactions [5]. In botany, researchers are interested in assessing and comparing the branching structure of plants based on the surface topology. Such structures are important features to understand the phenotype of a plant (the interactions of genetic background with the environment).

However, the problem poses several unique challenges. First, the spatial extent of class segments follows surface contours in the topological space, regardless of their spatial shapes and directions. This violates the assumption by the widely popular deep convolutional neural networks that the spatial structure of class segments is regular on a Euclidean plane (a grid framework). Second, the topological structure exists in multiple spatial scales based on different surface resolutions. Specifically, on a fine resolution, a surface will show more local fluctuations; but on a coarse resolution, a surface will only capture the global trend ignoring local details.

Existing techniques for 3D surface segmentation can be categorized into traditional non-deep learning approaches and deep learning approaches. Non-deep learning approaches include region growing, hierarchical clustering, iterative clustering, and graph algorithms [11].

*The University of Alabama, Tuscaloosa, AL, 35487 (whe11@crimson.ua.edu, asainju@crimson.ua.edu, zjiang@cs.ua.edu).

†The University of Alabama at Birmingham, Birmingham, AL, 35294 (yanda@uab.edu).

The region growing approach [12] starts with one or several seed elements and greedily expands the seed into regions by adding in other likely elements. Among the region's growing methods, one important technique is called the watershed method [13], which grows seeds based on a height function defined on the surface (e.g., from a local minimum to a ridge). The hierarchical clustering approach differs from the region growing approach in that it considers all vertices as individual seeds (clusters) and greedily selects two small clusters to merge into a bigger cluster in each iteration [14]. The iterative clustering algorithm (e.g., K means) can iteratively assign vertices into different clusters based on objective criteria. The graph-based approach considers the surface as a graph and uses graph-cut algorithms to segment the surface [15]. Deep learning has achieved great progress for image segmentation [16]. The most popular technique is to learn a fully convolutional network [17, 18] to extract high-level semantic features together with deconvolution or upsampling layers to combine features at multiple scales for detailed segmentation [19, 20]. These methods often require partitioning the image into smaller patches (e.g., 224 by 224) and learning a fully convolutional network for each patch with the depth channel as an additional feature [21, 22], without explicitly modeling the topological constraint. Some works aim to resolve this issue by adding spatial transformation in convolution kernels through a depth or Gaussian term, e.g., bilateral filters [23] and depth-aware CNN [24], but they still do not fully capture the topological structure for the entire surface. Other methods incorporate topological constraints into image segmentation in two ways: enforcing MRF or CRF-based topological constraints in the inference step [25], which is unable to fully utilize a topological prior to train a model; or using a topology-aware loss function to train the neural network by leveraging persistent homology to define a topological loss on the predicted class image [26, 27], which only focuses on basic topology constraint (e.g., the number of connected components or holes). In summary, existing methods do not fully incorporate the topological structures in the form of surface contours.

In contrast, we propose a novel graph neural network model for 3D surface segmentation by representing surface topological structure as a *contour tree skeleton*. A contour tree is a polytree capturing the evolution of surface contours at different elevation levels in computational topology. We further design a graph neural network based on the *contour tree hierarchy* to model surface topological structure at different spatial scales. We design downsampling and upsampling paths in the graph neural network based on the contour tree hierarchy to extract features at different scales. Experimental evaluations based on real-world hydrological datasets show that our model outperforms several baseline methods in classification accuracy.

2 Problem Statement

3D surface: A 3D surface is defined as an elevation (or depth) function on a 2D grid. We denote the elevation function by an array $\mathbf{E} \in \mathbb{R}^{N \times N}$, where N is the number of pixels in each spatial dimension. On a 3D surface, there can be m non-spatial explanatory features as well as a target class layer. We denote each pixel as $\mathbf{s}_n = (\mathbf{x}_n, e_n, y_n)$, where $\mathbf{x}_n$, e_n, and y_n represent the m explanatory features, elevation value, and the class of the pixel respectively. Examples of 3D surface include the geographic terrains on the Earth's surface based on digital elevation and local protein elevation surface in biochemistry. Note that in these cases, the underlying 2D grid is actually on a sphere, but it can be considered as a flat in a small local area.

Given a 3D surface with non-spatial explanatory features and target classes, the problem aims to learn a model to predict surface classes based on the explanatory features and the surface topological structure. For example, in hydrology, scientists are interested in mapping the surface water extent from remote sensing imagery based on not only the spectral signature of pixels but also the geographic terrain in a digital elevation map.

