AN ABSTRACT OF THE THESIS OF

Michael Lam for the degree of Master of Science in Computer Science presented on May 15, 2014.

Title: Object Detection in Biological Images Using a Search-Based Framework

Abstract approved: __________________________________________________________

               Sinisa Todorovic        Thomas Dietterich

This thesis addresses a basic problem in computer vision, that of semantic labeling of images. Our work is aimed at object detection in biological images for evolutionary biology research. In particular, our goal is to detect nematocysts in Scanning Electron Microscope (SEM) images. This biological domain presents challenges for existing approaches developed to address other domains (e.g. natural scenes). An image may show more than one nematocyst under partial occlusion and amidst background clutter (e.g., cellular debris). We formulate the detection of nematocysts as labeling of a regular grid of patches, where patches occupied by a nematocyst are labeled as foreground and the other patches are labeled as background. This structured prediction problem is addressed using the $\mathcal{HC}$-Search framework. $\mathcal{HC}$-Search seeks a solution in the space of candidate labelings of image patches. It employs a heuristic function ($H$) to uncover high-quality candidate labelings, and then applies a cost function ($C$) to select the best-scoring prediction among the candidates. The heuristic and cost functions are learned on
training data using imitation learning. Our key novelty is a formulation of a stochastic search space and pruning function that improve the accuracy and efficiency of $HC$-search. The stochastic search space is generated by a successor function that proposes random candidate labelings based on clusterings of patches in the image instead of employing the traditional method of flipping the labels of individual patches. Additionally, we introduce a pruning function ($P$) to remove bad candidate labelings generated by the successor function. This induces a sparser search space, which improves the efficiency of search. We compare our approach with the Conditional Random Field (CRF) model, a well-understood framework for scene labeling. While CRF inference typically yields good performance on natural scenes, our results demonstrate that CRFs perform poorly on the nematocyst images and that $HC$-Search outperforms CRFs.
Object Detection in Biological Images Using a Search-Based Framework

by

Michael Lam

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented May 15, 2014
Commencement June 2014
Master of Science thesis of Michael Lam presented on May 15, 2014.

APPROVED:

________________________________________
Co-Major Professor, representing Computer Science

________________________________________
Co-Major Professor, representing Computer Science

________________________________________
Director of the School of Electrical Engineering and Computer Science

________________________________________
Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

________________________________________
Michael Lam, Author
ACKNOWLEDGEMENTS

I would like to thank everyone who has contributed to this thesis. This includes my advisors Sinisa Todorovic and Tom Dietterich for their professional support and mentorship; my committee members for their time; graduate student Jana Doppa for his close mentorship and extensive help for this project; graduate student Shell Hu for assisting with code for this project; collaborators Abigail Reft and Marymegan Daly for collecting biological images; former student Yaofei Feng for annotating the dataset; and my graduate labmates for healthy, scholarly discussions. I would also like to acknowledge the ARCS Portland Foundation for their scholarship support.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Overview of Approach</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Contributions</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Literature Review</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Structured Prediction</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Scene Labeling</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>Search-Based Structured Prediction</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Problem Background</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Baseline Approaches</td>
<td>13</td>
</tr>
<tr>
<td>4.1</td>
<td>Conditional Random Field</td>
<td>13</td>
</tr>
<tr>
<td>4.2</td>
<td>IID Classifier</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>Proposed Approach</td>
<td>17</td>
</tr>
<tr>
<td>5.1</td>
<td>Search Space</td>
<td>17</td>
</tr>
<tr>
<td>5.2</td>
<td>Stochastic Successor Function</td>
<td>20</td>
</tr>
<tr>
<td>5.3</td>
<td>Search Strategy</td>
<td>22</td>
</tr>
<tr>
<td>5.4</td>
<td>Heuristic and Cost Function Learning</td>
<td>22</td>
</tr>
<tr>
<td>5.5</td>
<td>Pruning Function</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Results</td>
<td>27</td>
</tr>
<tr>
<td>6.1</td>
<td>Dataset Description</td>
<td>27</td>
</tr>
<tr>
<td>6.2</td>
<td>Evaluation Setup and Metrics</td>
<td>28</td>
</tr>
<tr>
<td>6.3</td>
<td>Methods</td>
<td>29</td>
</tr>
<tr>
<td>6.4</td>
<td>Quantitative Results: Baselines</td>
<td>30</td>
</tr>
<tr>
<td>6.5</td>
<td>Quantitative Results: Our Extensions</td>
<td>34</td>
</tr>
<tr>
<td>6.6</td>
<td>Qualitative Results</td>
<td>38</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>44</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Example images of nematocysts</td>
<td>11</td>
</tr>
<tr>
<td>5.1</td>
<td>High level overview of $\mathcal{HC}$-Search</td>
<td>19</td>
</tr>
<tr>
<td>6.1</td>
<td>Curves for Flipbit and stochastic spaces</td>
<td>33</td>
</tr>
<tr>
<td>6.2</td>
<td>Pruning parameter experiment</td>
<td>36</td>
</tr>
<tr>
<td>6.3</td>
<td>Comparison of Flipbit and stochastic spaces and pruning</td>
<td>37</td>
</tr>
<tr>
<td>6.4</td>
<td>Example good detections of nematocysts</td>
<td>40</td>
</tr>
<tr>
<td>6.5</td>
<td>Example good challenging detections of nematocysts</td>
<td>41</td>
</tr>
<tr>
<td>6.6</td>
<td>Qualitative comparison of detections with $\mathcal{HC}$-Search</td>
<td>42</td>
</tr>
<tr>
<td>6.7</td>
<td>Example bad challenging detections of nematocysts</td>
<td>43</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Baseline results</td>
<td>31</td>
</tr>
<tr>
<td>6.2</td>
<td>Error decomposition of $\mathcal{H}C$-Search</td>
<td>34</td>
</tr>
<tr>
<td>6.3</td>
<td>$\mathcal{H}C$-Search results with stochastic and pruning</td>
<td>35</td>
</tr>
<tr>
<td>6.4</td>
<td>Average number of candidates generated</td>
<td>38</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

One of the fundamental goals of computer vision is to automatically identify what objects are in an image. In particular, the goal is to label every pixel in an image with a semantic class label. For instance, we would like to find out if a particular pixel of a natural scene belongs to the tree, car or background class. This problem is known as scene labeling.

This computer vision problem arises in a wide range of domains, including the analysis of biological images. With the proliferation of digital cameras, biologists increasingly use images of specimens for various studies, including ecology, evolutionary biology and so on. In particular, evolutionary biologists are interested in identifying biologically relevant characters in images of different species. These characters typically correspond to object parts. So to automate this process, object detection approaches are needed for identifying a specimen in biological images.

Biological images present significant challenges to the state of the art. The imaging of organisms in their natural settings is not controllable enough for computer vision. For example, such images often contain significant background clutter that sometimes looks very similar to the specimens. This background clutter is unavoidable due to the imaging process. Specimens may also have variations in articulation and can be missing parts. They are imaged under partial occlusion from different viewpoints. Rarely do we see the aforementioned challenges in general images of natural scenes.
Existing approaches are not suitable for scene labeling of these biological images because they make restrictive assumptions about these scenes. For example, objects in natural scenes are prominently featured. In other words, there are enough pixels to allow an algorithm to see the details of the object. Also the object is salient and very different from the background. All these assumptions do not hold in biological images.

