A Hierarchical and Contextual Model for Aerial Image Parsing

Jake Porway[1], Kristy Wang[1,2], and Song-Chun Zhu[1,2]

Abstract

In this paper we present a hierarchical and contextual model for aerial image understanding. Our model organizes objects (cars, roofs, roads, trees, parking lots) in aerial scenes into hierarchical groups whose appearances and configurations are determined by statistical constraints (e.g. relative position, relative scale, etc.). Our hierarchy is a non-recursive grammar for objects in aerial images comprised of layers of nodes that can each decompose into a number of different configurations. This allows us to generate and recognize a vast number of scenes with relatively few rules. We present a minimax entropy framework for learning the statistical constraints between objects and show that this learned context allows us to rule out unlikely scene configurations and hallucinate undetected objects during inference. A similar algorithm was proposed for for texture synthesis [42] but didn't incorporate hierarchical information. We use a range of different bottom-up detectors (AdaBoost, TextonBoost, Compositional Boosting [7, 23, 37]) to propose locations of objects in new aerial images and employ a cluster sampling algorithm (C4 [22]) to choose the subset of detections that best explains the image according to our learned prior model. The C4 algorithm can quickly and efficiently switch between alternate competing sub-solutions, for example whether an image patch is better explained by a parking lot with cars or by a building with vents. We also show that our model can predict the locations of objects our detectors missed. We conclude by presenting parsed aerial images and experimental results showing that our cluster sampling and top-down prediction algorithms use the learned contextual cues from our model to improve detection results over traditional bottom-up detectors alone.

[1] Department of Statistics, University of California at Los Angeles, Los Angeles, CA 90095.

[2] Lotus Hill Research Institute, Ezhou, China.

1 Introduction

1.1 Objectives and Motivation

Aerial image understanding is an important field of research for tackling the problems of automated navigation, large scale 3D scene construction, and object tracking for use in event detection. Most of the tasks using aerial images need or would benefit from a full explanation of the scene, consisting of the locations and scales of detected objects and their relationships to one another. Being able to identify objects of many different types and understand their relationships to one another gives a deeper understanding of the data and allows subsequent algorithms to make smarter decisions faster.

There are difficulties in aerial image parsing that do not arise in more restricted recognition tasks. Three major obstacles to modeling aerial images are,

1. Highly Variant Configurations: Objects in aerial images can appear at many different locations, scales, and orientations in the image, creating a vast number of possible configurations. There may also be hundreds of objects present, making it infeasible to create a rigid model that can enumerate every possible spatial layout.

2. Multi-resolution: Objects in aerial images appear at a number of resolutions, from small cars about 10 pixels wide to massive roofs more than 800 pixels long. There is no single feature or detector that is likely to perform well across all categories and all sizes of object.

3. Coupling Constraints: Certain objects frequently appear together under certain constraints, for example cars often appear in parking lots. When performing inference to find the best explanation of the scene, one must make sure the explanation adheres to this strong coupling. If an image patch is explained by a roof with vents on top of it, changing that explanation to a parking lot with cars in it requires updating all of the objects involved at once. Single site sampling methods are insufficient for this task, as they would add/subtract single objects to and from the current explanation one at a time, each of which would be a very low-probability state on its own. We need some way to switch *all* the involved objects at once, removing the roof from the explanation while at the same time adding the cars and parking lot to the explanation together.

To motivate our solution to the problems above, let us look at an example of parsing an aerial image. Our goal is to simultaneously detect objects in images and organize them into a hierarchical contextual representation for the scene, i.e. which objects are grouped together and how they're related to one another. Figure 1 shows an example of what a parsed scene looks like. Figure 1(b) shows a flat configuration of detected objects in a typical aerial image. In Figure 1(c) these objects have been grouped hierarchically - nearby cars are aligned in rows, trees form treelines, and proximal buildings form city blocks, for example.

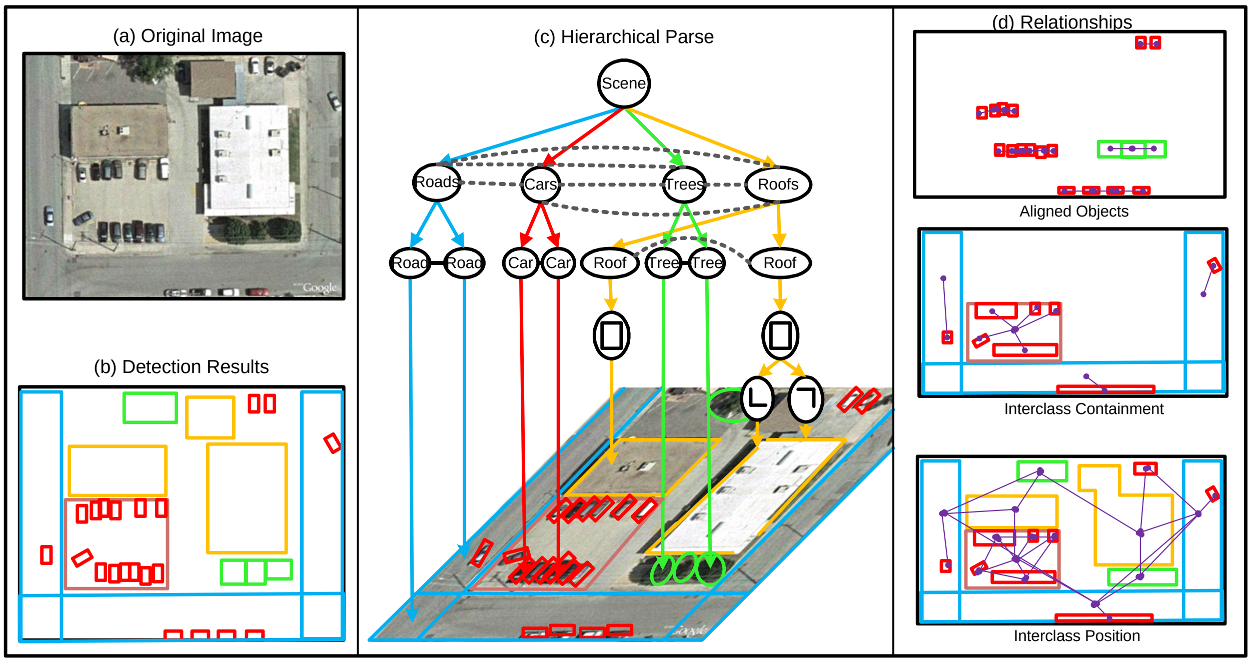


Figure 1: An example of a hierarchically parsed aerial image. (a) The original image. (b) A flat configuration of objects in the scene. (c) A hierarchical parse graph of the scene. (d) Three typical contextual relationships and the objects related by them.

This hierarchical representation explains the image at many resolutions and abstracts the objects into loosely related groups. It also models the number of groups we see in each image and the number of objects we observe in each group. Figure 2 shows a visualization of this representation.

Even though objects in groups don't appear in rigid formations the way the parts of a single object might, we recognize certain spatial and appearance constraints that they must obey. For example, we would never expect to see two cars on top of one another or a building smaller than a tree. These constraints between objects and groups are represented as horizontal lines between nodes in Figure 1(c). Figure 1(d) shows examples of which objects could be related by certain constraints. The first panel shows which cars and trees are aligned in a straight line. The second panel shows which objects contain other objects, and the last panel shows that almost all objects in this example are related by their relative position. We want our model to automatically learn which constraints between which objects are sufficient to represent the scene. If objects in a scene violate constraints that we expect to see or don't meet enough of them, then that interpretation of the scene will have a low probability. Such constraints would also rule out invalid object detections (e.g. cars on top of trees).

1.2 Major Contributions

We present a 3-layer hierarchy with embedded contextual constraints along with a 3-stage inference algorithm to solve the problems above and capture the natural hierarchical and contextual nature of aerial images.

1. Hierarchical and Contextual Model: To handle the large structural variations of aerial images we model scenes as groups of like objects, such as cars aligned in rows or roofs clustered into city blocks (see Figure 2). This abstracts the scene into loosely related neighborhoods. We then add statistical constraints within and between these groups to constrain their relative appearances, such as how close together they are or what size they are. We achieve this by embedding Markov random fields (MRFs) into a hierarchical grammar model. Our grammar model begins at a root scene node that then probabilistically decomposes into a number of group nodes, such as n groups of cars or m groups of roofs. Each of these groups can in turn decompose into a number of single nodes, such as j cars in one group, k cars in another. Alone this hierarchy only captures the frequency of objects, however, so we add contextual relationships via MRFs on the neighborhoods within each group and between groups. In this way we create a statistical model that uses a small set of decomposition rules to generate variable number of objects whose appearances are constrained by statistical relationships.

2. Automatic Learning of Context by Relationship Selection: Creating the constraints for our model by hand is infeasible due to the huge number of potential constraints that could exist, so we present an algorithm to automatically add statistical constraints to the hierarchy in a minimax entropy framework. This minimax entropy technique was used previously in texture modeling [42] and on more general graphical models [34] but we are now applying it to a hierarchical model. We seek to model the true distribution of aerial images, f, with our learned distribution, p, by iteratively matching feature statistics between f and p. This matching entails extracting features from a set of observed data (which follows f) and adjusting our model p such that it reproduces the statistics of these features. This learning method automatically selects the most important feature statistics to match and ignores low-information features. This allows us to add only relevant relationships from a large dictionary of potential constraints. By the end of this process, samples from p appear similar to true samples from f along the learned dimensions.

3. Flexible Detectors for Multiple Object Categories: The 3-level hierarchy terminates at object nodes, below which we may plug in any detectors that we like for each object type. Large textured regions, such as grass, trees, parking lots, water, and dirt, are detected using a Bag-of-Textons classifier. In this work, small patch-like objects, like cars, are detected using Haar features and AdaBoost. Roofs and roads, which have many different colors and shapes, are detected using edges as features and a compositional algorithm called Compositional Boosting [37] for detection. Compositional Boosting is itself a hierarchical detector and groups edges into larger and larger structures if they meet certain appearance constraints. In our work, edges detected in the image are grouped into corners, T-junctions, long lines, etc. that are then grouped into polygons. These polygonal boundaries are stronger indicators of a roof's presence than color or texture are and can model many different roof shapes.

4. Top-Down Bayesian Inference with Cluster Sampling and Prediction: To handle the coupling constraints that appear in aerial images we use a sampling algorithm inspired by Swendsen-Wang clustering [1, 28]. Swendsen-Wang cluster sampling was introduced to sample the Potts model more effectively by updating a cluster of sites at once instead of just a single site. Our variant of this algorithm, named **Clustering via Cooperative and Competitive Constraints (C4)**[22], updates multiple clusters at once, allowing us to move very rapidly in the solution space. The clusters represent competing explanations of the scene. For example, a patch explained by a parking lot with cars may be better explained by a single roof. To swap these explanations, we couple the cars with the parking lot and switch the whole cluster with the roof in a single step. This process results in an explanation of the scene with very few false positives. We also use the hierarchical nature of our top-down model to propose new objects our object detectors may have missed, thus increasing the number of true positives in our final result.

It bears noting that our training process uses hand-labeled images as input. This requires having a human identify objects of interest in images and label their boundaries. Thankfully only the boundaries of the objects need to be labeled, regardless of the number of contextual relationships being learned by the algorithm.

1.3 Related Work

Our work is related to two subfields of computer vision: aerial image parsing and hierarchical object recognition. In the aerial image parsing literature it is very rare to find work that detects multiple types of objects simultaneously. Most work focuses on detecting just one type of object, rarely using context or hierarchy to model the whole scene. In the object recognition category we often see complex hierarchical and contextual models. However, these models are often designed for rigid objects where appearance constraints are fairly constant between instances of the object.

Much work has been done on identifying single objects in aerial images, such as rooftops [16, 32, 36], cars [15, 39], or roads [19]. In these cases context plays little role, as single objects are detected without taking the support of surrounding objects into consideration. These works use similar object detectors to those we use, though they almost exclusively use one detector without considering the support from multiple detectors. These detectors include AdaBoost [7, 33], Bag of Words [2, 26] and TextonBoost [23].

Some aerial imaging works incorporate context and/or multiple object category detection into the same framework. SIGMA [17], a knowledge-based "expert system" for aerial images, was an attempt to model rule-based spatial relationships between objects. Unfortunately, as in much of the computer science based AI work of that time, relationships were often hardcoded and thus not generally extensible. On a smaller scale, Moissinac identified roads and city blocks in urban scenes using local context rules to decide how roads connect and how blocks should appear [18]. Hinz used positional relations to determine the likely positions of roads in aerial images [10]. A recent approach for parsing images of outdoor scenes by Berg [2] also seeks to model images as collections of regions that obey positional and relational constraints. As far as we know, however, these models require a good deal of hand-tuning and hardcoded logic in order to encode the relevant constraints. SIGMA relied on experts to identify relationships of interest to model, Moissinac knew exactly the domain he was working with (handdrawn maps) and designed relationships accordingly, and Berg et al. used domain knowledge of the objects they wanted to identify to design contextual cues. Our model improves upon this shortcoming by employing a minimax entropy learning framework to automatically select significant relationships from a bank of potential relationships that can be designed to work on many domains of data without constant user input.

Our model borrows concepts from grammars in natural language, so we would be remiss not to credit the large and growing field of object recognition using hierarchies and context in non-aerial image recognition. Growing out of the early grammar work by Fu and Ohta [8, 20] on grammars for line drawings, much recent work has attended to the process of learning compositional object structures that employ local context to resolve ambiguities. The constellation model [35], pictorial structures [5, 6], and patch hierarchies [31] all use learned statistical constraints to model the relative position of object parts to some reference frame. Higher level compositional structures for object categories [14, 29] have shown great performance on object detection and localization tasks and can even be learned from unlabeled images, while work on rule-based models of shape have shown the power of statistical composition [13, 24]. Some of these models can express the general relationships present between shared object parts of the same category, a very useful trait for generalization.

In later sections we define our model as an exponential model over a graphical structure. We employ minimax entropy techniques seen in [42] to learn the parameters of this model and pursue its contextual relationships. However, we must acknowledge the huge number of contributions made in modeling graphical models as exponential families from all walks of machine learning and computer vision. We refer to the tutorial by Jordan and Wainwright on this topic [34] on similar uses of maximum entropy for modeling distributions on graphical models.

More general grammars for full scene modeling have developed recently [3, 9, 11, 30, 40, 41]. The work in [9, 30] seek to explain an entire image by parsing it hierarchically into constituent regions and objects, while [3, 11, 40, 41] focus more on single objects. These models borrow closely from models used in natural language processing, and express structural and appearance variation as the result of production rules. Hierarchical models for objects that include scene-level constraints have been presented in [25, 27], which are very similar in spirit to our model. The contextual constraints, however, tend to strictly be relative position constraints. Moreover, while this entire corpus of work is hugely important for object recognition, many of the contributions here rely on the fairly constant arrangement of a relatively fixed number of parts in objects. The variation in part appearance, frequency, and location for a motorcycle is far lower than that of the arrangement of cars and roofs in a city scene.

In the field of natural language processing, much progress has also been made on Unification-Based Grammars (UBG), which seek to model sentences by augmenting a tree structure with additional features, such as pairwise frequencies of words or attributes of the sentence as a whole (e.g. number of direct objects) [12]. This work is very related to ours in that the researchers seek to extend a tree structure to include relationship constraints. While much successful work has been done using UBG's, the authors of this article have yet to encounter a straightforward way for automatically adding constraints to these models, much less adding constraints proportional to their importance. UBG's are mostly seen in the field of natural language processing, where the input data types are much more constrained than in vision and can thus be labeled and tagged more easily by hand. Our method provides a technique for automatically selecting the most representative pairwise features to be added to the model using minimax entropy.

We present an overview of our contextual hierarchy, learning algorithm, and Bayesian inference process in Section 2. This is followed by the formulation of our hierarchical and contextual model in Section 3 and by a description of the learning process in Section 4. We show that sampled aerial images drawn from our model are composed similarly to aerial images that we trained from in Sections 5 and 6. We finally present a Bayesian framework for a three-stage inference algorithm in Section 7 before closing with experiments in Section 8 and conclusions in Section 9.

2 Overview

In this section we give an overview of our 3-layer hierarchy and the learning algorithm for adding context to our representation. We also describe our Bayesian inference algorithm, a 3-step process that first detects objects using different detectors, then uses cluster sampling to remove false positives from our explanation before using top-down prediction to detect any objects our explanation is missing.

2.1 Hierarchical and Contextual Representation

Figure 2 shows our 3-level hierarchy. It consists of nodes divided into a root scene node, group nodes, and object nodes. Group nodes are collections of the same type of object, such as blocks of roofs or lines of cars, while object nodes are the single objects within each group. Below this level is the object representation level, which may be hierarchical in and of itself, as in the case of roofs and roads in our example, or may terminate at a one-layer representation for the object. The top 3-levels are representation agnostic, however, so we will put off a discussion of object detection and representation until Section 7.1. The thick arrow edges between the scene node and group nodes and between the group nodes and the object nodes indicate that a varying number of each group node may be present, and the number of object nodes they are comprised of can vary as well. The hierarchy is similar to a grammar, where the scene node decomposes into a variable number of object groups, which in turn decompose into a variable number of objects. This captures the loose, variable nature of aerial images with just a few compact rules. If we were to write these expansions in a grammar format, we would write

$$\begin{aligned}
&Scene \rightarrow (Roads^*) \cup (Roofs^*) \cup (Trees^*) \cup (ParkingLots^*) \cup (Cars^*) \qquad (1)\\
&Roads \rightarrow Road^*\\
&Roofs \rightarrow Roof^*\\
&Trees \rightarrow Trees^*\\
&Parking\ Lots \rightarrow Parking\ Lot^*\\
&Cars \rightarrow Car^*\ .
\end{aligned}$$

Here we're using "*" in the regular expression sense, meaning 0 or more of an object. One could rewrite the "*" operator by enumerating all cases, as in

$$Roads \rightarrow \emptyset | Road | (Road)(Road) | (Road)(Road)(Road) | ... \qquad (2)$$

On its own the hierarchy simply captures the number of object groups and objects in the scene. We also add statistical constraints between objects to ensure that their appearance and configuration obey certain statistical properties, such as relative scale, relative position, etc. These statistical constraints are represented as dotted horizontal lines in Figure 2 and can be any measurable statistic between some non-empty set of object nodes.

