$\mathcal{HC}$-Search for Structured Prediction in Computer Vision

Michael Lam, Janardhan Rao Doppa†, Sinisa Todorovic, and Thomas G. Dietterich
Oregon State University, Corvallis, OR
{lamm,sinisa,tgd}@eecs.oregonstate.edu

† Washington State University, Pullman, WA
jana@eecs.wsu.edu

Abstract

The mainstream approach to structured prediction problems in computer vision is to learn an energy function such that the solution minimizes that function. At prediction time, this approach must solve an often-challenging optimization problem. Search-based methods provide an alternative that has the potential to achieve higher performance. These methods learn to control a search procedure that constructs and evaluates candidate solutions. The recently-developed $\mathcal{HC}$-Search method has been shown to achieve state-of-the-art results in natural language processing, but mixed success when applied to vision problems. This paper studies whether $\mathcal{HC}$-Search can achieve similarly competitive performance on basic vision tasks such as object detection, scene labeling, and monocular depth estimation, where the leading paradigm is energy minimization. To this end, we introduce a search operator suited to the vision domain that improves a candidate solution by probabilistically sampling likely object configurations in the scene from the hierarchical Berkeley segmentation. We complement this search operator by applying the DAGGER *algorithm to robustly train the search heuristic so it learns from its previous mistakes. Our evaluation shows that these improvements reduce the branching factor and search depth, and thus give a significant performance boost. Our state-of-the-art results on scene labeling and depth estimation suggest that $\mathcal{HC}$-Search provides a suitable tool for learning and inference in vision.*

1. Introduction

This paper explores whether $\mathcal{HC}$-Search [7] can be effective in computer vision problems; it has already shown the state-of-the-art performance on structured prediction problems in natural language processing. A number of tractable heuristic methods for structured prediction have been used in vision, including Dual Decomposition [26], Graph-Cuts [6], Swendsen-Wang (SW) cut [4], Variational inference [49], Quadratic Programming (QP) [33, 19], Message passing [24], a hybrid of Linear programming (LP) and QP [25], and search-based methods [15, 11, 23, 32, 36]. Instead of training a single global energy function and then solving a global optimization problem, as is done in these approaches, $\mathcal{HC}$-Search decomposes the problem into three steps: (Step 1) Find an initial complete solution, (Step 2) Explore a search tree of alternative candidate solutions rooted at the initial solution, and (Step 3) Score each of these candidates to select the best one. Any existing method can do Step 1. Step 2 is guided by a learned heuristic function $\mathcal{H}$, and Step 3 is performed by a learned cost (energy) function $\mathcal{C}$.

Doppa et al. [7] claim several advantages for $\mathcal{HC}$-Search. First, in the standard approaches a global energy function must be trained to "defend against" the exponentially-large set of all possible wrong answers to the problem. This is expensive both computationally and in terms of sample complexity. It can require highly expressive representations (e.g., higher-order potentials). In contrast, the heuristic function $\mathcal{H}$ only needs to correctly rank the successors of each state that is expanded during the heuristic search in Step 2, and the cost function $\mathcal{C}$ only needs to correctly find the best of these in Step 3. These are much easier learning problems, and hence, simpler potential functions can be applied. Second, for making predictions there is no need to solve a global optimization problem at prediction time. In effect, the system learns not only how to score candidate solutions but also how to *find* good candidates—it learns to do inference more efficiently. Third, $\mathcal{HC}$-Search can be applied to non-decomposable loss functions. This is another consequence of using a search space formulation instead of a global optimization approach. Finally, $\mathcal{HC}$-Search provides a clean engineering methodology for determining which components of the system would most benefit from additional engineering effort.

Many computer vision problems can be formulated as structured prediction [40, 14, 26, 22, 2, 28, 48, 10, 41, 50,

30]. If $\mathcal{HC}$-Search can be applied to these, it could become a suitable inference tool for solving computer vision problems. Previous work has tested $\mathcal{HC}$-Search on two vision problems [27]. Results on semantic scene labeling were promising, but initial experiments on object detection against significant background clutter were disappointing. In this paper, we claim that the shortcoming of previous work was in Step 2: the formulation of the search space and the learning algorithm employed to train $\mathcal{H}$. This is because the $\mathcal{HC}$-Search specification in previous work [27] used a naive search space defined over relatively small image patches, and thus required an immense branching factor and a very deep search depth (to find a good solution). Previous work developed ad hoc search spaces that sometimes worked and sometimes did not.

In this paper, we present an elegant and general method based on a randomized segmentation search space. The search space is defined by probabilistic sampling of a plausible image segmentation from the hierarchy of Berkeley segmentations of the image (UCM [1, 3]). This sampling is realized by randomly picking a threshold on the saliency of region boundaries present in the segmentation.

Each search step then involves changing the label of the regions defined by intersecting the chosen segmentation with connected regions defined by the current candidate solution. By choosing different thresholds for UCM, we obtain segmentations at different scales, which when intersected with the candidate solution give us search steps at multiple scales.

This search space is a big improvement, but to successfully apply $\mathcal{HC}$-Search, we also complement this search space by training $\mathcal{H}$ using the advanced imitation learning algorithm DAGGER [35]. The deeper search depths of computer vision problems mean that training $\mathcal{H}$ using simple exact-imitation learning [34] is not sufficient, because exact-imitation learning does not "learn from its own mistakes." During learning, DAGGER blends the current heuristic $\mathcal{H}$ with an oracle heuristic, which is more effective at teaching $\mathcal{H}$ to recover from its errors.

We demonstrate that with these two improvements, $\mathcal{HC}$-Search is able to match or exceed the state of the art on three challenging problems: (1) semantic scene labeling, (2) monocular depth estimation, and (3) objected detection against highly-cluttered background. The paper also applies the $\mathcal{HC}$-Search engineering methodology to evaluate the improvements introduced by the segmentation search space and DAgger training. The results show that $\mathcal{HC}$-Search, when combined with these two improvements, is a suitable inference tool for solving computer vision problems.