We focus this work on detecting nematocysts in images. Nematocysts are explosive sub-cellular capsules with elongated surfaces characterized by spines whose properties represent important phenomic characters for evolutionary studies. We formulate the detection of nematocysts as labeling of a regular grid of patches, where patches occupied by a nematocyst are labeled as foreground and the other patches are labeled as background. Our goal is to identify image patches that are occupied by nematocysts.

1.1 Overview of Approach

To approach this problem, we first experimented with a well-understood framework for scene labeling called the Conditional Random Field (CRF). The CRF defines a parametric posterior distribution over the outputs (labels) given observed image features. CRFs use a single parametric model for inference, where inference is posed as finding the assignment that maximizes the joint posterior distribution as estimated by the CRF. However, inference is approximate since exact inference is usually intractable. Our results indicate that CRF inference on our biology domain is poor and call for a different approach.

Therefore we developed an approach for scene labeling that can handle the chal-
lenges of biological images. Our main hypothesis is that the challenges of this problem require a multi-component learning architecture where each computational step must be highly specialized for producing an ultimately good solution as opposed to having a single model. For instance, the CRF only learns one set of model parameters for the posterior distribution. We believe this is not strong enough in our domain. Thus, inspired by recent successes of search-based formulations of structured prediction, we employ one such multi-component architecture for scene labeling called HC-Search. This approach has never been used before for computer vision.

HC-Search employs two learned components in a multi-component architecture as opposed to a single component (the CRF scoring function) in a CRF. HC-Search learns a heuristic function $H$ that guides the search for high quality candidate labelings and a cost function $C$ that selects the best labeling from among these high quality candidates. While HC-Search inference is also approximate, prior work has shown that greedy search works quite well combined with an effective search space to yield good results. Therefore we formulate a stochastic search space for HC-Search that is specific for scene labeling. The stochastic search space is generated by a successor function that proposes random candidate labelings based on clusterings of patches in the image. This decreases the expected target depth of the solution. In addition, we introduce another extension to HC-Search. We introduce a pruning function $P$ that removes bad candidate labelings generated by the successor function. This induces a sparser search space to improve efficiency of search.

Our results indicate that HC-Search outperforms CRF on biological images. Quantitatively and qualitatively, HC-Search improves recall over CRF, and this improves
overall accuracy. With our added extensions, inference is also more efficient.

1.2 Contributions

We made several contributions. First of all, our evaluation demonstrates that our hypothesis is correct. A multi-component architecture such as $HC$-Search sufficiently addresses the challenges of biological images. Our results indicate that $HC$-Search outperforms CRF. We have obtained a level of detection accuracy that is useful for biologists. Secondly, we introduced two extensions to $HC$-Search that improve the efficiency and accuracy of search. The stochastic successor function proposes random candidate labelings based on clusterings of patches to flip more patches in a single search step instead of flipping a single patch. The pruning function eliminates bad candidate labelings to induce a sparser search space. Finally, we compiled a new dataset for computer vision. These are Scanning Electron Microscope (SEM) images that contain nematocysts. We also provide the groundtruth annotations.
Chapter 2: Literature Review

There is a large volume of literature on scene labeling and structured prediction. An exhaustive review is beyond our scope. In this chapter, we will first introduce the structured prediction framework. This framework generalizes our task. We will then present literature on scene labeling using Conditional Random Fields (CRF). The CRF is a well-understood framework for scene labeling. Finally, we will review search-based structured prediction approaches, which have recently gained traction in the artificial intelligence community for a wide range of problems.

2.1 Structured Prediction

In structured prediction problems, the predictor must produce a structured output $y$ given a structured input $x$. A standard approach to structured prediction is to learn a cost function $C(x, y)$ that scores a potential structured output given a structured input. Usually the cost function is a linear function of features and model parameters, in other words, $C(x, y) = w^T \phi(x, y)$ where $\phi(x, y)$ are features and $w$ are the model parameters. The goal then is to learn the model parameters from training data. For the task of inference, given a structured input and the cost function, the predictor for structured prediction finds the minimum cost output among all possible structured outputs given the structured input: $\hat{y} = \arg \min_y C(x, y)$
Approaches such as CRFs [13], Max-Margin Markov Networks [20] and Structured SVMs [22] represent the cost function as a linear model. However, the exact argmin problem is often intractable due to the exponential space of outputs. Exact inference is only possible when the dependency structure among features contains no loops, i.e. a tree or chain. Simplifying the features or dependency structure allows for tractable inference but at the cost of prediction accuracy. Alternatively, one can turn to heuristic approximate inference algorithms. However, their solutions are difficult to characterize and their behavior is not well understood. For our biology domain, which has not been explored before, it is necessary to obtain solutions with high prediction accuracy through a principled manner.

2.2 Scene Labeling

Scene labeling is an instance of structured prediction. It is the problem of assigning a semantic class label to every pixel in an image. In scene labeling, the structured input is the image and the structured output is a semantic labeling of the image.

The Conditional Random Field (CRF) [13] is one of the most popular and well-understood models in computer vision for scene labeling, object recognition and segmentation [10, 19, 23, 21, 17, 9, 16]. A CRF defines a parametric posterior distribution over the outputs (labels), $y$, given observed image features, $x$, in a factored form:

$$P(y|x, w) = \frac{1}{Z(x, w)} e^{w \cdot \phi(x, y)},$$

where $w$ are the parameters, $Z(x, w)$ is the partition function, and the features, $\phi(x, y)$, decompose over the cliques in the underlying graphical model. Observable image features, $x$, could be pixels [10, 19], image patches [23, 21],
or image regions [17, 9, 16]. These features at different sites of the image are encoded in the CRF as 1-cliques, also known unary potentials, $\phi_{\text{unary}}(x_i, y_i)$. CRFs are good at capturing pairwise smoothness and relationship between parts. These are encoded with 2-cliques, also known as pairwise potentials, $\phi_{\text{pairwise}}(x_i, x_j; y_i, y_j)$. CRFs can also capture more global cues such as object shapes. These are encoded with higher order cliques, $\phi_{\text{global}}(x_c, y_c)$, which are much more difficult to encode in CRFs.

Inference is typically posed as finding the joint MAP assignment that maximizes the posterior distribution: $\hat{y} = \arg\max_{y \in Y} P(y|x, w)$. Inference is generally intractable due to the exponential space of outputs. Therefore inference is generally approximate. Parameter learning is usually formulated as minimizing the negative conditional log-likelihood of the data. Learning is also generally intractable because it involves repeated calls to the inference procedure, which is approximate. It is not clear how approximate inference affects learning.

Well-known approximate inference algorithms in computer vision include Loopy Belief Propagation (LBP) and Iterated Conditional Modes (ICM). Belief Propagation (BP) is a message-passing algorithm for CRF inference, which yields exact inference for tree graphical models. LBP is an extension of BP for general graphs but yields approximate inference. ICM is an inference algorithm that iteratively maximizes the probability of each variable in the CRF conditioned on the rest. Another well-known inference algorithm is GraphCuts [2]. GraphCuts solves inference by converting the graph into a network flow. It is an exact inference algorithm only when all pairwise potentials in the graphical model involve submodular functions and the label set is binary. Otherwise GraphCuts remains an approximate inference algorithm. In summary, exact
inference is only possible when the dependency structure among features contains no loops or all pairwise dependencies involve submodular functions.