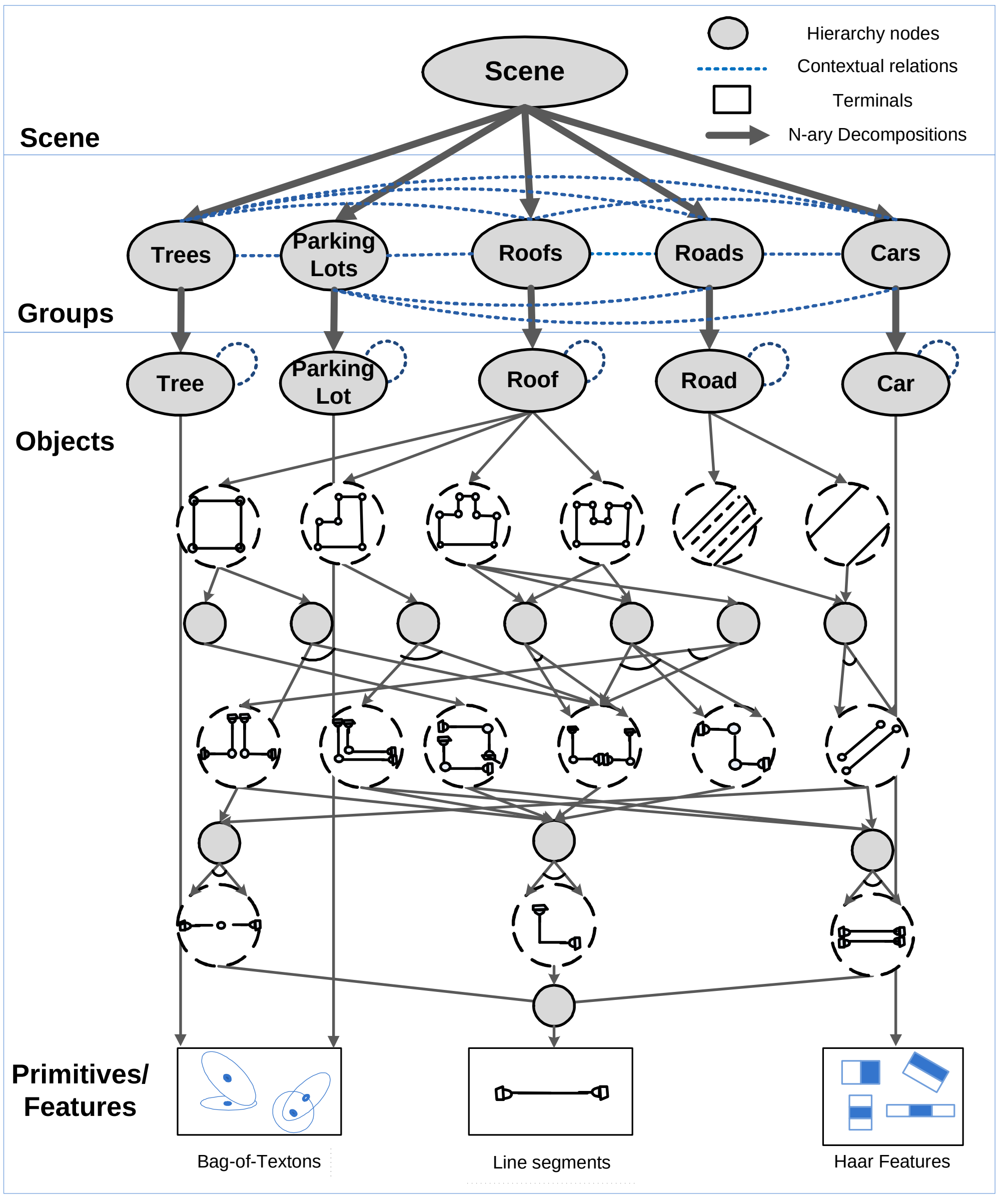


Figure 2: The 3-level contextual hierarchy. The scene is broken down into groups of objects of one of five categories, which are in turn broken down into individual objects of the same type. The thick vertical arrows between the scene and groups and between the groups and objects indicate that these nodes can decompose into a variable number of children. Objects are represented by detectors, some of which (roofs, roads) are hierarchical themselves. The horizontal lines between nodes at the same level represent statistical constraints on the nodes' appearance.

2.2 Minimax Entropy Learning

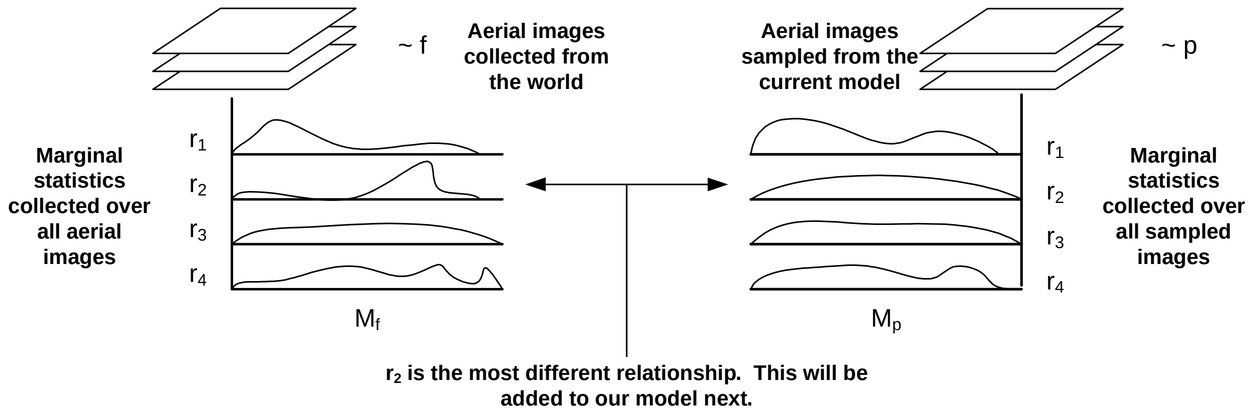

~ f
Aerial images collected from the world
Aerial images sampled from the current model
~ p
Marginal statistics collected over all aerial images
r1
r2
r3
r4
Mf
Marginal statistics collected over all sampled images
Mp
r2 is the most different relationship. This will be added to our model next.

Figure 3: A visualization of the learning process. Feature statistics M_f and M_p are computed over a set of aerial images and a set of aerial images sampled from our current model p, respectively. The most different of these, in this case r_2, is selected to be added to our model. During the next iteration r_2 will now match between M_f and M_p for newly sampled images from p. This process continues until no feature statistics differ significantly between the two sets of images.

We use a minimax entropy learning framework for automatically adding the horizontal constraints to the model during learning. The algorithm first gathers feature statistics M_f across a set of aerial images. For clarity we will use the term "relationship statistic" to mean feature statistic for the remainder of the paper. For example, a relationship statistic could be the distribution of the relative scale between every pair of cars in every image in our training set. The responses for each relationship statistic (e.g. relative scale between cars, relative position between roofs, etc.) are then modeled by a continuous parameterized distribution or just as 1-D histograms. We assume we can draw samples from our current model, which begins as just the hierarchy of object groups with no constraints. These samples will be aerial images themselves. We can gather the same relationship statistics M_p across our sampled images, again using the example of measuring the relative scale between every pair of cars in every image we sampled. We find the statistic r_i that differs the most between M_f and M_p, thus indicating its importance. We add this constraint to our model and then repeat the procedure until no more relationship statistics differ significantly. By the end of this process, the most important relationship statistics over samples from our model will match the most important relationship statistics over true aerial images. Figure 3 visualizes this process, and the results of learning the model are presented in Section 6.

2.3 Top-Down and Bottom-Up Bayesian Inference with the 3-Level Hierarchy

In this section we give an overview of the 3-stage algorithm for parsing new aerial images. In the first stage we detect objects in the image using bottom-up detectors. The next stage then prunes false positives using cluster sampling, followed by the third stage, which predicts missing objects based on our current explanation of the scene.

1. Bottom-Up Detection: In the first phase we collect bottom-up proposals for each object category of interest. The detectors used to find each of these objects can be any off-the-shelf detector and may detect many false positives in the scene. We use Bags of textons and edges, along with a number of boosting methods, to detect objects from multiple categories at multiple scales. Textured objects, like parking lots and trees, are detected at the pixel-level, while structured objects, like roofs and roads, are composed from edges in the image. By using different detectors for each object category, we ensure that we detect each category as well as we can, though we allow for false positives and false negatives, which will be handled in the next stages.

2. Top-Down Pruning of Inconsistent Detections: In the second phase we use the context relations that we learned in our model to prune out nodes that support unlikely interpretations of the scene. We use a cluster-sampling algorithm (C4) [22] for this phase. This algorithm helps us overcome the strong coupling between objects when we sample our different interpretations of the scene. For example, Figure 4 shows a case where we can explain a portion of the image as either a roof or a parking lot with cars in it. If we switch from one explanation to the other by adding or removing single objects to and from our current explanation, it will take an exponentially long time to move from one explanation to the other because the intermediate steps are so unlikely.

Alternatively, adding the roof in Figure 4(b) creates a very unlikely configuration (let's assume we never see parking lots on top of buildings in the training data), so it will be rejected. C4 clustering solves this problem by finding strongly coupled groups, like the parking lot and cars, and swapping them simultaneously with alternative explanations, as shown in Figure 4(c).

3. Top-Down Prediction and Verification of Missing Detections: The third phase of our inference algorithm uses the top-down model to predict any missing objects based on the learned prior model and detections from Stage 2. For example, if we detected 4 cars in a row with a gap in between them, it might be reasonable to predict that another car should be present there. Figure 5 shows an example of predicted roofs, cars, and roads based on our results from stage two and our prior model. The hallucinated objects are shown in green dashed rectangles, while the accepted detections from C4 are shown in black solid rectangles. These top-down predictions will then be pruned or accepted using a final round of C4.

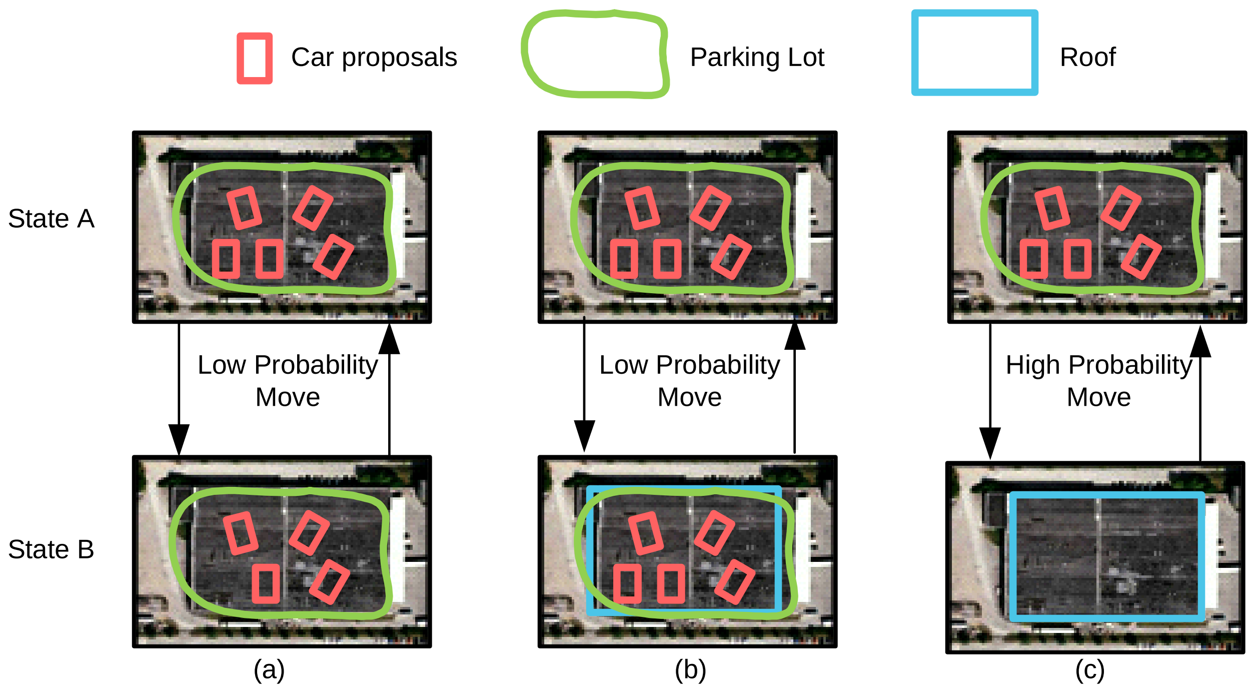


Figure 4: The problem with dealing with strong coupling in aerial images. Here we show an image patch that can be explained either by a roof or by a parking lot with cars (often time the vents on roofs are detected as cars). Traditional sampling methods fail because the intermediate steps to get from explaining the scene as a parking lot to explaining the scene as a roof have very low probability. (a) Removing cars one at a time just leaves an empty parking lot, which is a low probability state. (b) Adding the roof on top of the parking lot is a low probability state (we don't often see parking lots on roofs) (c) Switching the parking lot and cars for the roof in one move is a high probability move.

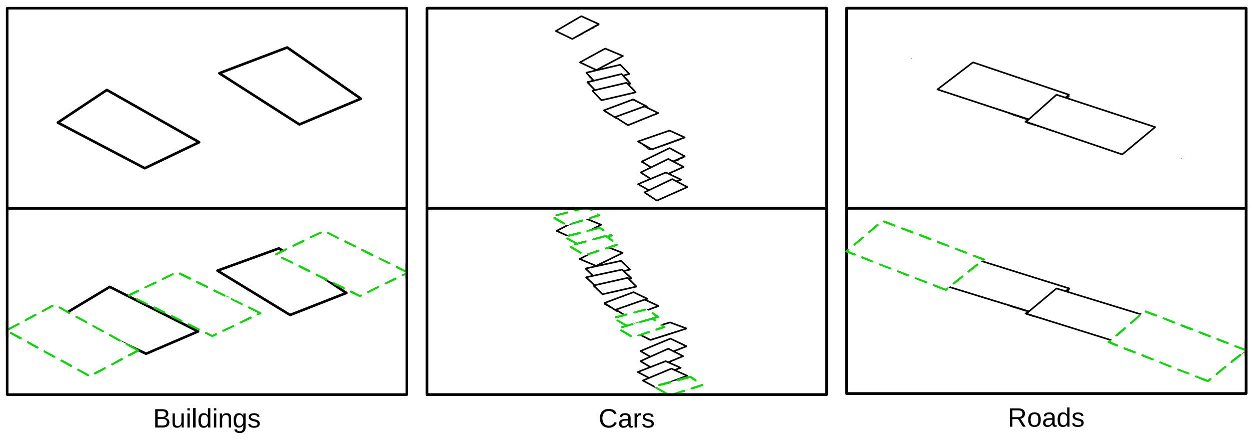


Figure 5: Top-down hallucinations of missing objects. Black rectangles indicate the detections from Stage 2, while the green dashed rectangles indicate hypotheses for missing objects proposed by the top-down part of our model.

3 Formulation

In this section we present the probabilistic formulation for our representation from Section 2.1.

3.1 Contextual Hierarchy Representation

Our representation $\mathcal{G}$ is a 3-tuple

$$\mathcal{G} = < V, R, P > \ , \tag{3}$$

where V are the nodes in the top 3 layers of Figure 2. R is a set of contextual relations and P is our probability model.

The hierarchical component formed from V consists of 3 types of nodes,

$$V = S \cup V^{Group} \cup V^{Object} \ . \tag{4}$$

S: The root Scene node.

V^{Group}: Groups of the same type of object, such as rows of cars or blocks of roofs.

V^{Object}: Individual objects in the image.

The Scene and group nodes may decompose into one of a variable number of children nodes. This makes these nodes similar to "Or" nodes, because node V_i^{Group} can decompose into 1 OR 2 OR 3 OR ... OR k objects. We define a variable $\omega(v)$ on $v \in V$ that takes an integer value for each number of children a node v decomposes into,

$$\omega(v) \in \{0, 1, 2, \dots, n(v)\} \ . \tag{5}$$

Each node $v_i \in V$ has a set of attributes $\phi(v_i)$ that describes its position, scale, and orientation,

$$\phi(v_i) = \{X_i, \sigma_i, \theta_i\} \ . \tag{6}$$

In Section 5 we present our implementation of these attributes for our experiments.

$R = \{r_1, r_2, \dots, r_{N(R)}\}$ is the set of relationships that exist between nodes at the same level of the hierarchy. A relationship r_i consists of a set of k nodes $V_k \in V$ that it acts on, a univariate function $f()$ over their attributes, and a model of the responses of $f()$, p:

$$r_i = \{V_k,\ f_i(\phi(V_k)),\ p_i\} \ . \tag{7}$$

For example, the relative position between cars and buildings could be expressed as

$$f(\phi(Cars), \phi(Buildings)) = X_{Cars} - X_{Buildings} \,. \tag{8}$$

If we believe relative position between cars and buildings is normally distributed with mean 5 and standard deviation 1, then the whole relationship is packaged as

$$r_i = \{(Cars, Buildings),\ f_i = X_{Cars} - X_{Buildings},\ f_i() \sim p_i = \mathcal{N}(5,1)\}. \tag{9}$$

We will discuss the relationship functions $f()$ and their distributions p_i in our implementation in Section 5. At this point it is enough to know that each relationship represents the distribution of a function response over a set of nodes. These distributions act as our statistical constraints. Figure 6 shows some examples of possible relationships.

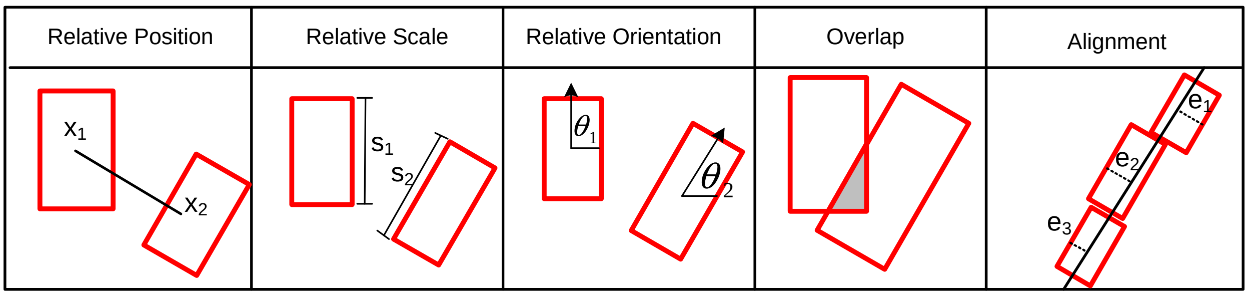


Figure 6: Examples of relationships/statistical constraints. A relationship can technically be any function over the attributes of some non-empty set of nodes.

P is our probability model, including the probability that nodes decompose into a certain number of child nodes, as well as the probability encoded in our statistical constraints. We will define this probability model in Section 3.3.

3.2 Parse Graphs

A *parse graph*, pg, is one instance drawn from the language of aerial images $\mathcal{G}$. This is like a sentence drawn from natural language and corresponds to a single aerial image. An example of a parse graph is shown in Figure 1(c). In a parse graph the production variables $w(v)$ have been decided for every node - we've selected some number of n groups from which to form the scene and selected m_i objects to exist within each group. In addition, every group of objects in pg that were constrained in $\mathcal{G}$ has inherited those appearance constraints. For example, if $\mathcal{G}$ contains a constraint on the relative distance between pairs of cars, every pair of cars in $pg \sim \mathcal{G}$ will have an edge between them constraining their relative distances.