3 Methods

This section introduces our proposed approach to address the unique challenges in topological surface segmentation. The key idea is to represent a topological surface by a contour tree skeleton. We design a graph neural network model with graph convolutional operations on contour tree nodes. In order to learn a topological structure at different spatial scales, we construct a contour-tree hierarchy and design downsampling and upsampling paths in the graph neural network.

3.1 Overview of deep model architecture Figure 1 provides an overview of the proposed model. The entire model architecture looks similar to the U-Net model [17] in traditional 2D image segmentation. The input of the model is a contour tree with extracted features on each contour node (through aggregating the features of individual pixels on the same contour). The node features are illustrated by a rectangle whose height represents the number of nodes and whose width represents the feature dimension. Two consecutive graph convolution operations are added at each level to further learn features on the topological skeleton. In the downsample path on the left, pooling operation is added by aggregating contour tree nodes based on node collapsing

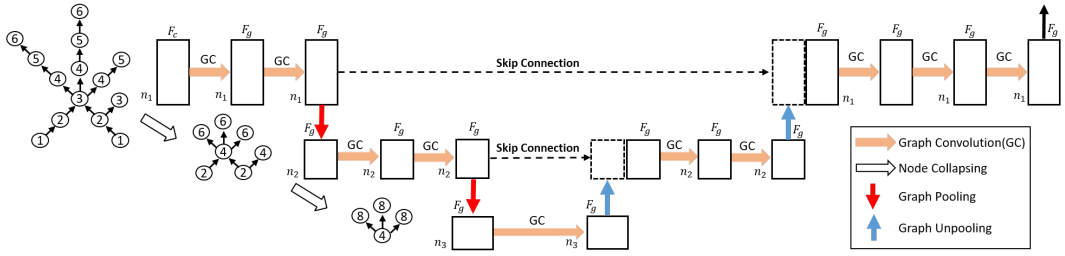


Figure 1: Overview of the model architecture

operations. In the upsample path on the right, unpooling operations are used to project features to a fine resolution. Pooled features at a coarse-scale are concatenated with features at a fine-scale from the downsample path through skip connection to combine both global and local features for detailed segmentation.

3.2 Surface representation by a contour tree

The key idea in the proposed approach is to represent the **topological skeleton** of a surface by a contour tree, which is a fundamental concept in computational topology. We first introduce several important concepts and formally define a contour tree.

Level set, contour: A level set on a 3D surface is the set of pixels with an equal elevation. Formally, $L(e_0) = \{\mathbf{s}_n | e_n = e_0\}$, where e_0 is an elevation threshold. Specifically, each level set on a 3D surface is composed of one or several connected components called contours, i.e., $L(e_0) = \bigcup_{k=1}^{K} C_k(e_0)$, where $C_k(e_0)$ is only self-connected (no connectivity between contours). For example, in a 3D surface defined by the elevation function in Figure 2(a), there are fourteen separate contour components in total. For instance, there are two contours $C_a(1)$ and $C_b(1)$ for elevation 1, as well as three contours $C_a(4)$ and $C_b(4)$ for elevation 4, shown in Figure 2(b).

Topological structure: In this paper, we define topological structure as the evolution of the topology of surface contours along with elevation levels. Specifically, as the elevation level rises, new contours can appear, merge, split, and disappear. Consider the example of Figure 2(a). As the elevation level rises from 1 to 6, two contours $C_a(1)$ and $C_b(1)$ first appear, then spread to $C_a(2)$ and $C_b(2)$ respectively, merge into $C_a(3)$ for elevation 3, split into three contours $C_a(4), C_b(4), C_c(4), C_d(4)$, and finally disappear after elevation 6. The topological structure provides a constraint on how the spatial extent of class segments spread over the surface. Pixels on the same contour have the same segment class. If we look at the evolution of a contour, we can find a topological order such as $C_a(1) \rightarrow C_a(2) \rightarrow C_a(3)$. For example, in hydrological applications, the spatial extent of water on the surface in Figure 2(a) is constrained by the contour patterns.