These are well-known issues with general graphical models that affect the performance of prediction. Unfortunately, our biological domain is even more difficult and these issues only further affect our performance. Therefore, we resort to a different framework for solving our problem, as reviewed in the next section.

2.3 Search-Based Structured Prediction

Search-based structured prediction approaches have recently gained traction in the artificial intelligence community for a wide range of problems. These approaches employ search to find the best predicted output from among a subset of all possible outputs.

One class of search-based structured prediction approaches relies on Monte Carlo Markov Chains (MCMC). This includes Gibbs sampling and the Metropolis-Hastings algorithm. The proposal distribution of the Metropolis-Hastings algorithm can be thought of as a “heuristic” that guides the “search,” which is a random walk. [1] proposes next states in the “search” by flipping labels of connected components in the image, which is faster than Gibbs sampling. MCMC approaches are appealing due to theoretical convergence guarantees resulting from reversible jumps in Markov Chains. However, MCMC approaches require a very long time to converge in practice in general.

By contrast, search can be done with a “learned” heuristic. A fairly new search-based structured prediction framework called $\mathcal{HC}$-Search places fewer restrictions on the cost function and uses a heuristic function to reduce the difficulty of the task performed by
the cost function [5, 6]. It is a multi-component architecture by employing two models, the heuristic and cost functions, rather than one model like the CRF. The key idea of the HC-Search approach is to search over the space of outputs by learning a heuristic function $H$ to guide the search toward high-quality outputs and a cost function $C$ to select the best candidate output generated by the search procedure. The framework can be re-interpreted as a search problem to correct the mistakes of an initial prediction, which may be suitable for challenging structured problems such as the biology domain. Their work shows state-of-the-art results on images of natural scenes. However, natural scenes have restrictive assumptions and their work does not address our more challenging biology domain. Therefore we investigate the biology domain with HC-Search.

Work has been done to improve the search space with Limited Discrepancy Search (LDS) [4] and a sparser version of it [7]. The method starts with a baseline greedy search algorithm. LDS assumes that this greedy method has made a small number of errors, but those errors could arise at any point in the search space. LDS defines a “discrepancy” as a one-step departure from the greedy path. LDS explores all 1-discrepancies, then all 2-discrepancies, and so on. Doppa, et al. in [8] explores a method similar to ours, but it does not use an IID classifier to construct the initial prediction. All these extensions do not address the inherent challenges of our biology domain and are not specific to scene labeling. We investigate our own extensions to HC-Search that specifically address scene labeling in our biology domain.

This thesis continues the work of [14]. Their work with HC-Search has shown some success on biological images but the heuristic function needed improvement. Our work addresses this issue with an improved search space and a pruning function.
Chapter 3: Problem Background

In this chapter we motivate our problem from a biology perspective. This thesis addresses the problem of object detection via scene labeling in scanning electron microscope (SEM) images for the purposes of morphological characterization of cnidae. This work focuses on nematocysts, one kind of cnida, illustrated in Figure 3.1.

A cnida (plural cnidae) is an explosive sub-cellular capsule that fires debris when it discharges. It is produced by a special cell called a cnidocyte. Because cnidae manifest both extreme morphological cell-level simplicity and wide biological diversity, cnidae provide a great opportunity to investigate fundamental questions in biology, including constraints and convergence in morphology [3]. Of particular interest is a morphological characterization of the basal tubules of nematocysts, marked yellow in the images shown in Figure 3.1. This is because surfaces of the basal tubules are characterized by spines whose shapes, lengths, and density of placement along the surface represent important phonemic characters for evolutionary studies [18].

Biological studies of nematocyst images are currently conducted through visual inspection and manual annotation, requiring prohibitive amounts of costly expert time. Typically this limits the studies to small image collections and narrows their scope. In this paper, we explore an opportunity for computer vision to help biologists in their analysis of nematocyst images by automatically detecting the basal tubules. The detection of basal tubules readily gives information about the size and shape of the nematocyst
Figure 3.1: Example images of nematocysts from our dataset. Detecting textured, elongated, highly deformable basal tubules of nematocyst organisms (marked yellow) against background clutter is very challenging.

useful for morphological studies.

As can be seen in Figure 3.1, images of nematocysts present significant challenges to the state of the art in computer vision. The basal tubules are relatively thin, elongated, and highly deformable objects covered with spines. They are typically imaged against significant background clutter, consisting of mucus and cellular debris. The clutter is unavoidable, since it is extremely difficult to isolate individual nematocysts during image acquisition. Thin, elongated particles of debris appear very similar to the basal tubule. The texture of debris appears very similar to the texture of spines along the surface of the basal tubule. In addition, some images may not show the entire basal tubule, since the tubule may be partially occluded by clutter or may extend beyond the image frame. Nematocysts are often damaged naturally and sometimes damaged through preparation so that large parts of the basal tubules may not be physically present in the image. These challenges are rare in general images of natural scenes.

In this thesis, we formulate detection of the basal tubules as the problem of assign-
ing a binary labeling to a regular grid of image patches. Patches that fall on the basal tubule are to be assigned the label “1”, and patches that fall on background are to be assigned the label “0”. In other words, patches that fall on the basal tubule are considered foreground and everything else is considered background. Related work mostly focuses on image classification for accelerating biological studies [15]. In contrast, this thesis focuses on object detection and localization for accelerating biological studies.
Chapter 4: Baseline Approaches

In this chapter we present some baseline approaches to the scene labeling problem for our biological domain. We introduce the CRF and IID Classifier as baseline approaches. In the results chapter, we will show that these baseline experiments yield poor results, which motivates our proposed approach.

4.1 Conditional Random Field

The Conditional Random Field (CRF) is a well-understood framework for structured learning and inference in computer vision [12, 11]. A CRF defines a parametric posterior distribution over the outputs (labels), $y$, given observed image features, $x$, in a factored form: $P(y|x, w) = \frac{1}{Z(x, w)} e^{w \cdot \phi(x, y)}$, where $w$ are the parameters, $Z(x, w)$ is the partition function, and the features, $\phi(x, y)$, decompose over the cliques in the underlying graphical model.

The task of inference is to find the joint MAP assignment that maximizes the posterior distribution: $\hat{y} = \arg\max_{y \in Y} P(y|x, w)$. This is generally intractable since the space of outputs is exponential, and so inference is usually approximate. Parameter learning is usually formulated as finding the parameter values that minimize the negative conditional log-likelihood of the data. This requires repeated calls to the inference procedure, and therefore is also generally intractable and approximate. Well-known ap-
proximate inference algorithms in vision include Loopy Belief Propagation (LBP) and Iterated Conditional Modes (ICM). Graph Cuts is an exact inference algorithm only when the graphical model has a special structure, otherwise it is approximate [2].