Let us define some terminology on parse graphs, similar to that of $\mathcal{G}$:

1. $V_{pg} \subseteq V$: The nodes present in pg, which are a subset of the nodes possible in $\mathcal{G}$.

2. $\Omega_{pg} = \{\omega(v); v \in V_{pg}\}$: The values of the production rules selected to form pg. For example, if group node v_i consists of 6 cars, $\omega(v_i) = 6$

3. $R_{pg} \subseteq R$: The constraints, or edges, between nodes in V_{pg}. These edges are inherited from the relationships R present in $\mathcal{G}$.

We should note here that parse graphs can be formed either deterministically or probabilistically. During training, we will define some grouping rules to deterministically combine labeled objects into hierarchical groups. Once labeled objects are deterministically grouped, we can measure any relationships of interest across nodes at the same level. During inference, however, the algorithm stochastically determines the most likely groupings of objects into a parse tree based on their relative appearances. The deterministic grouping function used in training is up to the user, but of course any inference methods will try to maximize the probability of a scene interpretation based on the grouping function used in learning. We describe the implementation details we use for this process in Section 5.2.

3.3 Probability Model

We begin with a set of aerial images $I^{obs} = \{I_i^{obs} : i = 1, 2, \ldots, N^{obs}\}$ that have corresponding parse graph representations $PG^{obs} = \{pg_i^{obs} : i = 1, 2, \ldots, N^{obs}\}$. These parse graphs describe the hierarchy of labeled objects in the image and are deterministically constructed from labeled images as in Section 5.2. Each parse graph pg follows some true, unknown distribution $f(pg)$. We would like the statistics of our learned model $p(pg)$ to match the statistics of $f(pg)$ as closely as possible. The statistics of $f(pg)$ consist of

1. The distribution of $\omega(v_{(\alpha)})$, the number of children each node $v_{(\alpha)} \in V$ decomposes into.

2. The distribution of responses of $f_{(\beta)}()$ for each statistical relation $r_{(\beta)} \in R$ in $\mathcal{G}$.

Note that we are switching our indexing subscripts from i's to α's and β's for clarity. α subscripts will be used when we are referring to the distributions of node decompositions, and β subscripts will be used when we are referring to the distributions of relationship constraints.

We will model both node decomposition and relationship distributions as histograms for the remainder of the paper. Specific parametric models may fit the distributions of $\omega(v_{(\alpha)})$ and $f_{(\beta)}()$ more closely, but we use histograms so that we can focus the discussion on learning the model without additional parameters. Also, histograms measure the true continuous distributions of $f(pg)$ and $p(pg)$ in the limit. We approximate the continuous distribution by the piecewise-continuous representation of histograms for each node $v_{(\alpha)} \in V$ and each $f_{(\beta)} \in R$. Our observed statistics for each bin z of the histograms for $\omega(v_{(\alpha)})$ and $f_{(\beta)}()$ are then

$$H_{(\alpha)}(pg, z) = \frac{\sum_{i=1}^{N^{obs}} \#(\omega(v_{(\alpha)}) = z)}{\sum_{i=1}^{N^{obs}} \sum_{j=0}^{n(v_{(\alpha)})} \#(\omega(v_{(\alpha)}) = j)}, \alpha = 1, 2, \ldots, n(V) \tag{10}$$

$$H_{(\beta)}(pg, z) = \sum_{i=1}^{N^{obs}} \frac{\sum_{V_{(\beta)} \subseteq V_{pg(i)}} \#(f_{(\beta)}() = z)}{\sum_{V_{(\beta)} \subseteq V_{pg(i)}} \#(f_{(\beta)}())}, \beta = 1, 2, \ldots, n(R) \tag{11}$$

where $\#$ is a counting function representing the number of times that something occurs, and $\#(f_{(\beta)}())$ is the number of times $f_{(\beta)}$ takes any value. Each bin z in $H_{(\alpha)}(pg)$ is the number of times that node $v_{(\alpha)}$ decomposes into z children divided by the number of times we observe $v_{(\alpha)}$ decomposing any of its $n(v_{(\alpha)})$ values. Each bin z in $H_{(\beta)}(pg)$ is the number of times that relationship function $f_{(\beta)}$ returns z divided by the number of times $f_{(\beta)}$ returns anything.

We seek a distribution $p(pg)$ that matches the relationship statistics ($H_{(\alpha)}(pg)$, $H_{(\beta)}(pg)$) as closely as possible with $f(pg)$, while remaining as random as possible (unprejudiced) along all other dimensions. This is equivalent to making sure that the expectation of the number of objects in each group, $E_p[\omega(v_{(\alpha)})]$, and the expectation of each relationship function, $E_p[f_{(\beta)}(\phi(V_{(\beta)}))]$, matches between our model and the true distribution. By maximum entropy this becomes the following constraint satisfaction problem,

$$p(pg)^* = \arg\max\{-\sum p(pg) \log p(pg)\} \tag{12}$$

subject to

$$E_p[\omega(v_{(\alpha)})] = E_f[\omega(v_{(\alpha)})],\ \alpha = 1, 2, \ldots, n(V) \tag{13}$$

$$E_p[f_{(\beta)}(\phi(V_{(\beta)}))] = E_f[f_{(\beta)}(\phi(V_{(\beta)}))],\ \beta = 1, 2, \ldots, n(R) \tag{14}$$

$$E_f[\omega(v_{(\alpha)}] \approx H_{(\alpha)}(PG^{obs}) \tag{15}$$

$$E_f[f_{(\beta)}(\phi(V_{(\beta)}))] \approx H_{(\beta)}(PG^{obs}) \tag{16}$$

In other words, we want the histograms formed from sampled aerial images from our model to match the true distributions observed in the training data. The probability model that satisfies these constraints is the familiar Gibbs model

$$p(pg; \Theta, R) = \frac{1}{Z[\Theta]} \exp^{-\xi(pg)} \tag{17}$$

$$\xi(pg) = \sum_{\alpha=1}^{n(V)} < \lambda_{(\alpha)}, H_{(\alpha)}(pg) > + \sum_{\beta=1}^{n(R)} < \lambda_{(\beta)}, H_{(\beta)}(pg) > \tag{18}$$

$$Z[\Theta] = \sum_{pg \in L(\mathcal{G})} \exp^{-\xi(pg)} \ , \tag{19}$$

where $\Theta = \{\lambda_{(\alpha)}, \lambda_{(\beta)}\}$. The first term in $\xi(pg)$ is the energy of the decomposition rules and the second is the energy of the relationship constraints. If we have an unlikely number of objects in an image (say 0 objects), then the first term will have high energy and the interpretation will have low probability. If we have objects that do not obey the statistical constraints we learned during training, for example we observe a car on top of a tree, then the second term will have high energy and the interpretation will have low probability.

The Lagrange multipliers $\{\lambda_{(\alpha)}, \lambda_{(\beta)}\}$ are vectors of the same dimension as $H_{(\alpha)}$ or $H_{(\beta)}$, respectively, and $< \ldots >$ indicates an inner product. For example, if relationship $r_{(\beta)}$'s function $f_{(\beta)}$ evaluates to z on parse graph pg, then the energy from that relationship is $\lambda^z_{(\beta)} * H_{(\beta)}(pg, z)$. The λ's are the natural parameter set of the model and serve to weight histogram bins so that dependent relationship interactions are weighted correctly. These λ's will be learned in the following section.

4 Learning The Hierarchical Contextual Model

We begin with a set of N^{obs} aerial images $I^{obs} = \{I^{obs}_i : i = 1, 2, \ldots, N^{obs}\}$ and their corresponding parse graphs $PG^{obs} = \{pg^{obs}_i : i = 1, 2, \ldots, N^{obs}\}$. The parse graphs PG^{obs} follow the real-world, unknown target distribution, $f(pg)$, by definition,

$$pg^{obs} \sim f(pg) \ . \tag{20}$$

Matching our distribution $p(pg)$ to $f(pg)$ is equivalent to finding the values for Θ that minimize the KL divergence between the two distributions

$$\Theta^* = \operatorname*{argmin}_{\Theta} KL(f(pg) || p(pg; \Theta, R)) \tag{21}$$

$$= \operatorname*{argmin}_{\Theta} \sum_{pg} f(pg) \log \frac{f(pg)}{p(pg; \Theta, R)} \ , \tag{22}$$

which is equivalent to finding the maximum likelihood estimates for Θ and a set of relationships R constraining the model. Letting $\mathcal{L}(\Theta)$ be the log-likelihood function for our parameters,

$$\mathcal{L}(\Theta) = \sum_{pg \in PG^{obs}} \log p(pg; \Theta, R) \tag{23}$$

$$(\Theta, R)^* = \underset{(\Theta,R)}{\arg\max}\, \mathcal{L}(\Theta)\ . \tag{24}$$

Learning our parameters can then be broken down into two distinct stages:

1. Given a set of relationships R in the model, estimate $(\lambda_{(\alpha)}, \lambda_{(\beta)})$.

2. Pursue a set of relationships R one-by-one to constrain the model.

This may seem backwards, but it is easier to understand the process if we first show the process for parameter estimation given a set of relationships R followed by the process for pursuing R.

4.1 Learning $(\lambda_{(\alpha)}, \lambda_{(\beta)})$

We solve for $\Theta = (\lambda_{(\alpha)}, \lambda_{(\beta)})$ using straightforward maximum likelihood estimation (MLE). Setting $\frac{\partial \mathcal{L}}{\partial \Theta} = 0$, we can solve for both sets of λ's:

1. $\lambda_{(\alpha)}$: We make the assumption that node decompositions are independent of each other and of their appearance constraints. We can therefore model $H_{(\alpha)}(pg)$ as a multinomial and count the frequency with which each node decomposes into a number of children, as in Equation 10. This makes our estimate for each $\lambda_{(\alpha)}$

$$\lambda_{(\alpha)} = -\log H_{(\alpha)}(PG^{obs})\ . \tag{25}$$

This is the MLE estimate for a multinomial and can be used to estimate production rule probabilities in grammars given that they are independent of any cross-link relations, i.e. context-free [4].

2. $\lambda_{(\beta)}$: Setting $\frac{\partial \mathcal{L}}{\partial \Theta} = 0$ for $\lambda_{(\beta)}$ yields

$$E_p[f_{(\beta)}(\phi(V_{(\beta)}))] = H_{(\beta)}(PG^{obs}) \tag{26}$$

which are exactly the Lagrange constraints that resulted from deriving our maximum entropy model. This equation can also be written as

$$\frac{\partial \mathcal{L}}{\partial \lambda_{(\beta)}} = \frac{1}{N^{obs}} \sum_{pg \in PG^{obs}} H_{(\beta)}(pg) - E_p[H_{(\beta)}(PG^{obs})] \tag{27}$$

which can be approximated by

$$\frac{\partial \mathcal{L}}{\partial \lambda_{(\beta)}} \approx H_{(\beta)}(PG^{syn}) - H_{(\beta)}(PG^{obs}) \; . \tag{28}$$

$H_{(\beta)}(PG^{syn})$ is the histogram formed from a set of parse graphs $PG^{syn} = \{pg_i^{syn} : i = 1, 2, \ldots, N^{syn}\}$ that are synthesized from our current model $p(pg)$. The synthesized parse graphs are drawn by first sampling the number of children each node decomposes into according to the learned $\lambda_{(\alpha)}$ parameters. The appearances of the objects in the resulting parse tree are then Gibbs sampled according to the current $\lambda_{(\beta)}$ weights and the constraints in the model. These images will also be aerial images, so we can compute histograms for the same relationship function $f_{(\beta)}()$ over these parse graphs as we did over PG^{obs}.

Solving for the $\lambda_{(\beta)}$'s such that $H_{(\beta)}(PG^{syn}) = H_{(\beta)}(PG^{obs})$ can then be done using gradient descent. We initialize the $\lambda_{(\beta)}$ weights to 0

$$\lambda_{(\beta)} = 0, \; \beta = 1, 2, \ldots, n(R). \tag{29}$$

In the first stage when $\lambda_{(\beta)} = 0$, $H_{(\beta)}(PG^{syn})$ will be close to uniform. Gradient descent is then used to update the $\lambda_{(\beta)}$'s,

$$\lambda_{(\beta)}^{t+1} = \lambda_{(\beta)}^{t} - \eta(H_{(\beta)}(PG^{syn}) - H_{(\beta)}(PG^{obs})) \; , \tag{30}$$

where η is a step factor that can depend on the iteration t. This update reweights the $\lambda_{(\beta)}$'s for each histogram based on how much $H_{(\beta)}(PG^{syn})$ differs from $H_{(\beta)}(PG^{obs})$. It reduces the energy for choosing underrepresented bins and increases the energy for choosing overrepresented bins during the next iteration of Gibbs sampling. After a number of iterations the $\lambda_{(\beta)}$'s will be weighted such that the synthesized distributions match the observed distributions, and thus $p(pg)$ will match $f(pg)$ along these dimensions. In other words,

$$|H_{(\beta)}(PG^{syn}) - H_{(\beta)}(PG^{obs})| < \varepsilon_2, \; \beta = 1, 2, \ldots, n(R). \tag{31}$$

Figure 7 shows a toy example of the $\lambda_{(\beta)}$ learning process. We begin with one observed histogram H^{obs} for relative car size and its corresponding λ weight vector, which begins as a vector of all 0's. Because this weight is uniform, the images we sample in Step (2) look fairly random. In Step (3) we compute H^{syn}, the distribution of relative car sizes across these sampled images. This, unsurprisingly, is fairly uniform as well because λ was uniform. The figure shows these H^{syn} and H^{obs} superimposed below to emphasize the difference in their bin counts. In Step (4) we update the λ weights according to how much each bin differs. Step (5) shows the sampled images resulting from the updated λ weighting, which have much more appropriate relative car sizes. This process is carried out simultaneously for each $\lambda_{(\beta)}$.

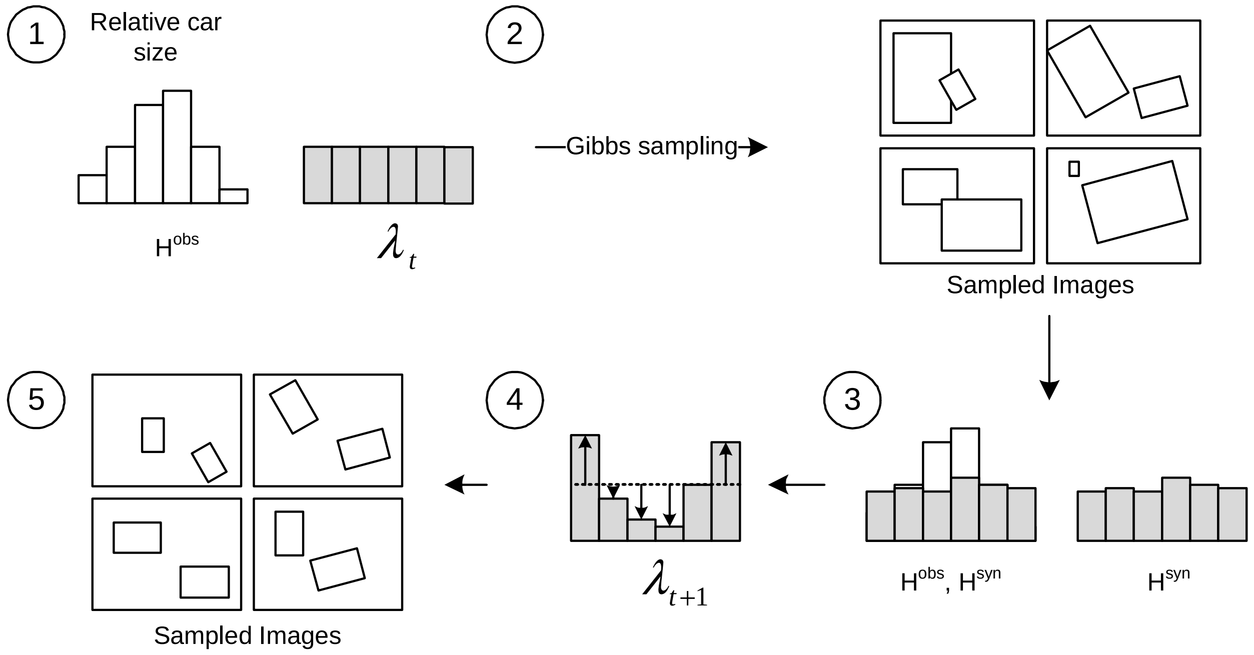


Figure 7: Examples of learning the relationship parameters, $\lambda_{(\beta)}$. (1) We begin with an observed histogram H^{obs}, in this case the relative size between cars. λ begins uniform. (2) Sampled images PG^{syn} are drawn from the model. (3) H^{syn} is computed for relative car size over the sampled images. (4) λ is reweighted according to the difference between H^{obs} and H^{syn}. (5) Newly sampled images appear scaled correctly.

4.2 Relationship Pursuit

The $\lambda_{(\beta)}$'s above were learned given that we already knew which relationships R existed in the model. We now show how to select the relationship constraints for the model. Because our dictionary of potential relationships Δ_R could be combinatorially huge, we will iteratively add relationships according to their importance instead of fitting the full model. Fitting the full model would require later attempts to sample from the model or perform inference with the model to check redundant relationships, making the model overly complex and slower to compute with.