In the following, Sec. 2 places $\mathcal{HC}$-Search in the context of prior work; Sec. 3 provides an overview of $\mathcal{HC}$-Search; Sec. 4 specifies our two improvements; Sec. 5 presents our experiments; and Sec. 6 concludes the paper.

2. Closely Related Prior Work in Vision

Inference complexity in computer vision has been addressed with cascade architectures which perform multiple runs of inference from coarse to fine levels of abstraction [12, 44, 43, 31, 45]. These make inference more efficient, but they place strong restrictions on the form of the cost functions to facilitate "cascading," and they typically require that the loss function be decomposable in a way that supports "loss augmented inference." $\mathcal{HC}$-Search places minimal restrictions on the cost function.

Classifier-based structured prediction algorithms [16, 46] make a series of local decisions towards producing a complete output y (e.g., sequential labeling of superpixels [32]). However, they typically apply the classifier in a greedy manner. This is not robust, because some greedy decisions are hard to make correctly, but they are crucial for good performance. In contrast, $\mathcal{HC}$-Search searches over complete candidate solutions, which allows it to recover from errors made during greedy or local search.

Our work is also related to work on learning for inference including cost-sensitive inference [9], reinforcement learning [42, 21], and speedup learning [13]. In these paradigms, the cost function is typically known and the goal is to learn a heuristic function for directing a search algorithm to a low-cost terminal node in the search space. $\mathcal{HC}$-Search learns the cost function too, and the goal is to find good solutions at any depth rather than only at well-defined terminal nodes.

Yadollahpour et al. [5, 47] proposed a re-ranking approach that is similar in spirit to $\mathcal{HC}$-Search. They generate a diverse set of plausible segmentations and learn a ranking function to score them. However, their approach does not provide any guarantees for the quality of the generated outputs. In contrast, $\mathcal{HC}$-Search imitates the search behavior of an oracle heuristic on training data. Based on standard learning theory considerations, it can be proved that the learned heuristic and cost function will generalize and perform well on new instances [8, 7, 35].

3. Overview of the $\mathcal{HC}$-Search Framework

The key elements of the $\mathcal{HC}$-Search framework include the Search space over complete outputs $\mathcal{S}$; Search strategy $\mathcal{A}$; Heuristic function $\mathcal{H} : \mathcal{X} \times \mathcal{Y} \mapsto \mathfrak{R}$ to guide the search towards high-quality outputs; and Cost function $\mathcal{C} : \mathcal{X} \times \mathcal{Y} \mapsto \mathfrak{R}$ to score the candidate outputs generated by the search procedure. An overview of $\mathcal{HC}$-Search is shown in Fig. 1.

Search Space. Every state in $\mathcal{S}$ consists of an input-output pair, $s = (x, y)$, representing the possibility of predicting y as the output for input image x. Such a search space is defined in terms of two functions: (1) *Initial state function*, I, such that $s_0 = I(x)$ is the initial state; 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

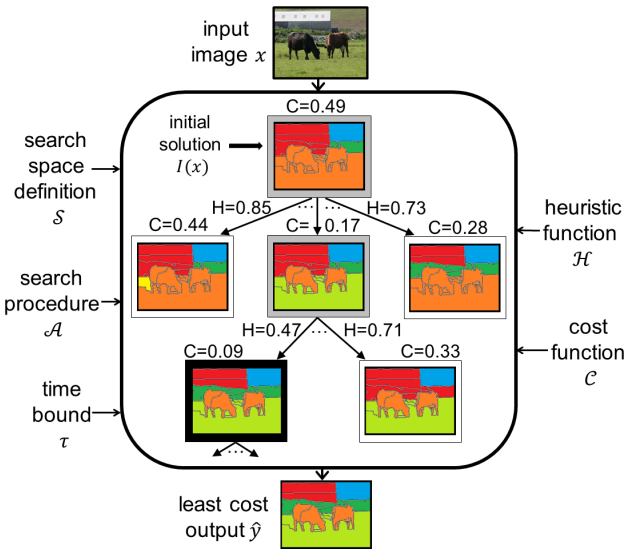

Figure 1: An overview of $\mathcal{HC}$-Search for the semantic scene labeling problem. Given an input x, an initial solution $s_0 = I(x)$ is computed by an (unspecified) procedure I. This forms the starting state in a search space $\mathcal{S}$. Each node in $\mathcal{S}$ consists of a candidate labeling y for x. Nodes are expanded according to search procedure $\mathcal{A}$, which is guided by a learned heuristic $\mathcal{H}$. The search continues until time bound τ. In this figure, $\mathcal{A}$ is greedy search, and the nodes with grey outlines are the nodes visited by $\mathcal{A}$. These nodes are then evaluated by learned cost function $\mathcal{C}$, and the node with the lowest cost is selected as the predicted output $\hat{y}$.

same input x. The initial state function can be any method that produces a complete candidate solution. In our work, I will be implemented by a classifier that has been trained to predict the label of each primitive region independently.

Search Strategy. Guided by the heuristic function $\mathcal{H}$, the search strategy $\mathcal{A}$ seeks high-quality candidate solutions. In this paper, we will use greedy search. Greedy search traverses a path of length τ through the search space, selecting as the next state, $s_{i+1} = \arg\min_{s \in S(s_i)} \mathcal{H}(s)$, the best successor of the current state s_i according to $\mathcal{H}$.

Making Predictions. Given a input x and time bound τ, $\mathcal{HC}$-Search first runs the search strategy $\mathcal{A}$ guided by the heuristic $\mathcal{H}$ until the time bound is exceeded. Let $\mathcal{Y}_{\mathcal{H}}(x)$ be the set of states visited by the search strategy $\mathcal{A}$ within time bound τ. The next step in $\mathcal{HC}$-Search is to compute the cost $\mathcal{C}(s)$ of each state $s \in \mathcal{Y}_{\mathcal{H}}(x)$. The final prediction $\hat{y} = \arg\min_{s \in \mathcal{Y}_{\mathcal{H}}(x)} \mathcal{C}(s)$.