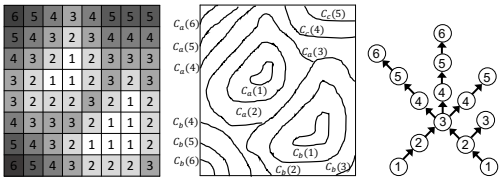


Figure 2: An example of an elevation surface (a), its contours (b) and contour tree (c)

Contour tree: A contour tree is a polytree whose nodes are surface contours and whose edges are the topological order (based on elevation gradient directions) between two adjacent contours. Formally, a contour tree can be denoted by $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of contours denoted by $\mathcal{V} = \{C_k(e) | e_{min} \leq e \leq e_{max}\}$ (e_{min} and e_{max} are the minimum and maximum elevation), and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ are the topological order between adjacent contours based on the elevation gradient. Contour tree is a fundamental tool to represent a surface topological structure in computational topology. For example, the contour tree corresponding to the 3D surface in Figure 2(a) is shown in Figure 2(c), where the tree branches show the evolution of contours at different elevation levels. The contour tree structure provides a topological structural constraint on how a potential class segment evolves on the surface. Since pixels on the same contour have the same segment class based on the topological constraint, we can aggregate features and classes from individual pixels to their corresponding contour tree nodes. We denote the node features and classes in a contour tree as $\mathbf{X} \in \mathbb{R}^{n \times m}$ and $\mathbf{Y} \in \mathbb{R}^{n \times 1}$ respectively, where n is the total number of nodes in a contour tree.

Its more general definition in the field of topology is *Reeb graph* [28], which captures how the level sets of a smooth and differential function evolves on a manifold (more details can be found in Morse theory [29]). A contour tree is a special case of Reeb graph when the function is defined on a 2D plane such that its level sets are in the form of loops called contours.

Representing the topological skeleton of a surface using a contour tree is important for our surface segmentation problem. The contour tree provides a global topological structure of a surface that often reflect the physical constraints of the target surface class segments. In other words, the contour tree structure provides an opportunity to learn structural features on a surface in the topological space. For example, in hydrology, the extent of water distribution on an elevation surface (also called geographic terrain surface) follows the topological structure of contours due to gravity. In botany, the branching structure of a planet also follows the topology of the plant body along with the height. Such topological structural features are very hard, if possible, to learn in the original surface representation (2D grid map) due to a lack of a regular shape or direction. Extensive research has been done in the field of computational topology to study efficient algorithms for contour tree construction [29, 28]. We use the algorithm in [30], which involves scanning all locations by an decreasing (and increasing) order of elevation values to create a joint tree (and a split tree) and then merging the two trees together. Its overall time complexity is approximately $O(n \log n)$, where n is the number of vertices [30]. To apply the algorithm, we added random perturbation to make pixel elevation unique and then collapsed pixels on the same contour into one tree node.

3.3 Graph convolution on a contour tree After representing a topological surface by a contour tree skeleton, we can conduct graph convolution operations on the contour tree. A graph convolution layer generalizes the traditional convolutional layer from 2D images to a graph. This is non-trivial since a graph usually does not have a fixed neighborhood size. There exists extensive research on designing convolutional layers on graphs [31, 32, 33, 34, 35]. Techniques can be categorized into spectral methods and spatial methods. The former utilizes graph Laplacian to aggregate features from neighboring nodes. The latter re-samples a neighborhood to a fixed size so that a traditional convolutional operator can be applied. It is worth to note that graph convolution does not consider the geometric structure of the mesh surface. But since the previous curvature filter layer already extracts geometric features at local vertices, the graph convolution layer simply focuses on incorporating spatial contiguity into extracted geometric features based on graph connectivity.