We pursue the relationship set R by beginning with just an empty hierarchy,

$$p_0(pg; \Theta_0, R_0);\ R_0 = \{\emptyset\}\ . \tag{32}$$

This is the model with no parameters learned at all. We then learn the $\lambda_{(\alpha)}$'s, or tree parameters, allowing us to sample images with the correct distributions of objects, but without spatial or appearance constraints. Sampling the model at this stage would produce parse graphs with the correct number of objects, but without horizontal constraints, causing the resulting image to look more like an "alphabet soup" of objects that are big, small, overlapping, etc. We then iteratively add a new relationship r_+ from a dictionary of potential relationships Δ_R at each iteration to get a new distribution $p_+(pg; \Theta_+, R_+),\ R_+ = R \cup \{r_+\}$. We choose r_+ such that we minimize $KL(f(pg)||p_+(pg; \Theta_+, R_+))$ at each step:

$$p_0(pg; \Theta_0, R_0) \to p_1(pg; \Theta_1, R_1) \to \ldots \to p_k(pg; \Theta_k, R_k)\ . \tag{33}$$

At each iteration we want to select the relation r_+ that brings our new model p_+ closest to f, thereby reducing the KL divergence between the two distributions. This is equivalent to finding the new p_+ that is maximally KL divergent from our current p, as visualized in Figure 3. Because we are guaranteed to monotonically decrease $KL(f(pg)||p_+(pg; \Theta, R_+))$ with every added relation [42], the r_+ that is maximally far away from p must be maximally close to f and thus represents the largest decrease in KL divergence:

$$\begin{aligned} r_+ &= \underset{r}{\operatorname{argmax}}\, KL(f(pg)||p(pg; \Theta, R)) - KL(f(pg)||p_+(pg; \Theta_+, R_+)) & (34)\\ &= \underset{r}{\operatorname{argmax}}\, KL(p_+(pg; \Theta_+, R_+)||p(pg; \Theta, R))\ . & (35) \end{aligned}$$

We can approximate this decrease in KL divergence, otherwise knows as the information gain, $\delta(r_+)$, using the Mahalanobis distance between the synthesized and observed histograms for the new potential relation r_+

$$\begin{aligned} \delta(r_+) &= KL(p_+(pg; \Theta_+, R_+)||p(pg; \Theta, R)) & (36)\\ &\approx d_{mahn}(H^{r+}_{(\beta)}(PG^{obs}), H^{r+}_{(\beta)}(PG^{syn}))\ . \end{aligned}$$

This holds due to a Taylor expansion around the relationship we're interested in adding, as shown in the Appendix of [42]. We measure $H_{(\beta)}(PG^{obs})$ and $H_{(\beta)}(PG^{syn})$ for all relations $r_{(\beta)} \in \Delta_R$ and compare their Mahalanobis distances. The $r_{(\beta)}$ with the largest Mahalanobis between the synthesized and observed histograms above some threshold is added to the model in the next iteration and its $\lambda_{(\beta)}$ parameters are learned as in the previous section.

4.3 Summary of Parameter Learning and Relationship Pursuit Algorithms

The algorithm for learning the parameters of the model proceeds in two steps. We first learn the $\lambda_{(\alpha)}$'s by MLE, which are just the sample frequencies of the decompositions of each node. We then iteratively add spatial and appearance relations one-by-one until no relation remaining in Δ_R has Mahalanobis distance greater than ε_1. After each relation is added, we iteratively update the $\lambda_{(\beta)}$'s for the current relation set to match $H_{(\beta)}(PG^{syn})$ to $H_{(\beta)}(PG^{obs})$, $\forall r_{(\beta)} \in R$. The algorithms are outlined below:

Algorithm 1. Relationship pursuit.

```
1. Begin with an empty model $p_0$ and observed parse graphs $PG^{obs} = \{pg_i^{obs} : i = 1, 2, \dots, N^{obs}\}$.
2. Compute observed histograms $H_{(\beta)}(PG^{obs})$ and $H_{(\alpha)}(PG^{obs})$ for all relationships in $\Delta_R$ and all
   node frequencies, respectively.
3. Approximate the $\lambda_{(\alpha)}$'s for the tree component using MLE, yielding $p_t$.
4. Repeat
   (a) Sample $N^{syn}$ parse graphs from the current model, $PG^{syn} = \{pg_i^{syn} : i = 1, 2, \dots, N^{syn}\}$.
   (b) Calculate $H_{(\beta)}(PG^{syn})$, $\beta = 1, 2, \dots, |\Delta_R|$.
   (c) Select the $r_{(\beta)}$ for which $d_{manh}(H_{(\beta)}(PG^{syn}), H_{(\beta)}(PG^{obs}))$ is maximal as $r_+$. Add $r_+$ to
       $R$.
   (d) Relearn the $\lambda_{(\beta)}$'s for the new set $R \cup \{r_+\}$ using Algorithm 2.
   until $d_{manh}(H_{(\beta)}(PG^{syn}), H_{(\beta)}(PG^{obs})) < \varepsilon_1, \beta = 1, 2, \dots, |\Delta_R|$.
```

```
Algorithm 2. Parameter estimation algorithm.

1. Given a set of relations $R$ and current model $p(pg; \Theta, R)$,
2. Repeat
    (a) Sample $n$ parse trees from the model, $PG^{syn} = \{pg_i^{syn} : i = 1, 2, \ldots, N^{syn}\}$.
    (b) Calculate $H_{(\beta)}(PG^{syn})$, $\beta = 1, 2, \ldots, n(R)$.
    (c) Update $\lambda_{(\beta)}^{t+1} = \lambda_{(\beta)}^{t} - (\eta(H_{(\beta)}(PG^{syn}) - H_{(\beta)}(PG^{obs})))$.
   until $|H_{(\beta)}(PG^{syn}) - H_{(\beta)}(PG^{obs})| < \varepsilon_2, \beta = 1, 2, \ldots, n(R)$.
```

5 Implementation

The learning algorithm above is independent of our choice of models for the relationship statistics and the object representations. In this section we present implementation details used to learn the model above in our experiments.

5.1 Object Representation

We represent our five object categories, (roads, trees, roofs, cars, parking lots), by their enclosing boundaries. Each object of interest is described by a boundary b that is defined as a graph,

$$b_i =< \nu_i, \zeta_i >, \tag{37}$$

where ν is a set of boundary points and ζ is a set of edges, along with a label l_i indicating what type of object it is. The boundaries and labels are hand-labeled in every observed image I^{obs}. These objects form the bottom layer of the hierarchy. We create a node v_i in the 3-layer hierarchy for each boundary b_i so that every boundary is represented by a bottom-level node in the hierarchy and every bottom-level node in the hierarchy has a corresponding boundary representation.

From these boundaries we can derive the appearance attributes $\phi(v_i) = \phi(b_i) = \{X_i, \theta_i, \sigma_i\}$ of each boundary b_i for each bottom-level node v_i in the hierarchy. The smallest enclosing bounding box, box_i, was computed for each b_i. The position of each object is its center of mass, its orientation is the major axis of box_i, and its scale is the length and width of box_i, treating the major axis as our measure of length and the minor axis our measure of width. If boundary b_i for node v_i consists of n vertices $\nu = \{\nu_1, \nu_2, \ldots, \nu_n\}$ and ($left_i$, $right_i$, top_i, $bottom_i$) describe the center points of the edges of b_i's bounding box box_i, we can compute each object's appearance attributes as,

$$X_i = (\frac{\sum_{j=1}^{n} x(\nu_j)}{n}, \frac{\sum_{j=1}^{n} y(\nu_j)}{n}) \tag{38}$$

$$M_{axis} = max((right_i{-}left_i), (top_i - bottom_i)) \tag{39}$$

$$m_{axis} = min((right_i{-}left_i), (top_i - bottom_i)) \tag{40}$$

$$\theta = cos^{-1}(M_{axis}, (1, 0)) \tag{41}$$

$$\sigma = (|M_{axis}|, |m_{axis}|)\ . \tag{42}$$

Figure 8 shows an example of the representation of a building based on these features.

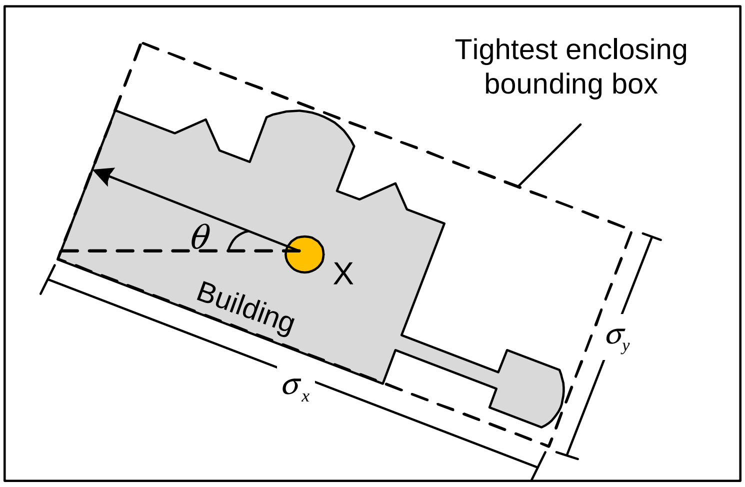


Figure 8: An example of the features computed for a single object. The boundary graph and smallest enclosing bounding box are used to compute position, scale, and orientation.

5.2 Deterministically Forming Parse Graphs

Each I^{obs} has a corresponding parse graph pg^{obs} describing the hierarchical arrangement of its objects. We are given the boundaries of every object as described in Section 5.1. We touched briefly on the deterministic formation of parse graphs in Section 3.2, but now go into more details of our implementation for deterministically converting labeled objects into parse graphs.

To form parse graphs from a collection of labeled objects (the leaf nodes) we make the following stipulation:

Proposition 1 Boundaries within distance ϱ of each other that are of the same object label will be considered members of the same group.

In other words, objects are deterministically assigned to groups according to their distance between one another. This provides two benefits:

Table 1: Category-dependent distance thresholds ϱ for deterministic grouping based on an object's aspect ratio s.

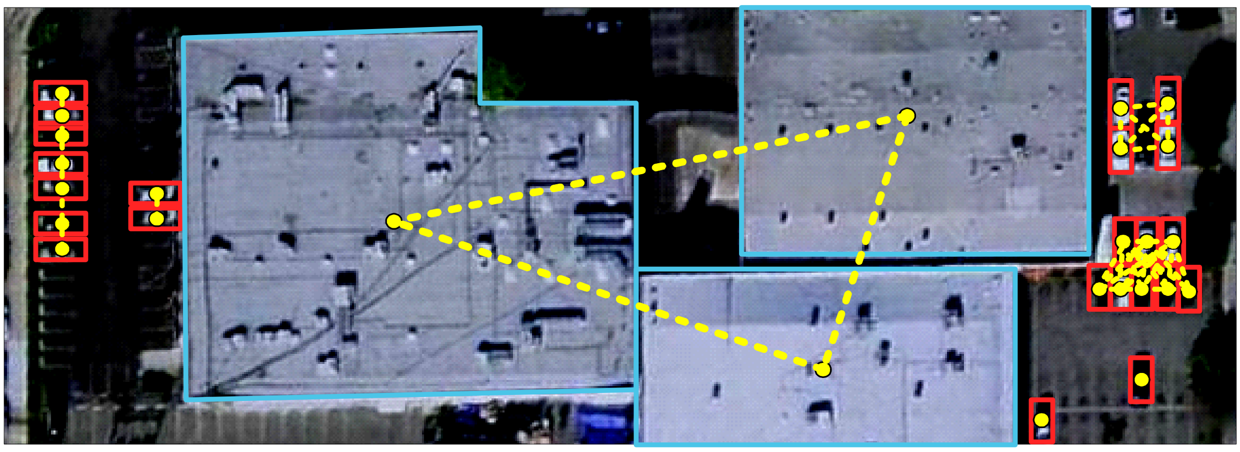
Object Label	ϱ
Roof(s)	s
Car(s)	0.5*s
Tree(s)	1.2*s
Road(s)	0.5*s
Parking Lot(s)	s

1) We can deterministically form a hierarchy from a flat set of objects.
2) We only measure relationship statistics within and between groups of objects, so limiting the distance at which two objects are related prevents us from calculating and learning statistics of objects that are very far away.

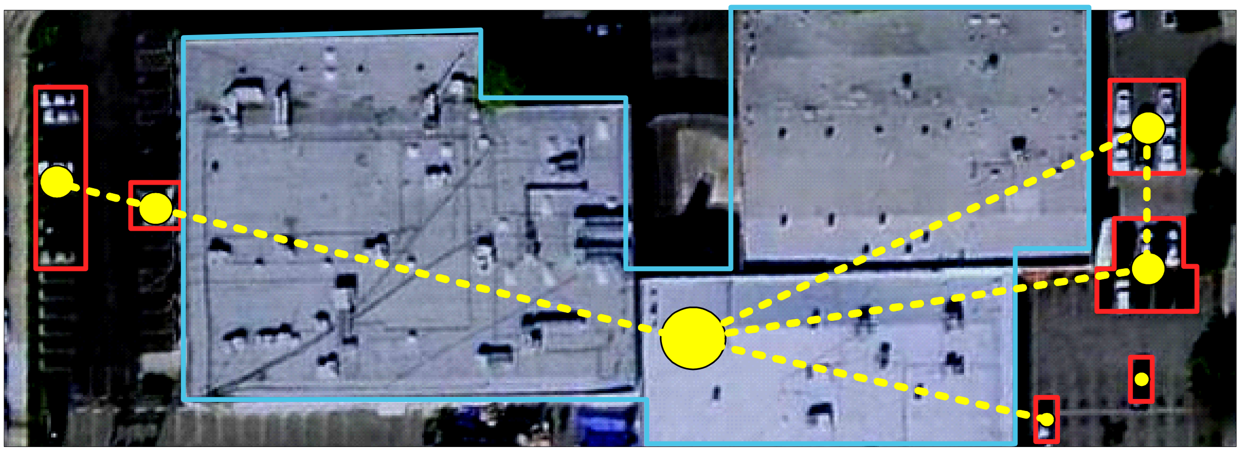
In our experiments we set ϱ to be label-dependent. If we let s be b_i's aspect ratio, we can define a set of distance thresholds as in Table 1. For example, a tree would need to be within 1.2 of its aspect ratio of another tree to be considered in the same group. Because some objects within the same label may vary significantly in size, one may also choose to consider two objects proximal only if they are within $c * min(s_1, s_2)$ of each other, where (s_1, s_2) are the aspect ratios of the two objects in question. This is particularly useful when determining which groups of objects should be associated, as their sizes can vary significantly more than those of single objects.

Figure 9 shows an example of deterministically forming a parse graph from a set of labeled objects. In this example, cars that are nearby one another are grouped together, as shown in (a). The same goes for roofs labeled in (a). Figure 9(b) shows the resulting groups from this first step and their distance-based relationships as well.

An important point to note here is that, though we form parse graphs deterministically for our observed images, we do **not** form them deterministically when inferring the best explanation of a new image. In our training images I^{obs} we are making the assumption that proximal objects are grouped and that this grouping defines the number of objects that the group consists of (decomposes into). There may, however, be proximal groups in our testing images that have, for example, a different number of objects than we expect to see based on our training data. In this case, it may make more sense to split the group into subgroups that match our learned decomposition frequencies than grouping them all under a hard-coded proximity condition. In a world where we've only seen sets of three cars, a row of six cars is more consistently explained as two sets



(a) Network structure between single objects



(b) Network structure between grouped objects

Object/Group boundary
Object/Group center
Neighborhood Edge

Figure 9: An example of deterministically forming the neighborhood structure for a parse graph from labeled objects. (a) Cars and roofs that are within a certain distance of each other are grouped together. (b) Groups that are within a certain distance of each other have group-level constraints applied.

Table 2: Relationship function definitions.

Relationship	n Nodes	Function $f_i()$
Aspect ratio	1	σ_y/σ_x
Relative position (in image coordinate frame)	2	$(X_2 - X_1)/s_1$
Radial position (in object coordinate frame)	2	$\{ \|X_2 - X_1\|, \theta(\|X_2 - X_1\|)$ - θ_1 $\}$
Relative scale	2	s_1/s_2
Relative orientation	2	$\theta_2 - \theta_1$
Percentage overlap	2	$Area_{overlap}/Area_1$
Alignment	n	SSE of least squares fit

of three by our model.

5.3 Relationship Functions

The functions f_i for each relationship are defined over the attributes of sets of nodes, $\phi(V_i)$. We implemented the relationship functions listed in Table 2. The relationship functions should be fairly explanatory, with the exception of the position functions. Relative position returns the vector between the centers of the two objects, which is relative to the coordinate frame of the image. This is not particularly useful as aerial images rarely have a well-defined "top" or "bottom". Radial position attempts to deal with this by measuring relative distance in polar coordinates. These two-dimensional features can also be decoupled and collected as if they were independent, e.g. a relationship of just relative X or Y positions or of distance and relative angle. The responses of these functions form the histograms $H_{(\beta)}$ that we match in the learning stage.

5.4 Histograms

We have chosen to represent our relationship statistic distributions as histograms. This is intended to save us the trouble of fitting specific distributions to each new relationship. In order to use histograms, we made the following design decisions:

1. The range of the histograms are determined by the maximum and minimum values observed in the training data. We model values outside of this range by a decreasing gradient function. Define the maximum probability value allowed for this gradient function as p and the width of the histogram bin as w. Then a point that is distance k bin widths beyond the edge of the histogram (i.e. $k * w$

distance from the edge) can be assigned probability $p - (k/n) * p$, where n is as many extra bins as we'd like to add to either side of our histogram. Thus, values that are beyond the histogram edge are assigned a probability that is a fraction of the maximum probability p, depending on how far away they are. n is usually set to be something large, e.g. 10000, and p is usually set to be the edge bin probability, which prevents the tail from being greater probability than the probability in the edgemost bins. Values greater than n are assigned probability 0 (or some minimum probability). These tails allow us to model values outside of the histogram range while guaranteeing that their probability is never greater than the probability for the edge closest to that side.

2. We divide our histograms into 10 bins each. We were surprised to find that, empirically, any number of bins above 6-7 were sufficient to produce samples from the model that are perceptually similar to real aerial images. Obviously we will never perfectly match the distributions with this discretization until we approach the limit, but we find suitable results with even as few as 10 bins.

3. Our training set is intentionally small, so that little human intervention is needed. At first it may appear that we are therefore reducing the size of our training set and will not have sufficient data to model the distributions. However, as most of our images are large images containing hundreds of the objects, one image often provides a large number of data points.

6 Experiments on Learning and Sampling

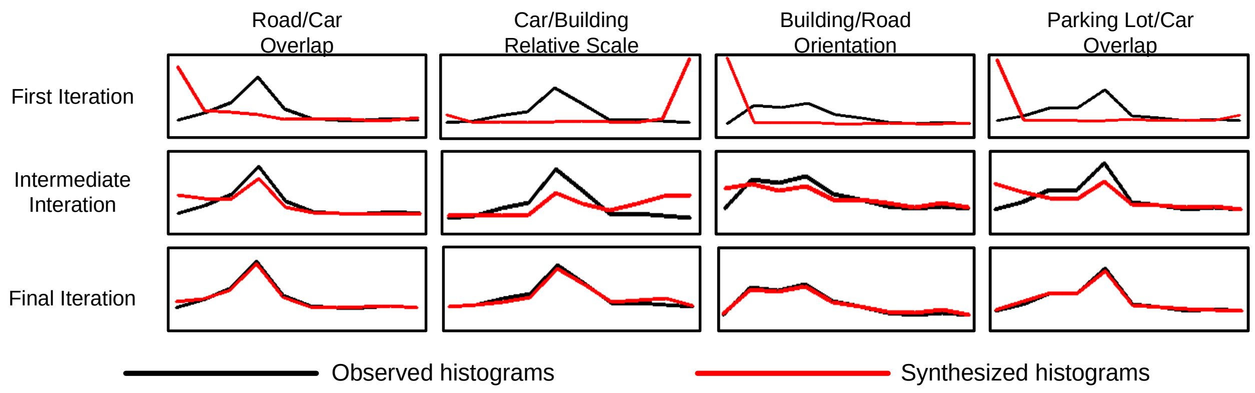


Figure 10: Histograms for four typical relations over the course of the learning algorithm. The black lines are the histograms of the observed data, $H_{(\beta)}(PG^{obs})$, and the red lines are the histograms of the synthesized data, $H_{(\beta)}(PG^{syn})$, at each iteration. At first the statistics of the synthesized data are so far off from the truth that most values are out of bounds. Halfway through the learning process the histograms look close to matching and by the final iteration the histograms match nearly perfectly.