In our model, the patches are organized in a graph, $G = (V, E)$, where $V$ and $E$ are sets of nodes and edges. The nodes $i = 1, 2, \cdots, |V|$ correspond to patches in the image, and edges $(i, j) \in E$ capture their spatial relations. For our biological domain, we treat the patch as organized in a regular grid with 4-connected neighbors. For other domains, the connection of neighboring patches or superpixels is typically defined by an image-specific adjacency matrix. The variable $x_i$ is the 128-dimensional SIFT descriptor vector for node $i$, $y_i$ is the proposed label for node $i$, and $\Psi_u(x_i, y_i)$ is the (learned) potential function that measures the compatibility of $x_i$ and $y_i$. Every edge $(i, j) \in E$ is described by a pairwise feature, $\Psi_{\text{pair}}(x_i, x_j, y_i, y_j)$, indicating the compatibility between patches $i$ and $j$ with the corresponding labeling $y_i$ and $y_j$

$$\Psi_{\text{pair}}(x_i, x_j, y_i, y_j) = \begin{cases} 0 & \text{if } y_i = y_j, \\ \exp(-\beta |x_i - x_j|^2) & \text{if } y_i \neq y_j, \end{cases}$$ (4.1)

where $\beta$ is a parameter. The pairwise feature $\Psi_{\text{pair}}(x_i, x_j, y_i, y_j)$ encourages neighboring patches to take the same label. Larger values of $\Psi_{\text{pair}}(x_i, x_j, y_i, y_j)$ correspond to lower compatibility of $x_i$ and $y_i$.

Let the set of all patch descriptors be denoted $x = \{x_i : i = 1, \cdots, |V|\}$, and let the set of all patch labels be denoted $y = \{y_i : i = 1, \cdots, |V|\}$, where $y_i \in \{0, 1\}$. We investigate two different CRF formulations, referred to as pairwise CRFs and pyramid CRFs, as explained below.
**Pairwise CRF.** The pairwise CRF, given by (4.2), corresponds to the formulation that contains the unary and pairwise features of image patches, with the standard 4-connected neighborhood of every patch on the image lattice:

\[
w \cdot \phi(x, y) = \sum_{i \in V} w_u \cdot \Psi_u(x_i, y_i) + \sum_{i \in V} \sum_{j \in N_i} w_{\text{pair}} \cdot \Psi_{\text{pair}}(x_i, x_j, y_i, y_j),
\]

(4.2)

**Pyramid CRFs.** The pyramid CRF, given by (4.3), contains additional pyramid features, \( \Psi_{\text{pyr}}(x_i, x_j, y_i, y_j) \). The graphical model now contains a grid of patches from a downsampled image by a factor of 2, in order to approximate higher-order features. Each node \( i \) from the downsampled layer is connected to its four corresponding child nodes \( k \in C_i \) in the original image. The purpose of this formulation is to account for multiple scales in the image.

\[
w \cdot \phi(x, y) = \sum_{i \in V} w_u \cdot \Psi_u(x_i, y_i) + \sum_{i \in V} \sum_{j \in N_i} w_{\text{pair}} \cdot \Psi_{\text{pair}}(x_i, x_j, y_i, y_j)
+ \sum_{i \in V, k \in C_i} w_{\text{pyr}} \cdot \Psi_{\text{pyr}}(x_i, x_k, y_i, y_k).
\]

(4.3)

We investigate these two CRF models combined with the well-known inference algorithms: ICM, LBP, and Graph-Cuts.

### 4.2 IID Classifier

A simpler baseline approach for scene labeling is to learn an identically and independently distributed (IID) classifier on patch or superpixel features and employ this classi-
fier to make independent label predictions for every image patch or superpixel. Example IID classifiers include Logistic Regression and Support Vector Machines (SVM). In fact, Logistic Regression is a special case of the CRF. Applying an IID Classifier over all patches can be thought of as a CRF with only unary potentials.

This solution is potentially unsatisfactory, because it does not take advantage of the relationships among neighboring patches. Structured approaches such as CRFs [12, 11] and $H_C$-Search [5] leverage the structure in the problem by accounting for relationships between inputs and outputs. However, in some domains an IID classifier can be sufficient. Therefore it is necessary to investigate how much CRF benefits over IID classifier for our biological domain, where it has not been well studied before.
Chapter 5: Proposed Approach

This chapter presents our proposed approach based on $\mathcal{HC}$-Search [5]. The key elements of $\mathcal{HC}$-Search include the search space over complete outputs $S_o$; the search strategy $A$; the heuristic function $\mathcal{H} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ to guide the search towards high-quality outputs; and the cost function $\mathcal{C} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ to score the candidate outputs generated by the search procedure. A high level overview of $\mathcal{HC}$-Search framework is shown in Figure 5.1.

Our extension to $\mathcal{HC}$-Search includes a pruning function $\mathcal{P} : \mathcal{X} \times \mathcal{Y} \times \mathcal{A} \mapsto \mathbb{R}$ that eliminates bad actions from the successor function specified by the search space (see section 5.5). Because it is computationally cheap, it can speed up the search process. Furthermore, because it shrinks the number of nodes explored in the search, it can simplify the task of both the heuristic and cost functions. We also specify a stochastic successor function for $\mathcal{HC}$-Search that proposes random candidates based on clusterings of the patches (see section 5.2). Below we explain all these elements and then describe how to learn the heuristic, cost and pruning functions.

5.1 Search Space

Every state in $S_o$ consists of an input-output pair, $(x, y)$, representing the possibility of predicting $y$ as the output for input image $x$ (see Figure 5.1). Such a search space is
defined in terms of two functions: 1) Initial state function, $I$, such that $I(x)$ returns an initial state for input $x$; and 2) Successor function, $S$, such that for any state $(x, y)$, $S((x, y))$ returns a set of next states $\{(x, y_1), \cdots, (x, y_k)\}$ that share the same input $x$.

The effectiveness of HC-Search depends critically on the quality of the search space. The quality of a search space can be understood in terms of the expected number of search steps needed to uncover the target output $y^*$. For most search procedures, the time required to find $y^*$ will grow as the depth of the target in the search space increases. Thus, one way to quantify the expected amount of search, independently of the specific search strategy, is by considering the expected depth of target outputs $y^*$. In particular, for a given input-output pair $(x, y^*)$, the target depth $d$ is defined as the minimum depth at which we can find a state corresponding to the target output $y^*$. By this definition, the expected target depth of our search space is equal to the expected number of errors in the output corresponding to the initial state.

The specific search space that we investigate leverages the IID classifier. Our initial state $I(x)$ corresponds to the predictions made by a logistic regression classifier. The logistic regression classifier also provides a confidence value for each label prediction, i.e. a label distribution, which we will leverage in the successor function.

There are several ways the successor function $S$ can generate a set of next states. We describe some simple ways before discussing our stochastic approach in section 5.2. The simplest way is to flip the label of every image patch; this search space is similar to the Flipbit space defined in [4]. However, this yields a very large search space when there are many nodes and when the label set is large. To reduce the search space, we can consider a subset of these candidates by computing a set of image patches where
Figure 5.1: A high level overview of \( H\mathcal{C}\)-Search. Given input \( x \) and a search space, \( S_o \), we first instantiate a search space over complete outputs. Each search node in this space consists of a input-output pair (i.e., input image and basal tubule detection). Next, we run a search procedure \( A \) guided by the heuristic function \( H \) for a time bound \( \tau \) (no. of search steps). The highlighted nodes correspond to the search trajectory traversed by the search procedure, in this case greedy search. We return the least cost output \( \hat{y} \) (basal tubule detection) that is uncovered during the search as the prediction for input \( x \).
the classifier has low confidence and generating one successor for each patch with the corresponding $y$ value flipped. We employ the conditional probability of the logistic regression IID classifier as the confidence measure.

