GE, QIAN. Robust Image Segmentation: Applications to Autonomous Car and Foraminifera Morphology Identification. (Under the direction of Dr. Edgar Lobaton.)

Image segmentation, as one of the most crucial low-level operations in computer vision, is often used as a pre-processing step in various high-level tasks, such as semantic segmentation, object detection and recognition. State-of-the-art segmentation algorithms are able to produce accurate segmentation results for some specific dataset. However, due to variations between images, it is less likely that a single algorithm with the same parameter setting is able to segment all the images successfully. In this dissertation, we propose several image segmentation algorithms robust to parameter selection, changes in image conditions and qualities using tools including persistent homology and deep neural networks.

First, we propose a new approach to capture the consensus information from a set of segmentation results. The probability of a segmentation curve being present is estimated based on our probabilistic image segmentation model. The persistent segments are extracted by applying topological persistence to the probability map. A robust segmentation is obtained with the detection of certain segmentation curves guaranteed. The experiments demonstrate our algorithm is able to consistently capture the curves present within the segmentation set.

Then, a new approach to robust obstacle segmentation from stereo disparity maps of outdoor scenes is proposed. An occupancy map is first computed from the disparity map for segmentation. Traditionally, a simple threshold is applied to segment obstacles from the occupancy map. However, this outcome is sensitive to the choice of parameter value. In our proposed method, instead of simple thresholding, we perform a topological persistence analysis on the constructed occupancy map, which leads to a more robust segmentation.

To further improve the robustness of the obstacle segmentation, we propose an approach to compute multi-valued disparity maps from stereo image pairs, because the single-valued disparity map often suffers heavily from reflections, lack of texture and repetitive patterns of objects. The candidate disparity values for each pixel location are selected based on the matching cost function of the semi-global block matching algorithm. Then an aggregate occupancy map is computed from the multi-valued disparity map for obstacle segmentation. Experiments show that our approach can recover the correct structure of obstacles on the scene when traditional estimation approaches based on single-valued disparity maps fail.

Finally, we study edge-based image segmentation algorithms on a microscopic fossil dataset collected by ourselves. First, a coarse-to-fine edge detection strategy is proposed to obtain clear edges from low quality images with vague edges. In order to produce more accurate segmentation results, then we propose a topology-aware edge detection approach based on a novel topology-aware evaluation metric to train an edge detection model to focus more on preserving topological structures such as closing gaps on edges.
DEDICATION

To my parents, Huixiang Ge and Jie Bai.
BIOGRAPHY

Qian Ge received the B.S. and M.S. degree in electrical engineering from University of Electronic Science and Technology of China, Chengdu, China. Then she joined the Ph.D. program of the Department of Electrical and Computer Engineering, NC State University, Raleigh, NC, USA, under Dr. Edgar Lobaton. Her research interests include robust image segmentation using deep neural networks and topological data analysis.
ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Edgar Lobaton, who provided me the great opportunity to join the ARoS lab and constant support during my PhD. Thank him for sharing his expertise experience in research with me and giving me the freedom to pursue various research directions, and I am also very grateful for his patience in helping me improve my communication and writing skills. I would also like to thank my committee members Dr. Wesley Snyder, Dr. Hamid Krim and Dr. Radmila Sazdanovic for their invaluable feedback for my work and dissertation. Thanks for serving in my committee board.

I would like to thank our collaborators Dr. Paul Fyfe at Department of English, Dr. Thomas M. Marchitto at University of Colorado Boulder, and Dr. Ritayan Mitra at IIT Bombay for their collaborative work and helpful inputs. I would also like to thank NSF (CNS-1239323) and NSF (OCE-1637039) for funding my research.

I would like to thank my parents for their endless love and encouragement. Without their continued support, I would never have had chance to start and finish my PhD journey. I still remember the moment that I decided to pursue a PhD with the encouragement from my father.

I would like to thank all the members of ARoS lab, Dr. Alireza Dirafzoon, Dr. Namita Lokare (for her help with experiments), Jeremy Cole (for his helpful feedback for my writing), Somrita Chattopadhyay, Chunpeng Wei, Boxuan Zhong (for his valuable suggestions and encouragement), Turner Richmond, Laura Gonzalez, Nathan Starliper and Rafael L. da Silva, for their collaborative work and helpful discussions. I would also like to thank all friends I made over the years who made my stay in NC State meaningful and memorable.

Finally I would like to thank my best friends in high school, Jia Luo, Dan Li, Xiaomo Wan and Lingqing Xing, who provided me selfless friendship, continued support and encouragement; Shanshan Qu in UESTC, who gave me the necessary distractions from preparing the GRE exam by inviting me to watch funny Japanese dramas every week; Xin Chen in NC State, who is the first friend I made in the US, and provided me invaluable help and suggestions making my life easier here. Special thanks to my best friend Feier Lian for her tolerating me and listening to all my nonsense.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
</tbody>
</table>

## Chapter 1  Introduction

1. Contributions ........................................ 3
1.2 Dissertation Outline .................................. 6

## Chapter 2  Background

2.1 Topological Data Analysis ..................................... 7
  2.1.1 Simplicial Complexes .................................. 7
  2.1.2 Persistent Homology ................................... 8
2.2 Stereo Vision ........................................... 11
  2.2.1 Data Collection ....................................... 12
  2.2.2 Camera Calibration .................................... 13
  2.2.3 Disparity Map Computation ............................ 15
2.3 Deep Neural Networks ...................................... 16
  2.3.1 Neural Networks ....................................... 16
  2.3.2 Fully Convolutional Networks .......................... 20
  2.3.3 FCN based Dense Prediction ............................ 21
2.4 Related Works ............................................ 21
  2.4.1 Consensus Image Segmentation .......................... 22
  2.4.2 Outdoor Scene Image Segmentation ...................... 22
  2.4.3 Edge Detection ........................................ 24
  2.4.4 Class Imbalance in Dense Predictions .................. 24

## Chapter 3  Consensus-Based Image Segmentation via Topological Persistence

3.1 Image Segmentation Model ................................... 27
3.2 Probability Map Construction ............................... 29
  3.2.1 Boundary Characterization .............................. 29
  3.2.2 Choice of Parameter $n$ ................................. 30
3.3 Extract Persistence Segmentation ........................... 32
  3.3.1 Disconnection Threshold ............................... 33
  3.3.2 Persistence Diagram .................................... 33
  3.3.3 Persistence Region Extraction ........................ 34
  3.3.4 Edge Position Estimation using Region Growing ....... 35
3.4 Experiment ................................................ 37
  3.4.1 Generation of Segmentation Set ....................... 39
  3.4.2 Results ................................................ 39
3.5 Conclusion ............................................... 41

## Chapter 4  Robust Obstacle Segmentation based on Topological Persistence in Outdoor Traffic Scenes

4.1 Probabilistic Occupancy Grid Computation .................. 45
  4.1.1 UV-Disparity Map ....................................... 45
  4.1.2 Ground Fitting ......................................... 46
### LIST OF TABLES

| Table 3.1 | Performance evaluation of the proposed method (Consensus-based) against input methods over 20 images from the Berkeley Segmentation Database | 41 |
| Table 6.1 | Edge Detection Results using Different Features and Classifiers | 82 |
| Table 7.1 | Methods for comparison. The first three methods are baseline methods and the last two are variants of our methods. | 98 |
| Table 7.2 | Evaluation of the methods used for refinement. *Correct*, *Complete*, and *LongShort* denote the evaluation metrics *Correctness*, *Completeness* and *2Long2Short* mentioned in section 7.2.4. | 101 |
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Applications of image segmentation. Images are from [Arb11; Der16; Zha16; BR11; Gup15; Wu12].</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Framework of robust image segmentation based on topological persistence applied to natural images and depth images.</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>Pipeline of robust obstacle segmentation based on multi-valued disparity maps.</td>
<td>5</td>
</tr>
<tr>
<td>1.4</td>
<td>Overview of the topology-aware edge detection.</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Persistent homology. (a) Two spaces $M_1$ and $M_2$ (images of letter D and B) with sample point cloud. (b) Sample point cloud $X_1$ and $X_2$ corresponding to letter D and B. (c) Corresponding persistence diagram of connected components (red points) and holes (blue points). (d) Examples of simplex. (e) Filtration on point cloud $X_1$.</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Stability of persistence diagram [CS07]. Left: two functions closed to each other. Right: the persistence diagram of both functions and the bijection between them.</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Persistence Analysis of 2D probability image. (a) Original grayscale image, (b) images after thresholding, and (c) persistence diagram. Each point in the diagram corresponds to clusters that are born and die at specific threshold values. Points near the diagonal are sensitive to small variations of the original image.</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td>Stereo vision with two cameras.</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>Stereo setup in our lab. (a) Camera used for data capture. (b) The stereo setup mounted on a car. (c) Stereo setup with two cameras and a trigger circuit.</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Neural network layers. Blue color indicates the input and white color indicates the output. (a) Fully connected layer. (b) Convolutional layer. (c) Transposed convolutional layer. (d) Max pooling. (e) Global average pooling.</td>
<td>17</td>
</tr>
<tr>
<td>2.7</td>
<td>Dilated 2D kernels with $l = 1, 2, 3$.</td>
<td>20</td>
</tr>
<tr>
<td>2.8</td>
<td>Encoder-decoder architecture for dense prediction.</td>
<td>21</td>
</tr>
<tr>
<td>3.1</td>
<td>Pipeline of the consensus-based image segmentation. (a) Original image. (b) Four samples from the segmentation set. (c) Probability map. (d) A set of upper level sets of the probability map. (e) Persistence diagram of the filtration shown in (d) where each point in the diagram corresponds to a connected component in the filtration. (f) Segmentation result.</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Illustration of the model used for segmentations: The representation model for a segmentation $\mathcal{S}$ (left) and the bounded perturbation model around $\mathcal{S}_\sigma$.</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Example of segment curves captured by patches. (a) An image with a soft edge and a sharp edge. (b)-(d) show three possible segmentation results. Red lines are the detected segment curves. Brown, blue and green squares are the patches used for probability map construction.</td>
<td>30</td>
</tr>
<tr>
<td>3.4</td>
<td>Connection probability map of different patch size $n = 3, 5, 7$ and 9. Darker color indicates a higher probability of edge. (a) $n = 3$. (b) $n = 5$. (c) $n = 7$. (d) $n = 9$.</td>
<td>31</td>
</tr>
</tbody>
</table>
Figure 3.5 Illustration of the effect of the choice of $n$. Vertical dash lines with the same color indicate the variation regions of one segment curve $\gamma$. (a) The case of $n - 2 \geq 2\delta$. The variation region can be covered by one patch. (b) The case of $n - 2 < 2\delta$. The variation region can be covered by four patches in this example. (c) The case of closed segment curves influence the approximation.

Figure 3.6 Filtration at $\sigma = 0, 0.25, 0.5, 0.75$ and 1 of each connection probability map shown in Fig. 3.4. The regions with $f(x) < 1 - \sigma$ are labeled using black color.

Figure 3.7 Persistence diagrams using patch size $n = 3, 5, 7$ and 9 (from top to bottom) and corresponding extracted persistence regions. The areas for segment curves being present with probability greater than 0.8 within a range $2\delta \leq n - 2$ are labeled with black color in column (a) and (b). (a) Labeling image. Different colors indicate different regions. (b) Labeling image labeled with the mean color of each region. (c) Persistence diagrams. Read dash lines parallel to the diagonal are the persistence thresholds $\tau$. From top to bottom: $\tau = 0.25, 0.40, 0.45$ and 0.55. The horizontal red dash lines are $\sigma = 0.8$ for all of the four diagrams.

Figure 3.8 Segmentation results after region growing using patch size $n = 3, 5, 7$ and 9 (from top to bottom). (a) Labeling image. Different colors indicate different regions. (b) Labeling image labeled with the mean color of each region. (c) Segmentation with edges.

Figure 3.9 (a)-(d) Results with best coverage score for base segmentation approaches in the input segmentation set: SAS [Li12], Normalized Cuts [SM00], Graph-based segmentation [FH04] and Mean Shift [CM02]. (e) Original image. (f) Connection probability map. (g) Persistence diagram of the connected components during filtration. The thresholds are $\sigma_{max} = 0.8$ and $\tau = 0.4$. (h) Our segmentation result. (i)-(l) Filtration with $\sigma = 0, 0.25, 0.5, 0.8$. With $\sigma$ increasing, the regions separated by segment curves with low probability are merged early.

Figure 3.10 Best segmentation results of the four base-algorithms in the segmentation set and result of proposed algorithm. Left to right: Original image, SAS [Li12], Normalized Cuts [SM00], Graph-based segmentation [FH04], Mean Shift [CM02], connection probability map, and proposed algorithm.

Figure 4.1 Proposed robust obstacle detection. (a) Left image of stereo pair image. (b) Disparity map. (c) Disparity map after ground segmentation. (d) Occupancy grid. (e) Persistence diagram according to the occupancy grid. (f) Robust segmentation in U-disparity space. (g) Robust segmentation results.

Figure 4.2 Disparity maps. (a) Disparity map in image space. (b) U-disparity map. (c) V-disparity map.

Figure 4.3 Occupancy grid. (a) Grayscale image. (b) Modified occupancy grid.

Figure 4.4 Birth and death of connected components during filtration. The first row shows a part of the original image and the corresponding probability of occupancy map. Second row shows a connected component in cyan which corresponds to a car in the image space for five different values of $\tau$, and third row shows the corresponding regions in image space.

Figure 4.5 Persistence diagram corresponding to Fig. 4.4. Each point indicates the lifespan of a connected component. The red line is the threshold line $\gamma = 0.5$. All the regions above this line have persistence interval greater than 0.5.
Figure 4.6  Segmentation results. (a) Clusters in u-disparity space, and (b) their corresponding regions in image space. Different colors indicate different obstacles. Ground is labeled as green.

Figure 4.7  Occupancy grid comparison. (a) Grayscale image. (b) Occupancy grid in [Per12]. (c) Modified occupancy grid. (d) Region of original occupancy map for vehicle close to the camera. (e) Region of modified occupancy map for vehicle close to the camera.

Figure 4.8  Segmentation results of both occupancy maps. (a)(b) Segmentation using original occupancy map. (c)(d) Segmentation using modified occupancy map. (a) Segmentation result using persistence threshold one step before part of the vehicle merges with background. (b) Right part of the vehicle merges with background before merges with other parts of the vehicle. (c) Segmentation result using persistence threshold one step before all parts of vehicle merge together. (d) All parts of vehicle merge together before merge with background.

Figure 4.9  Persistence diagrams of both types of occupancy maps. Red points are the detected obstacles in image space. (a) Persistence diagram computed from occupancy maps in [Per12]. (b) Persistence diagram computed from modified occupancy maps.

Figure 4.10  Segmentation results for persistence bound $\gamma$ ranging from 0.3 to 0.6. Trees on the left side of the image are separated when $\gamma = 0.3$ to 0.5. They are merged when $\gamma = 0.6$. Note that two vehicles are always detected properly for all persistence thresholds.

Figure 4.11  Comparison between thresholding and persistence methods of different parameters. Left column: Results of thresholding approach for $\tau = 0.25$, 0.30, 0.35, 0.40 and 0.45. Right column: Results of persistence approach for $\gamma = 0.35$, 0.40, 0.45, 0.50 and 0.55. The ground labeling is removed for clear illustration.

Figure 4.12  Comparison between thresholding and persistence methods of different frames. Left column: Results of thresholding approach for four successive frames with $\tau = 0.35$. Right column: Results of persistence approach for four successive frames with $\gamma = 0.45$. The ground labeling is removed for clear illustration.

Figure 4.13  Segmentation results with $\gamma = 0.45$ and corresponding persistence diagrams. Ground is labeled as green. Red points in persistence diagram are the visible segments in image space.

Figure 5.1  (a) Environment model. Different colors denote different depth of obstacles detected. Only obstacles above the ground plane and below a maximum-height plane are detected. (b) Corresponding obstacle detection in u-disparity space. Images used for analysis are from the KITTI dataset [Gei12; Gei13; Fri13].

Figure 5.2  Overview of proposed approach.

Figure 5.3  Example of wrong disparity estimation by SGBM due to the repetitive structure presenting in the fence. Disparity map shows the wrong disparity estimation in some regions of the fence (top). Right image (bottom) shows the estimated correspondence (red triangle) and ground truth (green square) for the test point in left image (middle).
Figure 5.4 Cost function $C_p$ of the test point highlighted in Fig. 5.3. The estimation is near the global minimum, but the ground truth is near a local minimum. Notice that each local minimum corresponds to a bar of the fence.

Figure 5.5 Threshold choice for disparity candidate selection. (a) $\epsilon^{(0.95)}$ as a function of parameter $\tau$. (b) Average number of candidates as a function of parameter $\tau$. A choice of $\tau = 0.4$ guarantees that $\epsilon^{(0.95)} < 5$ and that the number of candidates will be below 5 as well. A total of 120 training stereo pairs were used for selecting this threshold.

Figure 5.6 (a) $\epsilon^{(0.95)}(0.4)$ of each stereo pair in test set. (b) Average number of candidates per pixel of each stereo pair in test set. For most of pairs, $\epsilon^{(0.95)}(0.4) < 5$ and average number of candidates per pixel is less than 6.

Figure 5.7 (a) Left image of the stereo pair. (b) Disparity map from OpenCV-SGBM. (c) Aggregate occupancy map from multi-valued disparity map. (d) Occupancy map from OpenCV-SGBM disparity map. (e) Obstacle detection using aggregate occupancy map in $u$-disparity domain. (f) Obstacle detection using OpenCV-SGBM occupancy map in $u$-disparity domain. (g) Obstacle detection using aggregate occupancy map in image domain. (h) Obstacle detection using OpenCV-SGBM occupancy map in image domain. In this case, both approaches give good results. (c) and (d) show similar occupancy grid maps, since pixels with good disparity estimation have a small number of candidates.

Figure 5.8 (a)-(h) are the same as Fig. 5.7. In this case, the disparity map in (b) is wrong in some regions of ground due to the brightness. The occupancy map computed using this disparity map is noisy in the corresponding regions highlighted by the black rectangle shown in (d). This leads to the wrong obstacle detection on the ground shown in (h). (g) shows our approach is able to avoid those false negatives.

Figure 5.9 (a)-(h) are the same as Fig. 5.7. In this case, the disparity map in (b) is wrong in some regions of ground due to the brightness. The occupancy map computed using this disparity map is noisy in the corresponding regions highlighted by the black rectangle shown in (d). This leads to the wrong obstacle detection on the ground shown in (h). (g) shows our approach is able to avoid those false negatives.

Figure 5.10 (a)-(h) are the same as Fig. 5.7. In this case, the disparity map in (b) are wrong in the right region of ground due to the shadow. The occupancy map computed using this disparity map has wrong detection in the corresponding regions highlighted by the black rectangle closed to the black car shown in (d). This leads to the larger detected obstacle in the right part of the image shown in (h). (g) shows our approach is able to avoid those false negatives and gives the correct detection of the car.
Figure 5.11 (a)-(h) are the same as Fig. 5.7. In this case, the disparity map in (b) is wrong in some regions of the fence due to the repetitive pattern of the fence. The occupancy map shown in (d) has wrong detection in the region of fence highlighted by the black rectangle and the obstacle detection in (h) shows most parts of the fence are detected as farther obstacle. The occupancy map obtained by our approach in (c) shows several detections in the region of fence including the correct detection, since the pixels with bad disparity estimation have large number of candidates. (g) shows although our approach does not obtain the completely correct detection of the fence, it is able to obtain much better result than OpenCV-SGBM.

Figure 6.1 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.

Figure 6.2 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.

Figure 6.3 Setup of sample image capture [Mit19].

Figure 6.4 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.

Figure 6.5 Selected hand-crafted features of the second sample in Fig. 6.4. (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.
Figure 6.6  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.

Figure 6.7  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.

Figure 6.8  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.

Figure 6.9  Additional results. (a) Using patch size 57 × 57 for random forest. (b) Using deep features computed by a CNN from a 16-channel image.
Figure 7.1  Localized Topology Conditions (LTC) applied to foraminifera segmentation. (a) Illustration of the segmentation task. A simple image [top] from the dataset [Ge17]. Our aim is to estimate the edges between chambers and segment the sample based on the edge map. Groundtruth labels [bottom-left] and sample segmentation output [bottom-right] are shown. (b) Comparing pixel-based difference and topological detection for sample output in (a). Metrics such as cross-entry, are based on image difference [left] which can highlight areas where edges map are just misaligned. The proposed Deformation Violation (DV)-Mask [right] ignores small deformations and highlights locations where the local topological structure does not match. This mask is used to produce a Localized Topology-Aware (LTA) loss for training. (c) Illustration of topological conditions. On the [left], neighborhoods in which the localized topological conditions on the local complement sets \( \bar{E}_r \) are satisfied even though a small perturbation is present. On the [right], a case in which the conditions are not satisfied due to a missing edge in the prediction.

Figure 7.2  D-LinkNet architecture. Both the input and output are 128 × 128 grayscale images.

Figure 7.3  Iterative refinement edge detection.

Figure 7.4  Examples of synthetic dataset. Top row: ground truth images; bottom row: synthetic edge probability map with random gaps.

Figure 7.5  (a) Average size of DV-mask for fixed \( r \) as a function of \( K_d \) in the foram dataset. (b) Precision-recall curve for \( r = 9 \) when compared to manual labels of errors. The blue dots indicate \( r = 9 \) and \( K_d = 4 \) used in our experiments for the LTC.

Figure 7.6  Pixel difference and DV-mask of estimated edge maps obtained by baseline approach DLinkNet. More results can be found in Supplement.

Figure 7.7  Sample of synthetic image dataset [left] including ground truth [top], edge probability maps with one gap [middle] and two gaps [bottom]. Comparison of proposed local topology metric (left y-axis) and PathError (right y-axis) [right]. Best viewed in color.

Figure 7.8  Performance on synthetic validation set during training. Curves with the same color correspond to the same method through all three plots. Curves have been smoothed with MATLAB function. Evaluation metrics shown include: (a) Pixel-wise F1 score; (b) Region-level unweighted IoU; and (c) Region recall. Best viewed in color.

Figure 7.9  Sample results. (a) Input image. (b) DLinkNet output. (c)Feat output. (d) LTA output. (e) Ground truth labeling. The red boxes highlight the regions the proposed LTA loss obtains clearer and more closed edges than DLinkNet.

