Visual Chunking: A List Prediction Framework for Region-based Object Detection

Nicholas Rhinehart, Jiaji Zhou, Martial Hebert, and J. Andrew Bagnell
The Robotics Institute, Carnegie Mellon University
{nrhineha, jiajiz, hebert, dbagnell}@cs.cmu.edu

***Abstract*— We consider detecting objects in an image by iteratively selecting from a set of arbitrarily shaped candidate regions. Our generic approach, which we term visual chunking, reasons about the locations of multiple object instances in an image while expressively describing object boundaries. We design an optimization criterion for measuring the performance of a list of such detections as a natural extension to a common per-instance metric. We present an efficient algorithm with provable performance for building a high-quality list of detections from any candidate set of region-based proposals. We also develop a simple class-specific algorithm to generate a candidate region instance in near-linear time in the number of low-level superpixels that outperforms other region generating methods. In order to make predictions on novel images at testing time without access to ground truth, we develop learning approaches to emulate these algorithms' behaviors. We demonstrate that our new approach outperforms sophisticated baselines on benchmark datasets.**

I. Introduction

We consider the problem of object detection, where the goal is to identify parts of an image corresponding to objects of a particular semantic type, e.g. "car". In recent years, machine learning-based approaches have become de-rigueur for addressing this difficult problem; one classical approach is to transform the problem into one of binary classification, either on bounding boxes [1], [2], or regions. Such approaches (see Section II for a detailed discussion) typically follow a two stage procedure:

1) generate independent proposals to provide coverage across object instances
2) improve precision and reduce redundancy by pruning out highly overlapping proposals

Intuitively, the first step returns a set of proposals with high recall and the second step improves the precision. For the second step, traditional approaches rely on a combination of thresholds and arbitration techniques like Non-Max Suppression (NMS) to produce a final output. Such methods, while remarkably effective at identifying sufficiently separated objects, still have difficulty simultaneously detecting objects that are close together or overlap while preventing multiple detections of the same object (see Fig. 6). While we provide contributions to both stages, our focus is on formalizing and improving the second stage.

We formulate the objective of the second step as that of producing a diverse *list* of detections in the image. We propose an optimization criterion on this list of detections as a natural extension of the intersection over union metric (IoU) (described in Section III-A), and develop an algorithm that targets this criterion. This approach uses recent work on building performance-bounded lists of predictions [3], [4]. Our algorithm shares information across all candidate detections to build a list of detections, specifically exploiting this information to perform well even when object instances are adjacent. Each decision of appending to the list of detections is made with contextual information from all previous detections. Importantly, our list prediction algorithm is agnostic to the source of candidate detections. This provides our approach with the ability to use *any* candidate generating method as input for constructing a list.

Each candidate detection is treated as a union of superpixels with no adjacency constraints. We call these unions "chunks," inspired by a well-known task in Natural Language Processing: "chunking," which involves grouping many words together into meaningful semantic instances / entities. We use "region" to refer to a contiguous group of superpixels, and reserve "chunk" to refer to a group of superpixels corresponding to a single semantic instance.

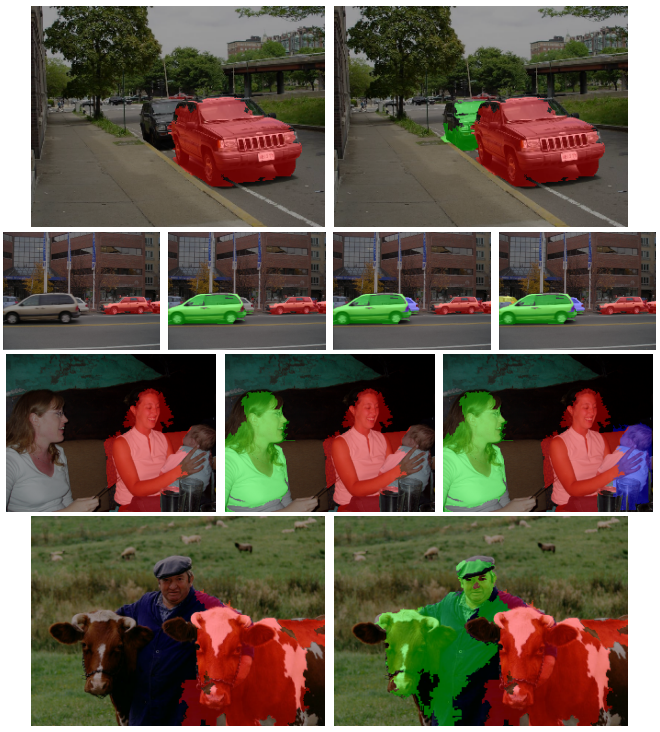

Fig. 1: Visual Chunking run on test data. The first prediction is shown in red, the second in green, the third in blue, and the fourth in yellow.

The analogy is particularly apt when object instances are adjacent, as in Fig. 1.

For the first step, we develop a class-specific supervised approach of region-based object proposal by iteratively grouping superpixels produced by a low-level segmentation algorithm [5] to form chunks. This helps build a high-recall candidate set. This algorithm learns to "grow" by utilizing class-specific ground-truth labeling by emulating an algorithm that optimizes a chunk's IoU score with an object, which we present in Algorithm 2. This strategy follows from imitation learning approaches [6], [7].

Our technique for building the list of detections can be run for arbitrary list lengths, or budgets. This enables several use cases: building very short lists of highly confident object predictions (high precision), long lists of many candidate regions (high recall), and dynamic length lists tuned by some heuristic(s) (e.g., the highest predicted IoU score of the remaining candidates).

II. Related Work

Much work has been done in the combined areas of object detection and semantic labeling. Object detection approaches often seek to place bounding boxes around all instances of objects [1], [8]. [9] casts the multi-class (and multi-instance) detection problem as a structured prediction task instead of NMS as post processing. However, the resulting detections are still bounding boxes.