### 5.2 Stochastic Successor Function

We formulate a stochastic search space for $HC$-Search. The stochastic search space is generated by a successor function that proposes random candidate labelings based on clusterings of patches in the image. Whereas the Flipbit successor function only flips one patch for each search step, the stochastic successor function jointly flips multiple patches in each search step while explicitly accounting for the spatial extent of objects in the image. This is done in the spirit of [1]. The purpose of flipping multiple patches at a time is to speed up search to reduce the expected target depth of the solution. The stochastic successor function proposes larger moves by 1) cutting edges probabilistically to form connected components and 2) flipping labels within a connected component.

We first probabilistically cut edges in $E$ whose pairwise potentials or weights are below a threshold drawn at random from a distribution. Recall that $\Psi_u(x_i)$ is a feature descriptor vector that describes patch $i$. Note that since the normalized feature descriptor is a histogram, we treat it as a multinomial distribution. Then the strength or weight of the edge $e_{ij} \in E$ between patches $i$ and $j$ can be measured as

$$
\Psi_{\text{pair}}(x_i, x_j) = \exp \left\{ -\frac{T}{2} \left[ D_{KL} (\Psi_u(x_i)||\Psi_u(x_j)) + D_{KL} (\Psi_u(x_j)||\Psi_u(x_i)) \right] \right\} \quad (5.1)
$$
where $D_{KL}(P||Q)$ measures the KL divergence of probability distribution $Q$ from probability distribution $P$, and $T$ is a parameter. The intuition of this equation is to measure the “similarity” between two normalized feature descriptors (treated as probability distributions) by averaging the KL divergence from both directions. In practice one can use a strict measure such as the chi-squared distance between two histograms, but this equation empirically worked best for us.

An edge is cut if its weight falls below a random threshold $\theta$, where $\theta$ is drawn from a uniform distribution on $[0, 1]$. Therefore an edge with smaller weight is more likely to be cut than an edge with larger weight. This is meaningful as we should expect that edges with small weights should connect two patches that are dissimilar while edges with large weights should connect two patches that are similar. This is appropriate for computer vision, because we are likely to cut edges that are at the boundaries of objects in the image.

After cutting edges randomly, the graph is now partitioned into connected components. We denote a connected component as patches that are connected through the edges that survived the cut. Furthermore we denote a connected domain as patches that are reachable to one another through the edges that survived the cut and additionally all the patches have the same label. A connected component contains connected domains. The goal now is to flip the label of a connected domain. For the binary case such as our biological domain, this is simple, as we only have one other label. Thus our candidate set consists of all connected domains with their label flipped.
5.3 Search Strategy

The role of the search procedure is to uncover high-quality outputs, guided by the heuristic function $\mathcal{H}$. Prior work [4, 5] has shown that greedy search works quite well when combined with an effective search space. We investigate $\mathcal{HC}$-Search with greedy search. Given an input $x$, greedy search traverses a path of length $\tau$ through the search space, selecting as the next state, the best successor of the current state according to the heuristic. Specifically, if $s_i$ is the state at search step $i$, greedy search selects

$$s_{i+1} = \arg \min_{s \in S(s_i)} \mathcal{H}(s), \text{ where } s_0 = I(x).$$

Making Predictions. Given an input image $x$, and a prediction time bound $\tau$, $\mathcal{HC}$-Search traverses the search space starting at $I(x)$, using the search procedure $A$, guided by the heuristic function $\mathcal{H}$, until the time bound is exceeded. It then scores each visited state $s$ according to $C(s)$ and returns the $\hat{y}$ of the lowest-cost state as the predicted output.

Let $y^*_\mathcal{H}$ denote the best output that $\mathcal{HC}$-Search could possibly return when using $\mathcal{H}$, and let $\hat{y}$ denote the output that it actually returns. Also, let $\mathcal{Y}_\mathcal{H}(x)$ be the set of candidate outputs generated using heuristic $\mathcal{H}$ for a given input $x$. Then, we define

$$y^*_\mathcal{H} = \arg \min_{y \in \mathcal{Y}_\mathcal{H}(x)} L(x, y, y^*), \quad \hat{y} = \arg \min_{y \in \mathcal{Y}_\mathcal{H}(x)} C(x, y).$$

(5.2)

5.4 Heuristic and Cost Function Learning

The error of $\mathcal{HC}$-Search, $\epsilon_{\mathcal{HC}}$, for a given $\mathcal{H}$ and $C$ can be decomposed into two parts: 1) Generation error, $\epsilon_{\mathcal{H}}$, due to $\mathcal{H}$ not generating high-quality outputs; and 2) Selection error, $\epsilon_{C|\mathcal{H}}$, the additional error (conditional on $\mathcal{H}$) due to $C$ not selecting the best loss
output generated by \( \mathcal{H} \). Guided by the error decomposition in (5.3), the learning approach optimizes the overall error, \( \epsilon_{\mathcal{HC}} \), in a greedy stage-wise manner by first training \( \mathcal{H} \) to minimize \( \epsilon_{\mathcal{H}} \), and then, training \( \mathcal{C} \) to minimize \( \epsilon_{\mathcal{C}|\mathcal{H}} \) conditioned on \( \mathcal{H} \).

\[
\epsilon_{\mathcal{HC}} = \frac{L(x, y_{\mathcal{H}}, y^*)}{\epsilon_{\mathcal{H}}} + \frac{L(x, \hat{y}, y^*)}{\epsilon_{\mathcal{C}|\mathcal{H}}} - L(x, y_{\mathcal{H}}, y^*) \tag{5.3}
\]

\( \mathcal{H} \) is trained by imitating the search decisions made by the true loss function (available only for training data). We run the search procedure \( A \) for a time bound of \( \tau \) for input \( x \) using a heuristic equal to the true loss function, i.e. \( \mathcal{H}(x, y) = L(x, y, y^*) \), and record a set of ranking constraints that are sufficient to reproduce the search behavior. For greedy search, at every search step \( i \), we include one ranking constraint for every node \( (x, y) \in C_i \setminus (x, y_{best}) \), such that \( \mathcal{H}(x, y_{best}) < \mathcal{H}(x, y) \), where \( (x, y_{best}) \) is the best node in the candidate set \( C_i \) (ties are broken by a random tie breaker). The aggregate set of ranking examples is given to a rank learner (e.g., SVM-Rank) to learn \( \mathcal{H} \).

\( \mathcal{C} \) is trained to score the outputs \( \mathcal{Y}_{\mathcal{H}}(x) \) generated by \( \mathcal{H} \) according to their true losses. Specifically, this training is formulated as a bi-partite ranking problem to rank all the best loss outputs \( \mathcal{Y}_{best} \) higher than all the non-best loss outputs \( \mathcal{Y}_{\mathcal{H}}(x) \setminus \mathcal{Y}_{best} \).

**Advantages of \( \mathcal{HC} \)-Search** relative to other structured prediction approaches, including CRFs, are as follows. First, it scales gracefully with the complexity of the dependency structure of features. In particular, we are free to increase the complexity of \( \mathcal{H} \) and \( \mathcal{C} \) (e.g., by including higher-order features) without considering its impact on the inference complexity. [4, 5] show that employing higher-order features results in significant improvements. Second, the terms of the error decomposition in (5.3) can be easily
measured for a learned \((\mathcal{H},\mathcal{C})\) pair, which allows for an assessment of which function is more responsible for the overall error. Third, \(\mathcal{HC}\)-Search makes minimal assumptions about the loss function, requiring only that we have a “blackbox” evaluation of any candidate output. Theoretically, it can even work with non-decomposable loss functions, such as F1 loss.