Figure 7.10  Localized topology evaluation metrics. The fraction of images that have any errors in the prediction (top) and the normalized (over the area of the image) average area of the DV-Mask (bottom) as a function of threshold \( \tau \) for edge detection. These curves are obtained after training the model five times and averaging the results.
CHAPTER 1

INTRODUCTION

Image segmentation is one of the most important low-level operations in computer vision. The function of image segmentation is to cluster the image pixels into a set of groups visually distinct and uniform with respect to some properties, such as gray level, texture or color [Fre02]. It is quite useful in many applications including medical imaging [Der16], autonomous driving [Zha16], security surveillance [OS15; BR11], and object detection, recognition and tracking [Gup15; Wu12]. Some example applications are shown in Fig. 1.1. The level of segmentation depends on the region of interest of the applications. Medical imaging aims to extract anatomical structures from radiology images such as CT or MR images. For autonomous driving, the regions of interest are traffic signs, obstacles and pedestrians on the road. Segmentations for security surveillance focus on human detection in a sequence of images. Object detection, recognition and tracking require individual objects to be separated from background first, so the features of those objects can be extracted properly for further processing.

A variety of algorithms have been proposed for image segmentation tasks. Grouped by their methodology, segmentation algorithms can be mainly divided into these classes: clustering segmentation approaches partition image into $K$ clusters by using clustering algorithms, including K-means [Dha15] and Mean Shift [CM02; Tao07]; edge-based algorithms [Can86; Dol06; Kon03; Mar04] first predict edges on the image and then get segmentation regions based on the predicted edges; region growing approaches [SC05; Man; Fan05] initialize regions by a set of seeds, then each region is grown from a seed based on the similarity between nearby pixels and the current region; superpixel-based approaches [RM03; Yan07; DY10; Vez12; Zha10a; Zha10b; Wan08; Zha13; Li12] use superpixels as initialization of segmentation to make use of superpixel cues and reduce computational complexity; graph-based algorithms [WL93; SM00; Rot04; AC12; Zha10a; FL12; Vez12; Yao12;
Figure 1.1 Applications of image segmentation. Images are from [Arb11; Der16; Zha16; BR11; Gup15; Wu12].

Zha10b] represent an image as a weighted undirected graph for which the segmentation problem is treated as a graph partitioning problem. With the success of deep neural networks (DNNs) in computer vision, numerous deep learning based methods have been proposed recently. Generally DNN-based image segmentation approaches have an encoder-decoder architecture, with an encoder to learn features from the image, and a decoder to up sample the feature maps as well as make predictions. Various approaches are proposed to improve the performance by better balancing the global and local context information [Gar17]. For example, DeepLab [Che16] applies fully connected conditional random fields as a post-processing step to model the relationship between output labels in order to gain more global information. In ParseNet [Liu15], global and local features are merged by concatenating feature maps at multiple scales.

One desirable property of image segmentation is robustness to noise, parameter selection, changes in image quality and resolution. Segmentation algorithms for medical images are required to be robust to noise, since those kind of images obtained from medical imaging sensors are often polluted by severe noise and blurred edges [BS09]. Interactive image segmentation algorithms are expected to be robust to user inputs [Ngu12]. In aspect of autonomous driving, the robustness of image segmentation is crucial for safety, as the quality of images varies over time due to the change of lighting condition and the environment.
State-of-the-art segmentation algorithms are able to successfully capture different features from images and produce accurate segmentation results for some specific dataset. However, it is difficult for a single algorithm with the same parameters to segment all the images successfully due to variations between images. Many segmentation algorithms are sensitive to small variations in images or changing of parameters due to the behaviors of these algorithms. Graph-cut based approaches are sensitive to the input parameter of region numbers. Edge-based approaches are always influenced by blurring edges. DNN-based segmentation approaches are generally more robust than traditional approaches because of the large training set and the powerful representation learning capacity, but generalization of a deep learning model is still a big issue when the training set is limited, which commonly happens in image segmentation tasks. In order to obtain the robustness, a number of robust image segmentation algorithms have been proposed. In [Der16], by including the distribution of the pixels rather than relying solely on the intensity values, an automated robust image segmentation based on level set method is able to be robust to noise, initial conditions, and intensity inhomogeneity. An interactive image segmentation proposed in [Ngu12] is robust to different user inputs and different initializations by making use of both the boundary and the regional information to find a global optimal solution. In [Gra11], a robust algorithm of low depth of field image segmentation is proposed, which does not need to set any parameters by hand as all as necessary parameters are determined fully automatically or preset. The DNN-based approaches are able to gain more robustness by including more training data under various conditions. However, obtaining labeling data for image segmentation is always a high cost.

In this dissertation, we propose several novel algorithms for robust image segmentation, including a robust image segmentation framework based on persistence homology, a robust obstacle segmentation using a novel multi-valued disparity map, a topology-aware metric to measure topological difference between two edge maps and a topology-aware loss to train a topology preserving edge detection network for edge-based image segmentation.

1.1 Contributions

In this section, we overview the contributions of this dissertation.

1. An innovative framework for robust image segmentation and its applications to consensus-based image segmentation and obstacle detection of outdoor scenes. Fig. 1.2 illustrates the overview of this framework:

- Propose a novel robust image segmentation framework to extract persistent regions from region/edge probability maps based on topological persistence, which is robust to changes in image conditions and parameter selection.

- Propose a consensus-based image segmentation approach to extract consensus information from an input image segmentation set by applying this framework. Experiments in Section 3.4 demonstrate that our algorithm is able to capture the curves consistently present within the input segmentation set.
• Propose an approach to robustly segment obstacle from stereo disparity maps. Experiments in Section 4.4 demonstrate that the algorithm is able to segment each obstacle on the road and is robust to parameter selection.

2. A novel algorithm to compute a multi-valued disparity map from stereo image pairs and its application to obstacle detection of outdoor scenes. Fig. 1.3 shows the pipeline of the multi-valued disparity map based obstacle detection:

• Propose an algorithm to compute the multi-valued disparity map from stereo image pairs by selecting candidate disparity values based on a statistical analysis of the cost function of semi global block matching [Hir08; Ope].

• Develop an obstacle detection pipeline based on the multi-valued disparity map. Experiments in Section 5.3 demonstrate that the multi-valued disparity map is able to provide a more reliable occupancy map for obstacle detection than the traditional single-valued stereo disparity map.

3. A set of novel localized topology conditions that identify structural difference between two edge maps and a novel topology-aware loss used to train a topology-aware edge detection network. Fig. 1.4 illustrates the overview of the topology-aware edge detection pipeline:

• Create a foraminifera dataset for study of visual species identification and structure segmentation with over 1400 foraminifera sample images of 6 species and over 450 labeled images for study of segmentation.

• Propose a coarse-to-fine edge detection strategy for limited computation resource. Experiments in Section 6.3 demonstrate that this approach is able to obtain a clear edge map from a low quality image with vague edges.
Propose a novel topology-aware evaluation metric based on localized topology conditions to measure the topological difference between two edge maps and indicate the regions causing the difference, which is demonstrated by experiments in Section 7.2.6.

Propose a topology-aware (LTA) loss to force an edge detection network to focus more on preserving topological structures. Experiments in Section 7.2.7 and 7.2.8 demonstrate that LTA loss is able to force the network to preserve topological structures such as closing gaps on edges and suppressing noises.

Figure 1.3 Pipeline of robust obstacle segmentation based on multi-valued disparity maps.

Figure 1.4 Overview of the topology-aware edge detection.
1.2 Dissertation Outline

The remainder of this dissertation is organized as follows: Chapter 2 provides the background knowledge of this dissertation, including topology data analysis, stereo vision for autonomous driving and deep neural networks for dense prediction, followed by works related to this dissertation. Chapter 3 describes the consensus-based image segmentation algorithm by applying the robust image segmentation framework. Chapter 4 gives a detailed description of the robust obstacle segmentation algorithm for autonomous driving based on the robust image segmentation framework. Chapter 5 discusses the computation of multi-valued disparity map and the application on obstacle detection. Chapter 6 introduces the coarse-to-fine edge detection strategy for low quality images with vague edges. Chapter 7 describes the topology-aware evaluation metric and the topology-aware loss based on this metric for training an edge detection network preserving topological structures. Finally, chapter 8 concludes this dissertation and discusses the future directions.
In this chapter, we provide a background of topology data analysis, stereo vision for autonomous driving and deep neural networks, followed by the overview of related works.

2.1 Topological Data Analysis

Topological Data Analysis [Ede02] is a field of study which employs tools from persistent homology theory [EH08]. This analysis is commonly used for the extraction of topological features from functions or point cloud data. These features are captured in a compact visual representation called the persistence diagram [EH10]. Persistent homology has been used for segmentation [LF07] of natural images and clustering [Cha11]. This section provides a brief overview of some of the concepts of persistent homology in the context of image analysis. A comprehensive review of this topic can be found in [EH10].

2.1.1 Simplicial Complexes

A point cloud can be described by homology which operates on Simplicial Complexes. Let \( \mathcal{X} = x_1, x_2, \ldots, x_n \) be a set of sample points of a space. A \( k \)-simplex is a \( k \)-dimensional convex combination of its \( k + 1 \) vertices, which can be written as

\[
\sigma = \text{conv}\{x_0, x_1, \ldots, x_k\},
\]

where \( \text{conv}\{x_1, \ldots, x_n\} \) is the convex hull of \( n \) vertices. Simplex has special names for the first few dimensions: a 0-simplex is a vertex, a 1-simplex is an edge and a 2-simplex is a triangle. Fig. 2.1 (d)
shows an example of those simplices. A face of a k-simplex, which is a (k-1)-simplex, is the convex hull of a non-empty subset of the vertices of the k-simplex. Then a simplicial complex is defined by a finite collection of simplices if faces of each simplex are also included in the collection.

To build the complexes of a point cloud, one can place a ball on each vertex and then construct simplices based on their pairwise distance relative to the radius of the ball. Vietoris-Rips complex [Hau95], which is computationally efficient, can be used for the construction. The simplices of Vietoris-Rips complex with parameter $\epsilon$, denoted as $VR_\epsilon$, are formed as follows: for each subset $S \in \mathcal{X}$, form an $\epsilon/2$-ball around each vertex in $S$, and include $S$ as a simplex of dimension $|S|$ if all the balls in $S$ have pairwise intersections. That is, any subset $S \in \mathcal{X}$ is a simplex in $VR_\epsilon$ if and only if $d(x_i, x_j) \leq \epsilon$, for all $x_i, x_j \in S$.

### 2.1.2 Persistent Homology

![Persistent homology diagram](image)

**Figure 2.1** Persistent homology. (a) Two spaces $M_1$ and $M_2$ (images of letter D and B) with sample point cloud. (b) Sample point cloud $\mathcal{X}_1$ and $\mathcal{X}_2$ corresponding to letter D and B. (c) Corresponding persistence diagram of connected components (red points) and holes (blue points). (d) Examples of simplex. (e) Filtration on point cloud $\mathcal{X}_1$.

Persistent homology [EH10] is the main tool of the topological data analysis. The definition of homology is motivated by the observation that two shapes can be distinguished by their holes. For example, as shown in Fig. 2.1, letter D has only one holes while letter B has two. Then letters D and B can be distinguished by the number of holes. Homology of a topological space can be summarized using the Betti numbers, which are the ranks of topological invariants called homology groups. The $n$-th Betti number, $\beta_n$, is the number of $n$-dimensional holes on a topological surface. For example, $\beta_0$ is the number of connected components, $\beta_1$ is the number of holes and $\beta_2$ is the number of voids.
of the surface. One may find a scale $\varepsilon$ to make the constructed simplicial complex of the same type of the structure invariant. However, the proper value of $\varepsilon$ is not fixed for all the cases.

Persistent homology theory computes a sequence of simplicial complex of a space at multiple scales $\varepsilon$, which is called filtration, to detect more persistent features over a wide range of scale and then is more likely to explore the true features of the data. A filtration $\{S_\varepsilon\}$ can be obtained by increasing $\varepsilon$ in a range of interest and it satisfies the following property:

$$S_{\varepsilon_1} \subseteq S_{\varepsilon_2} \quad \text{whenever} \quad \varepsilon_1 \leq \varepsilon_2.$$ (2.2)

Persistent homology computes the values of $\varepsilon$ for which topological features appear ($b^k_n$) and disappear ($d^k_n$) during a filtration, referred to as the birth and death values of the $k$-th feature in dimension $n$ (e.g., connected components if $n = 0$ and holes in the space if $n = 1$). We will focus on connected components in order to characterize persistence regions in image segmentation. This information is encoded into a multiset of points ($b^k_n, d^k_n$), called a persistence diagram [CS07]. Each point is referred to as a persistence interval with corresponding length equal to $d^k_n - b^k_n$. Fig. 2.1 (c) shows two persistence diagrams for $n = 0$ and 1. Algorithms for the efficient computation of persistent homology can be found in [Ede02; ZC04].

An example of a sampled data $\mathcal{X}$, filtration $S_\varepsilon$, and persistence diagram is shown in Fig. 2.1. Two sample point clouds $\mathcal{X}_1$ and $\mathcal{X}_2$ in (a) are sampled from two spaces, letter D and B, respectively. The corresponding persistence diagrams for connected components and holes are shown in (c). The filtration on point cloud $\mathcal{X}_1$ is illustrated in (e). Space $M_1$ can be described as having one connected component and one hole, meaning $\beta_0 = 1$ and $\beta_1 = 1$. On the other hand, $M_2$ can be described as a space with $\beta_0 = 1$ and $\beta_1 = 2$. These features are correctly represented by the persistence diagrams in (c). Although there are some other features shown in the diagrams, those features are distinguished from the persistent features based on the persistence interval.

In our segmentation framework, the probability map is the probability of regions or edges. The desired segments from the probability map should be the regions with high probability values or the regions enclosed by edges with high probability values. One way to obtain those regions is to apply a threshold to the probability map and all the connected components in the upper level set of the probability map are regarded as the segmentation result. We can get reasonable segmentations if we tune the threshold for each image. However, this threshold value cannot be the same for all the images due to variations between images. Also, the single threshold may introduce some noise which is a small perturbation in the probability map. Since persistence homology is used to extract persistence features of the data, we borrow the concept of it to extract persistence regions which describe the probability map as our segmentation result. The noise caused by small perturbation in probability map is removed from the segmentation because those regions have low persistence interval.

As mentioned before, persistent homology computes a sequence of simplicial complexes at multiple scales, which is called a filtration. For our probability map, the filtration is computed by generating a set of upper level sets of the probability map from high threshold to low threshold.
value. The topological feature we focus on is the connected component in each upper level set. The birth time is defined as the threshold value at which the connected component first appears during filtration and the death time is defined as the threshold value at which the connected component merges with other connected component born before it during filtration. In the persistence diagram, we apply a threshold to only choose the connected components with high persistence interval to avoid small perturbations of the probability map. The size of each connected component is varying during filtration. To construct our segmentation, the largest set of points which is associated with its death time is selected as the segmentation result.

One fundamental property of persistence diagram is stability [CS07]. That is, small changes in the point cloud imply only small changes in the diagram. Take 1-D function as example, let $f$ and $g$ be two functions, $D(f)$ and $D(g)$ be the corresponding persistence diagrams. The stability can be represented as

$$d_B(D(f), D(g)) \leq \|f - g\|_\infty,$$

(2.3)

where $d_B$ is the bottleneck distance and $\|\cdot\|$ is the $L_\infty$-norm. Fig. 2.2 illustrates the stability of persistence diagram for two 1-D functions. More details about stability of persistence diagram can be found in [CS07].

![Figure 2.2 Stability of persistence diagram [CS07]. Left: two functions closed to each other. Right: the persistence diagram of both functions and the bijection between them.](image)

One example of extraction persistence connected components for image segmentation is shown in Fig. 2.3. The image in (a) is a function $f : \mathbb{R}^2 \rightarrow [0, 1]$. This image has one connected component which is a circle. Given a threshold value $\tau \in [0, 1]$, we compute the upper level set $S_\tau = f^{-1}[1 - \tau, 1]$. Some of them are shown in (b). In (c), there is a single cluster in the diagram for which its death time is much greater than its birth time (i.e., the point is furthest away from the diagonal). There is also...
Figure 2.3 Persistence Analysis of 2D probability image. (a) Original grayscale image, (b) images after thresholding, and (c) persistence diagram. Each point in the diagram corresponds to clusters that are born and die at specific threshold values. Points near the diagonal are sensitive to small variations of the original image.

one cluster near the diagonal which only exists for a short persistence interval. The thresholded images illustrate that the small cluster is formed around $\epsilon = 0.25$ and merges with a larger cluster soon after. Then for this image, there is only one persistence region can be extracted which is the circle shown in the middle of the image.

2.2 Stereo Vision

Stereo vision is the extraction of 3D information from stereo images, which is similar to the biological process stereopsis. In stereo vision, features in two or more images taken at the same time from separated stereo cameras are matched with the corresponding features in the other images, and the differences are analyzed to yield depth information [BK13].

Our data for obstacle segmentation are captured by two cameras mounted on a car (left and right cameras). In practice, stereo imaging involves the following steps when using two cameras: First, the intrinsic and extrinsic parameters of stereo cameras are estimated through the camera calibration process. By using those camera parameters, undistorted and rectified images are obtained through undistortion and rectification process, respectively. The undistortion refers to removing radial and tangential lens distortion from the image while rectification adjusts the angles and distances between cameras. Then by finding corresponding points in the left and right images, a disparity map is computed by the differences in x-coordinates on the image planes of the same feature viewed in the left and right cameras. Finally, the distance information can be transformed from disparity map by triangulation. An example of the stereo imaging is shown in Fig. 2.4.

In this section, we start from the setup of the stereo vision for autonomous driving with two cameras. Then give a brief introduction of camera calibration and image rectification, followed by disparity map computation.
2.2.1 Data Collection

The stereo setup used in our lab is shown in Fig. 2.5. It contains two cameras and a trigger circuit to synchronize the cameras. The cameras are fixed on a board and can be mounted on a car like Fig. 2.5 (b). The angle of cameras can be adjusted but remains unchanged during recording after calibration. The captured data are stored in a computer which connects two cameras through USB cables. To obtain some types of groundtruth, a laser scanner is also included for some setups such as KITTI dataset [Gei12; Gei13; Fri13].
2.2.2 Camera Calibration

The raw stereo pairs captured by stereo cameras are distorted and unrectified. The undistorted and rectified pairs which align properly between left and right images used for disparity computation are obtained through undistortion and rectification process, which require the camera parameters estimated from the calibration process. The aim of camera calibration is to estimate the relative location and orientation of the camera as well as camera intrinsic parameters. In this section, we introduce intrinsic and extrinsic parameters of cameras followed by the camera calibration process.

2.2.2.1 Lens Distortions

In practice, it is impossible to find a lens that introduces no distortions. There are two main lens distortions [BK13]. One is radial distortions, which arises as a result of the shape of lens. Another one is tangential distortions, which arises from assembly process of the camera as a whole.

Radial distortions are often noticeably observed in the location of pixels near the edge of the imager, as for some lenses, rays farther from the center of the lens are bent more than those closer in. The radial distortions can be modeled as

\[
x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)
\]

\[
y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)
\]
where $r$ is the distance from optical center of the imager, $k_i$, $i = 1, 2, 3$, are the distortion parameters needed to be estimated through calibration, $x$ and $y$ are the original location of the distorted point in raw image and $x_{\text{corrected}}$ and $y_{\text{corrected}}$ are the undistorted location of the point [BK13].

Tangential distortion is resulting from the lens not being exactly parallel to the imaging plane. It can be modeled as

\begin{align}
    x_{\text{corrected}} &= x + [2p_1 y + p_2 (r^2 + 2x^2)] \\
    y_{\text{corrected}} &= y + [p_1 (r^2 + 2y^2) + 2p_2 x]
\end{align}

(2.6) (2.7)

where $p_1$ and $p_2$ are distortion parameters [BK13].

Thus, there are five parameters $k_1, k_2, k_3, p_1$ and $p_2$ required for image undistortion, which are referred to as nonlinear intrinsic parameters.

### 2.2.2.2 Linear Intrinsic Parameters

Without distortion, the transformation from 2D image space to 3D physical world space is a projective mapping. Let $[u, v, 1]^T$ represent a 2D point in image space and $[X, Y, Z, 1]$ represent a 3D point in physical world. Then the mapping can be written as

\[
    z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K[R \ T] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix},
\]

(2.8)

where $K$ is the $3 \times 3$ intrinsic matrix which contains linear intrinsic parameters, $3 \times 3$ rotation matrix $R$ and $3 \times 1$ translation vector $T$ contain the extrinsic parameters.

The linear intrinsic parameters required for rectification are: focal length $F$, scale factors $s_x$ and $s_y$ (pixel to real world distance), and principal point $(u_0, v_0)$. The intrinsic matrix $K$ can be written as

\[
    K = \begin{pmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}
\]

(2.9)

where $a_x = F s_x$ and $a_y = F s_y$.

### 2.2.2.3 Extrinsic Parameters

The extrinsic parameters $R$ and $T$ denote the relative rotation and translation between the camera coordinates and the world coordinates.

Let $\psi$, $\phi$ and $\theta$ be rotation angles of $x-$, $y-$ and $z-$axes of world coordinates. Then the rotation
matrix can be written as \( R = R_x(\psi)R_y(\phi)R_z(\theta) \), where
\[
R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \psi & \sin \psi \\ 0 & -\sin \psi & \cos \psi \end{pmatrix},
R_y(\phi) = \begin{pmatrix} \cos \phi & 0 & -\sin \phi \\ 0 & 1 & 0 \\ \sin \phi & 0 & \cos \phi \end{pmatrix},
R_z(\theta) = \begin{pmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.
\]

The translation vector \( T \) is the offset between the origin of the two coordinate systems and can be computed by \( T = \text{origin}_{\text{world}} - \text{origin}_{\text{camera}} \).

### 2.2.2.4 Stereo Calibration

The stereo calibration is the process of estimating the intrinsic and extrinsic parameters of both cameras as well as the geometrical relationship between the two cameras in space, which is done through basic geometrical equations. The equations used depend on the chosen calibrating objects, such as chessboard or circle pattern.

The detailed description of estimation intrinsic and extrinsic parameters can be found in [BK13]. Generally speaking, first, a set of snapshots of the calibrating object in different distances and angles are taken by stereo cameras. Then find the calibrating object in each snapshot to construct a new equation. And the equation can be solved by a well-posed equation system formed by using at least a predetermined number of snapshots.

Then the rotation and translation between two cameras are computed as
\[
R = R_r^T R_l^T
\]
\[
T = T_r - R_l T_l,
\]
where \( R_l, R_r, T_l \) and \( T_r \) are the rotation and translation parameters of the left and right cameras.

By using those parameters, we can project the image planes of the two cameras so that they reside in the exact same plane, with image rows perfectly aligned into a frontal parallel configuration. An example of a rectified image pair is shown in Fig. 2.4 (b) and (c).