Intermediate approaches deform the regions inside the output of a detector to produce object segmentations [10], [11], [12], or, conversely, adjust bounding boxes based on low-level features such as boundaries, texture, and color [13], [14]. Again, these approaches refine individual detections relying on the initial detector output. In contrast, we attempt to find the best list of detections given a large collection of candidate detections and regions. Closer to our work, [15] proposes to use a deformable shape model to represent object categories in order to extract region level object detections. This approach reasons about occluders and overlapping detection by using depth layering and is designed for one specific shape model for region-based representation, while our approach is agnostic to the source of region segments and detection boxes.

Direct region-based techniques, such as [16], [17], [18], use region-based information to formulate detections, the produced detections are bounding boxes, and detection performance is analyzed using individual bounding box metrics. [19] produces region-wise segmentations, however they assume the existence of only one object in each image. [20] produces multiple region-wise segmentations, but contiguous and adjacent objects are not resolved, and ignore inter-class context. Other region-based techniques are segmentation algorithms that rely on combining low-level image features with class-specific models [21], [22], [23], [24], control segmentation parameters from object detection [25], or use the box-level detections as features for segmentation [26], [27]. These approaches attempt to find regions that best agree with both the region segments and individual detections but do not explicitly deal with the problem of finding the most consistent *list* of detections as we do.

Semantic systems such as [28], [29], [30], [31] do produce region-level labels, which can be grouped into detections, however *there is no notion of separate detections*; connected components of labeling are not grouped into their constituent object instances. [32] uses non-overlapping segmentation proposals in its first stage, thus allowing, in principle, the handling of multiple instances of the same class, without explicitly optimization for multi-instance settings. Although the evaluation criteria in [32] focuses on per-class overlap without accounting for multiple instances, the authors do note the possibility for multi-instance extension. Combining semantic labeling with object detectors has been explored in different ways. Several approaches were proposed to combine pixel-level classification labels and box-level detections into a single inference problem. For example, [33], [34], [35], [36] incorporate detections into a CRF model for semantic labeling. These techniques attempt to generate a holistic representation of the scene that combines objects and regions. These approaches rely on semantic segmentation. Our approach, while incorporating semantic segmentation, is agnostic to the input features, as well as to the source(s) from which candidate detections are generated.

Another group of approaches related to our work address the problem of generating proposals for regions or boxes that are likely to delineate objects, in a class-independent manner. The proposals can then be evaluated by a class-specific algorithm for object detection. They include, for example, generating regions by iterative superpixel grouping [37], [18], and ranking proposed regions [38] or boxes [39], [40] based on a learned objectness score. In [41], the authors investigate an iterative, class-specific region generation procedure that incorporates class-specific models at different scales, and requires bounding boxes as input. Our generation method, in comparison, directly optimizes the instance-based IoU metric, and we provide worst-case and probabilistic performance bounds. All of these approaches are complementary to our work in that we can potentially use any of them as input to our candidate generation step, thus, we incorporate and compare to several of them in our experiments.

III. Approach

Our task is to output a list of chunks, i.e., list of sets of superpixels as described in Section I, with high intersection over union (IoU) scores with each of the ground truth instances in the image. This metric is formalized in Section III-A. We decompose the task into two parts:

- **Generation of a set of candidate chunks** containing some elements that cover individual object instances.
- **Iterative construction of a list of chunks** by selecting from an arbitrarily generated set of candidate chunks so as to maximize a natural variant of intersection over union score for multiple object instances and multiple predictions.

In the second stage, the candidate chunks can be generated from any algorithm, providing our method with the ability to augment our set of grown candidates constructed by other means. We start by describing the method by which we build lists of detections for the second stage, and first define a natural scoring function to evaluate any input list of chunks

given ground truth on the pixels corresponding to objects of interest in a scene. We provide an efficient greedy algorithm that is guaranteed to optimize this metric to within a constant factor given access to ground-truth and this arbitrary set of (potentially overlapping) candidate chunks.

Our test-time approach, following recent work in structured prediction [4], [6], is to learn to emulate the sequential greedy strategy. The result is a predictor that takes a candidate set of chunks and iteratively builds a list of chunks that are likely to overlap well with separate objects in the scene.

We do not place assumptions on the given candidate set of chunks: the list predictor is agnostic to the way the candidate set of chunks is generated. Such a set can be heuristically generated in many ways, *e.g.*, those created from the baseline approaches described in Section IV. In Section III-C, we provide an algorithm designed to generate a candidate based on a fixed superpixel-based segmentation, and in Section III-D extend this algorithm to the case of growing multiple chunks per images.

A. Objective function and greedy optimization

We establish an objective function to evaluate the quality of any list, and devise a greedy algorithm to approximately maximize this objective function given access to the ground-truth. This will lead to the development of learning algorithm that produces a prediction procedure that operates on novel images.

Given an image with ground truth instance set $G = \{g_1, ..., g_m\}$ and candidate chunk set $C = \{c_1, ..., c_n\}$, our goal is to sequentially build a list of chunks out of C so as to maximize the sum of IoU's with respect to ground truth instances. Denoting $L = (c_i, c_j, \ldots, c_k)$ as a size-k list of chunks, we first establish correspondences between candidate chunks and ground truth instances to enable pairwise IoU computation. Note that each c_i is associated with at most one ground truth instance g_i, and each g_i is associated with at most one c_i. For analytic convenience, we augment G with $k - m$ dummy ground truth instances $\tilde{g}$ to deal with the case in which the length of the list is larger than the number of ground truth instances ($|L| > |G|$). Every chunk c has zero intersection with each $\tilde{g}$. Each feasible assignment corresponds to a permutation $\tilde{L} = (c_{p_1}, c_{p_2}, \ldots, c_{p_k})$ of L, and the sum of IoU scores for this permutation can be written as the following: $h(\tilde{L}; G) = \sum_{i=1}^{k} \frac{|c_{p_i} \cap g_i|}{|c_{p_i} \cup g_i|}$. It is natural to define the quality metric $f(L; G)$ of a list L to be the sum of IoU scores under the optimal assignment, i.e., $f(L; G) = \max_{\tilde{L} \in P(L)} h(\tilde{L}, G)$, where $P(L)$ denotes all permutations of L. With an abuse of notation, $L \subseteq C$ indicates all elements in L belong to C. Our goal during training is to find list L to maximize f:

$$\arg\max_{L \subseteq C} f(L; G) = \arg\max_{L \subseteq C} \{ \max_{\tilde{L} \in P(L)} h(\tilde{L}; G) \}. \quad (1)$$

This scoring metric, which is a natural generalization of the IoU metric common in segmentation and single instance detection [42], [43], encourages lists of a fixed length that contain chunks that are relevant and diverse in covering multiple ground truth instances. Unfortunately, the metric as written down does not possess a clear combinatorial structure like modularity or submodularity that would beget easy optimizability.