Heuristic and Cost Function Learning. To define the objective functions for learning $\mathcal{H}$ and $\mathcal{C}$, we first decompose the overall prediction error into the errors due to $\mathcal{H}$ and $\mathcal{C}$. The error of $\mathcal{HC}$-Search, $\varepsilon_{\mathcal{HC}}$, for a given $\mathcal{H}$, τ, and $\mathcal{C}$ can be decomposed into two parts: 1) *Generation error*, $\varepsilon_{\mathcal{H}}$, due to $\mathcal{H}$ not generating high-quality outputs within time limit τ; and 2) *Selection error*, $\varepsilon_{\mathcal{C}|\mathcal{H}}$, the additional error (conditional on $\mathcal{H}$) due to $\mathcal{C}$ not selecting the best loss output generated by $\mathcal{H}$. This is formalized as

$$\varepsilon_{\mathcal{HC}} = \underbrace{L(x, y^*_{\mathcal{H}}, y^*)}_{\varepsilon_{\mathcal{H}}} + \underbrace{L(x, \hat{y}, y^*) - L(x, y^*_{\mathcal{H}}, y^*)}_{\varepsilon_{\mathcal{C}|\mathcal{H}}}. \quad (1)$$

, where $y^*_{\mathcal{H}}$ denote the best output that $\mathcal{HC}$-Search could possibly return when using $\mathcal{H}$ with time limit τ. Guided by the error decomposition in (1), the learning approach optimizes the overall error, $\varepsilon_{\mathcal{HC}}$, in a greedy stage-wise manner by first training $\mathcal{H}$ to minimize $\varepsilon_{\mathcal{H}}$, and then, training $\mathcal{C}$ to minimize $\varepsilon_{\mathcal{C}|\mathcal{H}}$ conditioned on $\mathcal{H}$.

$\mathcal{H}$ is trained by imitating the search decisions made by the true loss function (only available with ground truth data). We run the search procedure $\mathcal{A}$ for a time bound of τ 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 at random). The aggregate set of ranking examples is given to a rank learner (e.g., SVM-Rank) to learn $\mathcal{H}$. This approach is called *exact imitation* training.

$\mathcal{C}$ is trained to score the outputs $\mathcal{Y}_{\mathcal{H}}(x)$ generated by $\mathcal{H}$ according to their true losses. Specifically, this 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 $\mathcal{Y}_{\mathcal{H}}(x) \setminus \mathcal{Y}_{best}$.

4. Improvements for Computer Vision

In this section, we describe our two main improvements to $\mathcal{HC}$-Search for computer vision: (1) The randomized segmentation space; and (2) Training robust heuristic functions via DAGGER.

4.1. The Randomized Segmentation Space

Previous work employed simple search spaces that cannot deal with the huge branching factors and search depths of computer vision problems. Lam et al. [27] employed a search space called "Flipbit." The Flipbit space generates one successor state by changing the current label of one primitive image region (i.e., superpixel). Because the number of such regions is huge, this gives an immense branching factor, even if the number of labels is only two (hence, the name "bit").

In our new approach, we apply the Berkeley segmentation algorithm to create an ultrametric contour map (UCM). The UCM assigns every pixel a probability (the UCM value) of belonging to an object boundary. These values

have the property that for any threshold θ, the regions resulting from thresholding the UCM values at θ form a set of *closed* regions. Regions defined by smaller thresholds are strict hierarchical refinements of the regions defined by larger thresholds. A threshold of 0 yields the finest-scale segmentation, while a threshold of 1 yields a single full-image segment.

Suppose s_i is a candidate state in the search space and we wish to generate its successors $S(s_i)$. We do this as follows. First, we choose a value θ uniformly at random in $[0,1]$ and threshold the UCM values to obtain a segmentation of the image. Second, within each segment, we define a subgraph whose nodes are the superpixels in the segment and whose edges connect adjacent superpixels. The current labeling specified by s_i defines a coloring of these nodes. We compute the connected components of this subgraph, such that all nodes in a single connected component have the same color (label). For each such connected component, we generate one successor s' by assigning a new label to that component.

Our approach differs from Swendsen-Wang Cuts (SW-Cut), because SW-Cut proposes merges of only *pairs* of neighboring image regions and proposes splits that only produce two previously-generated regions. The SW-Cut proposals are accepted based on the KL divergence of the two regions' likelihoods. In contrast, we leverage the UCM map to propose meaningful segments. These may result in merging more than two superpixels or may split a region in a novel way, even though the whole had been assigned a single label by s_i. Finally, instead of using Berkeley Segmentation, we could use any multiscale segmentation as input.

4.2. DAGGER for Training Robust Heuristics

Our second improvement is to apply DAGGER to learn robust heuristic functions when compared to the simple exact imitation training approach (see Section 3). DAGGER is an iterative algorithm that can be viewed as generating a sequence of heuristics (one per iteration), with the property that the further we go along the sequence, the greater the risk of overfitting.

Let $\mathcal{R}$ be the set of pairwise constraints that we will use to train $\mathcal{H}$. $\mathcal{R}$ is initialized with the training examples generated by the exact imitation approach. Thus, the initial heuristic $\hat{\mathcal{H}}_1$ corresponds to the function learned from exact imitation. In all subsequent iterations j, we perform search using a mixture of the learned heuristic from the previous iteration $\hat{\mathcal{H}}_j$ and the oracle heuristic $\mathcal{H}^*$, i.e., at each search step we make search decisions using $\mathcal{H}^*$ (oracle) with probability β_j and $\hat{\mathcal{H}}_j$ with probability $1-\beta_j$, and generate additional ranking constraints whenever we make search errors. This allows DAGGER to learn from states visited by its possibly erroneous learned heuristic and correct its mistakes using oracle heuristic input. In the end, we select the heuristic that performs best on the validation data from the sequence of heuristic functions $\mathcal{H}_1, \cdots, \mathcal{H}_{d_{max}}$. Algorithm 1 provides the pseudocode for greedy heuristic learning with DAGGER.