### 2.2.3 Disparity Map Computation

Given two images of stereo pair with rows aligned after rectification, that is, the stereo correspondences are on the same row of two images, the disparity is measured as \( d = x_l - x_r \), where \( x_l \) and \( x_r \) are \( x \) coordinates of the stereo correspondence in left and right images.

The crucial step for disparity map computation is finding the correspondence in properly aligned stereo pairs. One of the efficient algorithms is called Semi-Global Matching (SGM) [Hir08]. This method is based on the idea of pixelwise matching of mutual information and approximating a global, 2D smoothness constraint by combining many 1D constraints. More details can be found in [Hir08]. Another commonly used algorithm is OpenCV [Ope] implementation of Semi-Global
Matching (SGBM), which is modified from SGM by matching blocks instead of individual pixels. One example of disparity map computed from SGBM is shown in Fig. 2.4 (d). In our work of multi-valued disparity map, we select candidate disparity values based on a statistical analysis of the matching cost function of SGBM to provide more robust disparity computation.

2.3 Deep Neural Networks

Deep Neural Networks (DNNs) are a family of machine learning algorithms with powerful representation learning capacity. The most significant difference between DNN and other machine learning algorithms is that the DNN is able to learn how to extract abstract representations from raw data through a large training set. Thus, it requires less human effort on feature engineering than traditional machine learning algorithms, which makes it a powerful tool to model complicated data such as images and natural language data. The dense prediction in computer vision is a task of making pixel-wise predictions on input images. Some examples are image segmentation, depth estimation and edge detection. In this section, we will briefly overview deep learning concepts related to dense prediction tasks. A more comprehensive introduction of deep learning can be found in [Goo16].

2.3.1 Neural Networks

The basic idea of neural networks is to approximate some function by solving an optimization problem. A neural network can be generally written as

\[ y = f(x; \omega), \tag{2.13} \]

where \( x \) and \( y \) are the input and output respectively, and \( \omega \) is a set of trainable parameters. Those trainable parameters are often learned by gradient based optimization algorithms through back propagation [Rum88] to minimize some loss function. A typical neural network consists of one input layer, one output layer and one or more hidden layers in between, each of which followed by nonlinear activation functions. DNNs are referred to neural networks with more than one hidden layers. In most cases, deeper neural networks are able to approximate a function with a smaller number of parameters than shallow ones because of their nonlinearities. Thus, DNNs are heavily used in computer vision tasks, which are required to model complicated nonlinear relationships between input and output. In this subsection, we overview some commonly used layers, including fully connected layers, convolutional layers, pooling layers, transposed convolutional layers and dilated convolutional layers.

2.3.1.1 Fully Connected Layers

In fully connected layers, each pair of input and output units is connected. Fig. 2.6 (a) illustrates the structure of a fully connected layer. Let \( y_i \) and \( x_j \) be the \( i \)-th output and \( j \)-th input unit respectively,
and $w_i^j$ be the weight of edge connecting $y_i$ and $x_j$, then a fully connected layer with $M$ output and $N$ input units can be written as

$$
y_i = \sum_j x_j w_i^j, \text{ for } i = 1, \ldots, M.
$$

(2.14)

Because of the fully connectedness between input and output, fully connected layers are often prone to overfitting the training data. Thus, fully connected layers do not directly take the raw image data as input for most of the time. Instead, they are usually used as the last few layers of a neural network to classify features extracted by previous layers, because those features have a much lower dimension than the raw image data. Also, the fully connectedness disregards the local structure from the input, which makes fully connected layers less desirable in dense prediction.

### 2.3.1.2 Convolutional Layers

Convolutional layers are often used to process grid-like data, such as images. Instead of connecting all the pairs of input and output units, convolutional layers connect input and output sparsely. That is, each location of output is only computed from a small portion (usually a small patch around that location for 2D data) of the input. Fig. 2.6 (b) illustrates the structure of a convolutional layer. Let $y(i, j)$ and $x(i, j)$ be the 2D input and output units at location $(i, j)$ respectively, and $\omega(m, n)$ be the weight apply to the patch with size $M \times N$ at $(m, n)$, then the output $y(i, j)$ of a convolutional layer can be written as

$$
y(i, j) = \sum_m \sum_n x(i + m, j + n) \omega(m, n),
$$

(2.15)

where $\omega(m, n)$, for $m = 1, \ldots, M$ and $n = 1, \ldots, N$ is called the kernel function. Equation 2.15 is similar to the convolution operation without kernel flipping, which is actually a correlation.
why it is called the convolutional layer. The kernel function can be thought of as a feature extractor in image processing, like the edge detector. Then a feature map containing the features extracted from the entire input can be obtained by applying the same kernel on each location of the input. That is, the kernel function is independent on the location \((i, j)\). In practice, to learn a set of different features from the input, usually more than one kernels are applied to obtained several feature maps, then both input and output data are usually 3D data with an additional channel dimension (number of features), so each kernel is extended to 3D as well. If the number of input channels and output channels are \(C_{in}\) and \(C_{out}\) respectively, and the kernel size is \(k \times k\) (\(k\) is usually an odd number to center the kernel on a pixel of input), then the number of parameters is \(C_{in} \times k \times k \times C_{out}\). Note that the number of parameters does not depend on the spatial size of input and output. Thus it is possible to use arbitrary size of input for a trained convolutional layer. Besides kernel size and number of feature maps, there are two other hyper-parameters, stride and padding size, which control the spatial size of the output feature maps. The stride is the number of pixels to slide the kernel. When stride is 1, the kernel moves one pixel at a time, while stride is 2, the kernel slides 2 pixels at a time, which results feature maps with a smaller spatial size. Padding on border of input data is used to compensate the boundary pixels discarded by convolution operation in order to prevent the size of feature maps from being smaller and smaller as the number of convolutional layers increasing. For example, to retain the spatial size of input when using a \(k \times k\) kernel, the padding size should be \(k - 1\). In practice, zero-padding is commonly used. However, for some image generation tasks, mirror-padding is preferred.

Compared with fully connected layers, convolutional layers have a smaller number of parameters because of the sparse connection between input and output. Also, its local connectivity retains the local structure of the input, which is desired in dense prediction tasks.

2.3.1.3 Pooling Layers

The pooling layer is a special type of layer as it does not have trainable parameters. Each location of the output is a summary statistic of presence of features within a neighborhood in the input. Note that the pooling operation is applied on each feature map individually. Thus both the input and output feature maps have the same number of channels. Similar to the convolutional layer, the kernel size is referred to the size of the neighborhood and stride is the number of pixels to slide the kernel. Two pooling layers are commonly used in practice, including max pooling and global average pooling [Lin14]. Fig. 2.6 (d) and (e) illustrate the max pooling and the global average pooling. The max pooling picks the most activated feature within the neighborhood. It is usually applied after a convolutional layer with kernel size 2 and stride 2 to decrease the spatial size of the feature maps in order to reduce the computational complexity and progressively increase the receptive field of features. Also, this operation provides the local translation invariance of feature maps. However, as it only picks one feature out of a set of features, it discards detailed local context information which is crucial for dense prediction. The global average pooling layer [Lin14] outputs the average of each input feature map. It is often applied after the last convolutional layer to produce a fixed
size of features for the following fully connected layers.

### 2.3.1.4 Transposed Convolutional Layers

In dense prediction or image generation tasks, the algorithms usually involve up sampling of feature maps to a higher resolution. Several interpolation methods, such as nearest neighbor interpolation and bi-linear interpolation, can be used. Instead of up sampling through these manually parameter picking methods, transposed convolution operation is designed for learning optimal up sampling parameters to gain more powerful representations. Transposed convolutional layers are also called deconvolutional layers or fractionally-strided convolution layers. However, the name "deconvolutional" layer is not preferred because deconvolution is already defined as the inverse of convolution [DV16]. The transposed convolution operation can be thought of as the inverse direction of the convolution operation. Note that it is not the inverse operation of the convolution, it only recovers the shape of initial input of a convolutional layer instead of the values. While convolution is a many-to-one operation as shown in Fig. 2.6 (b), transposed convolution is a one-to-many operation as shown in Fig. 2.6 (c). If the input and output feature maps of a convolutional layer are unrolled into vectors, then the convolution operation on entire feature map can be represented as a matrix multiplication with a non-square matrix $C$ constructed by the kernel, and the transposed convolution operation from output to input can be again represented as a matrix multiplication with a matrix $C^T$ if the same kernel is used. This is how the name transposed convolution comes from. A more detailed explanation can be found in [DV16].

As the transposed convolution also involves weighted summation of input pixels, it can be always implemented as its equivalent convolution operation. Thus, transposed convolutional layers have the same set of hyper-parameters as convolutional layers, except the stride $s$, which controls the number of zeros inserting between two neighboring input pixels. When $s = 1$, the transposed convolution operation is the same as a convolution operation with $s = 1$. When $s > 1$, it is equivalent to a convolution operation with $s < 1$. This is how the name fractionally-strided convolution comes from [DV16].

### 2.3.1.5 Dilated Convolutional Layers

As mentioned in Section 1, global context information is crucial for dense prediction to resolve the local ambiguities. Increasing the receptive field of features by progressively applying max pooling layers after convolutional layers is one way to gather more global context. However, this produces low resolution feature maps which discards detailed local context containing in input and requires more transposed convolutional layers to up sample feature maps to the original resolution. The dilated convolutional layer [Che16], which is able to expand the receptive field while retaining the resolution of feature maps, is proposed to resolve this issue. Dilated convolutional layers are also called atrous convolutional layers. A dilated convolutional layer is similar to a convolutional layer, except that the kernel is expanded in width and height by inserting zeros between neighboring elements and the number of zeros is controlled by a dilated rate $l$. Fig. 2.7 illustrates the kernels with
different values of \( l \). Let \( \omega(m, n) \) be a kernel with original size \( K \times K \), then the dilated convolutional layer can be written as

\[
y(i, j) = \sum_{m}^{K} \sum_{n}^{K} x(i + l \cdot m, j + l \cdot n) \omega(m, n). \tag{2.16}
\]

The dilated convolutional layer is the same as the regular convolutional layer when \( l = 1 \). As \( l \) increasing, the number of parameters and the output resolution (with proper padding) are kept the same, but the receptive field gets larger. When properly stacked, the receptive field can be exponentially increased without loss of resolution \[\text{[YK15]}\]. One issue of dilated convolutional layers is when \( l \) gets larger, the number of valid weights (weights applied to features instead of padding regions) gets smaller, which leads to parameter inefficient or even degenerates the layer to a regular convolutional layer with smaller kernels \[\text{[Che17]}\].

\section*{2.3.2 Fully Convolutional Networks}

As discussed above, for image data, usually first a few layers are convolutional layers to learn the features from the data, these are then followed by one or more fully connected layers to make predictions. This works well in single or sparse prediction tasks. However, for the case of dense prediction, one can use a fully connected layer to directly make predictions for all the pixels but lose the local structure retained through the convolutional layers, or use a small fully connected layer to apply on a small patch of feature maps to make a prediction for each pixel one by one which introduces a large amount of unnecessary repetitive computation. To overcome this, the fully convolution network (FCN) \[\text{[Lon14]}\] replaces fully connected layers with convolutional layers with kernels covering the entire input nodes. By doing this, this layer is able to accept an input of arbitrary size. If the input has the same size as before, the output will remain the same. If the input has a larger spatial size, the output will have a larger size as well and each output feature only depends on the input covered by the original size of fully connected weights.
2.3.3 FCN based Dense Prediction

The FCN based dense prediction generally has an encoder-decoder architecture [Vei09] and sometimes has a middle part in between, as shown in Fig. 2.8. The encoder consists of several convolutional layers followed by max pooling layers, and is used to extract features from the input image and down sample the resolution of feature maps to limit the computational complexity. The decoder consists of corresponding number of transposed convolutional layers to up sample the feature maps to the original resolution. The lower resolution feature maps received by the decoder do not contain detailed local context information due to max pooling. Skip connections [Lon14] between the encoder and the decoder are used in some applications to directly provide original detailed local information to decoder. To further balance the local and global information, the middle part can also consist of several dilated convolutional layers with carefully designed architectures, including sequentially stacking [YK15], Atrous Spatial Pyramid Pooling (ASPP) [Che17] and denseASPP [Yan18]. In our work of topology-aware edge detection, instead of designing a new network architecture, we propose a new loss function to quantify the topological difference between the predicted edge map and the edge label.

2.4 Related Works

In this section, we overview previous works related to our work, including consensus-based image segmentation, outdoor scene image segmentation for obstacle detection, disparity map computation, edge detection and class imbalance in dense detection tasks.
2.4.1 Consensus Image Segmentation

Consensus image segmentation makes use of the consensus information of a set of segmentations to obtain a better performance. In [NN13], the segmentation set is generated by many runs of a randomized segmentation algorithm. The closed contours are then obtained by combining those segmentation through consensus region merging. In [Oza15], a consensus segmentation algorithm applied on remotely sensed images is introduced, using a stochastic optimization algorithm based on the Filtered Stochastic BOEM (Best One Element Move) method. Also, it gives a way to estimate the optimum number of the clusters in segmentation. An unsupervised approach of consensus segmentation based on the graph cuts using the consensus inferred from hierarchical segmentation ensembles for partitioning images into foreground and background regions is presented in [Kim14]. In [Fra11], the segmentation set is used for computing a superpixel image, which is used to generate consensus clustering. In [TS14], the authors propose a bi-clustering framework and perspective for reaching consensus in grouping problems, which can be used in consensus image segmentation. In [Mey13], a cluster ensemble is used to determine the number of clusters in a group of data, which can be used to estimate the number of regions in the segmentation.

2.4.2 Outdoor Scene Image Segmentation

Outdoor scene image segmentation is a crucial step of scene understanding for autonomous driving or other related tasks. Besides color information, depth information is often utilized as stereo cameras and LIDARs are usually mounted on cars. In [Che10b], Chen et al. segment the stereo disparity map by employing a depth slicing technique and then, accurately mark the object boundaries using a region growing method to improve on-road obstacle segmentation. Another region growing technique for vehicle detection is suggested by Kormann et al. [Kor10]. In the first step, vehicles, modeled as cuboids, are detected using mean shift clustering of planar segments. Then, a UV-disparity map [Kor10] is computed to generate hypotheses for vehicle appearance and disappearance.

Wang et al. present a method for robust obstacle detection and free space calculation based on efficient disparity map computation and G-disparity [Wan14b]. The obstacles are detected using UV-disparity maps and splines are used for the road model. In [LA12], Lefebvre et al. perform vehicle detection by applying mean shift segmentation directly on the 3D point cloud estimated from the dense disparity maps computed from a stereo pair.

Erbs et al., in [Erb11], compute dynamic stixels from stereo disparity map and use the Dynamic Stixel World representation for efficient and compact one-dimensional modeling of real world 3D road scenes. Optimal segmentation is performed by means of dynamic programming. In [Erb13], this group presents another method for traffic scene understanding and driver assistance system by incorporating a Bayesian segmentation approach. The stixel representation of images adds robustness to their algorithms and thus, their method works pretty well even in adverse weather conditions.
A dense disparity map based on-road obstacle detection is presented as a constrained optimization problem in [Mil07]. The depth image here, is segmented based on surface orientation. In [GB06], two new obstacle detection algorithms based on disparity map segmentation for applications in intelligent vehicle systems are presented. The first algorithm assumes that the obstacles are located almost parallel to the image plane and directly segments them using a robust model fitting method applied to the quantised disparity space. The second method employs some morphological operations, followed by a robust model fitting technique to separate the ground regions.

Lee et al., in [Lee11a], perform vehicle detection using a road feature and disparity histogram. Road features are extracted from v-disparity maps and localized obstacles are divided into multiple obstacles using a disparity histogram and remerged using four criteria parameters - the obstacle size, distance, angle between the divided obstacles, and the difference of disparity values. In [Lee11b], they present another stereo-vision based obstacle detection approach using UV-disparity map and bird’s-eye view mapping.

Recently, map-based segmentation and navigation techniques for autonomous vehicles are also gaining popularity. In [Mar14], Martinez et al. propose a mapping algorithm based on probabilistic and heuristic methods to classify and predict the areas around an autonomous robot. Another path planning method for mobile robots is presented in [Che10a]. This paper employs an enhanced dynamic Delaunay Triangulation approach and a GPS tail technique for robot navigation. In [Pos11], Posada et al. present a robust method of floor-obstacle segmentation for mobile robot navigation. The method relies on fusing opinions of multiple heterogeneous classifiers generated from different segmentation schemes like graph cut and region growing to improve the overall classification rate.

Some other papers on traffic scene analysis and obstacle detection have incorporated techniques like watershed segmentation [Vei09], connected component analysis [Kha13], plane-fitting and edge based segmentations [Li04]. Literature has also addressed several surveys on intelligent transportation systems [ST13; Had14].

2.4.2.1 Disparity Map Computation

A disparity map is used to estimate the distance of each pixel away from cameras in stereo images. It usually has a lower accuracy estimation than the depth map captured by LIDARs, but is less expensive.

In [Hir08], Hirschmuller presents a classical method for disparity map computation. A pixelwise, mutual information based matching cost is used for matching between stereo pairs. This method is able to perform a fast pathwise optimization from all directions. In [Høi10], disparity map is estimated by applying an iconic Kalman filter and known ego-motion. It is able to reduce the variance of disparity of each pixel and increase the density of the disparity map. In [HK13], to improve the robustness of the correspondence matching, Hermann and Klette introduce an iterative semi-global matching algorithm. The search space for disparity is iteratively reduced by using a pre-evaluated disparity prior. In order to improve fusion of disparity information, Wang et al. in [Wan14a] propose a post-aggregation method based on the Dempster-Shafer Theory (DST). Recently, Luo et al. in
[Luo16] propose an efficient deep-learning based stereo matching method, which is able to provide high accurate disparity maps in less than a second of GPU computation.

### 2.4.3 Edge Detection

One of our works is edge-based segmentation, which makes use of edge clues to help with segmentation. So we also overview edge detection algorithms here. Many studies have been carried in the past 50 years in edge detection, contour prediction and edge-based segmentation. In early days, widely used model-based edge detectors, such as the Canny detector [Can86] and the Sobel detector [Kit83], only depend on the local gradient of color or texture in images. As they treat both object boundaries and texture edges with no difference, data driven and learning-based methods are more desirable in scenario of semantic region segmentation. Dollar et al. in [Dol06] 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 [Lim13] groups edge patches into sub-classes based on the edge shape and treat the edge detection as a multi-class classification task. In [DZ15], structured edge detection is proposed to predict labels of multiple pixels simultaneously.

Recently, with the recent success of deep neural network (DNN) in computer vision, numerous DNN-based methods have been proposed [Ber15b; Liu17; XT17; Xu17; Zhu18], significantly pushing the field forward. In [She15], 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 [Ber15b] and [Ber15a] use object-level cues from pre-trained image classification CNNs to improve the contour detection. Holistically-nested edge detection [XT17], achieves the state-of-the-art image-to-image edge detection performance via a novel holistically-nested architecture. Liu et al. [Liu17] fully exploit and utilize the multiscale and multilevel features from pre-trained convolutional networks to further improve the performance. However, most methods still rely on pixel-level losses, failing to encode high-level geometry and topology information. Several studies have been performed to explore this issue [BH16; RM17; Qu18; Mos18]. For example, StripNet [Qu18] segments long and continuous strip patterns by first considering the segmentation as a boundary-regression problem, then applying the topological constraints on the predicted boundaries. Also, Mosinska et al. [Mos18] propose a topology-aware loss to implicitly impose the topological constraints on predicted edges by minimizing the distance of high-level features between the prediction and the ground truth labeling.

### 2.4.4 Class Imbalance in Dense Predictions

Dense prediction tasks such as edge detection and object detection always face class imbalance during training as only a small part of training samples are positive. Class-balanced cross-entropy loss function proposed in [XT17] and class-balanced sampling discussed in [Ban17] are commonly used in edge detection tasks. For object detection tasks, hard example mining is used to select the hardest negative samples for class-balanced training [Shr16; Gir15]. Besides class-balanced sampling approaches, focal loss proposed in [Lin17] reshaped the loss function by adding a factor
to cross-entropy loss to make the training more focus on hard misclassified samples. Our localized topology-aware loss is similar to class-balanced cross-entropy loss and focal loss in weighting the loss for different pixels. However, the proposed topology-aware loss puts higher weights on pixels causing topology difference to train the model focusing on preserving the topological structure of edges.
As mentioned in Section 1, state-of-the-art segmentation algorithms are able to successfully capture different features from images. However, it is difficult for a single algorithm with the same parameters to segment all the images successfully due to variations between images. One feature that may be observed is that when comparing the segmentation results from different algorithms with different parameters, the desired boundaries are detected more consistently than other boundaries. One example is shown in Fig. 3.1 (b). The four segmentations are generated by four different algorithms. The boundary of the swan is detected in all the results with a small perturbation from each other and the undesired edges are detected randomly such as the edges within the water area. Thus, extracting consistent detections of a set of segmentations can give a better estimation of correct segmentation.

In this chapter, a new approach to capture the consensus segmentation information from a set of segmentations generated by varying parameters of different segmentation algorithms is proposed. Fig. 3.1 illustrates the proposed approach. First, an input image segmentation set is generated by varying parameters of four algorithms. Then, the probability of a segmentation curve present around a location \( x \) is estimated based on an image segmentation model and a probability map is constructed. By thresholding the probability map, a set of upper level sets of the probability map, which corresponds to the filtration for the topological persistence analysis, is generated. The persistent segments are then extracted by applying topological persistence to the probability map.
Finally, a robust segmentation is obtained with the guarantee of detection of certain segmentation curves.

The rest of this chapter is organized as follows: Section 3.1 introduces our image segmentation model; A detailed description of the probability map construction is introduced in Section 3.2; Section 3.3 discusses the persistence segmentation extraction and the choice of thresholds; Experiments of the proposed segmentation approach are discussed in Section 3.4; Finally, Section 4.5 concludes this chapter.

**Figure 3.1** Pipeline of the consensus-based image segmentation. (a) Original image. (b) Four samples from the segmentation set. (c) Probability map. (d) A set of upper level sets of the probability map. (e) Persistence diagram of the filtration shown in (d) where each point in the diagram corresponds to a connected component in the filtration. (f) Segmentation result.