Interestingly, however, Problem (1) can be cast as an equivalent maximum weighted bi-partite graph matching problem. This problem can be shown to be a submodular maximization problem under matroid partition constraints, and a greedy algorithm as shown in Algorithm 1 has multiplicative performance guarantees [44]. In addition to these guarantees, such a greedy algorithm is desirable as it is easily imitable at test time, and has a recursive solution: the $k+1$ length list is exactly the k length list with the next greedily chosen item appended. The greedy algorithm behaves as follows: at each iteration, it chooses the chunk with the highest IoU with one of the remaining ground truth instances. More precisely, a chunk's best overlap with each remaining ground truth is defined as $y(c; G_{re}) = \max_{g \in G_{re}} \frac{|c \cap g|}{|c \cup g|}$ (the "greedy marginal"), where G_{re} is the set of remaining unpaired ground truth instances. At each step, the algorithm chooses the chunk with the highest $y(c; G_{re})$ value, appends it to the list (L^{gr}), and removes its associated ground truth from the set of remaining ground truth. This associated ground truth element is given by $\pi_{gr}(c; G_{re}) = \arg\max_{g \in G_{re}} \frac{|c \cap g|}{|c \cup g|}$.

Algorithm 1 Greedy List Generation with Ground-Truth Access

Input: Set of candidate chunks C, set of ground truth instances G, size of predicted list k
Output: A near-optimal list L^{gr} of chunks
$L^{gr} = \emptyset$, $G_{re} = G$
for $i = 1$ **to** k **do**
  $c_i^{gr} = \arg\max_{c \in C} y(c; G_{re})$, $g_i^{gr} = \pi_{gr}(c; G_{re})$. ▷ *choose the highest scoring (chunk, GT) pair*
  $L^{gr} = L^{gr} \oplus c_i^{gr}$. ▷ *append the chunk to the list*
  $G_{re} = G_{re} \setminus g_i^{gr}$. ▷ *remove the associated GT*
end for
Return L^{gr}

Critically, the greedy algorithm is recursive, meaning longer lists of predictions always include shorter lists, and is within a constant factor of optimal[1]:

Theorem 1: Let L_i^{gr} be the list of the first i elements in L^{gr} and L_i^* be the optimal solution of Problem (1) among size-i lists

$$f(L_i^{gr}; G) \geq \frac{1}{2} f(L_i^*), \forall i = 1, \cdots, k. \quad (2)$$

See the appendix for proof of Theorem 1, which invokes results from [44]. Theorem 1 implies that if we are given a budget $|L|$ to build the list, then each L_i^{gr} scores within a constant factor of the optimal list among all lists of budget i, for $i = 1, \ldots, |L|$. This is an important property for producing good predictions earlier in the list and for producing the list of chunks rapidly. The empirical performance is usually much better than this bound suggests.

B. List prediction learning

In essence, the greedy strategy is a sequential list prediction process where at each round it maximizes the marginal

[1]Although Problem (1) can be solved exactly, it requires knowledge of the instances to be matched and does not possess a recursive structure that enables simple creation of longer lists of prediction.

benefit given the previous list of predictions and ground truth association. Maximization of the marginal benefit at each position of the list yields chunks that have high IoU with ground truth instances and minimal overlap with each other. At test time, however, there is no access to the ground-truth. Therefore, we take a learning approach to emulate the greedy algorithm. We train a predictor to imitate Algorithm 1, with the goal of preserving the ranking of all candidate chunks based on the greedy increments $y(c; G_{re})$. This predictor uses both information about the current chunk and information about the currently built list to inform its predictions. In our experiments, we train random forests as our regressor with features as $\Phi(c, L)$ (each chunk's feature is a function of itself and the currently built list, as described in Section IV-B), and regression targets $y(c; G_{re})$ (the score for a chunk at each iteration is the greedy marginal, or how much a chunk candidate covers a new object instance). This regression of "region IoU " is similar to that explored in [45], except it is explicitly reasoning about multiple objects, as well as the current contents of the predicted list. The prediction procedure is similar to the greedy list generation as in Algorithm 1, with the difference that there is no access to the ground truth.

C. Growing for a single instance

To generate a set of diverse chunks (output of stage 1 in the detection process), we develop a class-specific algorithm that "grows" chunks via iterative addition of superpixels, with the goal of producing diverse candidate detections that cover each ground truth object instance. We first analyze the case where there is only a single object of interest g in the image. We consider a chunk c to be a union of superpixels s, i.e., $c = \cup_{i=1}^{n}\{s_i\}$. Let $R(c)$ denote the IoU score between c and g.

To grow a chunk, Algorithm 2 starts with an empty chunk (no superpixels), and adds single superpixels to the current chunk sequentially. After each addition, the resulting chunk is copied and added to the set of candidate chunks. Let $\alpha_i = \frac{|s_i \cap g|}{|s_i|}$ be the ratio of intersection area with ground truth to the size of a superpixel s_i. The set of chunks generated by the greedy algorithm described in Algorithm 2 is guaranteed to contain the optimal chunk if the input predictor $\mathcal{G}$ returned the exact value of α_i, i.e., $\hat{\alpha_i} = \alpha_i$.