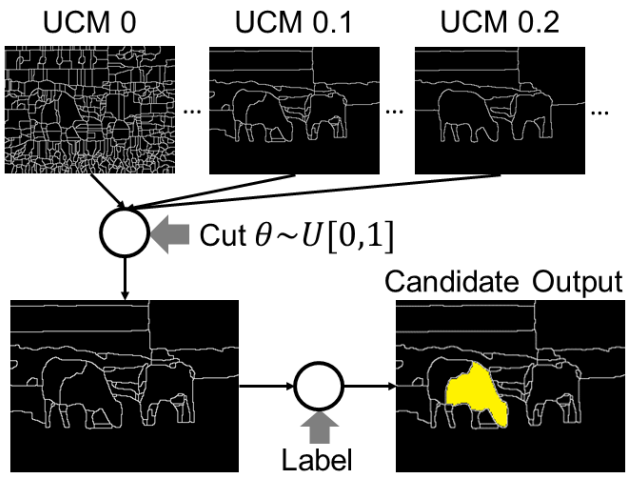

Figure 2: Overview of our successor function. Given an input state (x,y), Berkeley Segmentation extracts UCM from x. All edges that fall below a random threshold $\theta \sim U(0,1)$ are cut to yield subgraphs. Every connected component in every subgraph is relabeled to yield a set of candidate states $(x,y_1),(x,y_2),...,(x,y_n)$.

Algorithm 1 Greedy Heuristic Learning via DAGGER

Input: $\mathcal{D}$ = Training examples, (I,S) = Search space definition, L = Loss function, τ_{max} = search time bound, d_{max} = dagger iterations
Output: $\mathcal{H}$, the heuristic function

Initialization: $\mathcal{R} = \mathcal{R}_{ei}$ // *Exact imitation data*
$\hat{\mathcal{H}}_1$ =**Rank-Learner**($\mathcal{R}$)
for each dagger iteration $j = 1$ to d_{max} **do**
  Current Heuristic: $\mathcal{H}_j = \beta_j \mathcal{H}^* + (1-\beta_j)\hat{\mathcal{H}}_j$
  for each training example $(x,y^*) \in \mathcal{D}$ **do**
    $s_0 = I(x)$ // *initial search state*
    for each search step $t = 1$ to τ_{max} **do**
      Compute next states: C_t=$S(s_{t-1})$
      if $\mathcal{H}_j(C_t) \neq \mathcal{H}^*(C_t)$ **then**
        (x,y_{best})= the best state in C_t as per $\mathcal{H}^*$
        for each $(x,y) \in C_t \setminus (x,y_{best})$ **do**
          Add $\mathcal{H}(x,y_{best}) < \mathcal{H}(x,y)$ to $\mathcal{R}$
        end for
      end if
      s_t = the best state in C_t as per $\mathcal{H}_j$
    end for
  end for
  $\hat{\mathcal{H}}_{j+1}$ =**Rank-Learner**($\mathcal{R}$)
end for
return best heuristic $\hat{\mathcal{H}}_j$ on the validation data

5. Results

5.1. Setup

Datasets. We evaluate our approach on three datasets: (1) the Stanford Background dataset [14], (2) the Make3D

dataset [40, 38], and (3) the Nematocyst dataset [27]. The Stanford Background dataset contains 715 images of approximately 320×420 pixels. The ground truth assigns one of 8 semantic class labels to every pixel: sky, tree, road, grass, water, building, mountain and generic foreground object. The Make3D dataset contains 534 images of 2272×1704 pixels and their corresponding depth maps of resolution 55×305. Depth is measured in meters. The Nematocyst dataset consists of 130 grayscale images, each with a resolution of 1024×864 pixels. The image dataset was prepared by an expert biologist using a scanning electron microscope (SEM). The ground truth for each image is manually annotated by dividing the image into a regular grid of 32×32 pixel patches, and labeling each patch as belonging to the object class "basal tubule" or to "background." The dataset is challenging due to occlusion and heavily cluttered backgrounds.

Tasks. We evaluate our new successor function and DAGGER training on scene labeling, monocular depth estimation, and object detection against clutter. For scene labeling, the task is to assign the correct semantic class label to each pixel. For depth estimation, the task is to assign the correct real-valued depth to each pixel. For the object detection problem, the goal is to assign a label of "detected" or "not detected" to each primitive patch.

Evaluation Setup. For the Stanford Background dataset, we follow the five-fold cross validation experiment setup in prior work [14]. For the Make3D dataset, we employed 400 images for training and 134 images for testing as in previous work [40]. For the Nematocyst dataset, we followed the setup of previous work [27]: 80 images for training, 20 for hold-out validation, and 30 for testing.

Metrics. For scene labeling, our metric is accuracy, defined as the number of correctly labeled pixels divided by the total number of pixels. For depth estimation, our metrics are log-10 depth error and relative depth error. Log-10 depth error is defined as $|\log_{10} d - \log_{10} \hat{d}|$, and relative depth error is defined as $\frac{|d-\hat{d}|}{d}$, where d is the ground truth depth and $\hat{d}$ is the estimated depth. For object detection, our metrics are precision, recall, and F1 measure, where true positives are 32×32 patches that fall on the ground truth basal tubules.