### 3.1 Image Segmentation Model

Let $\Omega \subset \mathbb{R}^2$ be the image domain. We represent a segmentation $\mathcal{S}$ of an image as a set $X$ of nodes $x_i \in \Omega$, and a set $\Gamma$ of continuous curves $\gamma_{ij} : [0, 1] \to \Omega$ for which $\gamma_{ij}(0) = x_i$ and $\gamma_{ij}(1) = x_j$. Each curve is non-intersecting with any curve in the set and the number of curves incident with any point is greater than one, making this a valid segmentation. The segmentation $\mathcal{S}$ has a graphical structure corresponding to a planar graph $\mathcal{G} = (V, E)$ where $V = \{1, \ldots, N_X\}$, $N_X$ is the number of nodes in $X$, and $E = \{(i, j) | \gamma_{ij} \in \Gamma\}$. We define a segmentation $\mathcal{S}$ by the pair $(\mathcal{G}, \Gamma)$.

In order to define a probabilistic model for segmentations, the unknown generative model is
represented as:

\[ P[\mathcal{S} = (\mathcal{G}, \Gamma)] = P[\Gamma | \mathcal{G}] \cdot P[\mathcal{G}]. \]  

It is assumed that the graphical structure \( \mathcal{G} \) is obtained from an over-segmentation of the image with corresponding set of vertices \( V_o \) and edges \( E_o \). First, a subset of edges \( E \subset E_o \) is selected and the vertices \( V \subset V_o \) corresponding to the subgraph induced by \( E \) are chosen. We let

\[ p_{ij} := P[(i, j) \in E], \]  

which is a piece of information from this unknown distribution of graphical models that will be used in our study. The probabilities \( p_{ij} \) specify the probability of a curve segment to be present. We further assume that given a graphical structure, the set of curves \( \Gamma \) is generated using some process that does not violate the graphical structure (i.e., it does not introduce intercepts between the curves).

Let us define \( \mathcal{G}_\sigma \) as the graph obtained by selecting the set of edges \( E_\sigma = \{(i, j) \in E_o | p_{ij} > \sigma\} \) and corresponding nodes \( V_\sigma \). We assume that there exist constants \( \delta \) and an associated set \( \Gamma_\sigma \) of curves \( \gamma_{ij}^{\sigma} \) that satisfy

\[ |\gamma_{ij} - \gamma_{ij}^{\sigma}|_\infty \leq \delta \quad \text{for all} \quad (i, j) \in E_\sigma \cap E \]  

for any segmentation with graphical structure \( \mathcal{G} = (V, E) \), where \( |f_1 - f_2|_\infty = \sup_{s \in [0,1]} |f_1(s) - f_2(s)|_2 \).

This assumption requires the set of realizations of a given curve \( \gamma_{ij} \) to be concentrated within a band of radius \( \delta \) around the curve \( \gamma_{ij}^{\sigma} \). We define \( \mathcal{S}_\sigma = (\mathcal{G}_\sigma, \Gamma_\sigma) \).

Our objective is to come up with a procedure and a set of conditions under which the segmentation \( \mathcal{S}_\sigma \) can be estimated. In particular, we introduce a procedure under which we can construct a subset \( \mathcal{C} \subset \Omega \) such that \( \text{im}(\mathcal{S}_\sigma) \subset \mathcal{C} \), where \( \text{im}(\mathcal{S}_\sigma) = \bigcup_{(i, j) \in E_\sigma} \text{im}(\gamma_{ij}^{\sigma}) \) and \( \text{im}(\gamma_{ij}^{\sigma}) = \left\{ x = \gamma_{ij}^{\sigma}(s) \in \Omega | s \in [0,1] \right\} \).

Figure 3.2 Illustration of the model used for segmentations: The representation model for a segmentation \( \mathcal{S} \) (left) and the bounded perturbation model around \( \mathcal{S}_\sigma \).
Let \( D_\delta : \Omega \to [0, 1] \) be the function that measures the probability of having a segmentation overlapping with a ball \( B_\delta(x) \) of radius \( \delta \) centered at a point \( x \). That is,

\[
D_\delta(x) := P[im(S) \cap B_\delta(x) \neq \emptyset].
\] (3.4)

**Theorem 3.1.1.** The set \( \mathcal{C} = \{x | D_\delta(x) \geq \sigma\} \) satisfies:

\[
im(\mathcal{G}_\sigma) \subset \mathcal{C}
\] (3.5)

and

\[
im(\mathcal{G}_\sigma)^c \ominus B_{2\delta} \subset \mathcal{C}^c
\] (3.6)

where \( \mathcal{C}^c \) is the complement of set \( \mathcal{C} \) in \( \Omega \), \( B_{2\delta} \) is the ball of radius \( 2\delta \) and \( \ominus \) operator is the morphological erosion operator.

**Proof.** For the first inequality it is sufficient to show that every point in \( \mathcal{G}_\sigma \) has \( D_\delta(x) \geq \sigma \). Let \( x \in im(\mathcal{G}_\sigma) \). Then, we have that given \((i, j) \in E\) then \( B_\delta(x) \) will intersect with \( im(\mathcal{G}_{ij}) \) due to Eqn 3.3. Noting that \((i, j) \in E\) with probability greater that \( \sigma \) (by definition of \( S_\sigma \)) then \( D_\delta(x) \geq \sigma \).

For the second inequality, we note that any point \( x \) that is more than \( 2\delta \) from \( im(\mathcal{G}_\sigma) \) has \( D_\delta(x) = 0 \) since no curves will intersect with the ball \( B_\delta(x) \).

\[\square\]

### 3.2 Probability Map Construction

In this section, the construction of the probability map is described in detail. The probability map is constructed by using the connectivity of \( n \times n \) patches in image. The size \( n \) of the patch is a parameter associated with the perturbation bound \( \delta \) in the segmentation model. The effect of the patch size on the quality of the model approximation is also discussed in this section.

#### 3.2.1 Boundary Characterization

Given a set of segmentations \( \{\mathcal{G}_k, k = 1, \ldots, K\} \) generated by different segmentation algorithms with different parameters, we assume that the segment curves appearing in the set satisfy the probabilistic model we define in Section 3.1. That is, each segment curve in the segmentation results corresponds to an edge \((i, j) \in E_o\) of the over-segmentation.

In a segmentation \( \mathcal{G}_k \), an \( n \times n \) patch \( N(x) \) centered at \( x \) is called connected if all the pixels in the patch have the same label. Then the number of times it is disconnected over the set \( \{\mathcal{G}_k\} \), \( C_n^{dis} \), counts how many times one or more segment curves appear within that patch. As the examples shown in Fig. 3.3, a large patch may capture more than one edge in \( E_o \), while too small of a patch may miss some parts of the segment curves in \( E_o \). In Fig. 3.3 (b), the large brown patch covers two segment curves. The present probability of neither edge can be measured correctly by the patch of that size. The smaller blue patch is a suitable choice. In Fig. 3.3 (c) and (d), The small green patch cannot cover all the segment curves of the soft edge, which makes it have lower disconnection
probability than the actual value, whereas the blue patch is a suitable choice since it covers all the
possible positions of the curves. The choice of patch size $n$ will be discussed in Section 3.2.2. Based
on the segmentation model, we have $D_n^*(x) = C_n^{dis} / K$ as an estimate for $D_0(x)$. We assume that
segment curves of the desired segmentation boundary are more consistent than undesired ones
in set $\{S_k\}$ with a small perturbation around the correct boundary. Therefore, high values of $D_0(x)$
indicate the high confidence of a segment curve corresponding to $(i, j) \in E_o$ along the patch in the
groundtruth.

![Figure 3.3 Example of segment curves captured by patches. (a) An image with a soft edge and a sharp edge. (b)-(d) show three possible segmentation results. Red lines are the detected segment curves. Brown, blue and green squares are the patches used for probability map construction.](image)

Fig. 3.4 shows connection probability maps, $f(x) = 1 - D_n^*(x)$, of Fig. 3.1 (a) computed using
four different patch size $n = 3, 5, 7$ and 9. Darker color indicates the lower connection probability of
those regions. We note that all of these maps properly capture the contours of swan by assigning
them a low connection probability. The locations with high connection probabilities correspond
mostly to undesired segmentation curves. As the patch size increased, those curves become darker
and wider, as larger patch covers more segment curves which are further away from the center point.
Smaller patches can provide more details in the probability map than larger patch. In Fig. 3.4 (a)
and (b) ($n = 3$ and 5), the region of swan beak is clearly shown by the dark curves while in (c) and (d)
($n = 7$ and 9), the beak is filled by dark curves. The region highlighted by a red circle in Fig. 3.4 (a)
shows an example that segments curves of soft edges cannot be covered by a single small patch,
which leads to several lighter (higher connection probability) curves around the soft edge. Those
curves are almost removed when $n$ increases to 5.

### 3.2.2 Choice of Parameter $n$

As mentioned in Section 3.2.1, the quality of approximation of the segmentation model depends on
the parameter $n$. As shown in Fig. 3.9 (i)-(l), the connected components get larger and merged as $\sigma$
increases. Two connected components will merge under a certain $\sigma$ if there is a path of overlapped
patches connecting them. If $\gamma_{ij}$ for $(i, j) \in E_o$ is present between the two connected components
with probability $p_{ij}$, then $p_{ij}$ can be approximated by the minimum valued of $\sigma$ that merges the
In order to analyze the effect on the choice of the parameter $n$, consider an image with two segments separated by a soft vertical edge. Assume $\gamma_{ij}$ for $(i, j) \in E_o$ is present within a range of $2\delta$ pixels with probability $p_{ij}$ (i.e., a distance of $\delta$ from its centerline). Finding the minimum value of $\sigma$ that captures the separation between opposite sides of the regions separated by $\gamma_{ij}$ requires a value of $n$ that is big enough for this purpose. For the case $n - 2 \geq 2\delta$, the range of variations of the segments can be covered by a single $n \times n$ patch as shown in Fig. 3.5 (a). The pixel $x$ in the middle of the range has the disconnection probability $D_n^*(x) = p_{ij}$. The minimum probability threshold that makes the two segments connected is $\sigma = p_{ij}$. For the case $n - 2 < 2\delta$, a single $n \times n$ patch cannot cover the entire range, which makes the disconnection probability $D_n^*(x) < p_{ij}$. Note that the cross-sectional range can be covered by $m = \lceil \frac{2\delta + 1}{n - 1} \rceil$ patches overlapping with one pixel, where $[a]$ is the minimum integer greater or equal to $a$. Fig. 3.5 (b) shows one example of this case. Let $p_{dk}$ be the disconnected probability of $k$-th patch. The disconnection probabilities satisfy $\sum_{k=1}^{m} p_{dk} = p_{ij}$, where the sum is over the patches. There exists at least one pixel $x$ along the cross-sectional line for the range of variation of $\gamma_{ij}$ with $D_n^*(x) \geq p_{ij}/m$. The probability threshold that ensures that the
two segments are disconnected is $\sigma < p_{ij}/m$. Thus, if $n$ is chosen large enough, say $n - 2 \geq 2\delta$, then $(i, j) \in E_\sigma$ with $\gamma_{ij}$ appearing within range $2\delta$ in the set $\{S_k\}$ with probability $p_{ij}$ will be represented in all the labeled images with $\sigma < p_{ij}$. For $(i, j) \in E_\sigma$ with range $2\delta > n - 2$, they will be represented in the label image with $\sigma < \sigma'$, where $p_{ij}/m \leq \sigma' < p_{ij}$ and $m = \lceil \frac{2\delta + 1}{n-1} \rceil$.

Let us consider another case corresponding to two segment curves $\gamma_{ij}$ and $\gamma_{i'j'}$ where $(i, j)$ and $(i', j') \in E_\sigma$ are close to each other. An example is shown in Fig. 3.5 (c). The red dash lines represent the range for $\gamma_{ij}$ and the green is for $\gamma_{i'j'}$. Assume that, when $\gamma_{i'j'}$ is not present, $D^*_n(x) = \sigma_0$ where $x$ is the center of right patch in the figure and $\sigma_0$ is the minimum $\sigma$ making the two segments separated by red curve merge. After adding $\gamma_{i'j'}$, the probability of having a disconnected set increases, since $\gamma_{i'j'}$ is present in the area covered by the right patch. That is, $D^*_n(x) > \sigma_0$. Therefore, $\gamma_{i'j'}$ near $\gamma_{ij}$ may make the minimum connection threshold for $\gamma_{ij}$ greater than its actual value. For an $n \times n$ patch, segment curves which have perturbation regions with ranges that are at least $n - 1$ pixels away from each other can be identified without the influence of other segment curves.

![Figure 3.5](image.png)

**Figure 3.5** Illustration of the effect of the choice of $n$. Vertical dash lines with the same color indicate the variation regions of one segment curve $\gamma$. (a) The case of $n - 2 \geq 2\delta$. The variation region can be covered by one patch. (b) The case of $n - 2 < 2\delta$. The variation region can be covered by four patches in this example. (c) The case of closed segment curves influence the approximation.

### 3.3 Extract Persistence Segmentation

When applying a threshold $\sigma$ to $D^*_n(x)$, a labeled image $L(n, \sigma)$ is obtained by computing the connected components of the set $\mathcal{L}_\sigma^n = \{x | D^*_n(x) \leq \sigma\}$. Changing $\sigma$ from $\sigma_{min} = 0$ to $\sigma_{max} \leq 1$ produces a set of labeled image $\{L(n, \sigma) | \sigma \in [0, \sigma_{max}]\}$. 

32
3.3.1 Disconnection Threshold

The threshold $\sigma$ is used to get the segment curve $\gamma_{i,j}$ with probability $p_{i,j} \geq \sigma$. As discussed in Section 3.2.2, the quality of the approximation of the segmentation model is based on the size $n$ of patch. In order to capture all the segment curves that are present with probability greater than $p_{i,j}$, given that we only care about the curves appearing within a range $2\delta \leq n - 2$, we can set the threshold $\sigma = p_{i,j}$. If we consider the curves that appear within a larger range $2\delta > n - 2$, the threshold should be $\sigma = p_{i,j}/m$, where $m = \lceil \frac{2\delta + 1}{n - 1} \rceil$.

3.3.2 Persistence Diagram

If a single value of $\sigma$ is used for thresholding, the segment curves are estimated by only one labeled image $L(n, \sigma)$ and we throw out the information of other labeled images in the set. Also, the ideal value of $p_{i,j}$ and $\delta$ to get a reasonable segmentation may be different due to the variations of the image quality over the dataset. Images with blurred edges require large $\delta$ and low $p_{i,j}$ while images with sharp edges require small $\delta$ and have high $p_{i,j}$. Furthermore, the connected components we want to extract correspond to the peaks of the connectivity probability map $1 - D^*_n$, simply thresholding $D^*_n$ may not capture all the desired regions. To address this, we apply topological persistence to generate more robust segmentation estimation.

In order to extract the persistent connected components for our segmentation process, we first define a connection probability map $f(x) = 1 - D^*_n(x)$. Then, the labeled image set $\{L(n, \sigma)\}_{\sigma \in [0, \sigma_{max}]}$ form a filtration of the upper level set of $f$.

Fig. 3.6 shows the filtration at $\sigma = 0, 0.25, 0.5, 0.75$ and 1 of each connection probability map shown in Fig. 3.4. Different colors indicate different connected components. When increase $\sigma$ during filtration, the connected components become larger and those which are separated by curves of low disconnection probability get merged first. When $\sigma$ increases from 0.25 to 0.5 for $n > 3$, the undesired segmentation curves are almost removed. The connected components appearing at high $\sigma$ values indicate high probability of segment curves between them. As shown in Fig. 3.6, the connected components appear at $\sigma = 0$ are the area with connection probability 1. That is, segment curves never appear in those areas among the input segmentation set. At high values of $\sigma$, only the connected components separated by segment curves with high probability of being present among the input segmentation set are shown, such as the body of swan and the area of water. At $\sigma = 1$, all the connected components are merged to a single region. For different patch size $n$, at the same $\sigma$, the connected components get smaller and die later as $n$ increased. When $n = 3$, the swan body and the water area merge together at a small $\sigma$ before 0.5. However, those two connect components are still separated at $\sigma = 0.75$ for $n = 7$ and 9. The filtration of different patch size $n$ satisfies the following condition:

**Theorem 3.3.1.** Let $\mathcal{L}_\sigma^n$ be the filtration at $\sigma$ using patch size $n$, then

$$\mathcal{L}_\sigma^n \subseteq \mathcal{L}_\sigma^{n_2},$$

(3.7)
when \( n_1 > n_2 \).

**Proof.** The filtration is the upper level set of \( f_n(x) = 1 - D_n^*(x) \) and \( f_{n_1}(x) \leq f_{n_2}(x) \) for all pixels \( x \) in the image, if \( n_1 > n_2 \). Then \( \mathcal{L}_\sigma^{n_1} \subseteq \mathcal{L}_\sigma^{n_2} \). \( \square \)

Then the persistence diagram is obtained by encoding the birth and death time of all the connected components during filtration. Persistence diagrams computed from probability map shown in Fig. 3.4 using patch size \( n = 3, 5, 7 \) and 9 are shown in Fig. 3.7 (c). When \( n = 3 \), all of the connected components have persistence less than 0.6 and gather in low persistence region except the one with the highest persistence. For larger patch size, more connected components get higher persistence and persistence connected components are more separated from noise regions which have low persistence interval, especially for \( n = 9 \). However, this the patch size 9 is too large to capture some small details as shown in Fig. 3.4. Thus, \( n = 5 \) and 7 are suitable choice which are able to obtain desired persistence diagram and keep enough details.

![Figure 3.6](image)

**Figure 3.6** Filtration at \( \sigma = 0, 0.25, 0.5, 0.75 \) and 1 of each connection probability map shown in Fig. 3.4. The regions with \( f(x) < 1 - \sigma \) are labeled using black color.

### 3.3.3 Persistence Region Extraction

To obtain the persistent segmentation, only the regions with persistence intervals greater than a threshold \( \tau \) are kept to avoid the connected components generated by noise in the segmentation set \( \{S_k\} \). This threshold is illustrated by red dash lines parallel to the diagonal in Fig. 3.7 (c). The size of each region above the persistence threshold can vary, since it exists over a range of \( \tau \). To get the
largest size of the persistence region, the largest set of points which is associated with its death time is selected.

One advantage of the persistence diagram is its stability property [CS07]. Small changes in the function $f$ lead to small changes in the persistence diagram. This translates into the following for our scenario: we can obtain segmentation result that is robust to parameter value changes and small variations in the segmentation set $\{S_k\}$.

To make sure the segment curves $\gamma_{ij}$ present with probability greater than $p_{ij}$ within a range $2\delta \leq n - 2$ can be captured, we can set $\sigma_{\text{max}} = p_{ij}$. This ensures that regions with death time after $p_{ij}$ will never merge with other regions in filtration $\{L(n, \sigma)\}_{\sigma \in [0, p_{ij}]}$. This threshold is shown as the horizontal red dash line in Fig. 3.7 (c). This is equivalent to removing the labeled images with $\sigma > p_{ij}$ from the filtration. For this new diagram, the connected components separated by those segment curves will never get merged in the filtration and will be present in the final segmentation result. Note that this is not the same as simply thresholding $f(x) = 1 - D^*(n(x))$ by $\sigma = p_{ij}$, which will remove the regions with death before $\sigma$ as well as those that are born after $\sigma$.

Fig. 3.7 (a) and (b) shows the persistence regions extracted from persistence diagrams using $\sigma_{\text{max}} = 0.8$ and the best persistence threshold $\tau$ for each patch size $n$. The areas for segment curves being present with probability greater than 0.8 within a range $2\delta \leq n - 2$ are labeled with black color. Since the disconnection probability $D^*_n(x) \geq D^*_n(x)$, when $n_1 > n_2$, those areas get wider as $n$ increased. To obtain the similar persistence regions, smaller persistence threshold $\tau$ is used for smaller $n$. However, some noise regions are introduced if $n$ is too small. When $n = 3$, the desired regions and noise regions have similar persistence interval and get mixed up in the diagram. The best persistence threshold for $n = 3$ will provide desired persistence regions as well as noise regions, such as the small regions on the left of the swan neck. Those regions are removed when $n \geq 5$, since desired regions and noise regions are separated in the diagram in persistence direction. Also, the persistence regions obtained using $n = 7$ and 9 show examples that large $n$ fails to capture small details of the image. The swan beak can not be separated from other region using any of the persistence threshold.

### 3.3.4 Edge Position Estimation using Region Growing

In Section 3.3.3, persistence regions as well as the areas with segment curves being present within a range with probability greater than a threshold $\sigma_{\text{max}}$ are extracted. Since our aim is segmentation, we want not only the range of edges but also to estimate the position of edges. To obtain the segmentation, color-based region growing is used.

The color distance between a persistence region $k$ and its 8-neighborhood pixel $x$ is defined as

$$d = \sum_{i=1}^{3} |\bar{C}_i(k) - C_i(x)|,$$

where $i = 1, 2, 3$ are RGB channel of the color image, $\bar{C}_i(k)$ is the average color of persistence region $k$ in channel $i$, and $C_i(x)$ is the value of $i$th channel of pixel $x$. At each distance step $d_j$, persistence
Figure 3.7 Persistence diagrams using patch size $n = 3, 5, 7$ and $9$ (from top to bottom) and corresponding extracted persistence regions. The areas for segment curves being present with probability greater than 0.8 within a range $2\delta \leq n - 2$ are labeled with black color in column (a) and (b). (a) Labeling image. Different colors indicate different regions. (b) Labeling image labeled with the mean color of each region. (c) Persistence diagrams. Read dash lines parallel to the diagonal are the persistence thresholds $\tau$. From top to bottom: $\tau = 0.25, 0.40, 0.45$ and $0.55$. The horizontal red dash lines are $\sigma = 0.8$ for all of the four diagrams.

regions merge their neighboring pixels with color distance $d < d_j$ and update the average color for the new merged region. Then increase the distance by 1 and merge more neighboring pixels until all the pixels in the image have labels.

Fig. 3.8 shows the segmentations obtained by region growing from persistence regions in Fig.
3.7. The black area are labeled by persistence regions with similar color and position of edges is estimated within those areas.

Figure 3.8 Segmentation results after region growing using patch size $n = 3, 5, 7$ and $9$ (from top to bottom). (a) Labeling image. Different colors indicate different regions. (b) Labeling image labeled with the mean color of each region. (c) Segmentation with edges.

3.4 Experiment

The proposed approach is implemented in MATLAB. We use images from the Berkeley Segmentation Database [Mar01; Arb11] to test our algorithm. There are five groundtruth segmentations per image labeled by different human subjects.
We adapt the segmentation coverage score defined in [Arb11] and variation of information defined in [Mei05] to evaluate the segmentation result. The overlap score between regions \( R \) and \( R' \) is defined as
\[
O(R, R') = \frac{|R \cap R'|}{|R \cup R'|} \tag{3.9}
\]
where \( |R| \) is the area of \( R \). The coverage score which is the covering of a segmentation groundtruth \( \mathcal{G}_r \) by a segmentation \( \mathcal{G}' \) is defined as:
\[
C(\mathcal{G}' \rightarrow \mathcal{G}_r) = \frac{1}{N} \sum_{R \in \mathcal{G}_r} |R| \max_{R' \in \mathcal{G}'} O(R, R'), \tag{3.10}
\]
where \( N \) is the number of pixels in the image.