Algorithm 2 Single Instance Chunk Growing Algorithm

Input: Set of superpixels S, grower predictor $\mathcal{G}$.
Output: A set of chunks, $C_{\mathcal{G}}$.
$c = \emptyset$, $C_{\mathcal{G}} = \emptyset$
Sort elements in S by decreasing order of $\hat{\alpha_i} = \mathcal{G}(s_i)$
for $i = 1$ **to** $|S|$ **do**
  $c = c \cup \{s_i\}$, $C_{\mathcal{G}} = C_{\mathcal{G}} \cup \{c\}$
end for
Return $C_{\mathcal{G}}$

Theorem 2: Let $\mathcal{G}^*$ be an oracle growing predictor, i.e., $\mathcal{G}^*(s_i) = \alpha_i = \frac{|s_i \cap g|}{|s_i|}$. The output set of $C_{\mathcal{G}}$ from Algorithm 2 by setting $\mathcal{G} = \mathcal{G}^*$ contains the best chunk given the set of superpixels S.

Fig. 2: Selected images of the *best* grown chunks for images with single and multiple objects. Each chunk grows independently of the others. Given the initial seed, superpixels are iteratively added to the growing chunk. The predictor greedily adds superpixels that it believes make the highest contribution to the overall class-specific IoU score of the currently growing chunk.

See appendix for proof. At testing time, we must give an estimate of α_i, i.e., $\hat{\alpha_i}$. We train a random forest regressor as our predictor $\mathcal{G}$ with features θ for estimation. We analyze the performance of Algorithm 2 under approximation by relating the squared regression error of $\mathcal{G}$ to the IoU score of the grown best chunk in the chain. We note that the test-time performance depends on both the size of the squared error and the number of predictions made. Notably, the error bound has no explicit dependence on the area sizes of ground truth object instances and images. See appendix for proof.

Theorem 3: Given a regressor $\mathcal{G}$ that achieves no worse than absolute error ϵ uniformly across all superpixels, let $c^*_{\mathcal{G}}$ be the best chunk in the predicted set $C_{\mathcal{G}}$. The IoU score of $c^*_{\mathcal{G}}$ is no worse than 2ϵ of the IoU score of the optimal chunk c^*: $R(c^*_{\mathcal{G}}) > R(c^*) - 2\epsilon$.

Corollary 1: Suppose regressor $\mathcal{G}$ has expected sq. error δ over the distribution of superpixels, let n be the number of superpixels in the image, then we have for any $\eta \in (0, 1)$, with probability $1 - \eta$: $R(c^*_{\mathcal{G}}) > R(c^*) - 2\eta^{-1}\sqrt{n\delta}$.

D. Growing for multiple instances

We run the growing algorithm more than once to cover multiple objects. Instead of making predictions based solely on features of each individual superpixel, we augment the information available to the predictor by including a feature of the current grown chunk, $\theta(s_i, c)$ (see Section IV-B for more information about grower features). This yields predictors that prefer choosing superpixels in close proximity to the currently growing chunk, and allows us not to explicitly encode contiguity requirements, as objects may be partially occluded in a way that renders them discontiguous. We also modify Algorithm 2 by "seeding" the chunks at a set of superpixel locations, L (the initialization step, $c = \emptyset$, becomes $c = \{s_i\}\ \forall s_i \in L$), and running the growing procedure on each of these seeds separately. See appendix for the pseudo-code description modified from Algorithm 2. In practice, we choose a seeding grid interval and a maximum chunk size cutoff, yielding $|C_{\mathcal{G}}| \sim 700$. In Figure 2, we visualize the sequential growth of the best chunks for each object instance.

IV. Experiments

We describe our experiments and features in the next two sections, and discuss the results of each experiment in their

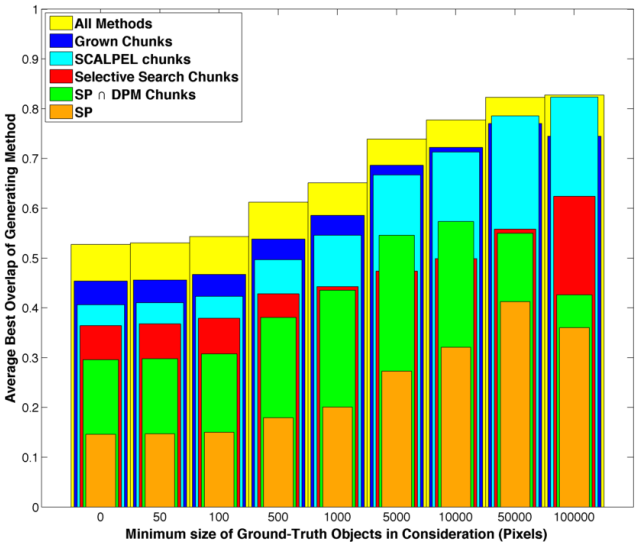

Fig. 3: LM+Sun Adjacent Cars candidate quality, measured by Average Best Overlap (ABO). We find that our grown chunks generally outperform the Selective Search and SCALPEL methods. On average, SCALPEL generated 893 regions per image, Selective Search generated 552 regions per image, SP ∩ DPM generated 8 chunks per image, and our grower generated 705 chunks per image.