Search Space. Our heuristic $\mathcal{H}$ and cost $\mathcal{C}$ functions are linear in the feature function $\psi(x,y)$ that we will specify shortly: $\mathcal{H}(x,y) = w_{\mathcal{H}}^{\top}\psi(x,y)$ and $\mathcal{C}(x,y) = w_{\mathcal{C}}^{\top}\psi(x,y)$. The feature function $\psi(x,y)$ consists of unary, pairwise and context features. The unary feature $\psi_{\text{unary}}(x,y)$ is the sum of superpixel descriptors $\phi(x_i)$ for each class $1...K$: $\psi_{\text{unary}}(x,y) = \left[\sum_{i=1}^{n} I(y_i = 1)\phi(x_i)\ ,\ \ldots,\ \sum_{i=1}^{n} I(y_i = K)\phi(x_i)\right]$. In the Nematocyst dataset, each superpixel is described by a 128-dimensional SIFT descriptor. In the Stanford background dataset, each superpixel is described by 64-dimensional texture features using 64 textons, 64-dimensional color features in LAB space of 64 bins and 12-dimensional normalized grid location features, totaling 140 dimensions, as similarly used in prior work [18]. In the Make3D dataset, we used the convolutional filter features from previous work [40]. The pairwise feature $\psi_{\text{pair}}(x,y)$ captures smoothness by summing all neighboring superpixel descriptors with different labels: $\psi_{\text{pair}}(x,y) = \sum_{i,j\in\mathcal{N}} I(y_i \neq y_j)|\phi(x_i) - \phi(x_j)|$. The context features $\psi_{\text{context}}(x,y)$ count the $\binom{K}{2}$ co-occurrences of different labels in four different spatial relationships: "left," "right," "above" and "below." Intuitively we are capturing, for example, whether a superpixel labeled "sky" is below another superpixel labeled "grass," which the heuristic and cost functions should learn to score less favorably. Define $\psi_{\text{single-context}}(x,y,l_1,l_2,c) = \sum_{i<j} I(y_i = l_1 \wedge y_j = l_2 \wedge config(i,j,c))$ where $config(i,j,c)$ is a function that evaluates to true if superpixels i and j are in the configuration of $c \in \{\text{left, right, above, below}\}$. Then the context features $\psi_{\text{context}}(x,y)$ is a concatenation of $\psi_{\text{single-context}}(x,y,l_1,l_2,c)$ features over all (l_1,l_2,c) combinations. The overall feature function is a concatenation of these unary, pairwise and context features: $\psi(x,y) = \left[\psi_{\text{unary}}(x,y)\ ,\ \psi_{\text{pair}}(x,y)\ ,\ \psi_{\text{context}}(x,y)\right]$.

In the object detection and scene labeling tasks, we employed the Hamming loss over all pixels: $L(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$. The initial state function applies an i.i.d. classifier to all superpixels in the object detection and scene labeling settings. For the successor function, we can keep the branching factor of search reasonable by employing smarter label proposals instead of using all possible labels. In the object detection task, our successor function chooses the neighboring label for all superpixel neighbors of a connected component. Since there are only two possible labels, this has the effect of "growing" or "shrinking" detection regions of a state and not introducing "islands" of detections. In the scene labeling task, for a particular connected component cc, our proposal label set consists of the top $B = 3$ confident labels of cc (provided by the initial state i.i.d. classifier) and the labels of the neighboring superpixels of cc. We found $B = 3$ to provide the best balance of accuracy versus efficiency.

In the depth estimation setting, we can treat the problem as scene labeling: each label corresponds to a depth. All depth values in the ground truth are clustered into $K = 20$ ordinal labels using K-means. A label corresponds to a depth and thus the larger the number of clusters K, the more accurate the prediction. We found $K = 20$ to provide the best trade-off of accuracy versus efficiency. There are several differences in the search space setup compared to scene labeling. The initial state function for depth estimation is an i.i.d. regressor that predicts a depth value for all superpixels. Each regression output is then mapped to the nearest label to yield an initial labeling for $\mathcal{HC}$-Search. We decided

on the regressor over an i.i.d. classifier with 20 labels since it was significantly faster without compromising much prediction accuracy. The loss function is the log-10 depth error averaged over all pixels: $L(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n} |\log_{10} d(y) - \log_{10} d(\hat{y})|$ where $d(y)$ is the depth value of label y. Each label is mapped to a depth so this loss can be computed. Finally the successor function proposes labels as follows: for each connected component and its label l, propose label $l+j$ where $j = 1,-1,5,-5$. This has the interpretation of increasing or decreasing the depth since the labels are ordinal; larger $|j|$ correspond to larger "jumps" for improving efficiency. The advantage of this setup is that despite having a large number of labels, the number of proposed labels for each connected component is not as many.

5.2. Baselines.

We define the following baseline methods.

B0. Edge Classifier: Instead of Berkeley segmentation, one can assign weights to superpixel graph edges similar in spirit to Swendsen-Wang Cuts. We generalize this to an i.i.d. edge classifier. Specifically, we train a logistic regression classifier on edge features and labels, where edge features are the difference of superpixel unary features on each side of the edge and labels indicate whether the edge belongs to an object boundary.

B1. Flipbit Space: The successor function in the Flipbit space proposes candidates by "flipping" the label of a superpixel to another label.

B2. Exact Imitation Learning: In exact imitation learning, the heuristic function $\mathcal{H}$ is learned by "imitating" the search trajectory of the oracle heuristic. DAGGER refines this $\mathcal{H}$ by training on additional examples that correct the trajectory mistakes from this $\mathcal{H}$. This baseline does not refine $\mathcal{H}$.

LL/$\mathcal{H}$L/L$\mathcal{C}$/$\mathcal{HC}$ Curves. To independently understand the quality of $\mathcal{H}$ and $\mathcal{C}$, we can replace either (or both) of them by an oracle that "cheats" by using the true loss function L and the ground truth labels. We can generate four performance curves. *LL* replaces both $\mathcal{H}$ and $\mathcal{C}$ with oracles, so it shows the best performance that can be obtained from the given search space S and time limit τ. $L\mathcal{C}$ generates the best possible search candidates, but evaluates them using the learned $\mathcal{C}$, so it tells us how good $\mathcal{C}$ is. $\mathcal{H}L$ uses the learned $\mathcal{H}$ to generate the search candidates, but then evaluates them using the oracle, so it allows us to assess the learned heuristic. And of course $\mathcal{HC}$ shows the actual performance of the system. By comparing these curves, we can understand which design choices (search space, heuristic, or cost function) would most benefit from more engineering.