Specifically, we propose to use two popular graph convolutional layers: ChebyNet [36] and diffusion graph convolution [37]. ChebyNet is a spectral-based method that uses graph Laplacian matrix to average neighbor features into a node. It uses Chebyshev polynomial to avoid eigenvalue decomposition for the graph Laplacian. The order of the Chebyshev polynomial corresponds to the number of neighbor hops being incorporated. Specifically, the graph convolution operator in ChebyNet can be expressed as $\mathbf{X}' = \sigma(\sum g_\theta(\mathbf{L})\mathbf{X} + \mathbf{b})$, where $\mathbf{X}'$ and $\mathbf{X}$ are node feature matrices before and after the graph convolution, $g_\theta(\mathbf{L}) = \sum_0^{K-1} \theta_k \mathbf{T}_k(\hat{\mathbf{L}})$, $\mathbf{T}_k(\hat{\mathbf{L}})$ is the ChebyShev polynomial of order k, evaluated at the scaled Laplacian $\hat{\mathbf{L}}$ with θ_k as kernel weights [36]. The diffusion graph convolution uses the indegree and outdegree normalized adjacency matrix to aggregate neighbor node features. Specifically, $\mathbf{X}' = \sigma(\sum_0^{K-1}(\theta_{k,1}(\mathbf{D}_O^{-1}\mathbf{W})^k + \theta_{k,2}(\mathbf{D}_I^{-1}\mathbf{W}^T)^k)\mathbf{X} + \mathbf{b})$, where $\mathbf{D}_I$ and $\mathbf{D}_O$ are diagonal indegree matrix and outdegree matrix, $\mathbf{W}$ is the node adjacency matrix, $\theta_{k,1}$, $\theta_{k,2}$ and $\mathbf{b}$ are kernel weights and bias parameters [37]. Multiplying the degree normalized adjacency matrix multiple times is equivalent to conducting the diffusion multiple hops. We can use multiple filters together to output multiple feature channels. Diffusion graph convolution defines convolutional operations on a directed graph based on two different node adjacency matrices on both directions.

3.4 Downsample and upsample paths based on contour tree hierarchy The purpose of the downsampling path is to extract semantic features of locations on the surface topology at a coarser resolution so that the extracted feature at each location has a global topological context. The task is non-trivial since the traditional max-pooling operation on 2D image data cannot be directly applied to a contour tree due to the lack of a regular grid structure. There exist some research on designing pooling operators on graphs. For example, one method is based on greedy node clustering (also called Graclus [38]), which iteratively merges two nodes with a high edge weight and low degrees. Another method called DIFFPOOL learns a cluster assignment matrix over graph nodes on top of the output of a GNN model [39]. Another work uses a trainable projection vector to perform k-max-pooling (i.e., selecting nodes based on the projected values from node features) [40]. However, the existing pooling operations can not preserve the topological structures of the graph. To fill the gap, we propose to do pooling operation based on a **multi-scale contour tree hierarchy**. The main intuition is to look at the surface height function at different precision levels. When the precision is high, more local fluctuations of the surface show up and the contour tree structure provides more local topological details. When the precision is low, the surface will look smoother and the contour tree structure focuses more on global

topological trends.

Moreover, as the surface height precision keeps decreasing, previously separate contours (tree nodes) will merge together into one, creating a collapsing hierarchy. The process can be considered as a bottom-up hierarchical clustering of contour tree nodes. Figure 1 provides an example, where the original contour tree is downsampled twice. The original contour tree is based on elevation values rounded to integers, the second contour tree has elevation rounds to even integers, and the third contour tree is based on elevation values rounded to multiples of fours. The one to many mapping relationships between nodes on a coarse-scale contour tree to those on a fine-scale contour tree can be expressed in a pooling matrix $\mathbf{T}_l$, where $\mathbf{T}_{ij} = 1$ if node i at the level l is collapsed into node j in level $l+1$, and $\mathbf{T}_{ij} = 0$ otherwise. Based on the pooling matrix, we can design an average-pooling layer on contour tree node features, as shown by $\mathbf{X}_{l+1} = \tilde{\mathbf{T}}_l \mathbf{X}_l$, where $\tilde{\mathbf{T}}_l$ is the row-normalized pooling matrix from level l to level $l+1$, and $\mathbf{X}_l$ and $\mathbf{X}_{l+1}$ are node feature matrices at level l and level $l+1$ respectively. This can be easily implemented based on sparse matrix multiplication.

The purpose of the upsampling path is to concatenate global and local topological features and to project those features to a surface at a fine scale. The traditional transposed convolution cannot be applied due to two reasons. First, there is no regular window (neighborhood) structure on the contour tree as in the traditional 2D image. Second, the mapping of pixel locations from a coarser scale to a fine-scale is not as regular in a contour tree as in a traditional 2D image. Thus, we cannot easily design a pooling operation with learnable parameters by transposed weight matrix multiplication as in U-Net. To fill the gap, we propose to use simple unpooling based on the same one to many mapping between nodes in the hierarchical clustering. Specifically, we can upsample node features from a coarse contour tree to a fine-scale contour tree based on the transpose of the pooling matrix $\mathbf{T}_l^T$. The unpooling operation can be expressed as $\tilde{\mathbf{X}}_l = \mathbf{T}_l^T \tilde{\mathbf{X}}_{l+1}$, where $\mathbf{T}_l^T$ is the transpose of the pooling matrix from level l to level $l+1$, and $\tilde{\mathbf{X}}_l$ is the upsampled feature matrix at level l from the level $l+1$.