We selected 120 aerial images from the Lotus Hill Database [38], which included labeled boundaries of roofs, roads, parking lots, tree regions, and cars to use as our training data [1]. As mentioned in Section 1, these boundaries are hand-labeled. The images ranged in size from 640x480 pixels to 1000x1000 pixels. We set $\varepsilon_1 = 4$ and $\varepsilon_2 = 0.2$ for Algorithms 1 and 2 and then learned a hierarchical contextual model of objects in aerial images.

Figure 10 shows $H_{(\beta)}(PG^{obs})$ and $H_{(\beta)}(PG^{syn})$ for four typical relations at three iterations of the parameter learning algorithm. In the first iteration, the histograms from our synthesized images are so far away from the true histograms that most of their data is out of bounds. Halfway through the learning, however, the histograms start to look coarsely similar. By the final iteration, the histograms have matched nearly perfectly. This assures us that the $\lambda_{(\beta)}$'s are reweighting the histogram bins correctly such that, over time, the images we synthesize using our model match the statistics of true aerial images.

We used 5 object categories in our model (car, roof, road, parking lot, tree) and their 5 corresponding group categories. We used 7 relationship functions in our model, resulting in a relationship dictionary Δ_R consisted of 360 possible relationships (10 aspect ratio relations + (5 objects)*(5 objects)*(7 relationships) + (5 groups)*(5 groups)*(7 relationships)). Of those 360 possible relationships our model selected 27, consisting mostly of overlap relations (car/car overlap, building/tree overlap, car/parking lot overlap), relative scale relations (car/car relative scale, roof/road relative scale), and alignment relations (car/car alignment). There were also a few orientation relations added, though they were only slightly better than noise (roof/road orientation) and could probably be weeded out by adjusting ϵ_1.

Figure 11 shows samples from our final model $p(pg; \Theta, R)$. The resulting images appear similar to true aerial images, with objects obeying many of the same spatial and appearance constraints that we observe in the real data. We see cars appearing on roads, roofs arranged in blocks, and few or no spurious overlaps. Note that these samples are not representative of a specific aerial image from the training data or elsewhere. These are simply object boundaries that have been scaled, positioned, and oriented such that they minimize the energy in our prior. Nevertheless, we see that the relationship histograms match between the two models and the sampled images are perceptually similar to true aerial images. This shows that our learned model is in fact capturing the relationship statistics present in true aerial images and can thus recreate believable aerial image configurations.

[1]Dataset available from http://www.imageparsing.com. More data will be released after the publication of this paper, but sample data is available free for downloading now.

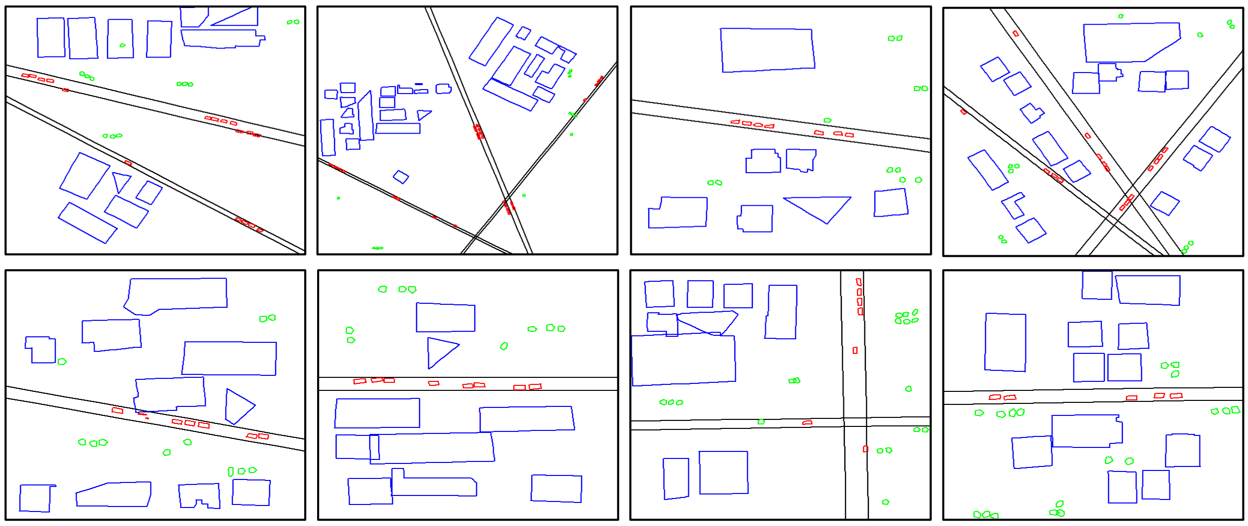


Figure 11: Samples from our learned model (blue = roofs, red = cars, black = roads, green = trees). These are not images directly sampled from the training data, but collections of objects obeying the statistics of our learned model. We can create a vast amount of unique object configurations even though we've never observed them directly.

7 Bottom-Up/Top-Down Bayesian Inference

In this section we present the Bayesian formulation for finding the highest probability explanation of an aerial image. We first run bottom-up detectors for each of the object categories. After this first stage of object detection, we use a cluster sampling algorithm inspired by Swendsen-Wang sampling called C4 to prune out false positives and inconsistent detections. In the third stage of the algorithm we propose locations of objects that may have been missed by the bottom-up detectors or incorrectly pruned in the first stage.

7.1 Bottom-Up Detections

We use different bottom-up detectors for each type of object:

Cars: We trained a discriminative AdaBoost classifier [7] to detect cars. We collected 3000 positive examples of cars, selected by hand as patches containing a single car from aerial images, as well as 3000 negative images for training, comprised of patches of training images in which no car is present. Figure 12(a) shows car detections using the learned classifier. Unfortunately, we do find that this method results in many false positives, which we will address later. AdaBoost is a commonly cited and described algorithm, so we refer the reader to [7] for further details.

Parking lots and Trees: Parking lots and trees are characterized by their textures. Color information is highly variable from one parking lot or grove of trees to the next, so color histograms are too simple to capture an appearance model for these classes. We resolve this problem by using TextonBoost [23], an

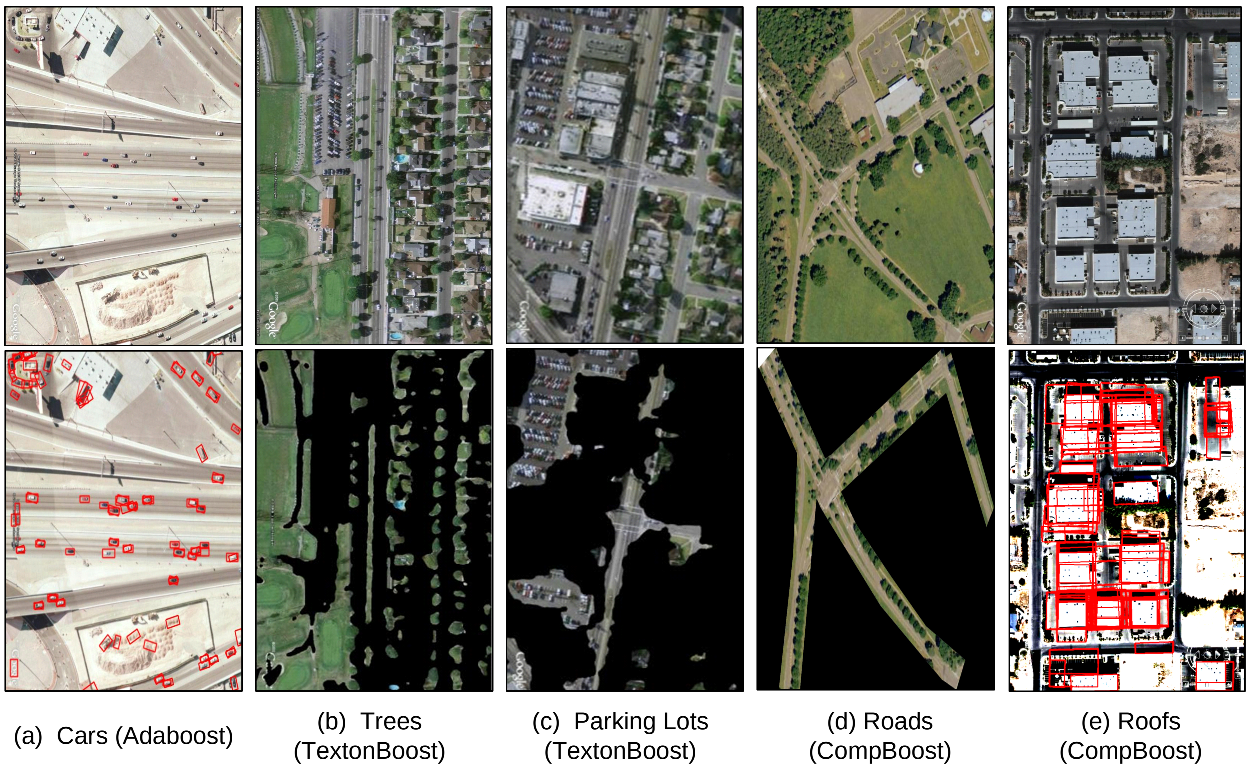


Figure 12: Single object detections using our bottom-up detectors.

algorithm for combining texture and shape cues in a boosting framework to create a discriminative classifier. TextonBoost extracts textons (collections of filter responses) for each category and clusters them into a texton dictionary. These textons are then boosted using to arrive at a combined discriminative classifier. We provided TextonBoost with about 100 images in which the images are labeled (0/1) according to whether or not a pixel belongs to background or the category we're learning (parking lots and trees are learned separately). The pure bottom-up results are shown in Figure 12(b) and Figure 12(c).

Roofs and Roads: Roofs and roads present quite a different problem from the categories we've represented up until now. They are neither defined by a constant shape nor a constant texture. The most informative cues are the edges that define their boundaries. We use a recently developed algorithm called Compositional Boosting [37] that hierarchically combines low-level cues into higher-level structures. A more detailed explanation of Compositional Boosting is given in [37], but we will describe it at a high level here.

High-level Description of Compositional Boosting

Compositional Boosting learns a model by first defining a dictionary of low-level features (such as edges) along with some spatial rules of interest (e.g. parallelism, relative length, collinearity). These low-level features are first labeled in a number of training images and labeled as belonging to the structure of interest or not. For example, in this experiment, we labeled edges in the training data as belonging to a roof, a road, or neither. Compositional Boosting begins building a hierarchy from these labeled edges by testing the mutual information of edges under certain spatial constraints. For example, in the roof class we will see lines at right angles more frequently than in random noise. Any composition rules with mutual information greater than some threshold are added to the hierarchy (e.g. two lines nearly 90 degrees from one another should form a higher-level component). This process then repeats at the next highest level until some percentage of the labeled lines are modeled by the final composition. Figure 2 shows the Compositional Boosting hierarchies below the roof and road nodes. A roof can decompose into a number of different shapes, each of which is formed from lower-level components.

To detect structures in images, we first define detectors for Compositional Boosting. We begin with an edge detector for edges, since they are the lowest level nodes in our hierarchy. However, we may also define higher level detectors to find higher-level nodes (e.g. corner detectors). Let us define a possible set of detectors $T = \{t_i : i = 1, 2, \ldots, k\}$ at each node designed to detect that part directly from the image. We also add auxiliary data structures to each node, called “Open” and “Closed” lists. The open lists will store any current potential detections for that node. The closed list will store any accepted detections of the node the list resides at. Each proposal in an open list is weighted by a posterior probability ratio.

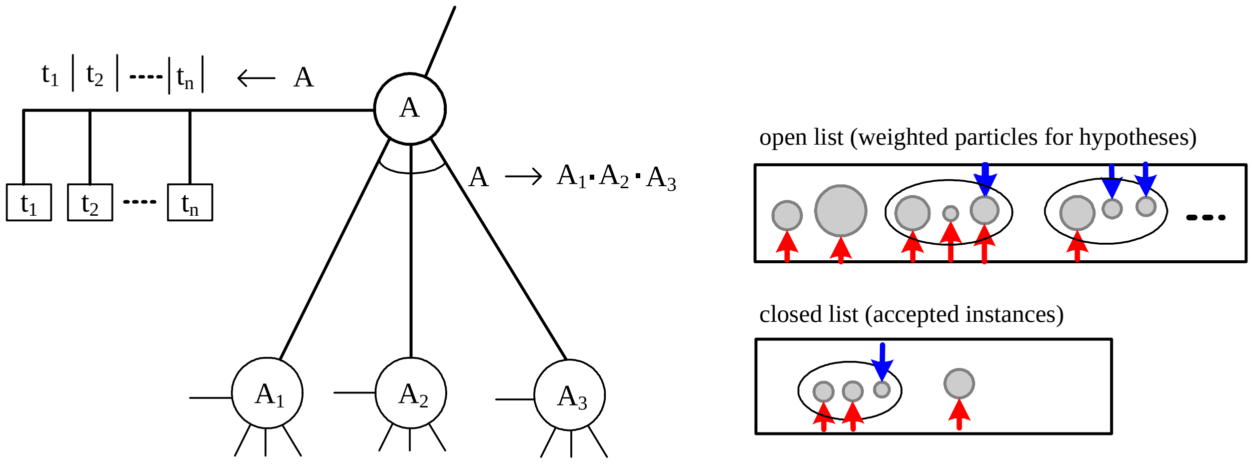

t1 | t2 | ---- | tn | ← A
A
t1
t2
----
tn
A → A1·A2·A3
A1
A2
A3
open list (weighted particles for hypotheses)
closed list (accepted instances)

Figure 13: A conceptualization of inference with Compositional Boosting. The left hand side shows an example of a node in a Compositional Boosting tree with parent node A and children nodes (A_1, A_2, A_3). $(t_1, \ldots, t_n)$ indicate proposals for A detected directly from the image, while $A = A_1 \cdot A_2 \cdot A_3$ indicates proposals for A detected as a product of child proposals. In the inference process, we store proposals at node A in open and closed lists, where particles in the open list are pending proposals and particles in the closed list have been accepted. The red and blue arrows in the figure indicate that there is evidence for each particle coming from both bottom-up and top-down channels.

Compositional Boosting first creates proposals for the open lists for an image I in one of two ways:

(1) Proposals for A are formed from local detectors T. The weight of each detection is the log-ratio of the local marginal posterior probability on an image patch Λ^i using some features of the image $F()$,

$$\hat{\omega^i_A} \approx \log \frac{p(A^i|F(I_{\Lambda^i}))}{p(\bar{A}^i|F(I_{\Lambda^i}))}\,, \tag{43}$$

where $\bar{A}$ is an alternative hypothesis.

(2) Proposals for A are formed by combining proposals for A's children from their Open and Closed lists. Proposals from each list are compared based on their compatibility, and highly compatible proposals are combined to propose the higher level node A. The weight on these hypotheses is the local conditional posterior probability ratio. Suppose a proposal A^i is formed from three of its child proposals A^i_1, A^i_2, and A^i_3, then the weight will be

$$\hat{\omega^i_A} \approx \log \frac{p(A^i_1, A^i_2, A^i_3|A^i)p(A^i)}{p(A^i_1, A^i_2, A^i_3|\bar{A}^i)p(\bar{A}^i)} \tag{44}$$

where $\bar{A}$ represents a competing hypothesis. In other words, we are measuring the probability that these proposals appeared as a result of A existing as opposed to some other node. The top-down process then greedily adds proposals from the Open lists to the Closed lists and updates the Open list weights until no weights are above a certain threshold.

Figure 13 shows a toy example of this process. Here we see that A can be formed from either its detections T, or by combining proposals from its children. This is where Compositional Boosting is particularly powerful, because weak detections of compatible children may be enough for us to propose the parent node.

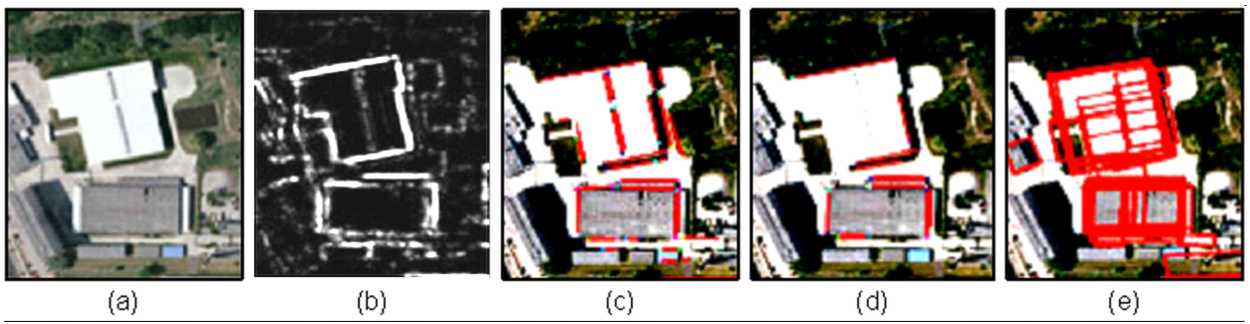


Figure 14: An example of detecting roofs with Compositional Boosting. We run a probabilistic edge detector to get image (b), after which the algorithm detects object parts, such as parallel lines and corners in (c) and (d). These act as evidence for the final roof proposals in (e).

Figure 14 shows an example of roof detection using Compositional Boosting. We begin with a probabilistic edge map formed from our source image. From this map we first extract edge segments using an edge detector. We next combine edges that are compatible according to the rules that we've learned from labeled roofs. Figure 14(c) shows parallel lines detected in the image, while Figure 14(d) shows corners detected in the image. Figure 14(e) shows the final roof detections inferred from the composition of these low-level features. We can see that the rectangular structures of roofs are detected, but there are also many false positives present.

7.2 Top-Down Bayesian Formulation

Using the approach above, we arrive at a set of N candidate proposals, $C_0 = \{c_i : i = 1, 2, \ldots, N\}$. Each candidate proposal consists of a boundary of an object that was detected using bottom-up detectors. We also use each detector's output as a measure of the object likelihood, $L_i = p(I|b_i, l_i)$. The computation of this likelihood varies from detector to detector, but in our experiments, for example, we computed color histograms for classes like trees, parking lots, and roads and used those to compute the likelihood of a bounded region based on its color distribution. Other methods like Compositional Boosting return a probability for the detected object, which we used as its likelihood score. For our AdaBoost candidates we used the number of overlapping proposals for each object as a measure of the likelihood of the object. Note that each likelihood is independent of the other classes, i.e. L_i measures the probability that a patch belongs to class A versus that it doesn't, not the probability that it belongs to class A versus class B. Ideally we would find a multiclass model that accounted for the probability that a patch belonged to class A over class B or class C, but for now we use a simple two-class approximation to measure the strength of each proposal. Each proposal is then $c_i = \{b_i, l_i, L_i\}$.