The variation of information measures the amount of randomness in one segmentation which cannot be explained in another segmentation [Mar01; Arb11]. Let \( P(k) \) denote the probability that a pixel being in region \( R_k \) in a segmentation and \( P(k', k_r) \) represent the probability that a point belongs to region \( R'_k \) in segmentation \( \mathcal{G}' \) and region \( R_{k_r} \) in segmentation \( \mathcal{G}_r \). Using the same notation as coverage score, these two probabilities can be written as
\[
P(k) = \frac{|R_k|}{N} \tag{3.11}
\]
\[
P(k', k_r) = \frac{|R'_k \cap R_{k_r}|}{N} \tag{3.12}
\]
Then the variation of information can be defined as
\[
I(\mathcal{G}', \mathcal{G}_r) = \sum_{k=1}^{K'} \sum_{k_r=1}^{K} P(k', k_r) \log \frac{P(k', k_r)P(k_r)}{P(k')} \tag{3.13}
\]
where \( K' \) and \( K \) are number of regions in segmentation \( \mathcal{G}' \) and \( \mathcal{G}_r \), respectively.

To combine the information of five groundtruth segmentations, the coverage score and variation of information for segmentation \( \mathcal{G}' \) is defined by
\[
C(\mathcal{G}') = \frac{1}{5} \sum_{k=1}^{5} C(\mathcal{G}' \rightarrow \mathcal{G}_r^k) \tag{3.14}
\]
\[
I(\mathcal{G}') = \frac{1}{5} \sum_{k=1}^{5} I(\mathcal{G}', \mathcal{G}_r^k) \tag{3.15}
\]
where \( \mathcal{G}_r^k \) is the \( k \)-th groundtruth segmentation.

For coverage score, higher score indicates better performance and for variation of information, smaller score is better.
3.4.1 Generation of Segmentation Set

We use segmentation results generated by four algorithms as our input. These four algorithms are SAS [Li12], Normalized Cuts [SM00], Graph-based segmentation [FH04] and Mean Shift [CM02]. The first two algorithms require the number of regions in the segmentation result as an input parameter. We get 26 segmentations from each of these two algorithms by varying the number of regions from 5 to 30. The Graph-based segmentation uses a Gaussian with standard deviation $\sigma$ to remove the digitization artifacts and a parameter $k$ to control the scale of observation which affects the size of the segments. We vary $\sigma$ from 0.4 to 0.8 with step size 0.1 and $k$ from 500 to 5000 with step size 100 to generate 230 segmentations. For Mean Shift, we generate 238 segmentations by changing the spatial search window size and bandwidth of the search window from 2 to 15 with step size 1, and from 7 to 15 with step size 0.5, respectively. These range of parameters were chosen because they generate reasonable but different segmentations for most of the test images.

When computing the disconnection probability map $D_n^*$, we first compute $D_n^*, i$ where $i = 1, 2, 3, 4$ for the $i$-th algorithm and $D_n^*$ is obtained by taking the average of the $D_n^*, i$. This process equally weights the contribution from each algorithm.

3.4.2 Results

![Figure 3.9](image)

Figure 3.9 (a)-(d) Results with best coverage score for base segmentation approaches in the input segmentation set: SAS [Li12], Normalized Cuts [SM00], Graph-based segmentation [FH04] and Mean Shift [CM02]. (e) Original image. (f) Connection probability map. (g) Persistence diagram of the connected components during filtration. The thresholds are $\sigma_{max} = 0.8$ and $\tau = 0.4$. (h) Our segmentation result. (i)-(l) Filtration with $\sigma = 0, 0.25, 0.5, 0.8$. With $\sigma$ increasing, the regions separated by segment curves with low probability are merged early.
Figure 3.10 Best segmentation results of the four base-algorithms in the segmentation set and result of proposed algorithm. Left to right: Original image, SAS \cite{Li12}, Normalized Cuts \cite{SM00}, Graph-based segmentation \cite{FH04}, Mean Shift \cite{CM02}, connection probability map, and proposed algorithm.

In our experiment, we use \( n = 5 \) as the patch size. The filtration is generated by varying \( \sigma \in [0, \sigma_{\text{max}}] \) with step size 0.1. The thresholds we use are \( \sigma_{\text{max}} = 0.8 \) and \( \tau = 0.4 \). Fig. 3.9 shows one segmentation result of the proposed approach. Fig. 3.9 (a)-(d) shows the best segmentation results in the segmentation set of the four algorithms mentioned in Section 3.4.1 based on the coverage score. As we can see, the result of the best score, 0.92, has some small noisy regions and it is not visually better than the result of SAS with score 0.90. This leads us to believe that the coverage score may not be the best evaluation metric in this scenario.

Fig. 3.9 (f) shows the connection probability map \( 1 - D_n^* \). This map properly captures the contours of the hawk and the branch by assigning them a low connection probability. The locations with high connection probabilities correspond mostly to undesired segmentation curves.

Fig. 3.9 (i)-(l) show the labeled images for \( \sigma = 0, 0.25, 0.5, 0.8 \). For low \( \sigma \), most of the segment curves appearing in the input segmentation set are captured. However, when \( \sigma \) increases from 0.25 to 0.5, the undesired segmentation curves are removed. For \( \sigma_{\text{max}} = 0.8 \), almost all the remaining segment curves are visually correct.

Fig. 3.9 (g) shows the persistence diagram during filtration. The thresholds are \( \sigma_{\text{max}} = 0.8 \) and \( \tau = 0.4 \). Connected components corresponding to the points in the blue region are selected as our segmentation result. Since the size of persistent connected components can vary in a range of \( \tau \), we select the set of points associated with the death time as the segmentation result. This maximizes the size of the \( k \)-th persistent region and captures the curves with probability greater than 0.8.

Fig. 3.9 (h) shows our segmentation result. It is observed that this approach maintains the segmentation curves that are visually desirable.

Fig. 3.10 shows more results of our proposed approach, which is able to extract the segment curves that appear with high probability in the input segmentation set. The coverage score of the...
Table 3.1 Performance evaluation of the proposed method (Consensus-based) against input methods over 20 images from the Berkeley Segmentation Database

<table>
<thead>
<tr>
<th>Methods</th>
<th>Graph</th>
<th>NCuts</th>
<th>SAS</th>
<th>Mean Shift</th>
<th>Consensus-based</th>
<th>Consensus-based</th>
<th>Consensus-based</th>
<th>Consensus-based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>τ = 0.30</td>
<td>τ = 0.35</td>
<td>τ = 0.40</td>
<td>τ = 0.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSC</td>
<td>0.5359</td>
<td>0.3970</td>
<td>0.5325</td>
<td>0.5540</td>
<td><strong>0.5982</strong></td>
<td><strong>0.6085</strong></td>
<td><strong>0.5725</strong></td>
<td><strong>0.5731</strong></td>
</tr>
<tr>
<td>VoI</td>
<td>2.1128</td>
<td>2.3515</td>
<td>1.8251</td>
<td>1.8946</td>
<td><strong>1.7070</strong></td>
<td><strong>1.6700</strong></td>
<td><strong>1.7600</strong></td>
<td><strong>1.7930</strong></td>
</tr>
</tbody>
</table>

proposed approach is not better than the best score in the segmentation set. However, our results are obtained by fixed parameters, which the best results of input segmentation set are obtained by different algorithms and parameters. Also, the proposed approach tends to segment single object into smaller number of regions which may be divided into more regions by the base approaches. This is because the edges are randomly detected inside an object due to the similarity within the region of that object. For the same reason, we can remove the curves generated in the background. However, if an undesired curve is always detected by all the algorithms, our approach will fail since our model assumes that frequently detected curves are valid segmentation curves.

To quantitatively evaluate our segmentation performance, we compute the coverage score and variation of information of four input segmentation algorithms and our proposed method for 20 images from the Berkeley Segmentation Database. For input segmentation algorithms, we compute the scores for all of the parameters in the input segmentation set. Then find the parameter providing the best average score. For the proposed method, we fix \(\sigma_{\text{max}} = 0.8\), and change the persistence threshold \(\tau\) from 0 to 1 with step size 0.5 to compute the average scores of 20 images for each threshold value. Table 3.1 shows the best average scores of four input segmentation algorithms and proposed method with persistence threshold \(\tau\) from 0.30 to 0.45. RSC denotes coverage score and VoI is variation of information. Note that, for the input segmentation algorithms, the best parameter for coverage score is not necessarily the same as the the best parameter for variation of information. But the scores of our method shown in the same column in the table are obtained by the same parameter. We can see, with persistence threshold range from 0.30 to 0.45, our method obtains better performance than all the input segmentation algorithms. Thus, our consensus-based segmentation is able to extract consensus information from input segmentation set and obtain better segmentation results.

3.5 Conclusion

In this chapter, we propose a new approach to capture the consensus segmentation information from a set of segmentation results generated by varying parameters of different algorithms. The probability of a segmentation curve being present is estimated based on our probabilistic image segmentation model. A connection probability map is constructed to characterize the segmentation curves with high probability. Then, persistence segments are extracted by applying topological persistence to the probability map. Finally, a robust segmentation is obtained with the detection of
certain segmentation curves guaranteed. The experiments demonstrate our algorithm is able to capture the curves present consistently within the segmentation set.
In recent years, there has been significant development in the sector of intelligent driver assistance systems and autonomous vehicles in order to enhance safety by monitoring the on-road environment. One of the foremost issues that needs to be addressed for these Advance Driver Assistance Systems (ADAS) is the interpretation of surroundings of the ego vehicle, i.e. dynamic scene analysis and on-road obstacle detection. Recently, vision-based approaches for traffic-scene analysis have become an increasingly popular research area. In this chapter, we focus on the problem of segmentation of obstacles in a disparity map obtained from a stereo-vision system. Real-time computation of dense disparity maps [VDMG06] has made stereo-vision an alternative to higher cost LIDAR systems. A driving scene analysis system typically extracts the road surface, segments the obstacles from the road, and computes the position of the obstacles relative to the ego vehicle.

The current state-of-the-art for solving this problem relies on parameter values that are carefully selected in order to provide good performance. Often these parameters rely on threshold values for segmentation of images. However, sensitivity of the detection results to these parameters can be a big concern. Small changes in threshold values can lead to large variations in the segmentation of the images. In this chapter, we introduce an approach for robust obstacle detection based on persistent topology.

Fig. 4.1 provides an overview of the proposed methodology. This work builds on recent advances
Figure 4.1 Proposed robust obstacle detection. (a) Left image of stereo pair image. (b) Disparity map. (c) Disparity map after ground segmentation. (d) Occupancy grid. (e) Persistence diagram according to the occupancy grid. (f) Robust segmentation in U-disparity space. (g) Robust segmentation results.

in ground segmentation and obstacle detection for safe autonomous driving. In particular, we employ an efficient UV-disparity approach to classify a 3D traffic scene into a ground plane and obstacles. Then, an occupancy grid, which is a probability map of obstacles, is constructed to enable segmentation of the obstacles. Finally, we apply a topological persistence technique on the occupancy map to perform robust obstacle detection. The analysis of this method shows how this approach improves robustness of the detection results.

A fundamental difference between our approach and other existing approaches is its hierarchical nature. Topological persistence-based segmentation does not rely on a single threshold value; instead, it keeps track of all the clustering results, corresponding to different thresholds. This is essentially providing a hierarchical clustering. A key advantage of our approach over traditional techniques is that the algorithm can be tuned for better performance through a few intuitive parameters leading to results that are less sensitive than simple thresholding. The proposed method also provides us with a persistence diagram that gives a compact visual representation of segmentation result corresponding to different threshold values. By analyzing this diagram, we can choose meaningful merging parameters for our segments and can also get a sense of the stability of the number of clusters under different choices of persistence parameters.

The remainder of this chapter is organized as follows: Section 4.1 introduces the probabilistic occupancy map computation based on UV-disparity map. A detailed description of our methodology
using persistence for obstacle segmentation is introduced in Section 4.2. Section 4.3 discusses the robustness of the segmentation by analyzing the effect of parameters. Results of our obstacle segmentation method are discussed in Section 4.4. Finally, Section 4.5 summarizes this chapter.

4.1 Probabilistic Occupancy Grid Computation

The occupancy grid for our application is a probability map of obstacles for each position in u-disparity space. This section presents the occupancy grid computation based on prior work in the literature [Per12], which makes use of the UV-disparity map methodology.

4.1.1 UV-Disparity Map

We assume that a disparity map has been computed from a stereo pair of images from the vehicle. In our experiments, we make use of the Semi Global Block Matching (SGBM) technique [Hir08]. This map can also be represented as a 3D point cloud where each point has coordinates \((u, v, d)\), where \((u, v)\) represent the x-y coordinates in the image domain and \(d\) represents the disparity value. By projecting all the points to the ud-plane and the vd-plane and accumulating the overlapping points, we generate two new 2D images, called the u-disparity and the v-disparity [HU05] maps, respectively. Fig. 4.2 illustrates these maps for a stereo pair from the KITTI dataset [Gei13].

![Disparity maps](image)

**Figure 4.2** Disparity maps. (a) Disparity map in image space. (b) U-disparity map. (c) V-disparity map.
4.1.2 Ground Fitting

The ground plane is computed by fitting a plane in the \(u \nu d\)-space, where \(p = (u, v)\) corresponds to the pixel location and \(d\) to the associated disparity value. For the fitting, we make use of a point cloud constructed by selecting the global minimum of the matching cost function \(C_p\) as a corresponding disparity value of the pixel. RANSAC [FB81] is used for a robust fitting of the plane.

We parameterize the plane using the form \(v = g_0(u, d)\) since we will be considering bounds on the heights of obstacles in the physical space for their detection. In particular, we will only consider candidate points \((u, v, d)\) that satisfy the constraint

\[
g_0(u, d) + \frac{a_u h_l}{a_v b} \cdot d \leq v \leq g_0(u, d) + \frac{a_u h_u}{a_v b} \cdot d
\]

(4.1)

where \(a_u\) and \(a_v\) are the intrinsic focal length parameters of the camera in pixels, \(b\) is the distance of the baseline of the stereo camera system, \(h_l\) and \(h_u\) are the heights above the estimated ground plane used to identify obstacle pixels. In our experiment, we use \(h_l = 200\) and \(h_u = 1700\) for KITTI dataset. Using this setup, the ground plane for ground segmentation is 0.2 meters higher than the fitting plane to make sure we remove all the ground pixels, and we cut everything above 1.7 meters from the fitting plane to avoid the influence of obstacles above the car. Then the height for obstacle detection is around 1.5 meters in real world space. Fig. 4.1 (c) shows one example of ground segmentation result.

4.1.3 Occupancy Grid Computation

A probabilistic occupancy map in the \(u\)-disparity domain [Per12] is computed in order to enable the segmentation of objects. This map is obtained by using only those points in the disparity map that are above the ground plane and below a plane of height \(h_{max}\), which specifies a maximum detection height for our approach.

Given a site \(s\) with coordinates \((s_u, s_d)\) in the \(u\)-disparity domain, we define two binary random variables \(V_s\) and \(C_s\) which represent the visibility of this site \((V_s = 1\) means that the site is visible) and the obstacle confidence \((C_s = 1\) means that an obstacle is seen at \(s\)). The probability of occupancy, \(O_s\), at site \(s\), can be represented as

\[
P(O_s) = \sum_{\nu, c} P(V_s = \nu, C_s = c) \cdot P(O_s | V_s = \nu, C_s = c).
\]

(4.2)

We assume independence between \(V_s\) and \(C_s\), which leads to the need for expressions for \(P(V_s = \nu)\) and \(P(C_s = c)\). In order to compute these probabilities, let \(N_P(s)\) be the total number of measured points in image domain at site \(s\), \(N_O(s)\) is the total number of obstacle points, and \(N_V(S)\)
is the total number of visible points. These three quantities can be obtained from

\[ A_P(s) = \{(u, v)|u = s, v \in [g_{lower}(s_d), g_{upper}(s_d)]\} \]
\[ A_O(s) = \{(u, v)|I_D(u, v) = s_d\} \cap A_P(s) \]
\[ A_V(s) = \{(u, v)|I_D(u, v) \leq s_d\} \cap A_P(s) \]

(4.3)

which leads to \( N_a(s) = |A_a(s)| \), where \( a \in \{P, O, V\} \), \( I_D \) is the disparity map, \((u, v)\) is a coordinate in the disparity image domain, \( g_{lower} \) and \( g_{upper} \) represent the \( v \)-coordinates of the pixels on the ground and the maximum height plane, as functions of the disparity values (i.e., these are the equations defining the lower and upper planes for obstacle detections).

The probability of visibility for the site \( s \) in \( u \)-disparity space is then defined as

\[ P(\bar{V}_s = 1) = \frac{N_V(s)}{N_P(s)}, \]  
(4.4)

and the probability of confidence of observation as

\[ P(\bar{C}_s = 1) = 1 - e^{-\lambda} \frac{N_{O(s)}}{N_{P(s)}} \], \]
(4.5)

where \( \lambda \) is a constant parameter.

Then define \( P(O_s|\bar{V}_s = v, \bar{C}_s = c) \) as

\[ P(O_s|\bar{V}_s = 0, \bar{C}_s = c) = 0 \quad \forall c \in \{0, 1\} \]
\[ P(O_s|\bar{V}_s = 1, \bar{C}_s = 1) = 1 - P_{FP} \]
\[ P(O_s|\bar{V}_s = 1, \bar{C}_s = 0) = P_{FN} \]

(4.6)

with \( P_{FP} \) and \( P_{FN} \) representing the false positive and false negative probabilities of occupancy detection respectively. Here we define \( P(O_s|\bar{V}_s = 0, \bar{C}_s = c) = 0 \) while it is 0.5 in [Per12]. Since our aim is the segmentation in 2D image space, we do not care about the invisible area. Also, the probability of invisible area in new occupancy map is 0, which makes it easier to separate individual obstacles. Let \( P_{FP} = 0.01 \) and \( P_{FN} = 0.05 \), the modified occupancy maps are shown in Fig. 4.3 (b). Some corresponding regions in between the grayscale image and the occupancy map are highlighted. The advantage of the new occupancy map will be discussed again in Section 4.3.

### 4.2 Persistence Regions Extraction

Let us begin by defining

\[ f(s) = P(O_s) \]
(4.7)

as the probability of occupancy function over the \( u \)-disparity space in order to draw a connection with the concepts introduced in Section 2.1.

Segmentation of \( f \) via simple thresholding is fast and easy to implement. However, the proper threshold value may not be easily selected due to variations on the probability map attributed
Figure 4.3 Occupancy grid. (a) Grayscale image. (b) Modified occupancy grid.

Figure 4.4 Birth and death of connected components during filtration. The first row shows a part of the original image and the corresponding probability of occupancy map. Second row shows a connected component in cyan which corresponds to a car in the image space for five different values of $\tau$, and third row shows the corresponding regions in image space.

to the quality of the disparity map. The latter is affected by external and internal factors such as illumination and the texture of objects. Thus, the ideal threshold may change between images, even in the same video sequence. This simple type of segmentation is very sensitive to the choice of threshold value. As we will see in the experimental section, small variations in the threshold value can lead to a large number of regions introduced and removed. Furthermore, obstacles are associated with peaks in probability of the occupancy map $f$, for which there may not be a single threshold value that includes all these peaks without merging obstacle regions that are not supposed
Figure 4.5 Persistence diagram corresponding to Fig. 4.4. Each point indicates the lifespan of a connected component. The red line is the threshold line \( \gamma = 0.5 \). All the regions above this line have persistence interval greater than 0.5.

to be merged. In order to address all of these issues, we make use of topological persistence to generate a more robust segmentation.

Fig. 4.4 illustrates the birth and death process of connected components during the filtration of upper level sets of \( f \). At \( \tau = 0.02 \), the cyan region is born. Another region in red is also born at \( \tau = 0.03 \) but dies at \( \tau = 0.1 \), because it merges with the cyan region which has an earlier birth time. The persistence interval of the red region is 0.01. At \( \tau = 0.53 \), the cyan region is still alive and its area increases. At \( \tau = 0.64 \), this region dies leading to a persistence interval of length 0.44. Note that by choosing regions with a persistence interval length greater than \( \gamma = 0.5 \), the cyan region would be selected, while the red region would be removed.

The birth and death of all regions obtained from \( f \) are captured in the persistence diagram, as displayed in Fig. 4.5. Each point in the persistence diagram represents the lifespan of a region. Note that this diagram contains hierarchical information about the merging of these regions as a function of \( \tau \). In order to obtain a robust segmentation of the obstacles, we keep only those regions with persistence interval greater than \( \gamma = 0.5 \). This bound is illustrated by a red line in Fig. 4.5. Finally, in order to obtain a labeling of the clusters in the u-disparity map, we need to determine the support of the selected persistent regions. However, since a region can exist over a range of values of \( \tau \), its size can vary. In this case, we select the largest set of points for each region which is associated with the value of \( \tau \) before the region dies. The only problem with this assignment is that the support of the regions may be overlapping, in which case a point in the u-disparity space is assigned to the region with the lowest death time.
Fig. 4.6 (a) illustrates the clustering result in u-disparity space for $\gamma = 0.5$. Each obstacle detected in the u-disparity domain needs to be mapped back to the original image plane for display. This is done by taking a site $s$ in the u-disparity space and identifying all points in the image domain with u-coordinate $s_u$ and disparity $s_d$ with the object label assigned to $s$. The corresponding segmentation result is shown in Fig. 4.6 (b).

(a)

(b)

Figure 4.6 Segmentation results. (a) Clusters in u-disparity space, and (b) their corresponding regions in image space. Different colors indicate different obstacles. Ground is labeled as green.

4.3 Effect of Parameters

In this section, the effect of two parameters, probability of invisible obstacles and persistence bound, is discussed.