5.5 Pruning Function

We extend \(\mathcal{HC}\)-Search with a pruning function. The goal of the pruning function is to improve the accuracy of \(\mathcal{H}\) and \(\mathcal{C}\) by making their learning problems easier. Another goal is to improve the efficiency of learning and inference. The pruning function induces a sparse search space by eliminating bad actions generated from the successor function.

Recall that the successor function \(S\) generates a set of actions from the original state. The difference between the original state and a state from the set of next states is called an action, which we will denote \(a\). In other words, whereas one can interpret the successor function as returning a set of next states, one can also interpret the successor function as returning a set of actions where an action \(a\) transforms the original state to the next state. The difference between the original state and the next state is usually small and local, only affecting a connected component of a handful of patches. Therefore the distinction between an action and a state is important because an action feature is different from a heuristic or cost feature, which operate on states and the entire labeling instead. Computing an action feature is generally less expensive than computing features across the entire labeling.
Given that the successor function generates a set of actions, the goal of the pruning function is to remove bad actions from the candidate set. Specifically, the pruning function $P : \mathcal{X} \times \mathcal{Y} \times \mathcal{A} \mapsto \mathbb{R}$ ranks actions generated from the successor function and “prunes away” a certain percentage of these actions, which we will call “bad actions.” This pruning percentage is a parameter. If the pruning percentage is 0%, then no actions are pruned and thus the problem reduces to $\mathcal{HC}$-Search with no pruning.

**Features.** Recall that an action $a$ consists of a set of patches that were flipped from the original state to the new state. A feature for this action can describe how this action affected the labeling. There are several features that we employed.

One simple feature that we employed is $1$ if the action introduces a hole and $0$ otherwise. This is happens when all neighbors of the action take the same label while the label of the action is different. Actions that introduce a hole are generally bad actions in nematocyst images.

Another feature that we employed is the spatial standard deviation. Intuitively, foreground patches of the same class should be close to one another; there should be some “center” for these patches. One way to capture this is to compute the standard deviation of the $x$ and $y$ positions of all patches of the same label.

**Learning.** The pruning function can be learned as follows. Let $y^*$ be the ground truth labeling and let $\hat{y}_t$ be the labeling at time step $t$. At time step $t$, the loss function (available only for training data) can be employed to compute the loss of the labeling $\hat{y}_t$, denoted as $L(x, \hat{y}_t, y^*)$, and the loss of the labeling $\hat{y}_{t+1}$ of the generated state from the successor function, denoted as $L(x, \hat{y}_{t+1}, y^*)$. If $L(x, \hat{y}_t, y^*) - L(x, \hat{y}_{t+1}, y^*) > 0$, then the loss has improved and thus the action is considered as “good” otherwise the
action is “bad.” Furthermore, \( L(x, \hat{y}_t, y^*) - L(x, \hat{y}_{t+1}, y^*) \) measures the amount of loss improved. We can create a ranking example where every “good” action is better than a ”bad” action. These along with the action features form training examples for the rank learner to learn the pruning function. By running \( \mathcal{HC}\)-Search with an oracle heuristic function and oracle cost function (using the loss function) on each training image, we can collect these training examples for every search step and train a rank learner.
Chapter 6: Results

We evaluate IID classifiers (Sec. 4.2), CRFs (Sec. 4.1), and $\mathcal{HC}$-Search (Sec. 5) on a dataset of SEM images containing nematocysts. For $\mathcal{HC}$-Search, we compare the Flipbit space with our stochastic space. We also compare the performance of our stochastic space when employing the pruning function.

6.1 Dataset Description

Prepared by an expert biologist, the image dataset consists of SEM images containing nematocysts. Fresh specimens of cnidarian tissue were (a) exposed to 1M sodium citrate for 10 minutes; (b) rinsed in water; (c) preserved in 70% ethanol; (d) dehydrated in a graded series; (e) sputter-coated with gold palladium in a Cressington sputter coater; and, finally, (f) imaged using a FEI NOVA nanoSEM microscope. The dataset consists of 130 images, each with resolution of $1024 \times 864$ pixels. The images often show multiple nematocyst specimens within cluttered background, as illustrated in Figures 3.1, 6.4 and 6.7. The dataset is very challenging. First, the background clutter consists of mucus and debris. These appear quite similar to the target basal tubules. Mucus and debris often latch onto parts of nematocysts, which may partially occlude the basal tubules or create foreground-background confusion even to the human eye. Parts of nematocysts may also be physically missing, or may simply be out of the field of view. SEM images
also suffer from low contrast. The ground truth for each image is manually annotated by dividing the image into a regular grid of 32x32 pixel patches and labeling each patch as belonging to the basal tubule of a nematocyst or background.

6.2 Evaluation Setup and Metrics

Given a test image, our structured prediction assigns one of the classes to each image patch on a regular grid. Performance is evaluated by precision, recall, and F1 measure, where true positives are patches that fall on the ground truth basal tubules. For $HC$-Search, we also evaluate our sensitivity to the time bound $\tau$ (i.e. the number of greedy search steps that are allowed before making the final prediction).

For the Flipbit space, we partition the 130 images into five folds $A = A_1 \cup \ldots \cup A_5$. For a particular fold $A_i$, fold $A_i$ is used for testing and the rest of the folds $A \setminus A_i$ are used for training. We train on five different folds and get five different test results. We compute the mean and standard deviation of these test results for each metric.

For the stochastic space, instead of using five folds, we fix 80 images for training, 20 for hold-out validation, and 30 for testing. Since the successor function is stochastic, training and inference will be different for each run. To address this issue, we run training and inference multiple times. For training, we collect training examples on 5 independent runs of search to increase the number and variety of training examples. For inference, we also run search 5 times independently, then compute the mean and standard deviation of the results for each metric. For instance, to evaluate the precision of $HC$-Search, we run $HC$-Search five times to obtain five different precision scores,
then compute the mean and standard deviation of these precision scores.

It should be noted that the standard deviation of the Flipbit results comes from the folds and the standard deviation of the stochastic results comes from the stochasticity of the successor function. To simplify the stochastic results, we only evaluate on one particular fold consisting of 80 images for training, 20 for hold-out validation, and 30 for testing.

6.3 Methods

An image is divided into a regular grid of patches. Each patch is described by a 128-dimensional SIFT descriptor. Assigning labels to the patches is performed using the following methods.

**IID Classifier** applies either SVM or Logistic Regression independently on each image patch. **Pairwise CRF** is the standard CRF that models the image using the unary and pairwise potentials of the image patches. **Pyramid CRF** augments the pairwise potentials with hierarchical relationships between (larger) parent patches and their (embedded smaller) children patches. The notations **w/ ICM**, **w/ LBP**, and **w/ GraphCuts** indicate that inference of CRF is conducted using ICM, LBP, or Graph-Cuts algorithms, respectively.

There are several experimental setups for $\mathcal{H}$-Search. The first setup investigates the Flipbit space with the following variants: **No Global**, **Max Global**, and **Sum Global**, which differ in the feature representation for the heuristic and cost functions. The goal is to study the effects of employing higher-order potentials in the heuristic and cost func-
tions. *No Global* uses only the unary and pairwise features of image patches, given by (4.1). *Max Global* additionally uses a higher-order feature describing the largest connected component of positive detections. *Sum Global* additionally uses a higher-order term describing all connected components of positive detections. The higher-order feature is defined as the standard Bag-of-Words (BoW) of 300 codewords, found by K-means over SIFTs of all image patches from the entire dataset. A connected component is represented by a 300-codeword histogram where each patch from the connected component is mapped to the nearest codeword.