5.3. Quantitative Results.

We first present results on the baselines. For baseline B0, we run *LL*-Search (serves as an upper bound) with time bound $\tau = 25$ and evaluate on the appropriate metrics for the Stanford Background and Make3D datasets. We do not apply Berkeley segmentation to the Nematocyst dataset because (1) the images contain so much background clutter that Berkeley segmentation does not yield good UCMs and (2) their groundtruth data are in terms of a regular grid of patches for evaluation. Table 1 demonstrates that using Berkeley segmentation outperforms the edge classifier. This makes sense because Berkeley segmentation's UCM yields hierarchical closed contours, whereas the locally trained i.i.d. classifier is not guaranteed to generate closed contours. Therefore, Berkeley segmentation is important to our search space for images of natural scenes.

Table 1: Comparison of i.i.d. edge classifier versus Berkeley segmentation measured using *LL*-Search performance with time bound $\tau = 25$, evaluated on the Stanford Background dataset (SBD) and Make3D dataset.

Dataset (Metric)	i.i.d. Edge Classifier	Berkeley Segmentation
SBD (Accuracy)	0.84	0.93
Make3D (Log-10 Error)	0.318	0.231

Figure 3 compares the *LL*-Search performance of the B1 baseline with our approach as a function of the time bound on the Nematocyst and Stanford Background datasets. Since the *LL*-Search serves as an upper bound, it is clear that the Flipbit space would require a larger time bound to reach the same accuracy as our randomized segmentation space. In fact our search space reaches about 94% accuracy in time bound $\tau = 25$ and 94% is the upper bound on the pixel-wise accuracy given our segmentation. Thus our randomized segmentation space reduces the expected target depth of the output and speeds up our inference. Figure 3 also compares the performance of the B2 baseline on the Nematocyst and Stanford Background datasets. Since DAGGER seeks to improve the heuristic function, we plot the performance of $\mathcal{H}L$-Search. For DAGGER we set the parameter $\beta_i = 0.1$ and used 5 DAGGER iterations. We see DAGGER indeed improves the heuristic function over exact imitation learning. In addition, the variance of $\mathcal{H}L$-Search decreases with DAGGER, demonstrating that DAGGER learns robust heuristics.

Figure 4 illustrates the performance of our approach in terms of *LL*-Search, $\mathcal{H}L$-Search, $L\mathcal{C}$-Search and $\mathcal{HC}$-Search for each dataset. For the Stanford Background dataset, we see that the learned heuristic and cost functions improve upon the initial state. The $\mathcal{HC}$-Search curve demonstrates that our approach—in the absence of deeply-learned features—meets the state of the art as shown in Table 2. For the Make3D dataset, we see that we achieve the state-of-the-art results as shown in 3. Despite our loss function

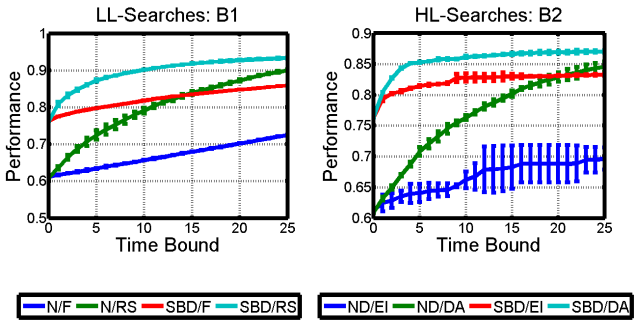

Figure 3: Baselines on the Nematocyst (ND) and Stanford Background (SBD) datasets. Performance for the Nematocyst dataset is the F1 score and the performance for SBD is accuracy. B1 (left): The *LL*-Search performance of Flipbit (F) versus our randomized segmentation space (RS) as a function of time bound. B2 (right): The $\mathcal{HL}$-Search performance of exact imitation learning (EI) versus DAGGER (DA) as a function of time bound.

Table 2: State of the art comparison of segmentation accuracy (%) on the Stanford Background dataset.

Method	Accuracy
Region Energy [14]	76.4
SHL [31]	76.9
RNN [41]	78.1
ConvNet [10]	78.8
ConvNet + NN [10]	80.4
ConvNet + CRF [10]	81.4
Pylon (No Bnd) [28]	81.3
Pylon [28]	81.9
Ours	81.4

optimizing the log-10 depth error, our relative depth error meets close to the state of the art. For the Nematocyst dataset, we see that the learned heuristic and cost functions perform nearly as well as *LL*-Search. Indeed in table 4 we see that we exceed the state-of-the-art results on the Nematocyst dataset.

We also study the average number of candidates generated by the successor function and compare to an upper bound. We compute this for the three datasets in table 5. The upper bound is computed by multiplying the average number of patches or superpixels per image, n, by the number of possible labels l minus 1 (the current label of a patch): $UB = n \times (l-1)$. This is a good upper bound because (1) the randomized segmentation space can select the finest segmentation, which is a collection of all patches or superpixels; and (2) the naive way of proposing labels is to consider all other possible labels. We see that the average number of candidates for all datasets is much less than the upper bound, demonstrating that our search space is more efficient.

Table 3: State of the art comparison of relative depth error and log-10 depth error on the Make3D dataset.

Method	Relative Error	Log-10 Error
Learn Depth: MRF [38, 37]	.530	.198
Pointwise MRF [40, 39]	.458	.149
Superpixel MRF [40, 39]	.370	.187
Surface Layout [17]	1.423	.320
Semantic Labels [29]	.375	.148
Depth Transfer [20]	.361	.148
Ours	.364	.148

Table 4: Comparison of precision, recall and F1 (%) on the Nematocyst dataset.

Method	Precision	Recall	F1
CRF [27]	43.2	36.0	39.3
$\mathcal{HC}$-Search Flipbit [27]	47.2	54.5	50.6
Ours	83.1	65.1	72.9

Table 5: Comparison of the number of candidates generated by the successor function averaged over all time steps and the upper bound of the branching factor.