4.3.1 Probability of Invisible Obstacles

In Section 4.1, $P(O_s|V_s = 0, C_s = c)$ is defined to be 0 instead of 0.5 as the original previous work in [Per12]. Fig. 4.7 (b) and (c) show the original occupancy map and modified occupancy map. Fig. 4.3 (d) and (e) shows the corresponding occupancy area for the vehicle on the right of the image which is close to the camera. In (b), the invisible areas have occupancy probability around 0.5, and in (c), the occupancy probability of those areas is closed to 0, which makes obstacles clearly shown in the probability map. As we can see, in the original occupancy map, the back windshield of the vehicle has a low probability of occupancy, which is even lower than the invisible area behind the vehicle. The low probability is because the surfaces of that part are not vertical. Then points of that
surface are dispersed over various sites on the u-disparity map. During filtration, the other part of the vehicle will first merge to the background behind the vehicle and then merge with the back windshield part, which leads to incorrect segmentation when applying topological persistence. In the modified occupancy map, the vehicle will first merge together to be one connected component then merge with the background at a later time. Fig. 4.8 shows segmentation of the vehicle on the right side of the image shown in Fig. 4.3 (a). Left column shows the results of original occupancy while right column shows the results of modified occupancy map. First row are segmentation results using persistence threshold one step size before the second row. The second row are the results showing parts of the vehicle merge with background or merge together. For the case in left column, it cannot obtain all the parts of vehicle as an individual region in the segmentation result, since part of the vehicle merges with background before all parts of the vehicle merge together. However, in right column, the vehicle can be separated individually as shown in (d).

Another difference between these two occupancy maps that is worth mentioning is illustrated in Fig. 4.9. Column (a) and (b) are persistence diagrams computed from original and modified occupancy maps, respectively. Red points are the detected obstacles in image space. Some points with high enough persistence are not in red, as the persistence regions are computed in u-disparity space and there are some regions invisible in image space. In the diagrams of column (b), most of the red points concentrate in the top-left of the diagram, which makes it easy to determine the persistence threshold and obtain a more accuracy segmentation.

### 4.3.2 Effect of Persistence Bound

The persistence bound $\gamma$ is used to select the most prominent regions in the hierarchical clustering of the data. This selection process can be visualized in the persistence diagram as selected features above a particular line (see Fig. 4.5 for an example). As the value of $\gamma$ increases, the line moves up, allowing fewer but larger regions to be selected. This is due to the merging of some of these regions. During this process, obstacles which are close to each other in the image space will get merged first.

Fig. 4.10 illustrates the changes in the segmentation results as $\gamma$ increases. When $\gamma = 0.3$, we can see that trees on the left side are divided into several small regions. When it increases to 0.6, trees on the left are merged into one region. The two vehicles are always detected and segmented properly between 0.3 and 0.6.

### 4.4 Experiments

The entire process is implemented in MATLAB on a 3.4 GHz computer with 16GB RAM. It takes approximately 9.5 seconds for a $1242 \times 375$ image for $\tau \in [0, 1]$. We make use of stereo image pairs from the KITTI Vision Benchmark Suite [Gei12; Gei13; Fri13] for analysis. For Occupancy grid computation, $P_{FP} = 0.01$, $P_{FN} = 0.05$, and $\lambda = 10$. The persistence analysis is performed by varying $\tau \in [0, 1]$ and letting $\gamma = 0.45$ unless otherwise specified. For both the thresholding method and the proposed persistence analysis method, a simple morphological post-processing step is applied to
the segmented images in order to remove the small detection regions as well the small gaps in the detection regions.

In this section, we first compare the persistence method with the traditional thresholding method and then show some segmentation results of proposed method.

### 4.4.1 Comparing Thresholding and Persistence Method

Although thresholding is a simple solution for segmentation of the obstacles in the probability of occupancy map, it is highly sensitive to the choice of parameter value. On the other hand, the persistence based method performs an analysis for a range of threshold values and keeps track of all the resulting segmentations in a hierarchical fashion. We exploit this property to obtain a more robust outcome.
Segmentation results of both occupancy maps. (a)(b) Segmentation using original occupancy map. (c)(d) Segmentation using modified occupancy map. (a) Segmentation result using persistence threshold one step before part of the vehicle merges with background. (b) Right part of the vehicle merges with background before merges with other parts of the vehicle. (c) Segmentation result using persistence threshold one step before all parts of vehicle merge together. (d) All parts of vehicle merge together before merge with background.

Fig. 4.11 provides sample results for both the thresholding and persistence methods when varying parameters. For thresholding, \( \tau \) is set to 0.25, 0.3, 0.35, 0.4 and 0.45. For persistence, \( \gamma \) is set to 0.35, 0.4, 0.45, 0.5 and 0.55. We picked these ranges in order to make the results for both methods comparable for the middle parameter value. It is observed that even over this small range, thresholding causes significant variation in its output. In particular, there are several detection regions that appear and disappear on the right side of the image. On the contrary, the results for persistence are very consistent over a similar range of parameters. Fig. 4.12 shows results of both methods for four successive frames using a single parameter. For thresholding, \( \tau = 0.35 \), which provides correct segmentation for frame 3, but merges two cars on the left in frame 1 and 2, and divided them into several small parts in frame 4. On the contrary, the persistence method provides correct segmentation of cars for all of the frames.

4.4.2 Results

Fig. 4.13 shows several segmentation results using our methodology. It is observed that our approach is able to correctly segment ground from obstacles. Furthermore, obstacle detection and segmentation results are qualitatively good. Cars that are not too far from the ego vehicle are detected consistently as single regions. In the second row, the method is also able to detect an individual driving a bike. Also, most trees and bushes are detected and segmented properly on both sides of
Figure 4.9 Persistence diagrams of both types of occupancy maps. Red points are the detected obstacles in image space. (a) Persistence diagram computed from occupancy maps in [Per12]. (b) Persistence diagram computed from modified occupancy maps.

the road. While the bushes on the left side of the road are always detected, sometimes they will be split or merged into a single region.

4.5 Conclusion

In this chapter, we propose a robust road extraction and obstacle segmentation approach from stereo disparity maps. We compute UV-disparity maps and an occupancy grid for easy obstacle
Figure 4.10 Segmentation results for persistence bound $\gamma$ ranging from 0.3 to 0.6. Trees on the left side of the image are separated when $\gamma = 0.3$ to 0.5. They are merged when $\gamma = 0.6$. Note that two vehicles are always detected properly for all persistence thresholds.

Segmentation and analyze the sensitivity of the segmentation results by using tools from persistent topology. This framework provides a persistence diagram that helps us explore all thresholding results in a hierarchical fashion and achieve a robust segmentation. The method has been tested on several stereo test images available from standard stereo databases and the results look promising.
Figure 4.11 Comparison between thresholding and persistence methods of different parameters. Left column: Results of thresholding approach for $\tau = 0.25, 0.30, 0.35, 0.40$ and $0.45$. Right column: Results of persistence approach for $\gamma = 0.35, 0.40, 0.45, 0.50$ and $0.55$. The ground labeling is removed for clear illustration.
Figure 4.12 Comparison between thresholding and persistence methods of different frames. Left column: Results of thresholding approach for four successive frames with $\tau = 0.35$. Right column: Results of persistence approach for four successive frames with $\gamma = 0.45$. The ground labeling is removed for clear illustration.
Figure 4.13 Segmentation results with $\gamma = 0.45$ and corresponding persistence diagrams. Ground is labeled as green. Red points in persistence diagram are the visible segments in image space.
Obstacle detection is a fundamental step for autonomous navigation. Correctness is of particular interest in applications areas such as autonomous driving systems [Dua13; Bro11] for which safety is a primary concern. However, obstacle detection often depends on reliable depth estimates. Stereo depth maps estimates have gained some popularity in the transportation area due to the affordable hardware and richness of contextual information. Nevertheless, the outcome from state of the art algorithms is not always reliable due to a number of visual artifacts. The stereo matching step, which requires finding correspondences between pairs of rectified images, is a challenging task due to reflections, lack of texture, weather conditions, and repetitive patterns present in objects. Most approaches [Hir08; HK13; Ito13] rely on the computation of a matching cost from which the minimum is often taken as a possible correspondence between stereo pairs. However, due to the nature of this problem, the true matches may not fall on the global minimum of this function but in a local minimum instead. This can lead to incorrect matches. Traditional approaches may also include some indicator function of the correctness or certainty of the match found, which in some cases is used to remove possible mismatches. This leads to depth values being missing, which can also cause a problem in the identification of obstacles.
There are a number of approaches aiming to make disparity computations more robust [HK13; Sam13; Ito13] and extracting obstacle detections from these disparity results in a robust way [Pfe13; Zha10c]. Our approach attempts to construct a more reliable occupancy map from an alternative representation of the disparity map, which keeps track of multiple-candidate disparity values. We rely on existing disparity computation algorithms to extract these candidates, and our occupancy map can be used as an input for approaches such as Stixels [Pfe13] for robust obstacle detection. In order to capture correct matches on the disparity computations, we make use of multiple-candidates associated with different local minima of the matching cost function instead of relying on a single estimate. These candidates are obtained from an statistical analysis of the performance of the disparity algorithm being used in order to guarantee that we capture the correct structure in our representation with high-confidence. This set of candidates is then used to construct an accumulated occupancy map which provides the probability of occupancy of an obstacle at any given location. Fig. 5.1 illustrates one result of our method.

The remainder of this chapter is organized as follows: Section 5.1 gives the problem formulation and overview of our approach. Section 5.2.1 describes the ground segmentation method used. Section 5.2.2 presents the generation of the multi-valued disparity map. Section 5.2.3 presents the aggregate occupancy map computation using multi-valued disparity map. Section 5.2.4 describe a simple obstacle detection approach from the aggregate occupancy map. Results of our proposed
method are discussed in Section 5.3. Finally, Section 5.4 summarizes this chapter.

5.1 Problem Formulation

In order to simplify the mathematical formulation, we make use of a very simplistic model of the world (see Fig. 5.1). However, we will show the applicability of the approach to real scenarios throughout this paper. Let the 3D environment be defined by the $xy$-plane (referred to as the ground plane) and prismatic obstacles erected perpendicularly from the ground to a height of $h$. The stereo cameras are parallel to each other with viewing direction along the $y$-axis and with displacement along the $x$-axis. We will further assume that an obstacle is always present along the $u$-axis of the image domain.

The stereo pair of rectified grayscale images $I_L : \Omega_L \to \mathbb{R}^+$ and $I_R : \Omega_R \to \mathbb{R}^+$ defined over the image domains $\Omega_L \subset \mathbb{R}^2$ and $\Omega_R \subset \mathbb{R}^2$ satisfy the relation:

$$I_L(u, v) = I_R(u, v + d(u, v)),$$

where $(u, v) \in \Omega_L$ and $d : \Omega_L \to \mathbb{R}^+$ is the disparity map associated with the distance of an object visible by images $I_L$ and $I_R$. For simplicity, we ignore the effect of occlusions and assume that $d$ is defined over its entire domain. We let $\mathcal{O} \subset \Omega_L$ be the set of points in the image domain that correspond to obstacles and not the road, and define the Obstacle Depth Function as $D(u) := d(u, v)$ where $(u, v) \in \mathcal{O}$. Note that due to our assumptions about the setup of the camera and the obstacles, $d(u, v)$ will be the same along the $v$-axis as long as the visible point corresponds to an obstacle. Our objective is a proper estimation of $D(u)$, which is dependent on a good estimate of the disparity map $d(u, v)$.

Stereo disparity algorithms often depend on the computation of a matching cost function $C_p : [a, b] \to \mathbb{R}^+$ for each pixel $p = (u, v) \in \Omega_l$, where $[a, b]$ denotes the disparity search range, and $a$ and $b$ are constants. The matching is challenging due to reflections, lack of texture and repetitive patterns of objects. In these cases, the correct disparity may fall in a local minimum of the matching

![Figure 5.2 Overview of proposed approach.](image-url)
cost curve instead of the global minimum cost. In order to capture the correct disparity values, we make use of multi-candidates per location in the image domain. These candidates are associated with local minima of the cost function, and form our multi-valued disparity map. We compute an aggregate occupancy map from this set of candidate matches. Fig. 5.2 shows the pipeline of our proposed approach: first, a ground plane is fitted robustly from the data by making use of the disparity values associated with the global minima from the matching cost functions (Section 5.2.1); then, the multi-valued disparity map is constructed (Section 5.2.2); followed by computation of the aggregate occupancy map (Section 5.2.3); and finally, an obstacle depth function is estimated (Section 5.2.4).

For our implementations, we make use of the matching cost defined for the Semi-Global Block Matching approach [Hir08] with $a = 0$ and $b = 127$. However, this can be extended to any other disparity computation algorithms based on a matching cost function.

5.2 Obstacle Detection based on Multi-Valued Disparity Maps

5.2.1 Ground Fitting

The ground plane is computed by the same approach and parameters described in 4.1.2. That is, the plane is parameterized by $v = g_0(u, d)$ and we only consider candidate points $(u, v, d)$ that satisfy the constraint

$$g_0(u, d) + \frac{a_u h_l}{a_v b} \cdot d \leq v \leq g_0(u, d) + \frac{a_u h_u}{a_v b} \cdot d$$

(5.2)

where $a_u$ and $a_v$ are the intrinsic focal length of the camera in pixels, $b$ is the distance of the baseline of the stereo camera system, $h_l$ and $h_u$ are the heights above the estimated ground plane. Parameters $h_l = 200$ and $h_u = 1700$ are used in our experiments in order to only segment obstacles within 0.2 to 1.7 meters higher than the fitting plane.

5.2.2 Multi-Valued Disparity Map

As mentioned above, finding correspondences in stereo pairs is heavily affected by visual artifacts such as reflections, lack of texture and repetitive patterns. Fig. 5.3 shows an example of wrong disparity estimation by Semi Global Block Matching (SGBM) [Hir08; Ope] implemented by OpenCV due to the repetitive structure present in the fence. Fig. 5.4 shows the cost function $C_p$ for the point highlighted by a red circle in the image. The cost function applied in OpenCV-SGBM is the Birchfield-Tomasi subpixel metric proposed in [BT98]. This cost is the measurement of absolute minimum difference of intensities between two pixels in each direction along the epipolar line [Hir08]. The SGBM algorithm selects a disparity value close to the global minimum of the function while the groundtruth is at one of the local minima. The periodic pattern in the cost function is due to the repetitive pattern of the fence, which makes it difficult (if not impossible) to determine which one is the correct matching using only local information.
Figure 5.3 Example of wrong disparity estimation by SGBM due to the repetitive structure presenting in the fence. Disparity map shows the wrong disparity estimation in some regions of the fence (top). Right image (bottom) shows the estimated correspondence (red triangle) and ground truth (green square) for the test point in left image (middle).

Before further discussion, we first define the local minimum of a cost function. Let $\bar{d}_{p,i}$ be a local minimum of the cost function $C_p$ of a pixel $p$ if it satisfies $C_p(\bar{d}_{p,i}) \leq C_p(\bar{d}_{p,i} - 1)$ and $C_p(\bar{d}_{p,i}) \leq C_p(\bar{d}_{p,i} + 1)$. Note that, although the disparity value is computed to sub-pixel accuracy, the local minimum is computed to pixel accuracy.

One way to capture the correct disparity value is to pick all the local minima, $\{\bar{d}_{p,i}\}$, of the cost function $C_p$ as possible disparity candidates for one pixel in the left image. However, this can lead to too many candidates. In order to select a subset of these candidates, we define the cost ratio of the $i$-th local minimum as the criteria of picking candidates by

$$r_{p,i} = \frac{C_p(\bar{d}_{p,i})}{\min_d C_p(d)}.$$ (5.3)

This ratio was selected as a criteria since the values of the cost functions can vary greatly between different pixel locations making it unrealistic to pick a global threshold for the entire image based on cost alone. Other than cost ratio, we also tested the prominence ratio as a criteria, which is defined by the ratio of maximal prominence among all local minimum and prominence of a certain local
Figure 5.4 Cost function $C_p$ of the test point highlighted in Fig. 5.3. The estimation is near the global minimum, but the ground truth is near a local minimum. Notice that each local minimum corresponds to a bar of the fence.

Figure 5.5 Threshold choice for disparity candidate selection. (a) $\epsilon^{(0.95)}$ as a function of parameter $\tau$. (b) Average number of candidates as a function of parameter $\tau$. A choice of $\tau = 0.4$ guarantees that $\epsilon^{(0.95)} < 5$ and that the number of candidates will be below 5 as well. A total of 120 training stereo pairs were used for selecting this threshold.

minimum. The experiment showed that cost ratio was a better choice. A subset $\{d_{p,j}\}$ of candidate disparity values is selected by keeping the local minima with cost ratio $r_{p,j}$ below a threshold $\tau$ and for which the point $(p, d_{p,j})$ satisfy the ground plane condition in Eqn. 5.2.
The value of $\tau$ is selected statistically from a set of 120 stereo pairs from the KITTI dataset (which includes groundtruth depth estimates) in order to make sure that the candidate points includes a point near the true disparity with high confidence, while minimizing the overall number of candidates. This is done by enforcing that 95% of the candidate sets have a candidate within a distance $\Delta$ of the true disparity. The value of $\Delta$ is empirically selected to be 5 pixels as it offers a good compromise between the number of candidates and the distance from the true disparity.

In order to perform the analysis described above, we compute for each pixel $p$ the minimum distance to the true disparity value

$$
\epsilon_p(\tau) = \min_j |d_p^{true} - d_{p,j}|,
$$

(5.4)

where $d_p^{true}$ is the true disparity value for the corresponding pixel, and $j$ is the index for the candidate disparity values. We also keep track of the number of candidates $n_p$ for corresponding pixel. These quantities are accumulated over all pixels and images in the dataset. Additionally, we compute $\epsilon(0.95)(\tau)$ as the distance associated with the 95% percentile of all $\epsilon_p$ values. We ensure that at least one candidate is within $\Delta$ from the true value for 95% of all the samples by picking a value of $\tau$ that satisfies $\epsilon(0.95)(\tau) \leq \Delta$. Fig. 5.5 illustrates the curves obtained from $\epsilon(0.95)(\tau)$ and the average number of candidates as a function of $\tau$. Since the number of candidates increases monotonically with the value of $\tau$, then it is sufficient to pick the largest value of $\tau$ for which $\epsilon(0.95)(\tau) \leq \Delta$. From our analysis, this corresponds to $\tau = 0.4$, which gives an average number of candidates less than 5.

As discussed above, the value of $\tau$ was selected using a set of 120 stereo pairs for training. A different set of 73 stereo pairs are used for testing the performance of this threshold. Fig. 5.6 (a) and (b) illustrate $\epsilon(0.95)$ and the average number of candidates for each test stereo pair for $\tau = 0.4$. It is observed that $\epsilon(0.95)(0.4)$ is less than 5 for most of the stereo pairs and the average number of candidates per pixel is less than 6.
candidates is less than 9 for all except one stereo pair. That is, for most pixels in each stereo pair, there exists at least one candidate disparity within 5 pixel distance of the ground truth and with average number of candidates less than 9.

Notice that this threshold may change if we use a dataset with a very distinct type of images or a different disparity computation algorithms. However, the value of $\tau$ can be tuned following this same process. This parameter choice is how our approach takes into account some statistical information about the environment and the performance of the cost function used for matching. Also notice that capturing the correct disparity values by multiple disparity values requires that the cost function reflects the correct matching cost between stereo pair. That is, it is likely to find the true matched point near a local minimum. There are no guarantees for the case of reflections since this artifacts yield cost function with misleading local minima.

5.2.3 Aggregate Occupancy Map Computation

The aggregate occupancy map is computed using a variation of the visibility based approach introduced in [Per12], in which a probability of occupancy is obtained for points in the so-called $u$-disparity space (or simply $ud$ space). The main difference for our approach is that we consider multiple candidate points per each pixel due to our multi-valued disparity map instead of a single point estimate.

In the original approach, a point $s = (u, d)$ in the $u$-disparity space is assigned a probability of occupancy by counting the number of visible and observed obstacle points falling within the site. Since our disparity map is multi-valued, a single pixel $p$ in the image domain may have multiple points in the point cloud. In order to compensate for this, we assign a weight to each candidate point equal to the $1/n_p$, where $n_p$ is the number of candidate points for that pixel. As our definition of local minimum in Section 5.2.2, all the points within the flat region of a cost function are considered as local minima as well. However, the contribution of this pixel to the occupancy grid computation is normalized by the number of candidates for this pixel. The end result is that pixels with large flat regions will not influence the occupancy grid. Please refer to [Per12] and Section 4.1.3 for more details on occupancy computation.

The second row of Fig. 5.7, 5.8, 5.9, 5.10 and 5.11 illustrate the aggregate occupancy map computed from multi-valued disparity map (left) and OpenCV-SGBM disparity map (right). Comparing the occupancy map from the two different approaches, we see that the aggregate occupancy map is able to capture the structure of obstacles as well as OpenCV-SGBM disparity map for the regions where OpenCV-SGBM disparity map does well, and for the regions where OpenCV-SGBM provides wrong estimation (which highlighted by the black rectangle), the aggregate occupancy map is able to avoid the error through the normalization process during occupancy computation. Because pixels with good disparity estimation have few candidates since the global minimum cost of those pixels is much smaller than other local minimum, and most of the bad disparity estimations are the outcome of several local minima close to the global minimum.
5.2.4 Obstacle Detection

A simple approach for obstacle detection from the aggregate occupancy map is to apply a threshold $\tau_p$ to the probability of occupancy $P_{\text{err}}$. Detections in the image domain are obtained by displaying a vertical segment for each location $(u, d)$ in which there was a detection in the occupancy map. The vertical segments are drawn between the lower and upper planes defined during the ground fitting procedure. We make use of a value of $\tau_p = 0.02$ for detections using our aggregate occupancy map, and a threshold of $\tau_p = 0.15$ for detections using the traditional occupancy map using SGBM.

5.3 Experiments

The disparity map and correspondence matching cost computation is implemented in C++ using OpenCV, and all the other processes are implemented in MATLAB. We use stereo pairs from the KITTI dataset [Gei12; Gei13; Fri13] to test our approach. We tested our approach on a complete dataset from the KITTI benchmark suite, and observed either comparable or improved results on the occupancy map between our approach and OpenCV-SGBM. In this section, we only highlight the most extreme cases. Figs. 5.7, 5.8, 5.9, 5.10 and 5.11 show five examples of the obstacle detection using the proposed multi-valued disparity map and the OpenCV-SGBM disparity map. Fig. 5.7 illustrates a case in which both approaches give good results. We noticed that since most images...
in the datasets are of good quality and feature well textured obstacles on the road, then disparity computation results are often correct. Our approach produces very similar results in these cases as well. Figs. 5.8, 5.9, 5.10 and 5.11 show four examples in which the multi-valued disparity map provides better obstacle detection than OpenCV-SGBM. Obstacle detections shown in the image domain are obtained by finding the obstacle with highest disparity value for each column in the $u$-disparity domain. That is, the closest obstacle among all detected obstacles at each $u$. We map these detection back to the image domain based on the height of the ground and sky plane at position $(u, d)$ obtained during the ground fitting process mentioned in Section 5.2.1.

In Fig. 5.7, both approaches detect the obstacle correctly. We can see little difference between the two occupancy maps. This is due to the fact that, as discussed in Section 5.2.3, pixels with good disparity estimation have a small number of candidates.