In addition to the Flipbit space, we also test \( HC \)-Search with our stochastic successor function and pruning function extensions to see how much performance gain we obtain. We first test \( HC \)-Search with our stochastic successor function and compare with the Flipbit successor function. Then we compare pruning versus no pruning while fixing the stochastic search space in \( HC \)-Search.

### 6.4 Quantitative Results: Baselines

Table 6.1 presents the detection results of IID Classifiers, CRF, and \( HC \)-Search. The results for Logistic Regression are reported for the detection threshold set at the maximum F1 score. The \( HC \)-Search results are obtained for time bound \( \tau = 100 \) (greedy search steps) for the Flipbit space and \( \tau = 30 \) for the stochastic space. Table 6.1 shows that \( HC \)-Search outperforms the two types of IID Classifiers, improving upon the initial prediction of logistic regression. \( HC \)-Search also yields higher recall and F1 than all variants of CRFs. Interestingly, the CRFs with ICM inference gave better recall and F1
than with LBP and Graph-Cuts inference. From Table 6.1, the inclusion of standard higher-order features (BoW) in $\mathcal{HC}$-Search does not lead to significant performance improvements. This contrasts with common reports in the literature and requires further investigation.

(a) IID Classifier Results

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>.675</td>
<td>.147</td>
<td>.241</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>.605</td>
<td>.129</td>
<td>.213</td>
</tr>
</tbody>
</table>

(b) CRF Results

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pairwise w/ ICM</td>
<td>.432</td>
<td>.360</td>
<td>.393</td>
</tr>
<tr>
<td>Pairwise w/ LBP</td>
<td>.545</td>
<td>.091</td>
<td>.156</td>
</tr>
<tr>
<td>Pairwise w/ GraphCuts</td>
<td>.537</td>
<td>.070</td>
<td>.124</td>
</tr>
<tr>
<td>Pyramid w/ ICM</td>
<td>.565</td>
<td>.258</td>
<td>.354</td>
</tr>
<tr>
<td>Pyramid w/ LBP</td>
<td>.500</td>
<td>.013</td>
<td>.025</td>
</tr>
<tr>
<td>Pyramid w/ Graph Cuts</td>
<td>.732</td>
<td>.013</td>
<td>.026</td>
</tr>
</tbody>
</table>

(c) $\mathcal{HC}$-Search Flipbit Results

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Global</td>
<td>.472</td>
<td>.545</td>
<td>.506</td>
</tr>
<tr>
<td>Max Global</td>
<td>.445</td>
<td>.508</td>
<td>.475</td>
</tr>
<tr>
<td>Sum Global</td>
<td>.457</td>
<td>.533</td>
<td>.492</td>
</tr>
</tbody>
</table>

Table 6.1: Baseline performance on the nematocyst images.

We also test sensitivity to (i) Image patch size, (ii) Choice of the descriptor for patches, and (iii) Training time bound $\tau$ for $\mathcal{HC}$-Search.

First, for patch sizes of 16x16 and 64x64 pixels, and appropriately adjusted ground truth, all the approaches underperform relative to the results presented in Table 6.1. For all the approaches, for 16x16 pixels, F1 decreases by 8–11 percentage points, and, for
64x64 pixels, F1 decreases by 8–9 percentage points. Thus, our default patch size of 32x32 pixels empirically works best.

Second, when replacing SIFTs with 496-dimensional HOG descriptors, the F1 of all the approaches decreases by 2–4 percentage points.

Finally, Figure 6.1 shows the plots of precision, recall, and F1 of \( HC \)-Search for increasing time bounds \( \tau \) with the Flipbit space. The plots show four types of curves: \( LL \)-Search, \( HL \)-Search, \( LC \)-Search, and \( HC \)-Search. \( LL \)-Search uses the loss function as both the heuristic and the cost function, and thus serves as an upper bound on the performance of the selected search architecture. \( HL \)-Search uses the learned heuristic function, and the loss function as cost function, and thus serves to illustrate how well the learned heuristic performs in terms of the quality of generated outputs. \( LC \)-Search uses the loss function as an oracle heuristic, and learns a cost function to score the outputs generated by the oracle heuristic. From Figure 6.1, for \( HC \)-Search, we see that as \( \tau \) increases, precision drops, but recall and F1 improve up to a certain point. This is understandable, because as \( \tau \) increases, the generation error \( (\epsilon_H) \) will monotonically decrease, since strictly more outputs will be encountered. Simultaneously, difficulty of cost function learning can increase as \( \tau \) grows, since it must learn to distinguish among a larger set of candidate outputs. In addition, we can see that the \( LC \)-Search curve is very close to the \( LL \)-Search curve, while the \( HL \)-Search curve is far below the \( LL \)-Search curve. This suggests that the overall error of \( HC \)-Search, \( \epsilon_{HC} \), is dominated by the heuristic error \( \epsilon_H \). A better heuristic is thus likely to lead to better performance overall.

We also report the error decomposition results of \( HC \)-Search in Table 6.2. Recall
Figure 6.1: Plots of precision, recall and F1 of $\mathcal{HC}$-Search versus the time step. The curves diagnose the heuristic and cost function learning. (a)-(c) are results with the Flipbit space (where error bars result from test folds) and (d)-(f) are results with our stochastic space (without pruning) for comparison (where error bars result from running inference many times). The $\mathcal{HC}$ curve (blue) is of interest. The stochastic space outperforms the Flipbit space in shorter time steps.
that from Equation 5.3, we can compute the decomposition of overall error $\varepsilon_{HC}$ in terms of the heuristic error $\varepsilon_H$ and cost function error $\varepsilon_{C|H}$ ($\varepsilon_{HC} = \varepsilon_H + \varepsilon_{C|H}$) for a given pair of heuristic and cost functions ($H, C$). As noted above, we can employ the true loss function $L$ as a heuristic function and/or cost function. Results in Table 6.2 more precisely indicate that heuristic error $\varepsilon_H$ dominates the overall error $\varepsilon_{HC}$ for the $HC$-Search approach and cost function error $\varepsilon_{C|H}$ for $LC$-Search is very small indicating that the cost function learner is able to leverage the better outputs produced by the oracle heuristic.

| Error     | $\varepsilon_{HC}$ | $\varepsilon_H$ | $\varepsilon_{C|H}$ |
|-----------|---------------------|-----------------|---------------------|
| $LL$-Search | .027               | .027            | 0                   |
| $HC$-Search | .116               | .075            | .041                |
| $LC$-Search | .034               | .027            | .007                |

Table 6.2: Error decomposition of $HC$-Search No Global for time bound $\tau = 100$.

6.5 Quantitative Results: Our Extensions

Having discovered that the heuristic function learning needs improvement in the Flipbit space, we investigate the effect of our stochastic successor function and pruning function extensions on $HC$-Search in this section.