4 Evaluation

The goal of the evaluation is to compare our proposed method with several baseline methods in classification performance on two real-world hydrological datasets. We also conducted self-comparison studies to evaluate the effect of different configurations in our model. Experiments were conducted on a workstation with four NVIDIA RTX 6000 GPUs. Unless specified otherwise, we used default parameters in open source tools for baseline methods. Candidate classification methods are listed below. The source codes, data samples, and more implementation details are provided in the supplementary materials.

- **Per-pixel classifiers**: We used random forest (**RF**), maximum likelihood classifier (**MLC**), and gradient boosted tree (**GBM**) from R randomForest and gbm packages.
- **Fully convolutional network (FCN):** We used U-Net [17] in Python. The model consists of ten double convolution layers with batch normalization, together with max-pooling layers and transposed convolution layers, for the downsample path and the upsample path. We used the Adam optimizer with a learning rate of 10^{-4} and a mini-batch size of 10.
- **Contour tree neural network (CTNN):** This is our proposed method. We implemented our codes in Python and Tensorflow. We used U-Net to extract pixel features by removing the last non-linear and softmax layers and aggregated pixel features as inputs into tree nodes. We used the Adam optimizer with a learning rate of 10^{-4} and a mini-batch size of 1.

Dataset description: We used two flood mapping datasets from the cities of Grimesland and Kinston in North Carolina during Hurricane Mathew in 2016. Explanatory features were red, green, blue bands in aerial imagery from NOAA National Geodetic Survey [41]. There are two classes: flood and dry. The digital elevation maps were from the University of North Carolina Libraries [42]. All data were resampled into 2-meter by 2-meter resolution. For the Grimland dataset, there are 80 training images, 120 validation images, and 16 test images. Each image size is 672 by 672 pixels, which are collapsed into 451584 contour tree nodes. For the Kinston dataset, there are 132 training images, 113 validation images, and 113 test images. Each image size is 3000 by 100, totally 3×10^5 nodes in the contour tree.

Evaluation Metric: For classification performance evaluation, we used precision, recall, and F-score. We also used overall accuracy given that the two classes are relatively balanced.

4.1 Classification Performance Comparison

We first compared different methods on their precision, recall, and F-score on the two real-world datasets. The results were summarized in Table 1 and Table 2 respectively. In this experiment, we used the default model architecture in CNTT with a four-level contour tree hierarchy based on different precision of elevation values at 0.01 meter,

Table 1: Classification on Real Dataset in Grimesland NC

Classifiers	Class	Precision	Recall	F	Avg. F	Accuracy
RF	Dry	0.68	0.75	0.71	0.73	0.73
	Flood	0.79	0.72	0.75		
MLC	Dry	0.69	0.67	0.68	0.72	0.73
	Flood	0.75	0.77	0.76		
GBM	Dry	0.68	0.75	0.71	0.73	0.73
	Flood	079.	0.72	0.75		
U-Net	Dry	0.99	0.71	0.82	0.85	0.86
	Flood	0.81	0.99	0.88		
CTNN	Dry	0.98	0.82	0.89	**0.90**	**0.90**
	Flood	0.84	0.99	0.91		

Table 2: Classification on Real Dataset in Kinston NC

Classifiers	Class	Precision	Recall	F	Avg. F	Accuracy
RF	Dry	0.82	0.73	0.77	0.74	0.74
	Flood	0.65	0.76	0.70		
MLC	Dry	0.87	0.63	0.73	0.72	0.72
	Flood	0.61	0.86	0.71		
GBM	Dry	0.82	0.73	0.77	0.74	0.74
	Flood	0.65	0.76	0.70		
U-Net	Dry	0.93	0.97	0.95	0.94	0.94
	Flood	0.96	0.92	0.94		
CTNN	Dry	0.95	0.98	0.96	**0.96**	**0.96**
	Flood	0.96	0.94	0.95		