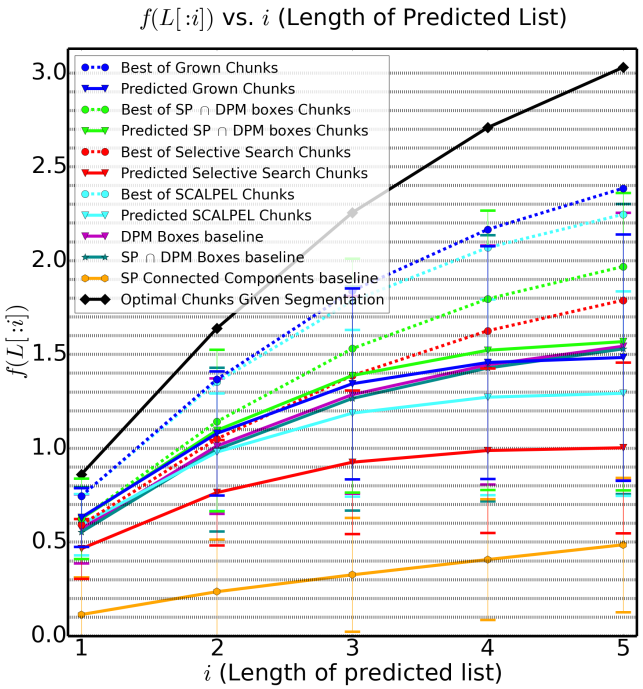

Fig. 4: $f(L;G)$ performance of lists constructed from best candidates from each prediction pool (dashed lines) and predicted candidates (solid lines) on a 50/50 split of LM+Sun Adjacent Cars dataset. Our chunk generating method (dark blue) generates candidates of similar quality to that of SCALPEL (light blue). On this dataset, our DPM-based baselines (magenta and dark cyan) perform quite well, but the best performing list prediction method (green line) is our list predictor that uses the SP ∩ DPM chunks as the candidate pool, and essentially has learned how to reorder them. This demonstrates how our approach can utilize and improve different candidate sources.

respective captions.

A. Datasets and baseline algorithms

We perform experiments on imagery from 3 different datasets. We refine the Stanford Background Dataset [46] labeling to include a `vehicle` class with instance labeling. We also perform experiments on PASCAL VOC 2012 (Fig. 5 and Tables II and I). This dataset possesses relatively few images containing adjacent and/or overlapping instances of the same class. Therefore, we created a subset of the LM+Sun dataset [31] of images containing at least 2 adjacent cars, consisting of 1,042 images.

B. Features

As discussed in III-B, the features $\Phi(c, L)$ should encode the *quality* of a chunk c (e.g. "Does the chunk look like a vehicle?") and *similarity* with the currently predicted list L (e.g. "Is this chunk similar to previously predicted chunks"). One of the quality features is built upon the superpixel-wise multi-class label distribution from [30], where we compute label distribution for each chunk via aggregating histograms of its constituent superpixels. The other quality features are shape features including central moments of the chunk, area, and scale relative to the image. The similarity features we use primarily encode spatial information between predictions. We use a candidate's $\frac{I}{U}$ with previous predictions, the spatial histogram used in [9] and the size of the current list. Chunks with high similarity with previously predicted chunks in the list are less favored.

The features $\theta(s, c')$ for the grower encode information about the quality of proposed chunk $c' = c \cup \{s\}$ by growing c with superpixel s. The grower uses the same quality features that characterize c' used by the list predictor, as well as several of the class-agnostic features described in [18], specifically color similarity (color histogram intersection), which encourages regions to be built from similarly colored regions, and region fill, which encourages growing compact chunks. See [18] for further details. As each superpixel is iteratively added to the chunk, similarity to the growing chunk for remaining candidate superpixels is recomputed.

We evaluate three methods[2] leveraging existing bounding box detections and superpixel-wise semantic labeling algorithm, all of which serve as our baseline systems for building lists of predictions: **1)** Bounding box detector output after NMS filtering **2)** Connected components of scene parsing ("SP") / semantic labeling **3)** A combination of **1)** and **2)**: intersection of connected components with bounding boxes, which creates chunks for every bounding box by extracting the labeled region inside ("SP ∩ DPM"). The third baseline

[2] We use the semantic labeling algorithm of [30] and the DPM detection method of [1] for bounding box output, with the default SVM threshold, and NMS threshold of 0.5. To generate the superpixels, we use the segmentation algorithm of [5]. For each experiment, separate semantic labeling systems and chunk growers were trained.

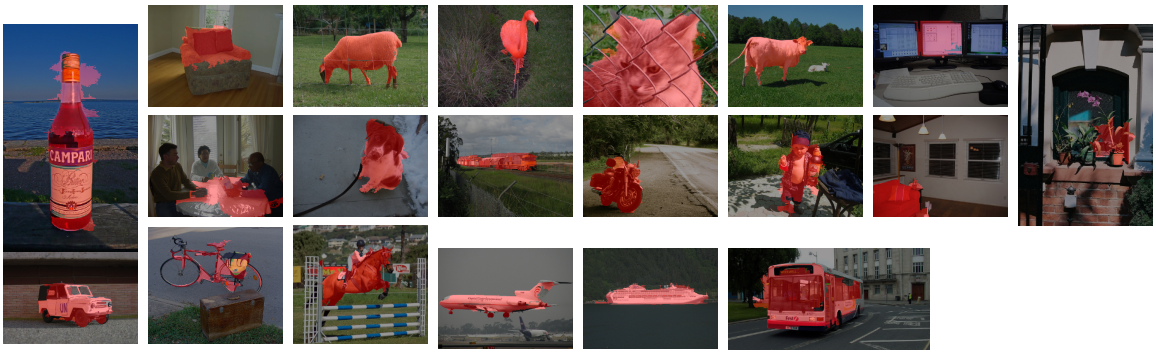

Fig. 5: Example single predictions on PASCAL VOC2012

TABLE I: Average instance-based accuracy (a metric proposed by [15]), and first slot scores (corresponding to the average overlap of the first prediction in each image) for systems trained and tested on the standard PASCAL 2012 `train` and `val` sets. We find that the very small amount of co-occurring instance training data was not sufficient to enable our system to perform as well as it did in our other experiments on images with co-occurring instances. While [15] provides experimental results of average instance-based accuracy on PASCAL 2010, their results are confined to verified correct DPM detections, rendering a fair comparison difficult.

	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow
inst_{acc}	.157	.066	.105	.132	.079	.228	.097	.155	.071	.211
$f(L[0])$	.521	.148	.375	.335	.186	.439	.190	.445	.141	.494
$f(L)$	.530	.158	.394	.347	.202	.509	.195	.461	.171	.581

	diningtable	dog	horse	motorbike	person	pottedplant	sheep	sofa	train	tvmonitor
inst_{acc}	.098	.165	.197	.166	.193	.078	.182	.139	.170	.109
$f(L[0])$	.260	.430	.437	.407	.362	.133	.466	.260	.403	.270
$f(L)$	.261	.454	.479	.441	.456	.160	.582	.272	.409	.278

TABLE II: List prediction and baseline performance on VOC2012 Person validation data and an 80/20 split of SBD Vehicles. Our list prediction outperforms all baselines in both experiments. In SBD Vehicles, the most competitive is the Scene Parsing intersected with DPM Bounding boxes. In VOC2012 Person, scene parsing was lower quality, and resulted in the DPM Boxes outperforming other baselines.