Figure 6.1 presents the performance of $HC$-Search with our stochastic search space (without the pruning function). Table 6.3 presents the final detection results. We observe that the curves with our stochastic space climb faster than the corresponding curves of the Flipbit space. This makes sense because the Flipbit space only flips one patch at a time. In addition, note that the final precision, recall and F1 of the stochastic space
outperforms that of the Flipbit space. We see that the precision from the stochastic space improves over the precision from the Flipbit space. Furthermore, note that all the curves are close to the $LL$-Search in the stochastic case whereas this is not true for the Flipbit case. Thus the stochastic search space improves the performance of search. In particular, we presented some evidence that the heuristic has improved.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic No Pruning</td>
<td>.831 ± .037</td>
<td>.651 ± .016</td>
<td>.729 ± .019</td>
</tr>
<tr>
<td>Stochastic Pruning</td>
<td>.825 ± .059</td>
<td>.526 ± .075</td>
<td>.641 ± .066</td>
</tr>
</tbody>
</table>

Table 6.3: $HC$-Search performance on the nematocyst images using our stochastic search space and pruning function.

We now investigate our pruning function. Since the percentage of candidates to prune away is a parameter of the pruning function, we first determine what is the best parameter. We train and run $HC$-Search with the pruning function parameter set at percentages between 0% and 90% in increments of 10%. Note that 0% pruning is equivalent to no pruning. Figure 6.2 shows a small peak around 50%, suggesting that the best pruning percentage is 50%. It is also interesting that after 50% the performance drops. This suggests that while pruning can be beneficial for heuristic and cost learning, pruning away too many candidates eventually harms heuristic and cost learning due to the lack of training examples. Further investigation is required to verify this hypothesis.

Having fixed our pruning parameter to 50%, we run $HC$-Search with the stochastic space and pruning function. Figure 6.3 compares the $HC$-Search performance of Flipbit space, our stochastic space without pruning and our stochastic space with pruning up to time bound $\tau = 30$. We observe that the stochastic space without pruning outperforms Flipbit as already reported. However, the stochastic space with pruning performs
slightly worse than without pruning. Table 6.3 also presents this observation. The recall curve is slightly worse. This suggests that the pruning function eliminated some good candidates. However, the overall recall curve continues to improve, and thus the pruning function is able to keep good candidates at the same time. On the other hand, this comparison is not entirely fair in the sense that no pruning uses more time (i.e. generates more candidates) than that of pruning. Future work would entail comparing the precision, recall and F1 of pruning and no pruning based on the actual amount of time instead of the time bound. Overall this requires further investigation.

Finally, we present some remarks on the number of candidates generated during every search step. Table 6.4 presents the average number of candidates generated at each search step for various settings. Flipbit (exhaustive) considers all possible flips so
Figure 6.3: The plots of precision, recall and F1 of $HC$-Search versus the training time bound comparing the Flipbit space, stochastic space without pruning and stochastic space with pruning. Our stochastic space outperforms the Flipbit space. Pruning yields slightly worse recall when compared with time bounds.
there are \(27 \times 32 - 1 = 863\) candidates. Flipbit (neighbors) only considers flipping neighboring patches (i.e. along the boundary) of connected components so that the next search state is restricted to only “growing” or “shrinking” connected components. This hard constraint eliminates a majority of bad candidates. However, the stochastic space further reduces the number of candidates from the Flipbit space since multiple patches can be flipped per search step. The pruning function even further reduces down to fewer than 100 candidates. Thus the pruning function and the stochastic successor function are able to significantly increase the efficiency of search by only considering a handful of good candidates. Indeed the overall amount of time to search is significantly reduced when employing the pruning function.

<table>
<thead>
<tr>
<th>Average No. of Candidates</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Flipbit (Exhaustive)</td>
<td>863</td>
</tr>
<tr>
<td>Flipbit (Neighbors)</td>
<td>260.2</td>
</tr>
<tr>
<td>Stochastic</td>
<td>194.6</td>
</tr>
<tr>
<td>Stochastic with 50% Pruning</td>
<td>97.3</td>
</tr>
</tbody>
</table>

Table 6.4: Average number of candidates generated at each search step.

6.6 Qualitative Results

Figure 6.4 illustrates detection results of CRF Pairwise w/ ICM and \(\mathcal{H}C\)-Search No Global for the time bounds of 5 and 50 steps. As can be seen, while the CRF takes advantage of pairwise interactions, the CRF still tends to produce islands of false positives. In contrast, the recall of \(\mathcal{H}C\)-Search improves by “growing” the initial connected component of positives as the time bound increases. Figure 6.5 illustrates another de-
tection result on a challenging cluttered image. Again, $HC$-Search is able to yield good detections as the time bound increases.

Figure 6.6 compares the detection results of $HC$-Search with different search spaces and pruning (set to 50%) fixed at time bound $\tau = 30$. We observe that the detection from the stochastic space improves recall over the Flipbit space. This makes sense as the stochastic space flips multiple patches at each search step to improve the recall faster than that of the Flipbit space. Qualitatively, we can see that $HC$-Search improves detection quality over CRFs and the stochastic space improves recall faster than that of the Flipbit space. Finally, we compare the pruning and no pruning detection results. Both detections are nearly the same. This suggests that the pruning function merely eliminates bad candidates from the successor function to improve efficiency of search, but does not drastically affect the detections.

Finally, figure 6.7 shows some false detections of $HC$-Search. Qualitatively, this occurs because the background clutter appears very similar to the textured surface of the basal tubule. As the time bound increases, $HC$-Search is still certain that particular false detections are okay and continue to “grow” them instead of eliminating them.
Figure 6.4: An example nematocyst with the basal tubule (green). HC-Search gives better precision and recall than CRF, and performance of HC-Search improves as the time bound ($\tau$) increases. Here we employ the Flipbit space.
Figure 6.5: Here is a good detection of a challenging image. The background is highly cluttered yet \( \mathcal{HC} \text{-Search} \) gives better precision and recall than CRF. The performance of \( \mathcal{HC} \text{-Search} \) improves as the time bound (\( \tau \)) increases.
Figure 6.6: Comparison of $\mathcal{HC}$-Search detections of basal tubules. We compare the Flipbit space with the stochastic space in addition to pruning and no pruning. The time bound is fixed at $\tau = 30$. All detections are good but the stochastic space is able to improve recall of patches faster than the Flipbit space. Given enough time bound, the Flipbit space will eventually catch up to the stochastic space. As for the pruning extension, there is not much difference in the labeling when employing pruning or no pruning.
Figure 6.7: The left column shows the ground truth labels of the basal tubule (green) for two example images of nematocysts with significant background clutter. The right column shows the corresponding $\mathcal{HC}$-Search detection results (green) with the time bound of 50. $\mathcal{HC}$-Search falsely detects other instances of basal tubules, which appear similar to the target basal tubules.
Chapter 7: Conclusion

We have evaluated our approach, baseline approaches and state-of-the-art structured prediction methods on the problem of detecting basal tubules of nematocysts appearing in SEM images. \( HC \)-Search yields better recall than both pairwise and pyramidal CRFs. We have shown that our hypothesis that \( HC \)-Search outperforms CRFs on our dataset is correct. A multi-component learning architecture such as \( HC \)-Search yields better results on our domain. Additionally, we have also extended \( HC \)-Search with a stochastic search space and pruning function. Our results indicate that these extensions improve precision, recall and efficiency over \( HC \)-Search with the Flipbit space. Thus overall our approach outperforms CRFs on our dataset. However, our biological domain remains difficult since we need a fairly large time bound for good results. This presents several opportunities for future work, including investigating a scheduling approach for the stochastic search space. Furthermore it would be interesting to evaluate our approach on images of general scenes.
Bibliography