Our goal is to find the parse graph pg that best describes the image. pg will consist of a set of candidates $C \subseteq C_0$ and grouping decisions such that the likelihood of the candidates and probability of their configuration is maximized. The Bayesian formulation is

$$pg^* = \operatorname*{argmax}_{pg} p(pg|I) = \operatorname*{argmax}_{pg} p(pg; \Theta, R) p(I|pg) \tag{45}$$

7.2.1 Previous Approach and Motivation

The conference version of this work [21] used a greedy algorithm to maximize this posterior. In that work, the authors used an iterative approach that first assigned a weight $\omega(c_i)$ to each of the currently unselected proposals based on how well it maximized the posterior. The object with the heighest weight was selected to be added to the running parse of the scene, pg, thus forming pg_+. The objects were then reweighted according to how much the remaining objects would improve the overall explanation of the scene and this process iterated until no objects above a certain weight remained.

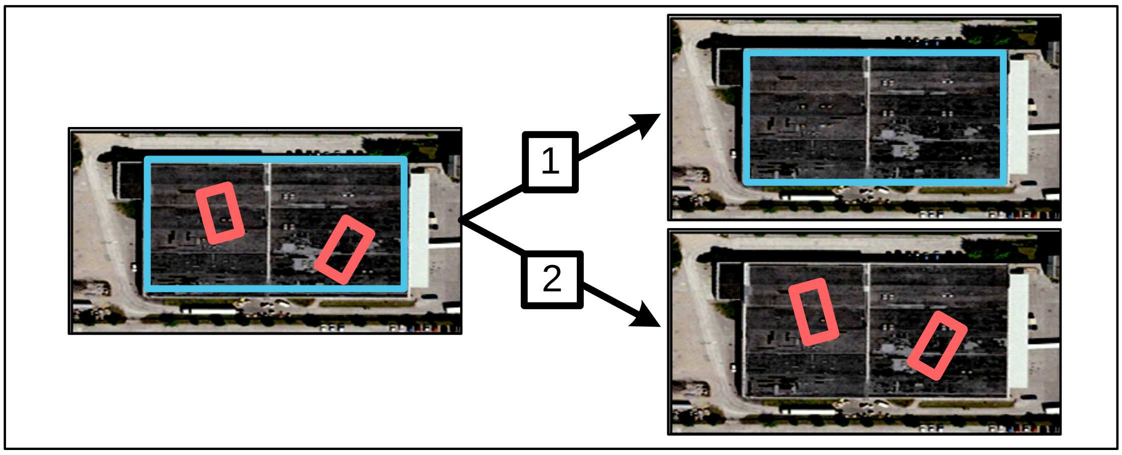


Figure 15: An example of two choices a greedy inference algorithm could make for interpretations of the scene. Because cars cannot appear on top of roofs (let us assume the data supports this) selecting the roof in decision 1 eliminates the car nodes while selecting the cars in decision 2 does the opposite. The algorithm is stuck with this decision no matter what later evidence it may find.

The problem with this approach is shown in Figure 15. Because the algorithm described in [21] is greedy, it cannot backtrack from a poor decision. By selecting the car node to be in the final parse of the scene in the second case, the algorithm will now give an exceedingly low weight to the enclosing roof, as we virtually never see cars on top of roofs. However, had the roof been selected first, as in scenario 1, the car would have been given a very low weight and we would have arrived at the correct interpretation. We would like our new algorithm to be able to backtrack from these mistakes.

7.2.2 Motivation: Swendsen-Wang Clustering

One solution to this problem is to use Swendsen-Wang clustering (SWC). SWC was designed to sample the Ising model by updating the labels of many sites at once instead of using Gibbs sampling to update one at a time[1, 28]. This is similar to our task of switching many objects in or out of the explanation at once. We'll look at SWC at a high level to motivate our solution to the problems in our specific task.

In classic Swendsen-Wang we have a number of sites with class labels,

$$X = \{x_1, x_2, \ldots, x_n\}, \;\; l(X_i) = \{l_1, l_2, \ldots, l_k\}\,, \tag{46}$$

that have a neighborhood structure of edges E connecting them. The goal of the algorithm is to assign the labels such that some energy term $\xi(X)$ is minimized. In the Ising model, this is simply a constraint that neighboring sites have the same label

$$p(X) = \frac{1}{Z} \exp^{-\xi(X)}; \;\; \xi(X) = \beta \sum_{<s,t> \in E} \delta(l(x_s) = l(x_t)), \; \beta > 0. \tag{47}$$

In our model, $p(X)$ would be our learned prior $p(pg; \Theta, R)$. SWC updates portions of X quickly by clustering neighboring nodes that have the same color and updating their labels simultaneously. It does this by first introducing an auxiliary variable into the Ising model, U, indicating whether an edge is "on" or "off" between two nodes,

$$U = \{u_{st} :< s, t > \in E\}, \;\; u_{st} \in \{0, 1\}. \tag{48}$$

Edges are turned on with probability ρ if the nodes they connect have the same value, and are turned off otherwise. These edges will be used to join nodes with the same labels into connected components.

The benefit of adding U to the formulation to get $\pi(X, U)$ was that Swendsen and Wang could now take large steps in the solution space. It was shown in [28] that sampling from $\pi(X|U)$ and $\pi(U|X)$ iteratively produced samples from $\pi(X, U)$. If this distribution was defined such that we can marginalize over U to get $p(X)$ back, then we can generate samples from $p(X)$, which is difficult and slow, by instead sampling from $\pi(X, U)$, which is easy and fast.

Without delving too deeply into the technical details, Swendsen and Wang's algorithm did just that in the following algorithm:

1. Sample from $\pi(U|X)$ to turn edges on and off. Edges between nodes with different labels are turned off w.p. 1. Edges between nodes with the same label are turned off w.p. $1 - \rho$, where $\rho = 1 - e^{-\beta}$ for the Ising model.

2. Form connected components CCP based on the edges that are left on at this iteration. All nodes within each CCP have the same label by the definition of $\pi(X|U)$.

3. Select a connected component CCP_i at random.

4. Sample from $\pi(X|U)$ to relabel the nodes in CCP_i according to the state of neighboring nodes. This quantity is often computed by Gibbs sampling the probabilities of every possible labeling.

It can be shown that, because $p(X)$ can be derived from $\pi(X, U)$ by summing over U, we can sample from $p(X)$.

Figure 16 shows an example of running Swendsen-Wang clustering on the Ising model. In Figure 16(a) edges are probabilistically turned on or off based on the labels of the nodes they connect to. This is

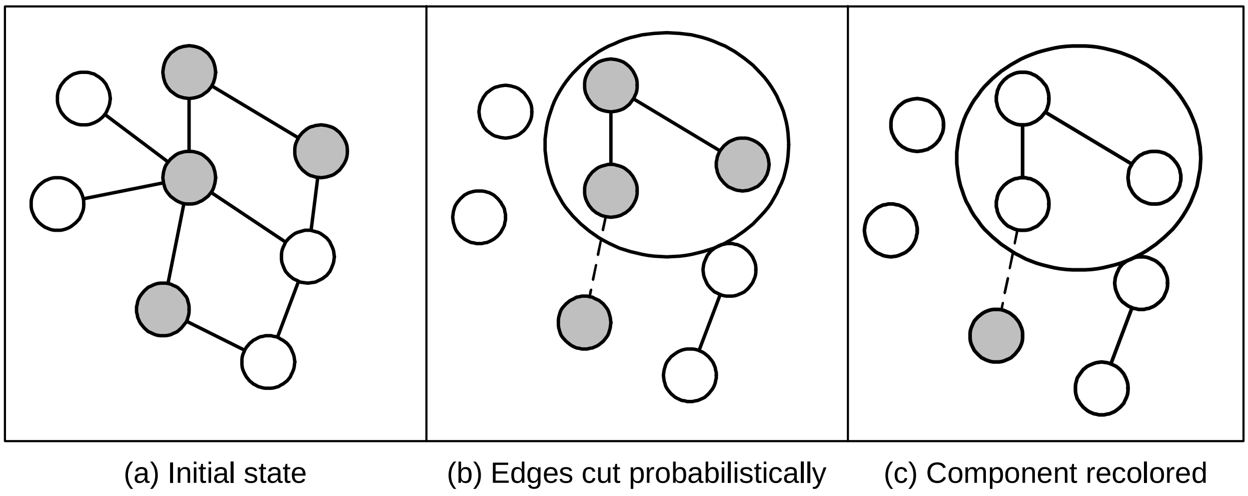


Figure 16: An example of Swendsen-Wang clustering on the Ising model. (a) The current state. (b) Nodes with different labels have their edges turned off, while nodes with the same label are connected w.p. $e^{-\beta}$. The dashed lines show edges that were probabilistically cut between nodes with the same label. The remaining edges form connected components. (c) Simultaneously relabel all nodes in a randomly selected connected component. This process updates large portions of the solution space quickly.

equivalent to sampling from $\pi(U|X)$. A connected component is then formed in Figure 16(b) based on the edges currently on and its labels are reassigned in Figure 16(c). This last step is equivalent to sampling from $\pi(X|U)$. The end result from this process are samples from $\pi(X, U)$, and thus $p(X)$.

This approach will not work for our task, however, for two main reasons:

1. Alternative explanations: In our task, each object proposal c_i can take a label $l_i \in \{0, 1\}$, indicating if it is in the current explanation or not. If we label a cluster using traditional SWC, we can remove a cluster of proposals from our explanation without adding any nodes back in. This may cause our posterior probability to decrease, as we have now explained less of the image. With no alternative explanation to replace these removed proposals we simply decrease the probability of our current system, which means this move is accepted with a very low probability.

2. Conflicting nodes: If we add an object to the solution set that conflicts with other objects, say a car on top of a tree, this will give the explanation such high energy that this move will never be accepted. This is the opposite problem from problem 1, in that we now need to remove parts of the current explanation simultaneously with adding the new explanation, otherwise the combined explanation might have a very low probability.

An example of this problem is shown in Figure 4. We need an algorithm that can exchange alternative explanations simultaneously without introducing conflicting objects to the explanation.

7.3 C4: Clustering via Cooperative and Competitive Constraints

We use an algorithm called Clustering via Cooperative and Competitive Constraints (C4)[22] to deal with these problems. It differs from Swendsen-Wang clustering in two major ways:

1. Negative edges: In addition to the "positive" edges in Swendsen-Wang clustering, in which nodes were encouraged to have the same label, C4 incorporates negative edges, dictating that neighboring sites should not be labeled similarly. We use them here to indicate that two explanations of the scene can't both exist at once. For example, we could have negative edges between two overlapping cars to indicate that they cannot both be in the same explanation at the same time.

2. Composite Flips: Traditional Swendsen-Wang updates the label of a single cluster in one step. In our model the new labels for one cluster may cause it to violate constraints with neighboring clusters, so we may need to update the labels of many clusters simultaneously. We thus form composite components consisting of conflicting clusters that all need their labels reassigned at once to remain consistent. This is like the switch shown in Figure 4(c), where an entire parking lot and cars are not only taken out of the candidate set, but are replaced with a roof simultaneously.

Figure 17 shows C4's results on a toy model. In this example, we have introduced a backbone of negative edges down the middle of the lattice, requiring that nodes on one side have the same color, but each side has a different color. Traditional Gibbs sampling attempts to update one site at a time, which creates a low probability state. SWC updates an entire side at one time, but only updates one cluster and ignores negative edges, thus creating another low probability state. C4 clusters the entire system and relabels the individual clusters subject to both positive and negative constraints, creating a high probability state.

We can extend the example in Figure 17 to our actual problem. Figure 18 shows a number of object proposals, which are like the nodes in Figure 17. Objects that have a high prior probability of being on together are grouped together with "positive" edges (e.g. the neighboring cars and the parking lot, the tree with the building), while objects that have low prior probability of being on together are grouped by "negative" edges (e.g. the building with the parking lot or tree with parking lot). Here we've added positive and negative edges based on pairwise energies from the exponent in $p(pg; \Theta, R)$. If the energy between two nodes is above a certain threshold, a negative edge is added between them. Otherwise, a positive edge is added.

The probability assigned to each edge is determined by using a squashing function of the energy between the two nodes. This is used to map the pairwise energy to the range $[0, 1]$. In our experiments we used an inverted logistic function $F(x) = \frac{1}{1+e^{-(x-u)/s}}$ where first $F(x) = 1 - F(x), x < 0$ and then we scale to $[0, 1]$ via $F(x)' = 2 * F(x) - 1$. This creates a symmetric function where energies that are much greater

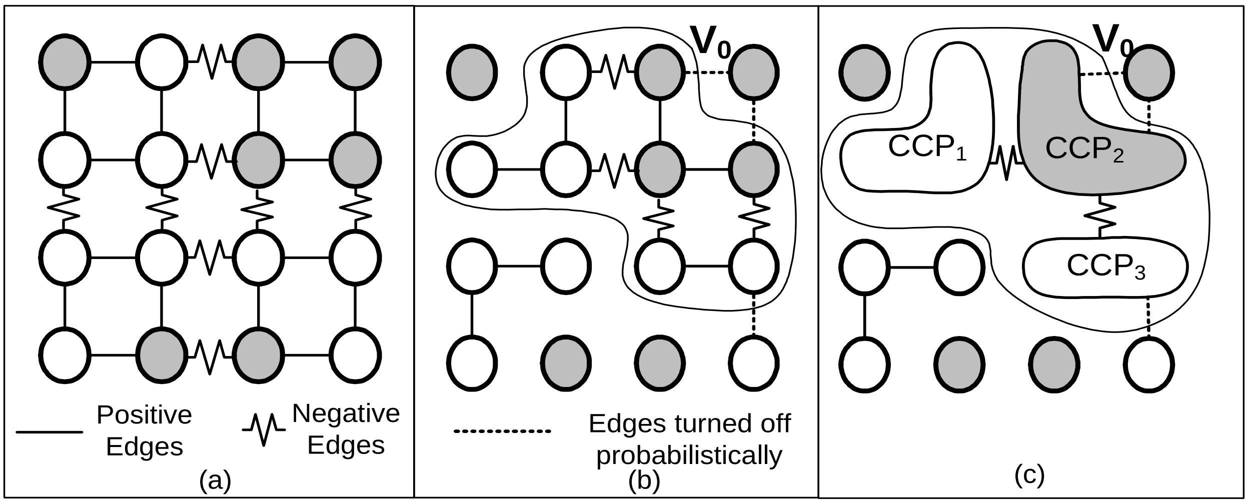


Figure 17: A toy example of C4 on the Ising model. Here we've added negative edges, indicated by jagged lines, that encourage nodes to have opposite labels. In (b), we probabilistically turn edges on and off based on whether they satisfy their current constraints (node labels same or different). The remaining edges form connected components (CCPs). (c) A connected component V_0, which consists of sub-components CCP_i of the same label connected by negative edges, is relabeled according to its constraints. Further details are provided in [22].

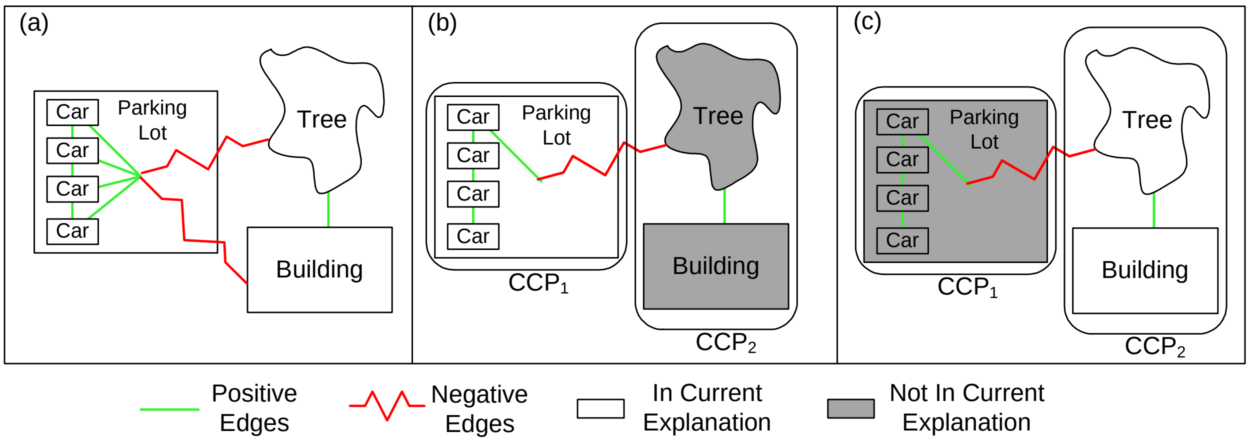


Figure 18: An application of C4 to a toy aerial image. (a) Compatible object proposals (cars in the parking lot, the tree next to the building) are connected by positive edges while non-compatible proposals (tree with parking lot, building with parking lot) are connected by negative edges. (b), (c) C4 groups objects based on these connections and updates the state of the system to swap between alternate explanations.

or much less than the center of the distribution u approach probability 1, while values near u approach 0, as they are just barely positive or negative. The parameters u and s can be adjusted to translate and scale the function, respectively. In this way we create data-driven probabilistic edges between proposals based on their pairwise energies.

In the same way that it does in Figure 17, C4 probabilistically turns positive and negative edges on and off, then proposes to relabel the selected cluster based on its constraints, giving a label of 0 to objects that shouldn't be in the current interpretation and a label of 1 to objects that should be in the current interpretation. Here C4 flips between the two possible explanations of the scene in Figures 18(b) and (c). The decision to flip the interpretation or not is reached by Gibbs sampling the possible component values (here 0 or 1), meaning that each interpretation is selected to be turned on proportionally to the value of $p(X)$ it results in. In the case of two equally likely interpretations, there is a 50% chance that the interpretation will swap at each step. At the conclusion of this process (which is sampling from $\pi(U|X)$ and $\pi(X|U)$ as in SWC) our samples from $p(pg|I)$ are sets of proposals for interpreting the scene. Further derivations and results can be found in [22].

7.4 Top-Down Prediction

In addition to reducing false positives using C4, we can also predict new instances of objects that may have gone undetected using the hierarchical aspect of our model.

Given our final parse pg^* from C4, we can Gibbs sample the group nodes to create new objects per group node or new group nodes in the scene. We can then Gibbs sample the appearance relationships for newly added nodes to form new proposals. As many objects are loosely constrained in the scene it is difficult to predict where objects should be exactly. We remedy this by just predicting additional aligned objects wherever we already have some aligned objects. This allows us to find new true positives in addition to pruning false positives. Figure 5 shows examples of hallucinated objects using the results from the C4 pruning. These final proposals will be added to a final round of C4 clustering to arrive at the final parse.