	SBD Vehicle					VOC2012 Person		
	$f_{\text{L}[0]}$	$f_{\text{L}[0:1]}$	$f_{\text{L}[0:2]}$	$f_{\text{L}[0:3]}$	$f_{\text{L}[0:4]}$	$f_{\text{L}[0]}$	$f_{\text{L}[0:1]}$	$f_{\text{L}[0:2]}$
$\overline{R(c^*)}$ (mean optimal chunks given segmentation)	0.82	1.43	1.87	2.19	2.44	0.83	1.17	1.33
$\overline{R(c^*_G)}$ (mean best grown chunks)	0.69	1.14	1.45	1.66	1.81	0.52	0.71	0.79
List Prediction with Grown and Baseline Chunks	**0.58**	**0.89**	**1.08**	**1.18**	**1.25**	**0.38**	**0.50**	**0.53**
List Prediction with Selective Search	-	-	-	-	-	0.27	0.36	0.41
Scene Parsing ∩ DPM Baseline	0.56	0.79	0.91	1.02	1.07	0.16	0.19	0.21
Connected Components Baseline	0.37	0.53	0.60	0.65	0.66	0.19	0.24	0.27
DPM Baseline	0.28	0.39	0.43	0.45	0.47	0.29	0.38	0.41

DPM SP ∩ DPM Chunking DPM SP ∩ DPM Chunking DPM SP ∩ DPM Chunking

Fig. 6: Comparison of list prediction versus other baselines. Each group of images contains, from left to right, the results of DPM, DPM intersected with Scene Parsing, and Visual Chunking. Note that while the Scene Parsing intersected with a bounding box detector can perform well, it fails in the case of poor NMS performance (group 1), and requires highly accurate Scene Parsing. Visual Chunking outperforms this baseline by instead building a list of detections.

is intended to capitalize on desirable properties of each component while avoiding their less desirable properties: boxes usually violate the object boundaries, and semantic labeling does not separate adjacent instances. The downside to this baseline is that it can suffer from compounding both detector and scene parsing errors. See Fig. 6 for a visual comparison.

We investigate the region generating methods of

SCALPEL [41] and Selective Search [18], and in Fig. 3 compare our chunk generating method against them on our LM+Sun Adjacent Cars dataset with the Average Best Object method suggested by [41], and additionally train our system by using these methods to fill the candidate pool. In Table II, we compare different list predictions methods on `vehicle` and `person` data, respectively.

V. Conclusion

We provide a novel method for producing region-based object detections in images, treating the problem as a list prediction from a set of candidate region proposals. We formulate a scoring criterion for multiple object instances and multiple predictions. We develop a list prediction algorithm that directly optimized the criterion. Our approach is agnostic to proposal generation method and provides a recursive solution for all list lengths, enabling it to easily produce any k best guesses for objects. We provide a method for *class-specific* candidate generation algorithm, yielding good coverage of objects. We demonstrate that our list prediction is a useful method for improving arbitrary candidate pools.

References

[1] P. F. Felzenszwalb *et al.*, "Object detection with discriminatively trained part-based models," *PAMI*, 2010.
[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.
[3] D. Debadeepta *et al.*, "Contextual sequence prediction with application to control library optimization," *ICML*, 2013.
[4] S. Ross *et al.*, "Learning policies for contextual submodular prediction," in *ICML*, 2013.
[5] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, 2004.
[6] H. Daumé III *et al.*, "Search-based structured prediction," *Machine Learning*, 2009.
[7] S. Ross *et al.*, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.
[8] A. Vedaldi *et al.*, "Multiple kernels for object detection," in *CVPR*, 2009.
[9] C. Desai *et al.*, "Discriminative models for multi-class object layout," *ICJV*, 2011.
[10] V. Lempitsky *et al.*, "Image segmentation with a bounding box prior," in *ICCV*, 2009.
[11] A. Monroy and B. Ommer, "Beyond bounding-boxes: Learning object shape by model-driven grouping," in *ECCV*, 2012.
[12] J. Z. Wang *et al.*, "Simplicity: Semantics-sensitive integrated matching for picture libraries," *PAMI*, 2001.
[13] Q. Dai and D. Hoiem, "Learning to localize detected objects," in *CVPR*, 2012.
[14] R. Mottaghi, "Augmenting deformable part models with irregular-shaped object patches," 2012.
[15] Y. Yang *et al.*, "Layered object models for image segmentation," *PAMI*, 2012.
[16] B. Leibe *et al.*, "Robust object detection with interleaved categorization and segmentation," *IJCV*, 2008.
[17] C. Gu *et al.*, "Recognition using regions," in *CVPR*, 2009.
[18] J. Uijlings *et al.*, "Selective search for object recognition," *IJCV*, 2013.
[19] E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," in *ECCV*, 2002.
[20] J. Carreira *et al.*, "Object recognition by sequential figure-ground ranking," *IJCV*, 2012.
[21] B. Leibe *et al.*, "Combined object categorization and segmentation with an implicit shape model," in *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.
[22] X. Y. Stella *et al.*, "Concurrent object recognition and segmentation by graph partitioning," in *NIPS*, 2002.
[23] A. Levin and Y. Weiss, "Learning to combine bottom-up and top-down segmentation," in *ECCV*, 2006.
[24] Z. Tu *et al.*, "Image parsing: Unifying segmentation, detection, and recognition," *IJCV*, 2005.
[25] M. P. o. Kumar, "Obj cut," in *CVPR*, 2005.
[26] J. M. Gonfaus *et al.*, "Harmony potentials for joint classification and segmentation," in *CVPR*, 2010.
[27] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *NIPS*, 2011.
[28] G. Heitz and D. Koller, "Learning spatial context: Using stuff to find things," in *ECCV*, 2008.
[29] S. Gould *et al.*, "Decomposing a scene into geometric and semantically consistent regions," in *ICCV*. IEEE, 2009.
[30] D. Munoz *et al.*, "Stacked hierarchical labeling," in *ECCV*, 2010.
[31] J. Tighe and S. Lazebnik, "Superparsing: scalable nonparametric image parsing with superpixels," in *ECCV*, 2010.
[32] A. Ion *et al.*, "Probabilistic joint image segmentation and labeling." in *NIPS*, 2011.
[33] L. Ladickỳ *et al.*, "What, where and how many? combining object detectors and crfs," in *ECCV*, 2010.
[34] S. Fidler *et al.*, "Bottom-up segmentation for top-down detection," in *CVPR*, 2013.
[35] G. Heitz *et al.*, "Cascaded classification models: Combining models for holistic scene understanding," in *NIPS*, 2008.
[36] J. Yao *et al.*, "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation," in *CVPR*, 2012.
[37] A. Levinshtein *et al.*, "Optimal contour closure by superpixel grouping," in *ECCV*, 2010.
[38] I. Endres and D. Hoiem, "Category independent object proposals," in *ECCV*, 2010.
[39] B. Alexe *et al.*, "What is an object?" in *CVPR*, 2010.
[40] ——, "Measuring the objectness of image windows," *PAMI*, 2012.
[41] D. Weiss and B. Taskar, "Scalpel: Segmentation cascades with localized priors and efficient learning," in *CVPR*, 2013.
[42] M. Everingham *et al.*, "The pascal visual object classes (voc) challenge," *IJCV*, 2010.
[43] P. Arbeláez *et al.*, "Semantic segmentation using regions and parts," in *CVPR*, 2012.
[44] M. L. Fisher *et al.*, "An analysis of approximations for maximizing submodular set functions ii," in *Polyhedral combinatorics*, 1978.
[45] J. Carreira and C. Sminchisescu, "Cpmc: Automatic object segmentation using constrained parametric min-cuts," *PAMI*, 2012.
[46] S. Gould *et al.*, "Decomposing a scene into geometric and semantically consistent regions," in *ICCV*, 2009.