0.1 meters, and 1 meter respectively. We used ChebyNet graph convolution operators at each level with the number of neighbor hops as 4 and 2 in each level, and the number of graph convolution operators as 16, 32, 64, and 128 at each level respectively. On the Grimesland dataset, random forest, maximum likelihood classifier, and gradient boosted tree achieved overall F-score around 0.73. The poor performance of per-pixel classifiers was due to class confusion from feature noise and obstacles. For example, pixels of tree canopies overlaying flood water have the same spectral features with those in dry areas. Without incorporating spatial dependency structure between pixel locations, these per-pixel classifiers cannot overcome the class confusion. U-Net performed much better with an average F-score of 0.85 due to its capability of learning complex spatial contextual features from input aerial imagery (e.g., color and texture of floodwater). Our CTNN model performed the best with an overall F-score of 0.90 (higher than the F-score of 0.85 in U-Net only). The reason was that the CTNN model can not only utilize the complex spatial contextual features extracted from U-Net but also incorporate topological structural dependency among class labels in the contour tree hierarchy. In this dataset, additional topological constraints made a significant difference because feature obstacles obscured some flood boundaries in the aerial imagery, making it difficult to delineate those boundaries in U-Net only.

Results on the Kinston dataset were summarized in Table 2. Similar to the Grimesland dataset, we can observe that most non-spatial classifiers performed poorly with an overall F-score below 0.74. Both U-Net and CTNN performed very well in this data because the flood boundaries in this dataset were distinguishable in comparison to the Grimesland dataset. Thus, U-Net could successfully identify most flood boundaries with high accuracy. Our CTNN model performed slightly better than U-Net only, indicating that learning additional topological structure along surface contours still added some value in this case.

4.2 Self-comparison on model configurations

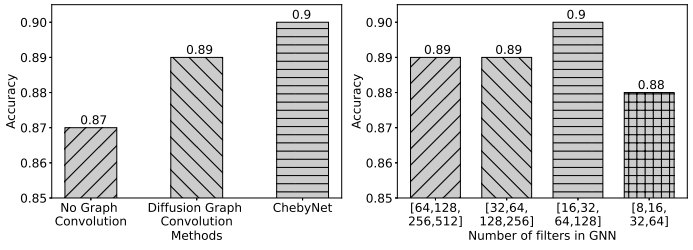
We also conducted a sensitivity analysis to analyze the effect of different configurations in our CTNN model,

such as the type of graph convolutional filters, the number of filters at each contour tree level, the number of neighbor hops in each graph convolution filter, and the number of contour tree hierarchical levels. Due to limited space, we only focused on the Grimesland dataset. Unless specified otherwise, we used the same configurations for our CTNN model as in Section 4.1. Note that the metrics were directly reported on test data for sensitivity analysis. This is different from the results in Section 4.1, in which we used independent validation data to select hyper-parameters.

First, we studied the effect of the type of graph convolutional filters in our CTNN model. We fixed the other configurations of the model and compared three different choices: no graph convolution (i.e., only use a non-linear function to map input channels to output channels for each node, or 1D convolution), diffusion graph convolution, and ChebyNet. The results were summarized in Figure 3(a). We can see that without graph convolution (i.e., only using 1D convolution on each node), the overall accuracy is 0.87. This configuration was equivalent to learning additional multi-layer perceptrons on the features of each node extracted from U-Net. Thus, the performance slightly improved over U-Net only. Adding diffusion convolution improved the overall accuracy from 0.87 to 0.89, indicating that the graph convolution filters along the contour tree structure helped better delineate flood boundaries. The accuracy of a (spectral based) ChebyNet graph convolutional filter was the highest. It was interesting that the undirected graph convolution in ChebyNet performed slightly better than the directed diffusion graph convolution on our contour tree (which is a directed graph). The reason was probably that most neighboring nodes should have the same classes due to spatial autocorrelation except for nodes near the flood boundary (the flood extent is contiguous).

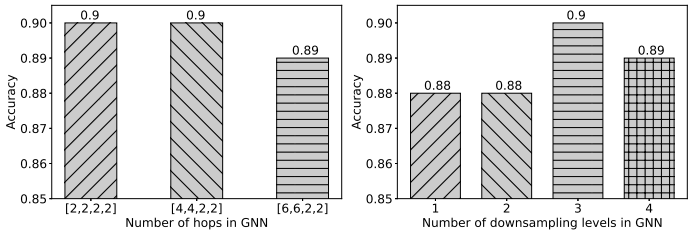
We also tested the effect of the number of neighbor hops in the ChebyNet graph convolutional operations. We used the default setting and varied the number of neighbor hops in three different settings. We first fixed the number of neighbor hops as 2 in all four downsample levels, and then increased the neighbor hops of the first two downsample levels to 4 and 6 respectively. The comparison was summarized in Figure 4(a). We can see that using a larger number of neighbor hops degraded the performance. This was due to the fact that a larger number of neighbor hops had an over-smoothing effect between the adjacent contours near the flood boundary.