8 Experiments

We ran our algorithm on 5 large (4000x4000) images collected from Google Earth. We learned a top-down model as in Section 4 and implemented detectors for each of the objects as described in Section 7.1. Figure 19 shows the process of our algorithm on one of these images. The first panel shows the original image, while the second panel shows an overlay of the initial bottom-up detections, which contains a huge number

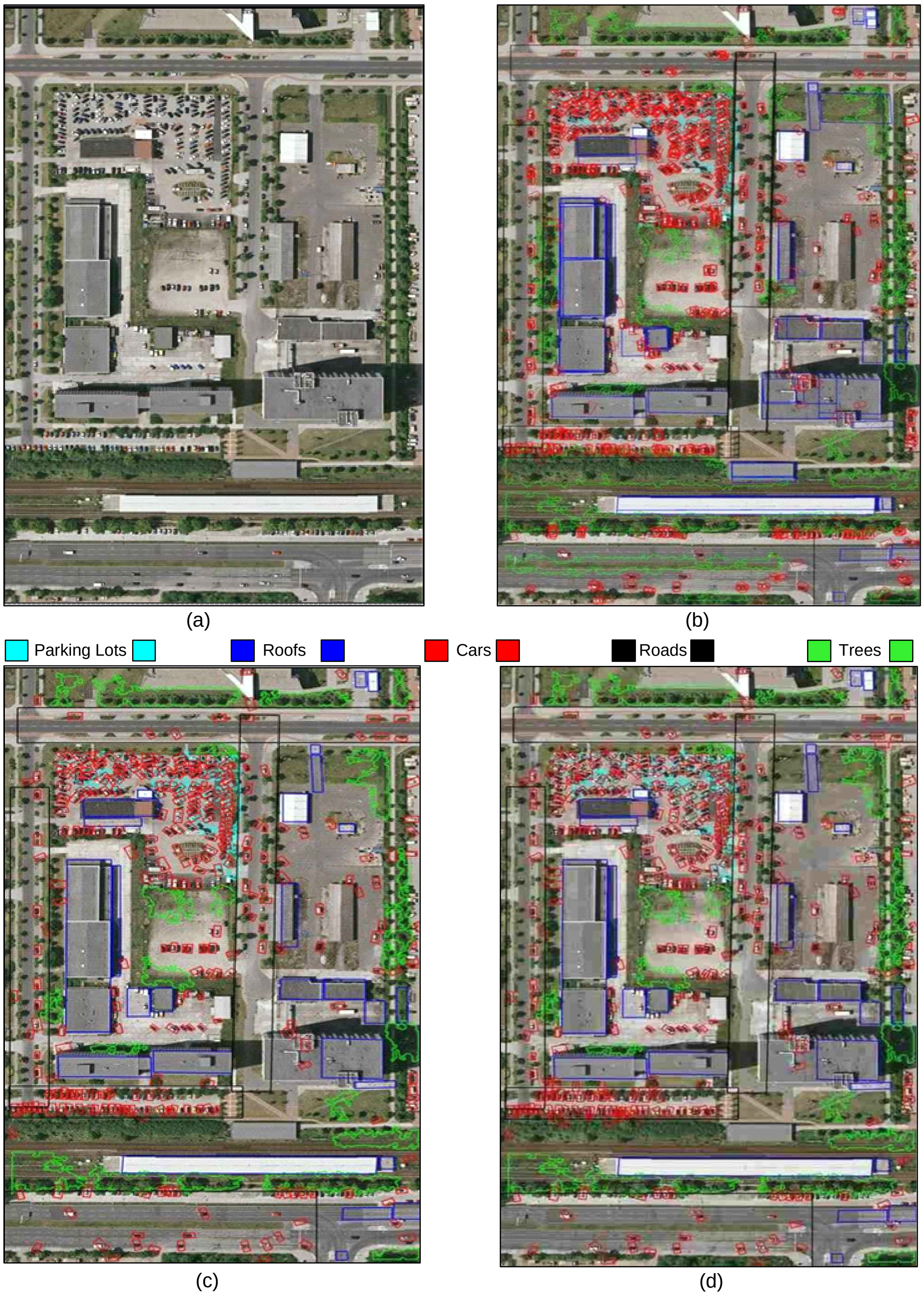


Figure 19: The bottom-up to top-down pipeline. (a) The original image. (b) The bottom-up detections. There are a huge number of overlapping and inconsistent detections. (c) The top-down pruning results using C4. Many false positives are removed. (d) The results given newly proposed nodes from the hierarchical prior.

of false positives (3 false positive roads, 71 false positive buildings, 623 false positive cars, 10 false positive trees). The third panel shows the results of using C4 clustering to find a high probability set of bottom-up detections to explain the scene. The fourth panel shows the final explanation of the image after some new proposals have been suggested and verified. The first step of the C4 algorithm shows the most dramatic improvement, with vast numbers of inconsistent detections (cars on roofs, trees on roads, overlapping roofs) being removed, leaving just single object boundaries for the important objects (we now have 0 false positive roads, 5 false positive buildings, 57 false positive cars and 0 false positive trees). The second step gives a slight improvement, though primarily just in finding missing cars, which are difficult to see at this resolution. Note that there are still some missed detections, either because our initial detectors did not detect the object or because the context may have inadvertently ruled out a valid explanation (e.g. accidentally favoring the shadow of a roof instead of the roof itself, thus suppressing the true roof).

The inference stage, given bottom-up proposals, takes about 10 seconds to run on a dual core 1.6GHz machine. The bottleneck in our pipeline is the detection phase, however. For example, our AdaBoost results take a mere couple of seconds to compute. The edge detector we used, on the other hand, can take upwards of a minute to process each image. Therefore, the speed of our approach is highly dependent upon the speed required to compute the initial bottom-up detections.

Because of the newness of Compositional Boosting, we first examined how much improvement we achieved in detecting low-level roof parts using Compositional Boosting. Figure 20 shows ROC curves for detecting U junctions, L junctions, parallel lines, and opposing L junctions. Using specific bottom-up detectors alone (the blue curves) causes us to miss a lot of the junctions present. By using Compositional Boosting, we are again able to leverage context and hierarchy to identify missing junctions to help us propose more roofs, as shown by the red curves.

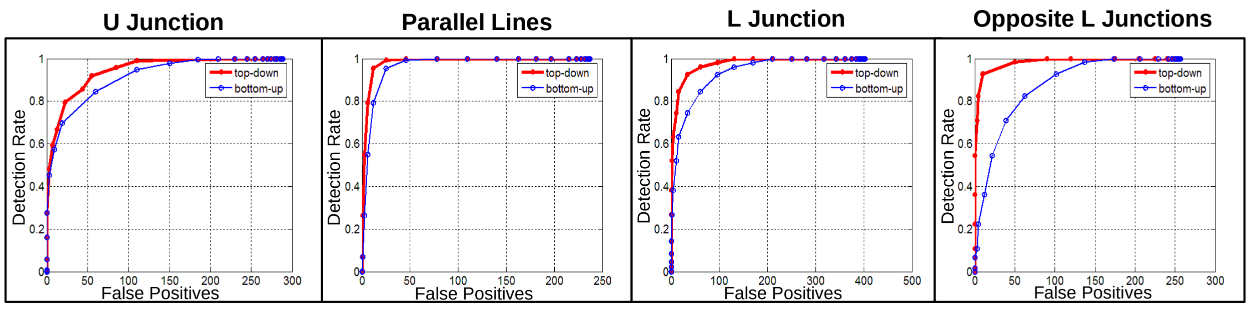


Figure 20: ROC curves for detecting U junctions, L junctions, parallel lines, and opposing L junctions using Compositional Boosting. We see that CompBoost helps us identify weakly detected junctions, which helps us propose better high-level detections.

Figure 21 shows a zoomed in view of our test images before and after pruning. Figure 21 shows that, at first, we have many conflicting proposals for the object boundaries, notably that a parking lot could be on top of the roof. After we enforce the contextual constraints we learned, however, we return to a sensible explanation of the scene, one in which there are no longer cars on top of roofs or overlapping proposals. Figure 22 shows a zoomed in view after we propose new cars. Initially we missed some cars in the rows of the parking lots. Because our model recognizes that cars appear in rows, however, it proposes cars of roughly the same shape and sizes of the neighboring cars around them, using a line grammar. Cars matching above a certain likelihood are accepted and the conflicting nodes are removed.

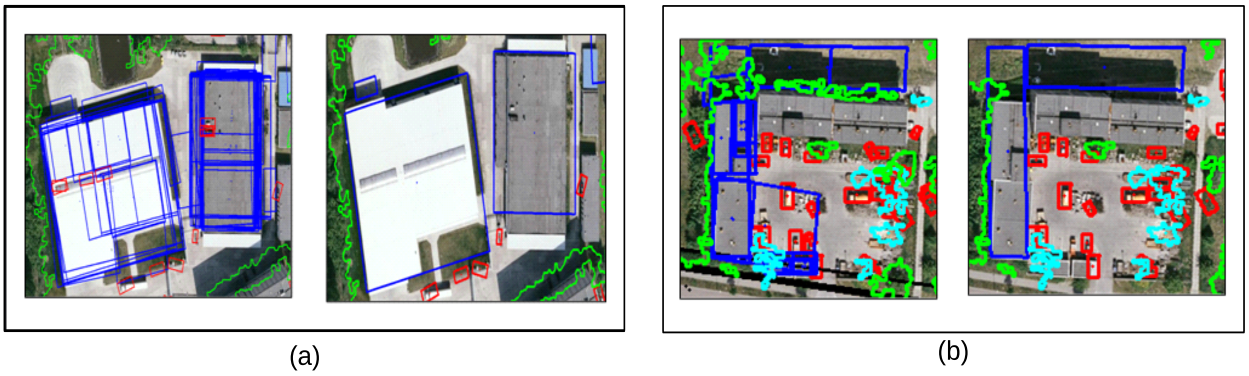


Figure 21: Close up views of our improvement during pruning. Notice that overlapping proposals and inconsistent explanations (cars in trees) have been removed.

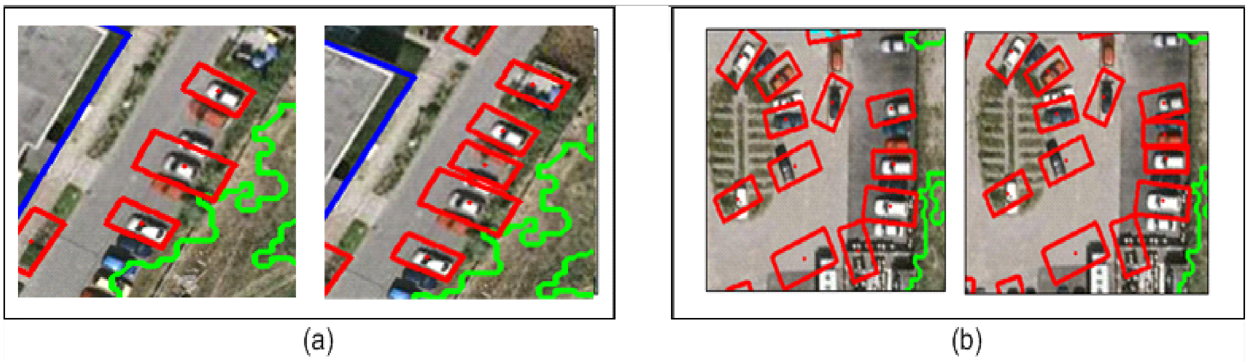


Figure 22: Close up views of our improvement during top-down prediction. Additional cars are added to the rows due to the presence of other collinear cars.

Table 3 shows the detection rate and false positives per image of each category using just that category's detector (shown in parentheses) vs. using the full hierarchical contextual model. We can see that the hierarchical contextual model greatly reduces the false positive rate from single-object bottom-up detectors because it can leverage context to remove false positives. Our detection rate is about the same, however, as

Table 3: False positives per image and detection rates for bottom-up detectors versus our method.

Detection method	False positives per image	Detection rate
Cars (AdaBoost)	242.33	88.1%
Cars (Ours)	**71.83**	**84.2%**
Parking Lots (TextonBoost)	1.17	84.3%
Parking Lots (Ours)	**0.16**	**84.3%**
Trees (TextonBoost)	14.5	88.8%
Trees (Ours)	**9.33**	**88.8%**
Roofs (CompBoost)	73.5	70.3%
Roofs (Ours)	**1.67**	**70.3%**
Roads (CompBoost)	5.67	95%
Roads (Ours)	**0.05**	**88.3%**
Combined (All Detectors)	337.17	93.1%
Combined (Ours)	**83.04**	**87.5%**

the pruning phase in the second stage serves mostly to rule out inconsistent detections. In the third stage we were able to identify a few extra cars (as shown in Figure 22), but the amount of extra detections was not enough to account for the inadvertent pruning of true positives from stage 2. Overall, our context allows us to achieve comparable detection rates to single-object detectors, but with far fewer false positives.

Figure 23 shows two different precision-recall curves for the bottom-up and top-down stages of our process. We show precision-recall as opposed to ROC curves because it is difficult to decide how to compute the number of true negatives for multi-category classification tasks, a decision that can drastically alter the appearance of the algorithm's performance. In Figure 23(a), we measure our accuracy at the pixel level. In this experiment, we labeled each pixel in the image as belonging to an object category or not and then converted our inferred boundaries to a similar labeling. In Figure 23(b) we measure our performance using object-level accuracy. In this case we considered an object to be detected if it had a boundary around it within some threshold of its true scale and position. In both cases we can see that the top-down improvements over the initial detections are substantial. While the initial detections give average performance, it is the introduction of the top-down pruning and prediction that flattens our curve, enabling us to keep a very high level of precision as the recall increases. Notice, however, that in Figure 23(b) the second stage actually degrades performance slightly for low values of recall, likely because it has pruned too many true positives, reducing our precision slightly. This could likely be improved by adjusting the likelihoods of our initial candidates so that we don't overprune them. Overall though the top-down performance far eclipses the initial bottom-up detection results on their own. We also looked at the benefits of using C4 to find solutions.

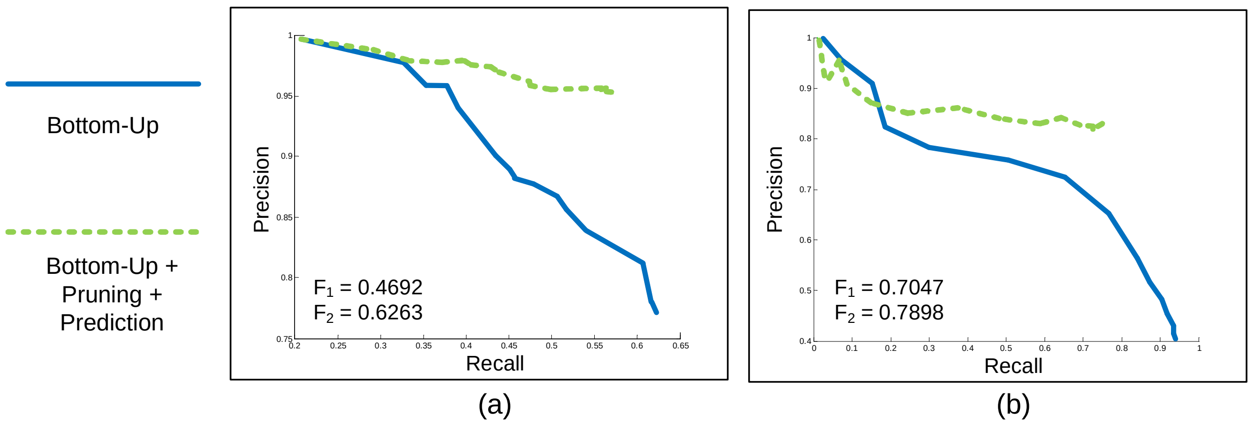


Figure 23: Precision-Recall curves for the bottom-up and top-down inference algorithm. In both cases we see a huge improvement by using CSW to prune out false positives and using our model to predict missing objects. F_1 and F_2 are the best F-measures for the bottom-up and for the full algorithm, respectively. (a) Precision-Recall curve using pixel-level accuracy, i.e. each pixel in the image is assigned a category label. (b) Precision-Recall curve using object-level accuracy, i.e. each object is considered detected if we infer an object of appropriate dimensions over it.

Figure 24 shows an example of a patch where C4 is extremely useful during inference. In the first panel, the algorithm has mistakenly interpreted the vents on top of the building as cars. This has thus ruled out the true explanation that there is a building there. Thanks to C4's ability to swap out all of the related cars while simultaneously adding the roof, we are able to arrive at the correct solution in panel (b). This solution is maintained because it has a higher probability than the previous explanation.

In Section 4.2 we mentioned that we choose to add relationships iteratively instead of fitting the full model, as this allows us to keep our model simple for sampling and performing inference. The question remains, however, about whether we need to learn as many relationships as we do. Figure 25 shows the inference results for an aerial image using a model with a high ε_1 (10) and a model with a standard ε_1 (4). We can see that the partially learned model is lacking contextual relationships for cars and buildings, as many cars appear on roofs, and cars appear on top of one another. The fully learned model does not make these mistakes. While ε_1 is definitely variable, we strive to select a value that produces good results while still minimizing the size of our relation set R.

Figures 26 and 27 show the final results of our algorithm on a number of other urban aerial images. We used our algorithm to find the best parse graph representations for each object and here just display the

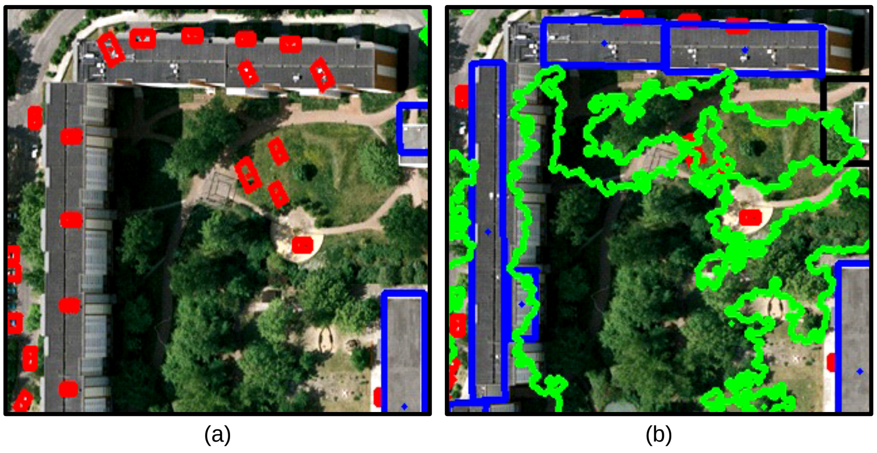


Figure 24: An example of swapping alternative solutions using C4. (a) Vents are incorrectly labeled as cars on top of the roof. (b) The cars are correctly swapped out for the roof simultaneously to arrive at the correct solution.