VI. Appendix

This appendix contains proofs of theoretical results and additional pseudo-code descriptions presented in the paper.

A. Proof for Theorem 1

Given an image with ground truth entities $G = \{g_1, g_2, \ldots, g_m\}$, candidate chunks set $C = \{c_1, c_2, \ldots, c_n\}$ and list size budget k, our goal is to select the optimal k chunks out of C and associate each with the ground truth entities so as to maximize the sum of intersection over union scores under the association. Such problem can be cast as maximum weighted bi-partite graph matching, a classic assignment problem in combinatorial optimization. The edge set E of the bi-partite graph is the Cartesian product of G and C, i.e., $E = C \times G$. The weight w_{ij} for each edge e_{ij} is the I/U score between chunk c_i and ground truth g_j. The defined quality metric $f(L, G)$ in Section 3.1 is equal to the optimal assignment score for subgraph $L \times G$.

Let $V^* \subseteq C$ be the optimal size-k subset of chunks, which can be computed in cubic time by Hungarian Algorithm [**?**]. Algorithm 1 can be seen as a greedy approach for maximum bi-partite graph matching with 1/2 approximation guarantee [**?**]. Furthur, let $L_{gr}(V)$ be the greedy match on graph $V \times G$, we can show that for any augmented graph $V' \times G$ where $V \subseteq V'$, $L_{gr}(V')$ obtained from running Algorithm 1 with k iterations is no worse than $L_{gr}(V)$. Hence, we can conclude that running Algorithm 1 on $C \times G$ has 1/2 approximation gurantee with respect to the optimal size-k subset of C. Together with the fact that greedy solution has recursive structure, i.e., shorter greedy list is the prefix list for longer greedy list under larger budget, we can prove Theorem 1.

B. Proof of Theorem 2

Consider superpixel s_i and ground truth g , let $\Delta_{x_i} = \|s_i \cap g\|$, $\Delta_{y_i} = \|s_i \cup g\| - \|g\|$ and $r_i = \frac{\Delta_{x_i}}{\Delta_{y_i}}$, we have that $\alpha_i = \frac{\Delta x_i}{s_i}$ is a montonic transformation of r_i, i.e., $r_i \geq r_j$ if and only if $\alpha_i \geq \alpha_j$. Therefore the rankings based on α_i or r_i are the same. This follows from the fact that $\frac{1}{\alpha_i} - 1 = \frac{1}{r_i}$.

Using the fact that superpixels are **non-overlapping**, given any superpixel s_i and a set of superpixel c, we have $R(c \cup \{s_i\}) = \frac{\cap(c,g)+\Delta_{x_i}}{\cup(c,g)+\Delta_{y_i}}$. Further, if $r_i \geq R(c)$, adding s_i to c would increase $R(c)$ and vice versa, since $r_i = \frac{\Delta_{x_i}}{\Delta_{y_i}} > \frac{\cap(c,g)}{\cup(c,g)}$ implies $r_i > R(c \cup \{s_i\}) > R(c) = \frac{\cap(c,g)}{\cup(c,g)}$. Therefore, suppose the optimal solution be c^*, then if $r_i > R(c^*)$, it must be true that $s_i \in c^*$, and otherwise $s_i \notin c^*$. This also implies the optimal set of superpixels is the first k elements based on a sorting of superpixels by r_i, where k is the smallest integer such that $r_{k+1} \leq R(c^*)$.