Finally, we also tested the effect of the number of levels in the contour tree hierarchy. We used the default setting for the other hyper-parameters and varied the number of downsampling operations from 1 to 4.



(a) (b)

Figure 3: The effect of graph convolutional filter types and the number of filters



(a) (b)

Figure 4: The effect of the number of neighbor hops and downsample levels in GNN

Specifically, when using four downsample operations (five different contour tree scales), we keep reducing the precision of elevation values four times from 0.01 to 0.1 meters, 1 meter, and 10 meters respectively. We found out that downsampling the contour tree 3 times (into four levels) had the best overall accuracy.

5 Discussion on Significance and Impact

This work has a significant potential societal impact in the flood mapping application. Flood extent mapping plays a crucial role in addressing grand societal challenges such as disaster management, national water forecasting, as well as energy and food security. For example, during Hurricane Harvey floods in 2017, first responders needed to know where flood water was in order to plan rescue efforts. In national water forecasting, detailed flood extent maps can be used to calibrate and validate the NOAA National Water Model [43], which can forecast the flow of over 2.7 million rivers and streams through the entire continental U.S. [44]. In current practice, flood extent maps are mostly generated by flood forecasting models, whose accuracy is often unsatisfactory in high spatial details [44]. Other ways to generate flood maps involve sending a field crew on the ground to record high-water marks, or visually interpreting earth observation imagery [45]. However, the process is both expensive and time-consuming. With a large amount of high-

resolution earth imagery being collected from satellites (e.g., DigitalGlobe, Planet Labs), aerial planes (e.g., NOAA National Geodetic Survey), and unmanned aerial vehicles, the cost of manually labeling flood extent becomes prohibitive. This paper develops a classification model that can automatically classify earth observation imagery pixels into flood extent maps. The results can be used by first responders to plan rescue efforts, by hydrologists to calibrate and validate water forecasting models, as well as by insurance companies to process claims. The proposed model advances the state of the art methods by modeling the surface contour structures on geographic terrains.

6 Conclusion and Future Work

In this paper, we propose a contour tree neural network for 3D surface segmentation in the context of hydrological applications. The model represents a 3D elevation surface by a contour tree skeleton from computational topology. Graph convolution and pooling operations are used in a multi-level hierarchy to learn node features at multiple surface scales. The unique advantage of the model compared with other existing models is that it can capture the topological structure of class segments along with surface contour patterns. Evaluations on real-world datasets show that the proposed model outperformed several baseline methods in classification accuracy.

One limitation in our current model is that the graph convolution filters assume specific contour tree structures, which often vary from one surface to another. Another limitation is that our model still relies on pre-extracted spatial contextual features from U-Net for each tree node due to the relatively small parameters in a graph neural network. We plan to address these issues in future work. For example, we can potentially study how to partition a geographic area along a river into pieces such that their contour tree structure is generally similar.

Acknowledgement

This material is based upon work supported by the NSF under Grant No. IIS-1850546, IIS-2008973, CNS-1951974, the National Oceanic and Atmospheric Administration (NOAA), and the University Corporation for Atmospheric Research (UCAR).

References

[1] Zhe Jiang. A survey on spatial prediction methods. *IEEE Transactions on Knowledge and Data Engineering*, 31(9):1645–1664, 2018.

[2] Shashi Shekhar, Zhe Jiang, Reem Y Ali, Emre Eftelioglu, Xun Tang, Venkata Gunturi, and Xun Zhou. Spatiotemporal data mining: a computational perspective. *ISPRS International Journal of Geo-Information*, 4(4):2306–2338, 2015.

[3] Zhe Jiang and Shashi Shekhar. Spatial big data science. *Schweiz: Springer International Publishing AG*, 2017.

[4] Zhe Jiang. Spatial structured prediction models: Applications, challenges, and techniques. *IEEE Access*, 8:38714–38727, 2020.