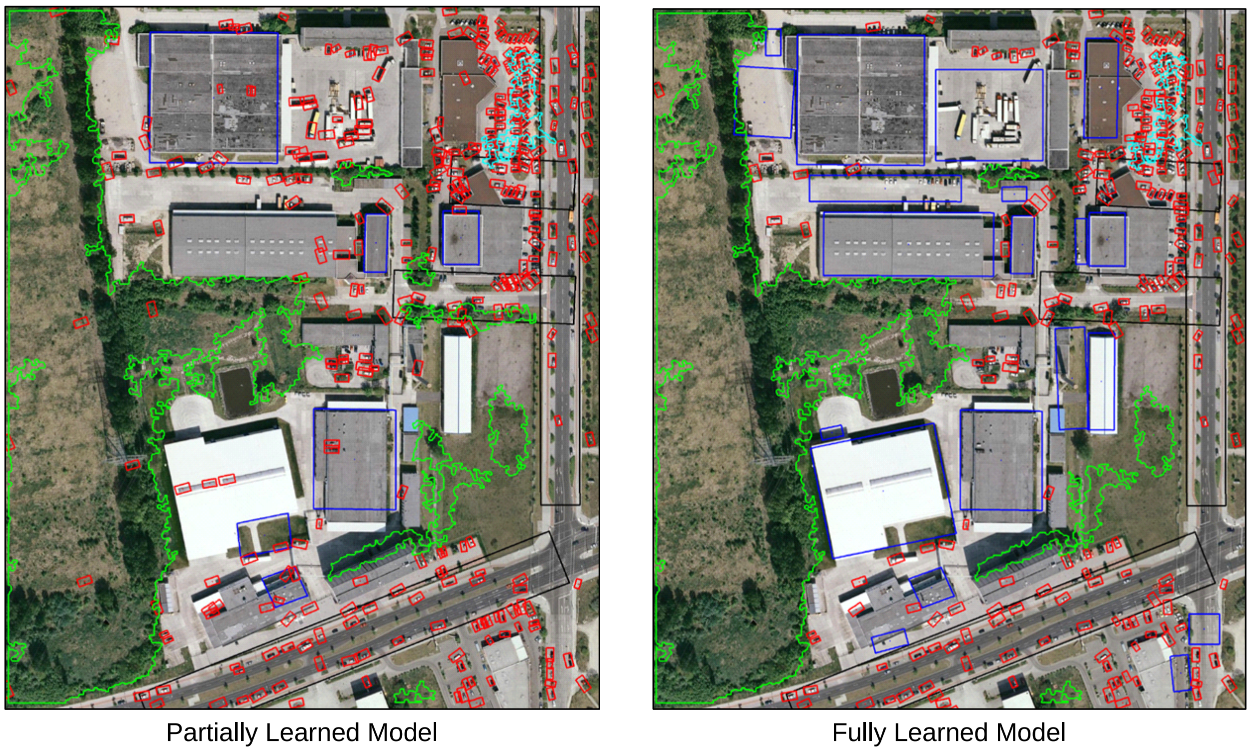


Figure 25: A comparison of inference results for the model learned with a very high ε_1 versus a rather low ε_1 (i.e. fewer relationships are added to the first model). The partially learned model is missing a lot of overlap constraints (e.g. cars on trees, cars on buildings), and so makes very poor decisions when parsing the scene. Many of the buildings have cars on top of them and cars readily overlap each other.

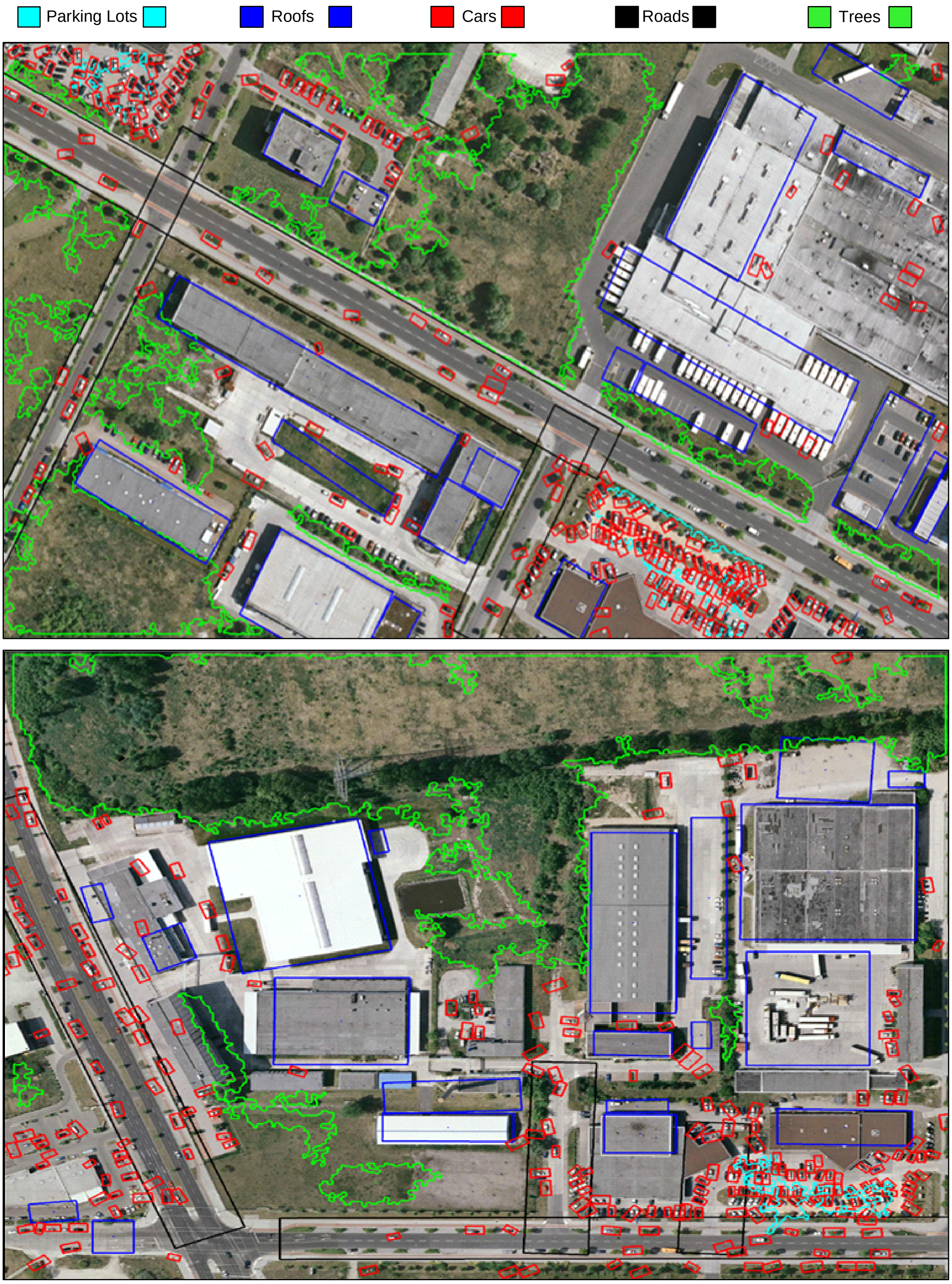


Figure 26: Flat configurations of parsed images.

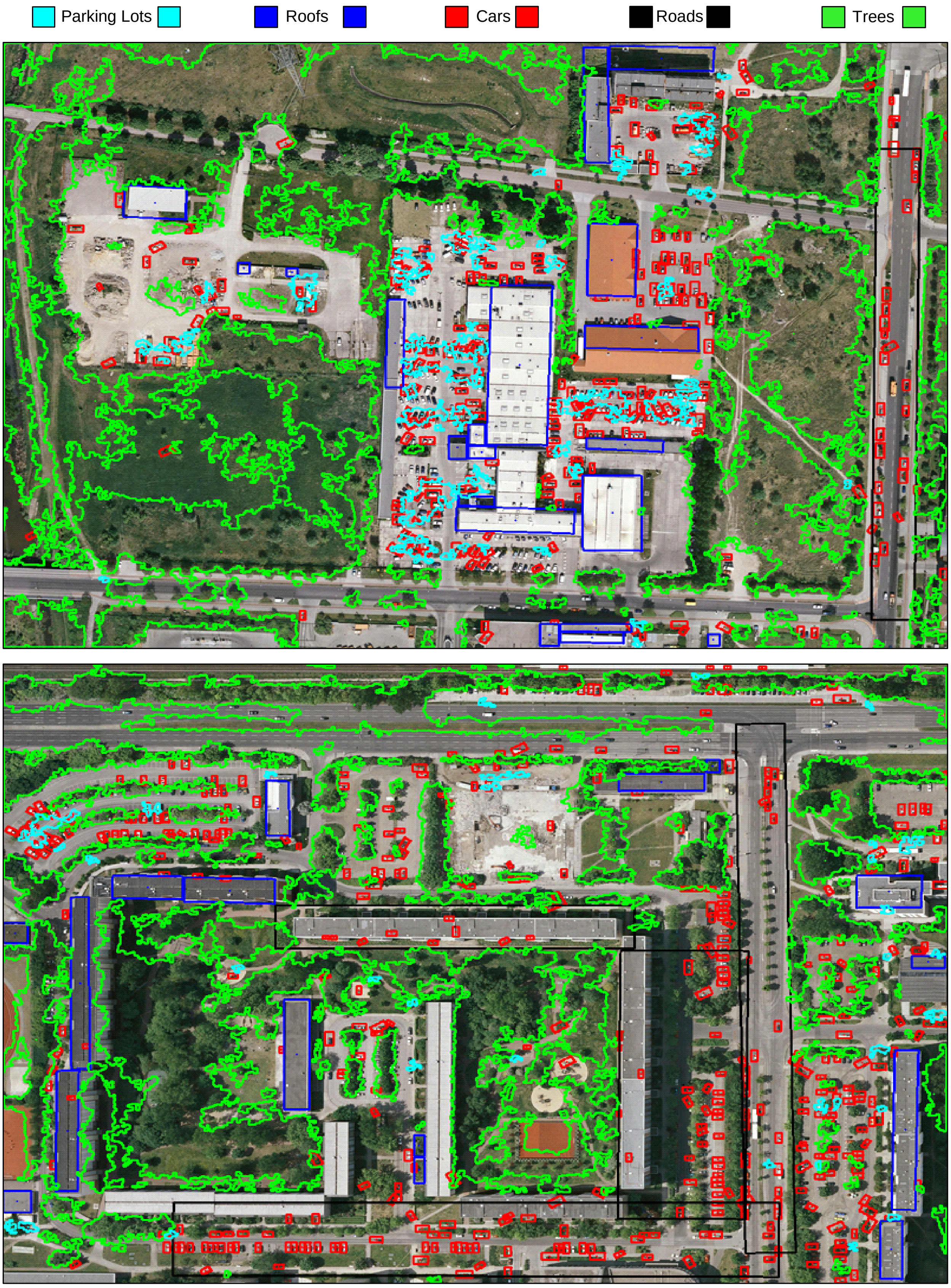


Figure 27: Flat configurations of parsed images.

flattened configurations of the highest probability parse graph for each scene. We can see that the majority of objects are detected accurately, though there are still a few false positives.

We would like to compare our methods to other works in the field, but, as mentioned in Section 1, we are hard-pressed to find competing algorithms that identify multiple categories of objects. Similarly it was quite difficult to find benchmarks on consistent datasets in the aerial imaging community, so we would like to offer these results as a benchmark on the aerial images we used from the Lotus Hill Database. These images will be available from the Lotus Hill Institute's website (http://www.imageparsing.com) and can be used freely by anyone else interested in testing on them.

9 Discussion and Future Work

In this paper we presented a 3-level contextual hierarchy for modeling aerial images that automatically augments a hierarchical model with relational constraints using a minimax entropy framework. The learning algorithm was able to iteratively add relationships from even large dictionaries of potential relationships in order to model the statistics of the aerial images. We found that this learning process was impeded only by the time needed to sample new images, which can be made tractable through a number of optimizations. In the end our model selected only a small subset of the dictionary of relationships, yet was still able to accurately recreate aerial images. The current work relied on hand-labeled data, but we would like to relax this assumption to first do unsupervised recognition of object types before learning the scene structure in future work.

The bottom-up methods we used along with our three-phase Bayesian inference algorithm were instrumental in obtaining the results that we did. Using C4 to swap interpretations of the scene was extremely helpful, and at times critical, in finding a good subset of detections to represent the scene. Our top-down prediction did not, however, significantly improve results much, as our algorithm only proposed objects that met the line grammar relationship rules (e.g. cars). As was seen in Figure 21, very few cars are actually added to the representation. This work served as a proof of concept that some combined bottom-up/top-down inference can definitely improve performance, but in this case there were so many objects in each scene that an improvement by 10-20 newly detected objects didn't significantly impact our classification rate. For specific tasks, for example one in which finding every car is important, this technique will be very useful.

In our experiments we hand-defined our 3-level hierarchy and added the context automatically. When learning the hierarchy automatically, however, the question remains as to when one uses hierarchy to group objects and when one simply uses context. We found that grouping objects made our representation simpler and allowed us to avoid the computational inefficiency of computing a fully pairwise graph between all objects. However, we certainly could have created a flat model and added constraints to that. The question of when to use hierarchy and when context on a flat model suffices is an interesting and unresolved one.

Overall we found that our method was able to learn relationships flexibly and out-performs commonly-used single-object detectors. We hope in the future to find a combination of initial detectors, improved bottom-down inference, and adjustments to C4 that will improve our results even further.

Acknowledgments

This work at UCLA is supported by an NSF grant IIS-0713652 and ONR grant N00014-05-01-0543. The work at Lotus Hill Research Institute is supported by 863 project 2008AA01Z126 and NSFC grant 60776793. We thank the Lotus Hill Dataset for groundtruth annotation [38] for the aerial images used in this paper and Benjamin Yao for his many helpful discussions and his assistance with preparing the dataset.

References

[1] A. Barbu and S.-C. Zhu. Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *Pattern Analysis and Machine Intelligence*, 27:1239–1253, 2005.

[2] A. Berg, F. Grabler, and J. Malik. Parsing images of architectural scenes. *IEEE 11th International Conference on Computer Vision*, 2007.

[3] H. Chen, Z. Xu, Z. Liu, and S.-C. Zhu. Composite templates for cloth modeling and sketching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1:943–950, 2006.

[4] Z. Chi and S. Geman. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2), 1998.

[5] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[6] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973.

[7] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, (55), 1997.

[8] K. Fu. *Syntactic Pattern Recognition and Applications*. Prentice Hall, 1981.

[9] F. Han and S.-C. Zhu. Bottom-up and top-down image parsing by attribute graph grammar. *Proceedings of the International Conference on Computer Vision*, 2, 2005.

[10] S. Hinz and A. Baumgartner. Road extraction in urban areas supported by context objects. *Intl. Archives of Photogrammetry and Remote Sensing*, 33, 2000.

[11] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2:2145–2152, 2006.

[12] M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. Estimators for stochastic unification-based grammars. *Proceedings ACL'99, Maryland*, 1999.

[13] Y. Keselman and S. Dickinson. Generic model abstraction from examples. *Pattern Analysis and Machine Intelligence*, 27:1141–1156, 2001.

[14] F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2:524–531, 2005.

[15] Y. Li, I. Atmosukarto, M. Kobashi, J. Yuen, and L. Shapiro. Object and event recognition for aerial surveillance. *SPIE - The International Society for Optical Engineering*, 2005.

[16] M. A. Maloof, P. Langley, T. Binford, R. Nevatia, and S. Sage. Improved rooftop detection in aerial images with machine learning. *Machine Learning*, 2003.

[17] T. Matsuyama and V. Hang. Sigma: A framework for image understanding integration of bottom-up and top-down analyses. *Plenum*, 1990.

[18] H. Moissinac, H. M. tre, and I. Bloch. Urban aerial image understanding using symbolic data. *Image and Signal Processing for Remote Sensing, Proc. SPIE*, 1994.

[19] B. Nicolas, J. Viglino, and J. Cocquerez. Knowledge based system for the automatic extraction of road intersections from aerial images. *International Archives of Photogrammetry and Remote Sensing*, 2000.

[20] Y. Ohta. *Knowledge-based interpretation of outdoor natural color scenes*. Pitman, 1985.

[21] J. Porway, K. Wang, B. Yao, and S.-C. Zhu. A hierarchical and contextual model for aerial image understanding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[22] J. Porway and S. Zhu. C4: Stochastic inference on graphical models with positive and negative edges for rapidly exploring competing solutions. *Technical Report*, 2009.

[23] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. *Proceedings of the European Conference on Computer Vision*, pages 1–15, 2006.

[24] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999.

[25] A. Singhal, J. Luo, and W. Zhu. Probabilistic spatial context models for scene content understanding. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 2003.

[26] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering objects and their location in images. *Tenth IEEE International Conference on Computer Vision*, 2005.

[27] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Describing visual scenes using transformed dirichlet processes. *Neural Information Processing Systems*, 2005.

[28] R. Swendsen and J. Wang. Nonuniversal critical dynamics in monte carlo simulations. *Physicl Review Letters*, 1987.

[29] S. Todorovic and N. Ahuja. Extracting subimages of an unknown category from a set of images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1:927–934, 2006.

[30] Z. Tu and S.-C. Zhu. Image segmentation by data-driven markov chain monte carlo. *PAMI*, 24(5):657–673, 2002.

[31] S. Ullman, E. Sali, and M. Vidal. A fragment-based approach to object representation and classification. *Proceedings of the 4th International Workshop on Visual Form*, 2001.

[32] C. Vestri and F. Devernay. Using robust methods for automatic extraction of buildings. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1, 2001.

[33] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[34] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1):1–305, 2008.

[35] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2:101–108, 2000.

[36] L. Wei and V. Prinet. Building detection from high-resolution satellite image using probability model. *Geoscience and Remote Sensing Symposium, IGARSS*, pages 25–29, 2005.

[37] T. F. Wu, G. Xia, and S.-C. Zhu. Compositional boosting for computing hierarchical image structures. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[38] B. Yao, X. Yang, and S.-C. Zhu. Introduction to a large scale general purpose groundtruth dataset: methodology, annotation tool, and benchmarks,. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 4697:169–183, 2007.

[39] T. Zhao and R. Nevatia. Car detection in low resolution aerial image. *IEEE International Conference on Computer Vision*, 1, 2001.

[40] L. Zhu, C. Lin, H. Huang, Y. Chen, and A. Yuille. Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. *Proceedings of the 10th European Conference on Computer Vision: Part II*, 2008.

[41] S.-C. Zhu and D. Mumford. A stochastic grammar of images. *Foundation and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006.

[42] S.-C. Zhu, Y.-N. Wu, and D. Mumford. Frame: Filters, random fields, and minimax entropy towards a unified theory for texture modeling. *International Journal of Computer Vision*, (2):107–126, 1998.