For Fig. 5.8, 5.9 and 5.10, it is observed that the OpenCV-SGBM disparity maps are bad in some regions due to the brightness or shadow on the ground. Because of it, the occupancy maps obtained using these disparity maps are noisy in the corresponding regions in $u$-disparity domain, which are highlighted by the black rectangle. This leads to the wrong obstacle detection results on the ground. For Fig. 5.11, the bad disparity is due to the repetitive pattern of the fence. The occupancy map in (d) shows some wrong detection in the fence area highlighted by the black rectangle and the obstacle detection (h) shows most parts of the fence are detected as farther obstacle. The occupancy map obtained by our approach in (c) shows several detections including the correction detection in the fence of region and in (g), it shows our approach obtains much better result than (h). Our method is able to mitigate these problem. This can be explained by noting that regions with wrong disparity
Figure 5.9 (a)-(h) are the same as Fig. 5.7. In this case, the disparity map in (b) is wrong in some regions of ground due to the brightness. The occupancy map computed using this disparity map is noisy in the corresponding regions highlighted by the black rectangle shown in (d). This leads to the wrong obstacle detection on the ground shown in (h). (g) shows our approach is able to avoid those false negatives.

Figure 5.10 (a)-(h) are the same as Fig. 5.7. In this case, the disparity map in (b) are wrong in the right region of ground due to the shadow. The occupancy map computed using this disparity map has wrong detection in the corresponding regions highlighted by the black rectangle closed to the black car shown in (d). This leads to the larger detected obstacle in the right part of the image shown in (h). (g) shows our approach is able to avoid those false negatives and gives the correct detection of the car.

Values often have multiple candidates, and their contribution to the occupancy map is distributed over all candidate matches.
Figure 5.11 (a)-(h) are the same as Fig. 5.7. In this case, the disparity map in (b) is wrong in some regions of the fence due to the repetitive pattern of the fence. The occupancy map shown in (d) has wrong detection in the region of fence highlighted by the black rectangle and the obstacle detection in (h) shows most parts of the fence are detected as farther obstacle. The occupancy map obtained by our approach in (c) shows several detections in the region of fence including the correct detection, since the pixels with bad disparity estimation have large number of candidates. (g) shows although our approach does not obtain the completely correct detection of the fence, it is able to obtain much better result than OpenCV-SGBM.

5.4 Conclusion

In this paper, we propose a methodology for obstacle detection in outdoor scenes using multi-valued stereo disparity maps. The multi-valued stereo disparity map is obtained by selecting the local minimum of the matching cost function using statistics from a training set of stereo pairs. An aggregate occupancy map is computed from this multi-valued disparity map. Finally, obstacles are detected by thresholding the aggregated occupancy map. The experiment shows that the proposed approach is able to provide more reliable occupancy maps for obstacle detection than traditional single-valued stereo disparity computations.
CHAPTER 6

COARSE-TO-FINE FORAMINIFERA IMAGE SEGMENTATION THROUGH DEEP FEATURES

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 [SG99]. Foraminifera have existed since the Cambrian period, and an estimated 10,000 species are still in existence [Vic92], 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 [BSG03], biostratigraphy [CE45], paleoecology [Ber92] and paleobiogeography [BER72]. 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 [CE45]. 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 [Liu94].

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.
Figure 6.1 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.

Figure 6.2 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.

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 [KS83]. An example of this is shown in Fig. 6.1. None of the proposed image-based, automatic classification approaches [MA06; Liu94] 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 foraminifera shells [Cor91; Bol91], 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 as diatoms.

In this chapter, 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 post-processing on the refined edge map. The entire pipeline used in our experiment is illustrated in Fig. 6.2 and some sample segmentation results are shown in Fig. 6.1. 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 chapter is organized as follows: Section 6.1 introduces the images used for segmentation from our foraminifera dataset; A detailed description of our segmentation pipeline is introduced in Section 6.2; Experiments of our approach as well as the analysis of the results are discussed in Section 6.3; Finally, Section 6.4 summarizes this chapter and discusses the potential applications of foraminifera segmentation.

### 6.1 Foraminifera Image Dataset

We created 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 x 450 pixels per sample. Fig. 6.3 shows the setup of the image capture system [Mit19]. Fig. 6.4 (a) shows 4 of 16 images of three samples in our dataset. A dataset of 1437 foraminifera samples, in which 457 samples are manually segmented, has been collected and can be found on the project website [Pro]. 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).
samples. The images of first six species (987 samples) are used for our study.

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. 6.4 (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. 6.4, 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.

In this chapter, we discuss a coarse-to-fine strategy using manually designed features to obtain the edge detection through a small training set when the computational resource is limited. In the next chapter, we will discuss a deep learning based approach. 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.

### 6.2 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 $\{I^i : \Omega \rightarrow \mathbb{R}, i = 1, \ldots, 16\}$, our goal is to predict the probability of a location $x \in \Omega$ being an edge, based on a set of image patches $\{I_{N_x}^i, i = 1, \ldots, 16\}$.

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
Figure 6.5 Selected hand-crafted features of the second sample in Fig. 6.4. (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.

Performance in edge detection tasks [Lim13]. 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.

6.2.1 Hand-Crafted Feature Extraction

At this stage, inspired by [Dol06] and [Lim13], we compute several types of edge features from all 16 images per sample to capture 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 [Lin98]. 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

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

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. 6.5 (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. 6.4, 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. 6.5 (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 [FP12] to reconstruct a 3D surface of each sample for better fusion of the input images, followed by a Bilateral filter [TM98] 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. 6.5 (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 [Gol05] 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

\[
\kappa = H + \sqrt{H^2 - K},
\]

where \( H \) is the mean curvature and \( K \) is the Gaussian curvature of the surface. Please refer to [Gol05] for more details. Fig. 6.5 (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 \( 15 \times 15 \) patch on that pixel and concatenating the features obtained from each of the 15 feature maps.

### 6.2.2 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 [Cri11]. 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. 6.8 (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. 6.5 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.

### 6.2.3 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. 6.8 (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.
6.2.3.1 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 summary, 12 feature maps are computed and a patch size $15 \times 15$ 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. 6.8 (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 [Dol06]. Better boundary completion performance can be achieved through careful selection of training data and use of larger patches.

6.2.3.2 Deep Features

The features we extract for refinement in Section 6.2.3.1 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 [Goo16]. In our experiment, we use four convolutional layers and two fully connected layers as used in [She15], which is sufficient for low-level feature extraction as suggested in [She15]. The only difference is that we do not use local response normalization layers (LRN). The input of this CNN is a $32 \times 32$ patch of coarse probability map and it is trained to label the center point of each patch as edge or non-edge. Dropout [Sri14] with probability 0.5 is used in the first fully connected layer (FC1) during training to reduce the risk of overfitting.

Similar to [She15], 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 [She15]. The comparison of hand-crafted features and deep features can be found in Fig. 6.6. 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 $11 \times 11$ 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. 6.8 (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
Figure 6.7 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 $15 \times 15$ patches in original images. (c) Each pixel in the deep feature maps is computed by a $32 \times 32$ patches in the coarse edge map and thus depending on a set of $46 \times 46$ patches in the original images. (d) Each pixel location in the refined edge map is predicted by a set of $11 \times 11$ patches in deep feature maps and then depending on a $43 \times 43$ 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 $57 \times 57$ patches from the original images.

details and also get more closed boundaries because of the more powerful features.

Fig. 6.7 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 $57 \times 57$ 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 highly limited training data.

6.2.4 Post-Processing for segmentation

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. 6.8 (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.

6.3 Experiment

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

6.3.1 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 pixels 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 [Aba15]. 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 significantly optimized through implementation of a fully convolutional network and parallel computing.

6.3.2 Qualitative and Quantitative Evaluation

Results of each stage of our approach are shown in Fig. 6.8. 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. 6.8, 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 evaluation metrics: the best weighted covering score (W), un-weighted covering score (Un-W) [Arb11] and recall of regions (Recall) for fixed thresholds. The edge recall is defined as the percentage of the labeling 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 6.1 Edge Detection Results using Different Features and Classifiers |
|-----------------------------|---------------------|---------------------|
|                            | 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     |

Table 6.1 presents the scores at each stage. The precision 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. 6.8. But we do observe the improvement of precision after
Figure 6.8 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.

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 unweighted 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 $43 \times 43$ patches in the coarse map, we report the scores of refinement with hand-crafted features using $43 \times 43$ patch (RF+RF(HC43)). The similar scores with $15 \times 15$ 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 two other settings with only one stage using 100 training sample images: using patch size $57 \times 57$ 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. 6.9. 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.

### 6.4 Conclusion

In this chapter, 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.
In Chapter 6, we discuss a coarse-to-fine strategy for foraminifera image segmentation when computational resources are limited. In this chapter, we discuss an edge-based segmentation approach using deep learning algorithms and introduce a topology-aware loss to force the model to put more focus on preserving the topological structure of edges.

As an intrinsic property of data-driven edge detection, the distribution of edge/non-edge pixels is highly imbalanced. For most of the recent deep learning-based approaches, the loss is usually balanced by either uniformly sampling positive and negative pixels with a fixed ratio [She15] or utilizing all pixels with a class-balancing cross entropy loss [XT17]. In spite of the great success in pixel level performance, these strategies are not suitable in application such as edge-based image segmentation where local topology and connectivity information is essential. As shown in Fig. 7.1 (b), in which two edge maps are compared, standard pixel-based metrics greatly penalize small shifts/deformation in edge maps as shown in the left plot, and do not differentiate them from gaps in the edge maps that are more essential for segmentation. This motivates our work in which (1) we develop a new error metric for determining how well two edge maps match, and (2) make use of this metric to enhance generalization for learning-based segmentation techniques.

In general, three types of pixels exist during training: (i) pixels that have the same classification as the reference image; (ii) pixels that are misclassified due to noisy or incorrect estimation, but
do not affect the topological structure of the edge map; and (iii) pixels that are misclassified and affect the topological structure. It is obvious that the loss associated with type (iii) pixels is more desired for training a topology-aware edge detection network. Since the uniform sampling or class-balancing cross-entropy loss cannot distinguish types (ii) and (iii), the loss associated with type (iii) is always washed out by type (i) and type (ii) through averaging over one mini-batch. This negative impact could be potentially reduced given large enough training sets. However, edge labeling can be expensive and noisy, which limits the number of good training samples for most of the edge detection datasets. Thus, a topology-aware loss function is desired for efficiently utilizing the training data.

In this chapter, we introduce Localized Topology Conditions (LTC) which can identify local topological difference between estimated edge and the reference edge maps to distinguish between type (ii) and (iii) pixels. By using this, we propose the Localized Topology-Aware (LTA) loss function by increasing the relative loss of type (iii) pixels in standard cross entropy to train an edge detection network putting more focus on preserving the topological structure of edges. We analyze these techniques using a synthetic dataset and a public foraminifera (forams) dataset [Zho17; Ge17] (see Fig. 7.1). We demonstrate that the proposed approach used as a refinement step of edge detection is able to obtain predicted edges with topological structures that better resemble the reference edges.

**Figure 7.1** Localized Topology Conditions (LTC) applied to foraminifera segmentation. (a) Illustration of the segmentation task. A simple image [top] from the dataset [Ge17]. Our aim is to estimate the edges between chambers and segment the sample based on the edge map. Groundtruth labels [bottom-left] and sample segmentation output [bottom-right] are shown. (b) Comparing pixel-based difference and topological detection for sample output in (a). Metrics such as cross-entry, are based on image difference [left] which can highlight areas were edges map are just misaligned. The proposed Deformation Violation (DV)-Mask [right] ignores small deformations and highlights locations where the local topological structure does not match. This mask is used to produce a Localized Topology-Aware (LTA) loss for training. (c) Illustration of topological conditions. On the [left], neighborhoods in which the localized topological conditions on the local complement sets $E_r$ are satisfied even though a small perturbation is present. On the [right], a case in which the conditions are not satisfied due to a missing edge in the prediction.
than the baseline algorithm.

The remainder of this chapter is organized as follows: Section 7.1 introduces our novel Localized Topology Conditions and a topology-aware mining strategy; experiments as well as the analysis of the results are discussed in Section 7.2; and Section 7.3 summarizes this chapter.

7.1 Localized Topology Aware Loss

As discussed above, a topology-aware edge detection network can be trained by paying more attention to error pixels causing topological errors (type (iii) pixels). We accomplish this by proposing the Localized Topology Aware (LTA) loss function, which is based on an evaluation metric that can be used to distinguish type (ii) and type (iii) pixels based on the local topological difference between two sets of edges. Using this topological difference, we can select the type (iii) pixels and weight them accordingly in the LTA loss function.

7.1.1 Localized Topology Conditions

Tools from topological data analysis (TDA) [Ede00] including persistence diagram [ZC05] and persistence landscape [Bub15] have previously been used as topological features of a set of points in a space. Both of these approaches provide robust global topological descriptors by summarizing the filtration (a sequence of topological features at different scales) of point clouds. However, for dense prediction applications such as edge detection, local descriptors are more desired. Although persistence diagram and the persistence landscape can be locally applied to a small neighborhood around each pixel in the image, high time complexity is a drawback. As such, we propose a novel low-complexity topology-based evaluation metric to locally indicate the difference between two sets of edges which is invariant to bounded deformations.

First, we define the edge map functions $I_k : \Omega_k \rightarrow \{0, 1\}$, where $k \in \{1, 2\}$, $\Omega_k \subset \mathbb{R}^2$ and a value of 1 indicates the presence of an edge. We will say that two maps match if

$$I_2(x) = I_1 \circ g(x)$$

for some deformation $g : \Omega_2 \rightarrow \Omega_1$ with $\|g(x) - g(x')\| \leq K_d \|x - x'|\|$, where $K_d$ is a Lipschitz deformation bound. Our objective is to identify any locations in which two edge maps fail to match. In this case, we interpret failing to match as not being able to find any deformations that locally match the structure of the edge maps. Our approach will verify a number of conditions based on local topological invariants that should be satisfied if a local deformation exists, and flag as violating locations those that do not satisfy these invariants. This methodology builds on a framework developed for occlusion detection [Lob10] and image feature matching [Lob11] using topological invariants. A localized homology-based evaluation metric was also proposed in [Ahm14] for comparison of roadmap reconstructions. However, this metric does not accommodate for local deformations.

To define the topological invariants, let us begin by defining the edge set $E_k = \{x \in \Omega_k \mid I(x) = 1\}$,
the ball of radius $r$ and centered at $x$ as $B_r(x) = \{x' \in \mathbb{R}^2 |||x - x'|| < r^2\}$, the local edge set $E_{k,r}(x) = E_k \cap B_r(x)$, and its local complement as $\tilde{E}_{k,r}(x) = E_k^c \cap B_r(x)$. We can check that if two edge maps match given a deformation $g$ then

$$E_{1,r-K_d}(x) \subset g(E_{2,r}(x)) \subset E_{1,r+K_d}(x).$$  \hfill (7.2)

This follows from the fact that the deformation is bounded. As such, expanding or shrinking a neighborhood by the Lipschitz constant ensures the inclusion of the deformed set. Let us further define the count $\alpha(S)$ of connected components over a set $S$, and the relative count $\alpha(S_-, S_+)$ for $S_- \subset S_+$. The relative count corresponds to the number of connected components in $S_-$ that belong to the same connected components in $S_+$ are identified. Then, we have that:

**Lemma 7.1.1.** Given two edge maps, $I_1$ and $I_2$, that match under a local deformation bounded by $K_d$ then for any $r > 0$ the following condition is satisfied:

$$\alpha\left(E_{1,r-K_d}(x), E_{1,r+K_d}(x)\right) \leq \alpha\left(E_{2,r}(x)\right).$$  \hfill (7.3)

**Proof.** Let $L(S)$ be the set of connected components over a set $S$, and $L(S_-, S_+)$ be the set of connected components in $S_-$ after they have been identified using the connected components in $S_+$. We let $S_- := E_{1,r-K_d}(x), S_+ := E_{1,r+K_d}(x)$ and $S := E_{2,r}(x)$. In order to prove the desired result, we just need to define an injective mapping $i : L(S_-, S_+) \rightarrow L(S)$. Let $l \in L(S_-, S_+)$, then we know that $l$ was formed from connected components in $S_-$; hence by equation 7.2, we have that these sets are all included in $g(S)$. This gives a candidate set of connected components in $S$ that can be associated with $l$. We assigned any of them to $l$ as part of our definition of $i$. Note that any component in $g(S)$ is included in a single component in $S_+$ (by equation 7.2). Since the list of components in $g(S)$ that were candidates for $l$ as part of our definition of $i$ were selected from components in $S_-$ that are included in a single component in $S_+$, then these candidates are not candidates for an element $l' \in L(S_-, S_+)$ for which $l' \neq l$ because they are associated with a different component in $S_+$. This shows that our mapping is one-to-one and concludes the proof. \hfill $\Box$

Note that this condition does not rely on $g$ anymore, and it is purely a topology constraint over a local neighborhood. Building on this condition, we can come up with the following result.

**Definition 7.1.1 (3.1).** Given two edge maps $I_1$ and $I_2$, we define an indicator map $\Gamma_r(x|I_1, I_2, K_d)$ at scale $r$ and bound $K_d$ as a binary map that has value 1 at location $x$ if the following conditions are not satisfied:

$$
\begin{align*}
\alpha\left(E_{1,r-K_d}(x), E_{1,r+K_d}(x)\right) &\leq \alpha\left(E_{2,r}(x)\right) \\
\alpha\left(E_{2,r-K_d}(x), E_{2,r+K_d}(x)\right) &\leq \alpha\left(E_{1,r}(x)\right) \\
\alpha\left(\tilde{E}_{1,r-K_d}(x), \tilde{E}_{1,r+K_d}(x)\right) &\leq \alpha\left(\tilde{E}_{2,r}(x)\right) \\
\alpha\left(\tilde{E}_{2,r-K_d}(x), \tilde{E}_{2,r+K_d}(x)\right) &\leq \alpha\left(\tilde{E}_{1,r}(x)\right)
\end{align*}$$  \hfill (7.4)

**Theorem 7.1.2 (3.1).** Two edge maps $I_1$ and $I_2$ cannot be matched under a local deformation bounded by $K_d$ if the indicator map $\Gamma_r(x|I_1, I_2, K_d)$ is not an all zero map.
Proof. If there exists a deformation \( g \) with Lipschitz deformation bound as defined in equation 7.1, then the first condition in equation 7.4 would need to be satisfied by the previous lemma. Since the deformation is also a deformation for the map \( \bar{I}_k = 1 - I_k \), then the third condition must also be true. Finally, since \( g^{-1} \) is also a deformation between \( \Omega_1 \rightarrow \Omega_2 \) with the same Lipschitz constant, the second and fourth conditions must also hold. If any of these conditions are violated at any point, then such deformation does not exist, and as such, the edge maps cannot be matched.

**Theorem 7.1.3.** Two edge maps \( I_1 \) and \( I_2 \) cannot be matched under a local deformation bounded by \( K_d \) if the indicator map \( \Gamma_r(x|I_1, I_2, K_d) \) is not an all zero map.

Fig. 7.1 (c) illustrates how the topological conditions are satisfied for small deformation and not for topological violations together with the corresponding indicator mask in Fig. 7.1 (b). The indicator \( \Gamma_r \) helps identify locations in which the neighborhood \( B_r(x) \) could not match between the two edge maps. Hence, we define a deformation violation mask (\( DV \)-mask) as

\[
DV_r(I_1, I_2; K_d) = \{ x \mid \Gamma_r(x|I_1, I_2, K_d) = 1 \} \oplus B_r,
\]

where \( A \oplus B_r = \cup_{a \in A} B_r(a) \) is the dilation of set \( A \) by \( B_r \). We will make use of the area of this mask as a measure of how well the two masks match. Alternatively, we could also compute the indicator over a range of \( K_d, [K_{min}, K_{max}] \), and then assign to each location \( x \) a value corresponding to the smallest \( K' \) such as \( \Gamma_r(x|I_1, I_2, K) = 0 \) for all \( K > K' \) in the range. If no value of \( K' \) is found then it means that no match can be found over the range of deformations. If a value of \( K' \) is found then it means that any smaller value of \( K \) guarantees that no deformation exists. This new function can also be used to specify other performance metrics for comparing edge maps.

As an implementation note, thick edge maps may include holes with single pixel or a few pixels within the edges, which would be counted on the lower bounds associated with \( \bar{E}_{k,r} \) in equation 7.4. In order to remove these artifacts, we erode by a pixel the sets \( \bar{E}_{k,r-K_d}(x) \) and \( \bar{E}_{2,r-K_d}(x) \). Furthermore, in order to handle narrow edges which (after sampling) can lead to disconnected edges maps with segments that may be one pixel apart from each other, we dilate the sets \( E_{1,r+K_d}(x) \) and \( E_{2,r+K_d}(x) \). These modifications do not affect the validity of the topological conditions, and provide a more robust detection by directly handling sampling issues of the edge map.

### 7.1.2 Localized Topology-Aware Loss

For simplicity, we consider the loss for a single pair of edge maps (groundtruth \( I_g \) and prediction \( I_p \)). We model the prediction as an edge detection network of the form \( I_p(x) = f_W(x) \in [0, 1] \), where \( W \) represents the set of network parameters. We use all the edge pixels as positive samples and uniformly sample the same number of non-edge pixels as negative samples. The pixel-level loss is defined as

\[
L_0(x; W) = -I_g(x) \cdot \log(I_p(x)) - (1 - I_g(x)) \cdot \log(1 - I_p(x)).
\]  

(7.6)
We also tried the class-balanced cross entropy loss and found equally sampling provided better performance. We define the deformation violation map $DV_r(I_g, I^r_p; K_d)$ for $I^r_p$, a binary version of $I_p$ obtained by setting pixels to 1 if $I_p(x) > \tau$ and 0 otherwise. Unfortunately, the $DV$-mask does not directly specify which pixels cause the topological conditions to be violated, but instead specifies a neighborhood containing these points. Hence, we use this map as a criterion of hard example mining during the training phase. That is, we put more weights on pixels with non-zero values in the $DV$-mask. Hence, the Localized Topology Aware (LTA) loss is defined as

$$L_{LTA}(x; W) = \begin{cases} \gamma L_0(x; W) & \text{if } x \in DV_r \\ L_0(x; W) & \text{otherwise}, \end{cases}$$  \quad (7.7)$$

where $\gamma \geq 1$ is a hyper parameter to adjust the influence of the topological metric. When $\gamma = 1$, the LTA loss is equivalent to the original cross entropy loss.

### 7.2 Experiments

In this section, we first introduce the edge detection network used in our experiments and the implementation details. Then we compare our topology-aware evaluation metric against other topology level evaluation metrics proposed in [Weg13] on a synthetic dataset. Finally, we apply our LTA loss for edge detection on a synthetic and real foram datasets, and compare it with two approaches.