Dataset	Avg. No. Candidates	Upper Bound
Nematocysts	126.2	864
SBD	362.7	3591.9
Make3D	1003.5	10497.5

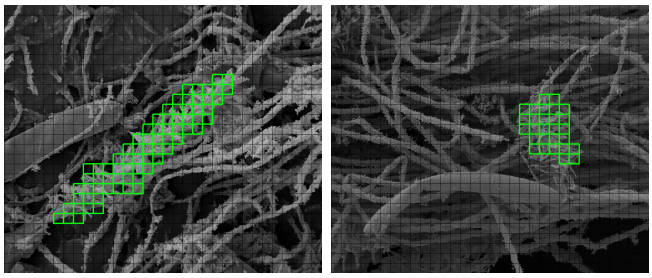

Figure 5: Qualitative results on the Nematocyst dataset. The green boxes indicate detected patches.

5.4. Qualitative Results.

Figures 5, 6, and 7 display qualitative results of our approach. We see that our approach yields excellent results. In the Stanford dataset, the backgrounds are mostly labeled correctly. In the Make3D dataset, the relative depths appear correct in general, although the depth for the sky could be closer to white (far away) than what is predicted. In the Nematocyst dataset, most of the patches are detected correctly, even in the presence of heavy background clutter.

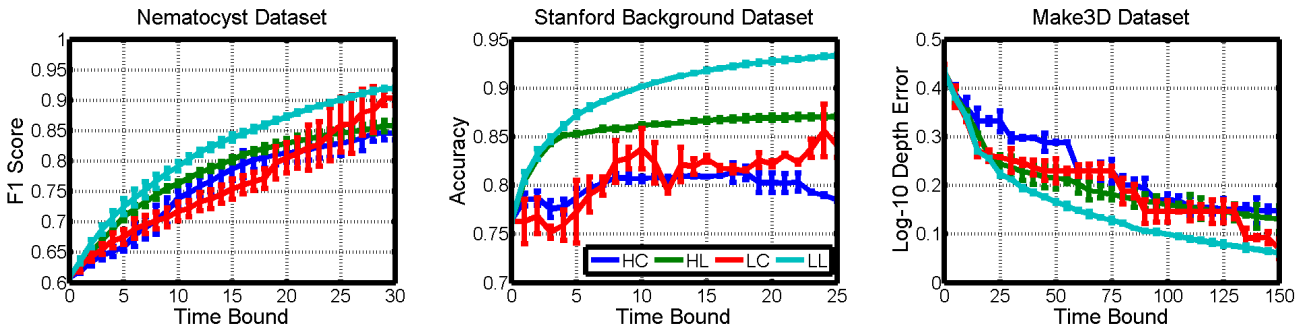

Figure 4: The performance of our approach as a function of time bound for *LL*-Search, $\mathcal{H}L$-Search, $L\mathcal{C}$-Search and $\mathcal{HC}$-Search on the Nematocysts (left), Stanford Background (middle) and Make3D (right) datasets.

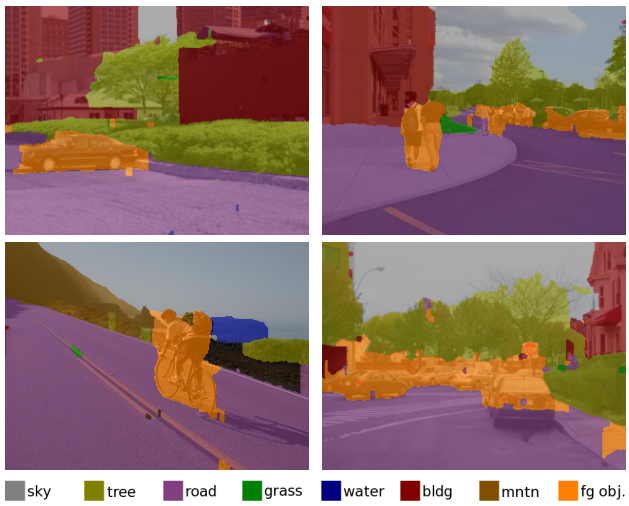

Figure 6: Qualitative results on the Stanford Background dataset. The highlighted colors correspond to semantic class labels.

Figure 7: Qualitative results on Make3D dataset. Darker shades indicate depths closer to the viewer than lighter ones.

6. Conclusion

We introduced two improvements to the $\mathcal{HC}$-Search framework for vision tasks: (1) a randomized segmentation space as a generic search space that leverages UCM segmentations; and (2) the application of DAGGER for training robust heuristic functions. We show that $\mathcal{HC}$-Search with these improvements gives performance comparable to or better than the state of the art across three diverse vision tasks: semantic scene labeling, monocular depth estimation, and object detection in biological images. Our error decomposition analysis demonstrates that our improvements achieve significant performance boosts over previous attempts to apply $\mathcal{HC}$-Search in computer vision.

Our investigation showed that there is still much room for improvement in the search space design for both scene labeling (oracle accuracy is 93.32%) and depth estimation (oracle error is 0.06). The cost functions are still making many ranking errors, which suggests that there is scope for improving the cost function representation. Yadollahpour et al. [47] observed similar phenomena in their re-ranking approach. Further future work includes designing high-quality search spaces for diverse computer vision problems, learning cost functions with rich representations to improve the accuracy and efficiency, and evaluating performance on larger datasets such as PASCAL VOC datasets.

Acknowledgements

The authors would like to acknowledge Xu Hu and Anirban Roy for helping with the datasets. This work was supported in part by NSF grants 1208272 and 1302700.