C. Proof of Theorem 3 and Corollary

Let $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_N$ and let the optimal set of superpixels to maximize I/U with ground truth g be the first k superpixels, i.e., $c^* = \{s_1, s_2, \ldots, s_k\}$; suppose the regressor makes bounded uniform error ϵ, i.e., $|\hat{\alpha}_i - \alpha_i| < \epsilon$, and let M be the largest number such that: $\alpha_j \geq \alpha_k - 2\epsilon$ for $j = 1, ..., M$. If the regressor makes bounded uniform error ϵ, then the worst case would be: it underestimates $\alpha_1, ..., \alpha_k$ by ϵ and overestimates $\alpha_{k+1}, ..., \alpha_M$ by ϵ. Therefore, some of the elements in $\alpha_{k+1}, ..., \alpha_M$ would rank higher than elements in $\alpha_1, ..., \alpha_k$. Denote $c^*_{\mathcal{G}}$ as the best solution among the chain of sets induced by the ranking output of the regressor $\mathcal{G}$.

$$R(c_{\mathcal{G}^*}) \geq R(\{s_1, ..., s_M\}) = \frac{\sum_{i=1}^{M} s_i \alpha_i}{g + \sum_{i=1}^{M} s_i(1 - \alpha_i)} \tag{3}$$

$$= \frac{\sum_{i=1}^{k} s_i \alpha_i + \sum_{j=k+1}^{M} s_j \alpha_j}{g + \sum_{i=1}^{k} s_i(1 - \alpha_i) + \sum_{j=k+1}^{M} s_j(1 - \alpha_j)} \tag{4}$$

$$\geq \min\{\frac{\sum_{i=1}^{k} s_i \alpha_i}{g + \sum_{i=1}^{k} s_i(1 - \alpha_i)}, \frac{\sum_{i=k+1}^{M} s_i \alpha_i}{\sum_{i=k+1}^{M} s_i(1 - \alpha_i)}\} \tag{5}$$

$$= \min\{R(c^*), \frac{\sum_{i=k+1}^{M} s_i \alpha_i}{\sum_{i=k+1}^{M} s_i(1 - \alpha_i)}\} \tag{6}$$

$$= \frac{\sum_{i=k+1}^{M} s_i \alpha_i}{\sum_{i=k+1}^{M} s_i(1 - \alpha_i)} \geq \frac{\sum_{i=k+1}^{M} s_i(\alpha_k - 2\epsilon)}{\sum_{i=k+1}^{M} s_i(1 - (\alpha_k - 2\epsilon))} \tag{7}$$

$$= \frac{(\alpha_k - 2\epsilon)}{(1 - \alpha_k + 2\epsilon)} \geq \frac{R(c^*)(\alpha_k - 2\epsilon)}{\alpha_k + 2\epsilon R(c^*)} \tag{8}$$

In 8, we are using the fact that $r_k = \frac{\alpha_k}{1-\alpha_k} \geq R(c^*)$ implies $1 - \alpha_k \leq \frac{\alpha_k}{R(c^*)}$. Rearrange the terms, we get:

$$\frac{R(c)}{R(c^*)} \geq \frac{\alpha_k - 2\epsilon}{\alpha_k + 2\epsilon R(c^*)} \geq 1 - \frac{2(1 + R(c^*))}{\alpha_k}\epsilon \tag{9}$$

$$\geq 1 - \frac{4}{(1/R(c^*) + 1)}\epsilon \tag{10}$$

From (9) to (10), we are using the fact that $\frac{1}{r_k} = \frac{1}{\alpha_k} - 1 \leq \frac{1}{R(c^*)}$ and $R(c^*) \leq 1$. We can proceed to have an additive bound:

$$R(c^*) - R(c) \leq \frac{4R(c^*)}{1/R(c) + 1}\epsilon \tag{11}$$

$$\leq 2\epsilon \tag{12}$$

A more natural assumption is to assume an expected square error ϵ over the distribution P of all superpixels. Denote $\delta = \mathbb{E}_{i \sim P}(\hat{r_i} - r_i)^2$, the expected uniform error bound $\mathbb{E}[\epsilon]$ satisfies:

$$\mathbb{E}[\epsilon] = \mathbb{E}[\max_i |\epsilon_i|] = \mathbb{E}[(\max_i \epsilon_i^2)^{\frac{1}{2}}] \tag{13}$$

$$\leq \mathbb{E}[(\sum_i \epsilon_i^2)^{\frac{1}{2}}] \leq \mathbb{E}[n\epsilon^2]^{1/2} \tag{14}$$

$$\leq \sqrt{n\delta} \tag{15}$$

In (14), we are applying Jensen's Inequality along with the fact that $\sqrt{x}$ is concave. Using Markov Inequality, for any $\eta \in (0, 1)$, with probability $1 - \eta$, we have that:

$$\epsilon \leq \frac{\sqrt{n\delta}}{\eta} \tag{16}$$

Together with 12, we have that: for any $\eta \in (0, 1)$, with probability $1 - \eta$

$$R(c^*_{\mathcal{G}}) > R(c^*) - 2\frac{\sqrt{n\delta}}{\eta} \tag{17}$$

D. Pseudocode for Multiple Grower Algorithm

Algorithm 3 Multiple Instance Chunk Growing Algorithm

Input: Set of superpixels S, grower predictor $\mathcal{G}$, seeding superpixel s'
Output: A set of chunks $C_{\mathcal{G}}$.
$c = \{s'\}$, $C_{\mathcal{G}} = \emptyset$
for $i = 1$ **to** $|S|$ **do**
  $s_i = \text{argmax}_{s \in S}\, \mathcal{G}(s, c)$
  $c = c \cup \{s_i\}$, $C_{\mathcal{G}} = C_{\mathcal{G}} \cup \{c\}$
end for
Return $C_{\mathcal{G}}$

The pseudocode in Algorithm 3 describes the growing algorithm with specified seeding superpixel s'. We run this algorithm for each $s' \in L$ in order to increase diversity, where L is the set of seeding superpixels. Two major differences from single instance chunk growing algorithm in Section 3.2 are addressed below:

1) A seeding superpixel s' needs to be given as input to initialize the chain of growth, i.e., $c = \{s'\}$.
2) Instead of just using features only based on the superpixel s itself, we also consider features including both the superpixel and the currently growing chunk c. We replace $\hat{\alpha_i} = \mathcal{G}(s_i)$ with $\hat{\alpha_i} = \mathcal{G}(s_i, c)$. These feature not only encode information about the quality of a superpixel but also encourage the grower to grow spatially compact chunks.