[5] Vijay Natarajan, Yusu Wang, Peer-Timo Bremer, Valerio Pascucci, and Bernd Hamann. Segmenting molecular surfaces. *Computer Aided Geometric Design*, 23(6):495–509, 2006.

[6] Miao Xie, Zhe Jiang, and Arpan Man Sainju. Geographical hidden markov tree for flood extent mapping. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2545–2554, 2018.

[7] Zhe Jiang, Miao Xie, and Arpan Man Sainju. Geographical hidden markov tree. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[8] Zhe Jiang and Arpan Man Sainju. Hidden markov contour tree: A spatial structured model for hydrological applications. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 804–813, 2019.

[9] Arpan Man Sainju, Wenchong He, and Zhe Jiang. A hidden markov contour tree model for spatial structured prediction. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[10] Wenchong He and Zhe Jiang. Semi-supervised learning with the em algorithm: A comparative study between unstructured and structured prediction. *arXiv preprint arXiv:2008.12442*, 2020.

[11] Ariel Shamir. A survey on mesh segmentation techniques. In *Computer graphics forum*, volume 27, pages 1539–1556. Wiley Online Library, 2008.

[12] Alan D Kalvin and Russell H Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, 16(3):64–77, 1996.

[13] Vicente Grau, AUJ Mewes, M Alcaniz, Ron Kikinis, and Simon K Warfield. Improved watershed transform for medical image segmentation using prior information. *IEEE transactions on medical imaging*, 23(4):447–458, 2004.

[14] Michael Garland, Andrew Willmott, and Paul S Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 49–58, 2001.

[15] Xinjian Chen, Meindert Niemeijer, Li Zhang, Kyungmoo Lee, Michael D Abràmoff, and Milan Sonka. Three-dimensional segmentation of fluid-associated abnormalities in retinal oct: probability constrained graph-search-graph-cut. *IEEE transactions on medical imaging*, 31(8):1521–1531, 2012.

[16] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey.

arXiv preprint arXiv:2001.05566, 2020.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[19] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[20] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[21] Jinghua Wang, Zhenhua Wang, Dacheng Tao, Simon See, and Gang Wang. Learning common and specific features for rgb-d semantic segmentation with deconvolutional networks. In *European Conference on Computer Vision*, pages 664–679. Springer, 2016.

[22] Lingni Ma, Jörg Stückler, Christian Kerl, and Daniel Cremers. Multi-view deep learning for consistent semantic mapping with rgb-d cameras. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 598–605. IEEE, 2017.

[23] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer, 2016.

[24] Weiyue Wang and Ulrich Neumann. Depth-aware cnn for rgb-d segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.

[25] Chao Chen, Daniel Freedman, and Christoph H Lampert. Enforcing topological constraints in random field image segmentation. In *CVPR 2011*, pages 2089–2096. IEEE, 2011.

[26] Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In *Advances in Neural Information Processing Systems*, pages 5658–5669, 2019.

[27] Agata Mosinska, Pablo Marquez-Neila, Mateusz Koziński, and Pascal Fua. Beyond the pixel-wise loss for topology-aware delineation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3136–3145, 2018.

[28] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[29] James R Munkres. *Topology*. Pearson, January 2000.

[30] Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003.

[31] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[32] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202*, 2018.

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[34] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, 2019.

[35] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1416–1424, 2018.

[36] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[37] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[38] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11):1944–1957, 2007.

[39] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pages 4800–4810, 2018.

[40] Hongyang Gao and Shuiwang Ji. Graph u-nets. *arXiv preprint arXiv:1905.05178*, 2019.

[41] National Oceanic and Atmospheric Administration. Data and imagery from noaa's national geodetic survey. `https://www.ngs.noaa.gov`.

[42] NCSU Libraries. LIDAR Based Elevation Data for North Carolina. `https://www.lib.ncsu.edu/gis/elevation`, 2018.

[43] National Oceanic and Atmospheric Administration. National Water Model: Improving NOAA's Water Prediction Services. `http://water.noaa.gov/documents/wrn-national-water-model.pdf`, 2018.

[44] Don Cline. Integrated water resources science and services: an integrated and adaptive roadmap for operational implementation. Technical report, National Oceanic and Atmospheric Administration, 2009.

[45] PA Brivio, R Colombo, M Maggi, and R Tomasoni. Integration of remote sensing data and gis for accurate mapping of flooded areas. *International Journal of Remote Sensing*, 23(3):429–441, 2002.