References

[1] P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *IEEE Workshop Perceptual Organization (POCV)*, page 182, 2006. 2

[2] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, and J. Malik. Semantic Segmentation using Regions and Parts. In *CVPR*, 2012. 2

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011. 2

[4] A. Barbu and S.-C. Zhu. Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities. *PAMI*, 27(8):1239–1253, 2005. 1

[5] D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich. Diverse M-best solutions in Markov Random Fields. In *ECCV*, 2012. 2

[6] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE TPAMI*, 23(11):1222–1239, 2001. 1

[7] J. R. Doppa, A. Fern, and P. Tadepalli. HC-search: A learning framework for search-based structured prediction. *J. Artif. Intell. Res. (JAIR)*, 50:369–407, 2014. 1, 2

[8] J. R. Doppa, A. Fern, and P. Tadepalli. Structured prediction via output space search. *Journal of Machine Learning Research*, 15(1):1317–1350, 2014. 2

[9] A. et al. Cost-sensitive top-down/bottom-up inference for multiscale activity recognition. In *ECCV*, 2012. 2

[10] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning Hierarchical Features for Scene Labeling. *IEEE Trans PAMI*, 35(8):1915–1929, 2013. 2, 7

[11] P. F. Felzenszwalb and D. Mcallester. The generalized A* architecture. *JAIR*, 29, 2007. 1

[12] P. F. Felzenszwalb and D. A. McAllester. The generalized A* architecture. *Journal of Artificial Intelligence Research (JAIR)*, 29:153–190, 2007. 2

[13] A. Fern. Speedup learning. In C. Sammut, editor, *Encyclopedia of Machine Learning*, pages 907–911. Springer-Verlag, New York, 2010. 2

[14] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, 2009. 2, 4, 5, 7

[15] P. Gupta, D. Doermann, and D. DeMenthon. Beam search for feature selection in automatic SVM defect classification. In *ICPR*, 2002. 1

[16] Hal Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning Journal (MLJ)*, 75(3):297–325, 2009. 2

[17] D. Hoiem, A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 2007. 7

[18] A. Kae, K. Sohn, H. Lee, and E. Learned-Miller. Augmenting CRFs with Boltzmann Machine Shape Priors for Image Labeling. *CVPR*, 2013. 5

[19] J. Kappes and C. Schnörr. MAP-inference for highly-connected graphs with DC-programming. In *Pattern Recognition*, pages 1–10. Springer, 2008. 1

[20] K. Karsch, C. Liu, and S. B. Kang. Depth extraction from video using non-parametric sampling. In *ECCV*, 2012. 7

[21] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012. 2

[22] P. Kohli, L. Ladický, and P. H. Torr. Robust Higher Order Potentials for Enforcing Label Consistency. *Int. J. Comput. Vision*, 82(3):302–324, 2009. 2

[23] I. Kokkinos. Rapid deformable object detection using dual-tree branch-and-bound. In *NIPS*, 2011. 1

[24] A. Kumar and S. Zilberstein. Message-passing algorithms for quadratic programming formulations of MAP estimation. In *UAI*, 2011. 1

[25] A. Kumar, S. Zilberstein, and M. Toussaint. Message-passing algorithms for MAP estimation using DC programming. In *AISTAT*, 2012. 1

[26] M. Kumar and D. Koller. Efficiently selecting regions for scene understanding. In *CVPR*, 2010. 1, 2

[27] M. Lam, J. R. Doppa, X. Hu, S. Todorovic, T. Dietterich, A. Reft, and M. Daly. Learning to detect basal tubules of nematocysts in sem images. In *ICCV 2013 Workshop on Computer Vision for Accelerated Biosciences*, 2013. 2, 3, 5, 7

[28] V. Lempitsky, A. Vedaldi, and A. Zisserman. Pylon model for semantic segmentation. In *NIPS*, 2011. 2, 7

[29] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *CVPR*, 2010. 7

[30] R. Mottaghi, X. Chen, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014. 2

[31] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *ECCV*, 2010. 2, 7

[32] N. Payet and S. Todorovic. SLEDGE: sequential labeling of image edges for boundary detection. *IJCV*, 104(1):15–37, 2013. 1, 2

[33] P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *ICML*, 2006. 1

[34] S. Ross and D. Bagnell. Efficient reductions for imitation learning. *Journal of Machine Learning Research - Proceedings Track*, 9:661–668, 2010. 2

[35] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research - Proceedings Track*, 15:627–635, 2011. 2

[36] A. Roy and S. Todorovic. Scene labeling using beam search under mutex constraints. In *CVPR*, 2014. 1

[37] A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *ICCV*, 2007. 7

[38] A. Saxena, C. S. H., and N. A. Y. Learning depth from single monocular images. In *NIPS*, 2005. 5, 7

[39] A. Saxena, M. Sun, and A. Y. Ng. Learning 3-d scene structure from a single still image. In *ICCV workshop on 3D Representation for Recognition (3dRR-07)*, 2007. 7

[40] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3-d scene structure from a single still image. *PAMI*, 2008. 2, 5, 7

[41] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *ICML*, 2011. 2, 7

[42] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape grammar parsing via reinforcement learning. In *CVPR*, 2011. 2

[43] D. Weiss, B. Sapp, and B. Taskar. Sidestepping intractable inference with structured ensemble cascades. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 2415–2423, 2010. 2

[44] D. Weiss and B. Taskar. Structured prediction cascades. *Journal of Machine Learning Research - Proceedings Track*, 9:916–923, 2010. 2

[45] D. J. Weiss and B. Taskar. Scalpel: Segmentation cascades with localized priors and efficient learning. In *CVPR*, 2013. 2

[46] J. Xie, C. Ma, J. R. Doppa, P. Mannem, X. Fern, T. Dietterich, and P. Tadepalli. Learning greedy policies for the easy-first framework. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2015. 2

[47] P. Yadollahpour, D. Batra, and G. Shakhnarovich. Discriminative re-ranking of diverse segmentations. In *CVPR*, 2013. 2, 8

[48] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012. 2

[49] B. Zhao, L. Fei-Fei, and E. Xing. Image segmentation with topic random field. In *ECCV*, 2010. 1

[50] L. Zhu, Y. Chen, Y. Lin, C. Lin, and A. Yuille. Recursive Segmentation and Recognition Templates for Image Parsing. *IEEE Trans PAMI*, 34(2):359–371, 2012. 2