#### 7.2.1 Edge Detection Network Architecture

The edge detection module adopted in our experiment is the D-LinkNet [Zho18], which is originally designed for road extraction. As shown in the Fig. 7.2, this module has an encoder-decoder architecture with skip connections as well as a center dilation part in between. Feature maps are down sampled through max pooling layers in the encoder and up sampled through transposed convolutional layers with stride 2 in the decoder. The center dilation part consists of four sequentially stacked dilated convolutional layers [YK15] with dilation rates 1, 2, 4, 8, respectively. The dilated convolutional layers are adapted to enlarge the receptive field of each feature point while retaining the resolution of the feature maps. This module has the advantage of extracting narrow and long span of regions in the image, such as road and edges.

In Chapter 6, a coarse-to-fine strategy is used to obtain better edge detection performance. Inspired by this, we adopt an iterative refinement approach similar to [Mos18]. Instead of obtaining the predicted edge map through a single step, the output edge probability map is again used as the input of an additional edge detection module. A cleaner and more accurate edge map can be achieved after several steps of the refinement. In our experiments, we use two steps of refinement, because we did not observe significant improvement with higher steps. As shown in Fig. 7.3, after getting the intermediate estimated edge map through a D-LinkNet module, another D-LinkNet
Figure 7.2 D-LinkNet architecture. Both the input and output are $128 \times 128$ grayscale images.

Figure 7.3 Iterative refinement edge detection.

module takes the intermediate results as input to obtain the final result.
7.2.2 Implementation Details

The local topological metric computation is implemented in C++ and the edge detection network is implemented in Python using the TensorFlow framework [Aba15]. All the models are trained by Adam optimizer [KB14] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

7.2.3 Dataset

7.2.3.1 NCSU-CUB Foram [Pro]

This dataset is used for visual foram species identification [Ge17; Zho17], which is introduced in Section 6.1. Images of each sample are taken under 16 different lighting directions via a microscope. It contains 457 manually segmented images which separate chambers and apertures of each sample. This dataset contains soft and hard edges, similar adjacent regions and non-closed edges, which makes accurate edge detection non-trivial and is suitable for evaluating the performance of edge closing. In our experiment, the images are rescaled to $128 \times 128$ to reduce the computational complexity while keeping enough edge information. Among all the labeled samples, 322 samples are used for training and the remaining 135 samples are used for testing. An example is shown in Fig. 7.1 (a).

7.2.3.2 Synthetic Dataset

This dataset is inspired by the Foram dataset and it is designed for the study of iterative edge detection refinement. First, a 3D object with three to six intersecting ellipsoids is randomly generated. Then, the ground truth image is created by taking a snapshot from a random viewpoint and extracting the edges of ellipsoids. Ellipsoids are distinguished by different colors, thus edges can be easily obtained by comparing the intensity of neighboring pixels. To mimic the imperfect predicted edge probability map, small gaps are randomly added to the inner edges, and each edge branch is randomly thickened or thinned and assigned a probability value in a range of $[0.2, 1.0]$. The outer boundaries are kept as the thinned version of the ground truth, as we assume the edge probability maps are computed from the original images similar to the images in NCSU-CUB Foram dataset (Fig. 7.1 (a)) and thus the boundary edges are straightforward to be obtained perfectly. Finally, images are rescaled to $128 \times 128$. This dataset is designed to directly quantify the power of the approaches to complete any missing local structure. Example samples are shown in Fig. 7.4.

7.2.4 Edge Detection Evaluation Metrics

We use three types of evaluation metrics (pixel-level metrics, region-level metrics and topology-level metrics) to quantitatively evaluate the edge detection results.

7.2.4.1 Pixel-Level Metrics

We first use the $F1$ score for evaluating the pixel wise edge classification performance. However, this metric is sensitive to small shifts of the edges while the edges still maintain the topological structure.
structure. Therefore, we also use correctness, completeness proposed in [Wie98] as the relaxed precision and recall, and use quality [Wie98] to summarize them. The true positives are determined by the predicted and target edge pixels within a certain threshold distance, which makes it less sensitive to small shifts. In our experiments, we use a threshold distance of 5 pixels. See [Wie98] for more details.

7.2.4.2 Region-Level Metrics

One benefit of the topology-aware edge detection is improving the accuracy of edge-based segmentation. So we also use the region-level performance to indicate the performance of the topological structure preservation. A binary edge map is first obtained by thresholding the edge probability map. Then a closing operation using a disk with a radius of 2 is used to refine the edge map. Finally the segmentation is obtained by extracting regions based on the skeleton of the edges. To evaluate the segmentation performance, the unweighted Intersection over Union (IoU) [Arb11] is adopted. We also use the recall of regions [Ge17] to further evaluate how well the edge closure performs. This metric quantifies the percentage of the target regions detected in the predicted segmentation and a target region is marked as detected if there exists a predicted region with IoU greater than a threshold. In our experiment, the region recall threshold is 0.7 and 0.5 for Synthetic Image and NCSU-CUB Foram respectively.

7.2.4.3 Topology-Level Metrics

To directly evaluate the topology preserving performance, the two metrics compared in section 7.2.6 are used, including a set of topology metrics proposed in [Weg13] and DV-masks. Metrics described in [Weg13] involve the following three steps: (1) randomly sample two points lying on edges existing in both edge maps; (2) find the shortest path along the edge connecting the two points in each edge map; and (3) compare the length of the path between the two edge maps. While sampling, we restrict that two sampled points are always connected in the ground truth edge map. Then, a pair of points is labelled as infeasible, if the two points are disconnected in the predicted edge map. Additionally, if the difference of path length is larger than 10% of the path on ground truth, the pair
is labelled as 2long2short, otherwise it is labelled as correct. To summarize these three quantities into a single value, we define PathError as

$$\text{PathError} = \frac{2\text{long2short} + \text{infeasible}}{\text{correct} + 2\text{long2short} + \text{infeasible}},$$

which quantifies the portion of sampled point pairs with different shortest paths in the ground truth and the predicted edge map. In our experiments, 300 paths are sampled per image for convergence of the evaluation results. For the proposed local topology metric, we use the non-zero area of the DV-mask computed from an image pair to quantify the topology difference under a bounded deformation. In this experiment, we use synthetic data and set $r = 6$ and $K_d = 2$.

### 7.2.5 Parameter Selection for LTC

As discussed above, the parameters $r$ and $K_d$ used in condition (7.4) need to be determined before computing $DV$-mask for measuring the topological differences. Fig. 7.5 (a) illustrates average size of non-zero regions of $DV$-mask as a function of $K_d$ when fixing $r$ from 5 to 10. These curves were computed from the edge detection results of the NCSU-CUB Foram testing set obtained by the baseline approach DLinkNet (see Section 7.2.7) with an edge threshold of 0.4. As $K_d$ controls the level of deformation, more pixels get detected using smaller $K_d$. Also, as $r$ controls how close a pixel needs to be to affect the $DV$-mask, more pixels get detected using larger $r$. To find a proper pair of $K_d$ and $r$, we also manually labelled the pixels causing topological difference between the predicted and ground truth edge maps, and compute precision and recall of the $DV$-masks with respect to the labeled mask by fixing $r$ from 1 to 10 and varying $K_d$ from 0 to $r - 1$ for each $r$. Then we pick $r = 10$ and 9 as they provide the top 2 average precision 0.37 and 0.35, respectively. Since we want to have a high recall to cover as many as possible pixels causing structure difference, as well as an acceptable precision to not to cover too many undesired pixels, two pairs of parameters $r = 9, K_d = 4$ and $r = 10, K_d = 4$ with recall/precision 0.74/0.29 and 0.75/0.26 are suitable choices. In order to have a low computational complexity, we use $DV$-mask with $r = 9$ and $K_d = 4$ in our experiments for evaluation. The precision-recall curve for $r = 9$ is shown in Fig. 7.5 (b).

For the LTA loss in Eq. (7.7), we treat $r$, $K_d$ and $\gamma$ as hyper-parameters. In our experiments, we use $\gamma = 3$ as we did not see significant improvement for $\gamma > 3$ after testing several values of $\gamma$. For $r$ and $K_d$, we empirically choose $r = 6$ and $K_d = 2$ for synthetic images, and $r = 8$ and $K_d = 5$ for the NCSU-CUB Foram dataset.

### 7.2.6 LTC Performance

In Fig. 7.6, we show pixel difference and DV-mask of estimated edge maps obtained by the baseline approach DLinkNet with $r = 9$ and $K_d = 4$. Large part of the pixel difference results from small shifts and width difference of edges. However, $DV$-masks only indicate the regions causing structure difference. In order to demonstrate that the proposed LTC have the capacity to quantify the topology...
Figure 7.5 (a) Average size of DV-mask for fixed $r$ as a function of $K_d$ in the foram dataset. (b) Precision-recall curve for $r = 9$ when compared to manual labels of errors. The blue dots indicate $r = 9$ and $K_d = 4$ used in our experiments for the LTC.

difference between two edge maps, we compare the proposed metric with a set of topology metrics proposed in [Weg13] as described in Section 7.2.4.

To compare both types of evaluation metrics, we create a set of synthetic image pairs with controlled topology difference. Each pair of images consists of a ground truth image created in the same way as the one in Synthetic Image and a corresponding gap image created by randomly adding gaps on the ground truth edge. Then the topology difference can be controlled by the number of gaps in gap images. That is, image pairs with more gaps have larger topology difference. We create ten sets of synthetic pairs, with the number of gaps ranging from 1 to 10 - each set contains 200 image pairs. The results of both metrics averaged over 200 images on each set are reported in Fig. 7.7. The quantities of both metrics consistently increase as the number of gaps increases as expected, which demonstrates that the proposed local topology metric is able to quantify the topology difference between edge maps as well as the PathError metric. Moreover, our local topology metric also provides the pixel locations causing the topology difference, which cannot be obtained from PathError.

7.2.7 Edge Detection Performance on Synthetic Images

First, we use the Synthetic dataset to compare the proposed LTA loss against a baseline approach and a higher-order loss approach. The methods used for comparison are listed in Table 7.1. All the methods use D-LinkNet [Zho18] as the edge detection network but with different settings of loss computation. Specifically, three types of losses are considered: Cross Entropy Loss, the loss defined in Eq. (7.6), Feature Matching, a topology-aware loss proposed in [Mos18] and Topology Weight, the cross entropy loss masked by the DV-mask. For Feature Matching, we use the same setting as [Mos18]. That is, minimize the VGG19 [SZ14] feature maps of conv1_2, conv2_2 and conv3_4.
Figure 7.6 Pixel difference and DV-mask of estimated edge maps obtained by baseline approach DLinkNet. More results can be found in Supplement.

between ground truth edge maps and predicted probability edge maps. We also use DLinkNetR to demonstrate the necessity of the hard example sampling using the proposed DV-mask. This method is the same as LTA except the DV-mask is replaced by a randomly generated fake mask. The fake mask contains a square with the size determined by the average size of DV-masks computed during the training of LTA at each epoch, and the square is randomly placed within the outer boundary of the edges for a fair comparison. The total loss of each method is the weighted summation of all applicable losses. When more than one type of loss is used, the Cross Entropy Loss is weighted by
Figure 7.7 Sample of synthetic image dataset [left] including groundtruth [top], edge probability maps with one gap [middle] and two gaps [bottom]. Comparison of proposed local topology metric (left y-axis) and PathError (right y-axis) [right]. Best viewed in color.

Figure 7.8 Performance on synthetic validation set during training. Curves with the same color correspond to the same method through all three plots. Curves have been smoothed with MATLAB function. Evaluation metrics shown include: (a) Pixel-wise F1 score; (b) Region-level unweighted IoU; and (c) Region recall. Best viewed in color.

1, the Feature Matching is weighted by 0.1 and the Topology Weight is set to be $\gamma = 3$. The scale to compute $DV$-Mask is set to be $r = 6$ and $K_d = 2$.

Table 7.1 Methods for comparison. The first three methods are baseline methods and the last two are variants of our methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cross Entropy Loss</th>
<th>Feature Matching</th>
<th>Topology Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLinkNet[Zho18]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DLinkNetR</td>
<td>Yes</td>
<td>No</td>
<td>Random</td>
</tr>
<tr>
<td>Feat[Mos18]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LTA</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>LTA</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
We create four different sets of synthetic images as training sets, each of which contains 100 synthetic image pairs. For each method listed in Table 7.1, four models are trained on those four training sets with batch size 2 and a learning rate 0.001 for 450 epochs. Those models are also validated on a separate 200 synthetic image pairs during the training phase. Fig. 7.8 shows the averaged performance (over the four training sets) of each method on the validation set during training. Here we use the F1 score of the edge pixel classification to quantify the pixel-level performance, and use region recall and unweighted overlap ratio to quantify the topology preserving performance. Overall, our LTA loss (LTA) achieves the best region-level performance. Feat obtains better region-level performance than DLinkNet, but still not better than LTA. This is surprising as Feat uses additional high-level features. We hypothesize that Feat is still sensitive to small shifts, which causes the errors due to edge gaps to be washed out by the larger contributions due to edge shifts captured by the pixel-level loss. Also, the synthetic images are less challenging in terms of visual patterns than the real images, which makes Feat less advantageous on this dataset. LTAPure only utilizes the cross entropy loss masked by the DV-masks and it has the worst F1 score while better or competitive region-level performance than other methods without topology weighting; however, it still worse than LTA. This suggests that: (1) Models trained using LTA loss tend to preserve the topological structure of edges, and in the case of the synthetic data, more closed edges are obtained from those models; (2) Loss associated with mis-classification which does not affect the topological structure is crucial for pixel-level performance; (3) Though less important, topological structure preservation still benefits from the pixel-level loss. Finally, DLinkNetR has a better performance than DLinkNet at some training steps, but the training is more unstable. Also it cannot reach the same region-level performance as LTA, which indicates that the improvement of LTA is due to the focus on the pixels causing topology difference between predicted edge maps and ground truth edge maps.

7.2.8 Edge Detection Performance on Foram Images

In this section, we apply the proposed LTA loss on the NCSU-CUB Foram dataset, and compare it with the baseline approaches including DLinkNet [Zho18] and Feat [Mos18]. As mentioned in Section 7.2.1, a two-step iterative refinement procedure is adopted on this challenging dataset where each step is an individual D-LinkNet model. The three approaches are all identical at the first iteration step while distinguished at the second iteration step. At the first step of the iteration, the cross entropy loss defined in equation 7.6 is applied. At the second step, three types of losses, the LTA loss defined in equation 7.7, the cross entropy loss defined in equation 7.6, and the feature matching loss proposed in [Mos18] are used and denoted as LTA, DLinkNet, and Feat, respectively.

During training, edge detection models at both steps are trained simultaneously. For each iteration step, cross entropy loss is weighted by 1, feature matching loss is weighted by 0.1 and the LTA loss is weighted by $\gamma = 3$. The scale to compute DV-Mask for the LTA loss is set to be $r = 8$ and $K_d = 5$. The total loss of the entire network is the weighted summation of both steps with weights 1 and 3 for the first and the second iteration step respectively. Since there are sixteen images for each sample in this dataset, to fully utilize the images, during the training phase, one of the sixteen
images of each sample is randomly picked as the input of each training epoch. When testing, the final result is obtained by averaging over the output of the sixteen images.

The three models are trained for 50 epochs with a learning rate of 0.001 and for additional 10 epochs with a learning rate of 0.0005. The batch size is 2 for all models. Sample results of the three refinement approaches are shown in Fig. 7.9. Overall, the results of Feat are closest to the ground truth labeling, as it encourages both the predicted and ground truth edge maps to be exactly the same through a more powerful higher-order feature as discussed before. DLinkNet and the proposed LTA have similar results, but the results of LTA are clearer (Fig. 7.9 w1 and w2) and have more closed edges (Fig. 7.9 w3 and w4). Although the predicted edges of LTA are not as clear as Feat, the edge probability values obtained by Feat are either pretty high for easy detected edges or pretty low for vague edges, while the edge probability values of LTA are within a relatively smaller range. This suggests that the LTA loss provides a similar regularization effect to label smoothing [Sze15].

The quantitatively evaluation results are summarized in Table 7.2. The edge thresholds are 0.4, 0.2 and 0.5 for DLinkNet, Feat and LTA respectively. The proposed LTA outperforms DLinkNet for

Figure 7.9 Sample results. (a) Input image. (b) DLinkNet output. (c) Feat output. (d) LTA output. (e) Ground truth labeling. The red boxes highlight the regions the proposed LTA loss obtains clearer and more closed edges than DLinkNet.
Table 7.2 Evaluation of the methods used for refinement. Correct, Complete, and LongShort denote the evaluation metrics Correctness, Completeness and 2Long2Short mentioned in section 7.2.4.

<table>
<thead>
<tr>
<th>Metric</th>
<th>DLinkNet</th>
<th>Feat</th>
<th>LTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge Correct</td>
<td>0.949</td>
<td>0.946</td>
<td>0.967</td>
</tr>
<tr>
<td>Correct</td>
<td>0.958</td>
<td>0.970</td>
<td>0.959</td>
</tr>
<tr>
<td>Quality</td>
<td>0.911</td>
<td>0.920</td>
<td>0.929</td>
</tr>
<tr>
<td>Region IoU</td>
<td>0.802</td>
<td>0.747</td>
<td>0.814</td>
</tr>
<tr>
<td>Recall</td>
<td>0.855</td>
<td>0.879</td>
<td>0.839</td>
</tr>
<tr>
<td>Topology Correct</td>
<td>90.82</td>
<td>93.57</td>
<td>91.07</td>
</tr>
<tr>
<td>Infeasible</td>
<td>0.12</td>
<td>0.18</td>
<td>0.19</td>
</tr>
<tr>
<td>LongShort</td>
<td>9.06</td>
<td>6.06</td>
<td>8.74</td>
</tr>
</tbody>
</table>

most of the metrics. Compared to Feat, our LTA does not provide better performance on topology-level metrics, as Feat uses an additional higher level feature. As observed in Fig. 7.10, Feat and DLinkNet have very different localized topology metric performance as a function of the threshold. From the plots, we observe that LTA provides an improvement over DLinkNet by 7.5% and Feat gives the best performance based on the metric.

7.3 Conclusion

We propose localized topology conditions to measure the structure difference between two edge maps, and a LTA loss based on these detection to guide the edge detection network to focus on preserving the topological structure. We evaluate our LTA loss on a synthetic dataset and apply to a public dataset NCSU-CUB Foram. The experiments demonstrate that the LTA loss is able to force the network to focus more on preserving topological structures including closing gaps on edges and suppressing noises between edges on images with vague and non-closed edges.
Figure 7.10 Localized topology evaluation metrics. The fraction of images that have any errors in the prediction (top) and the normalized (over the area of the image) average area of the DV-Mask (bottom) as a function of threshold $\tau$ for edge detection. These curves are obtained after training the model five times and averaging the results.
In this dissertation, several robust image segmentation algorithms are discussed including natural image segmentation, obstacle segmentation in outdoor scenes and topology-aware edge-based image segmentation. Our work focuses on developing image segmentation algorithms robust to parameter selection, changes in image qualities and conditions using tools including persistent homology and deep neural networks. Our contributions are summarized as follows:

1. Propose an innovative framework for robust image segmentation based on topological persistence and apply the framework to consensus-based image segmentation and obstacle detection in outdoor scenes;

2. Propose a novel algorithm to compute a multi-valued disparity map from stereo image pairs in order to provide a reliable occupancy map for obstacle detection under various image conditions;

3. Propose a topology-aware evaluation metric to identify structural difference between two edge maps and a topology-aware loss based on this metric to train an edge detection network focusing more on preserving topological structures.

Part of this dissertation was published, including [Wei14; GL16; Ge17; GL17]. Here, we discuss several potential future directions of our work:

1. In Chapter 3, the persistence threshold to extract the persistence region is fixed to $\tau = 0.4$. However, this threshold may be selected automatically by minimizing the cost of the changing of the threshold. According to the definition of persistence diagram for connected components, changing the persistence threshold to include or remove points on persistence diagram in the result
is equivalent to division or merge of connected components in the segmentation result. We may find a metric to measure the cost of the division and merge, so that an optimal persistence threshold can be obtained by minimizing the cost. For example, when consider the cost of points removal, that is, merge of connected components in the segmentation by increasing the persistence threshold, one potential cost can be measured by the size of connected components and the color difference between the two connected components before merged.

2. In Chapter 4, the boundary of detected obstacles is not smooth enough and the class of each obstacle is still unknown, since the entire segmentation process is only based on the disparity map. One potential extension can be the semantic segmentation by integrating color information using Conditional Random Fields (CRFs) \([\text{Laf01]}\). The unary potential can be obtained from pixel-wise classification of the color image. Let \(y\) be a node of the CRF and \(k\) be the \(k\)-th segments in the obstacle segmentation result. To integrate our segmentation result to a CRF model, we can define the pairwise potential as

\[
V_2(y_i, y_j) = \begin{cases} 
-\beta, & \text{if } y_i = y_j, \text{ and } y_i \in k_l, y_j \in k_m, l = m \\
\beta_1, & \text{if } y_i \neq y_j, \text{ and } y_i \in k_l, y_j \in k_m, l \neq m \\
\beta_2, & \text{O.W.} 
\end{cases}
\]

(8.1)

where \(\beta, \beta_1\) and \(\beta_2\) are pre-defined constants. This pairwise potential encourages the nodes within the same detected obstacle to be the same region in the semantic segmentation result.

3. In Chapter 7, the \(DV\)-Mask is used to indicate error pixels which cause the topology difference between the predicted edge map and the ground truth. Then the loss of those pixels are increased by a fixed weight. If a range of \(K_d\) is used, then a new \(DV\)-Mask can be obtained with each mask pixel assigned the maximum \(K_d\) detecting this pixel. Thus one potential extension is to use this new \(DV\)-Mask to provide a more gradual weighting of the loss. Another possible direction is an attention-based topology-aware edge refinement. The \(DV\)-Mask can be used as an automatic annotation tool to label on imperfect edge maps the regions causing a specific type of topological difference from the ground truth, such as edge gaps. Then an detection model, such as YOLOv3 \([\text{RF18]}\), can be trained to detect those regions on a imperfect edge map. Finally several small regions containing topological error are extracted and can be processed using a lighter module to recover the desirable edge structures, instead of using a heavier recovery module to directly apply on the entire edge map.
[Pro] https://research.ece.ncsu.edu/aros/forum-identification/.


Chazal, F. et al. “Persistence-Based Clustering in Riemannian Manifolds”. *Symp. on Computational Geometry (SoCG)*. 2011.


