6-1-2010

Detecting and Parsing Architecture at City Scale from Range Data

Alexander Toshev
University of Pennsylvania

Philippos Mordohai
Stevens Institute of Technology

Ben Taskar
University of Pennsylvania, taskar@cis.upenn.edu

Toshev, A.; Mordohai, P.; Taskar, B.; , "Detecting and parsing architecture at city scale from range data," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* , vol., no., pp.398-405, 13-18 June 2010 doi: 10.1109/CVPR.2010.5540187

Detecting and Parsing Architecture at City Scale from Range Data

Abstract
We present a method for detecting and parsing buildings from unorganized 3D point clouds into a compact, hierarchical representation that is useful for high-level tasks. The input is a set of range measurements that cover large-scale urban environment. The desired output is a set of parse trees, such that each tree represents a semantic decomposition of a building – the nodes are roof surfaces as well as volumetric parts inferred from the observable surfaces. We model the above problem using a simple and generic grammar and use an efficient dependency parsing algorithm to generate the desired semantic description. We show how to learn the parameters of this simple grammar in order to produce correct parses of complex structures. We are able to apply our model on large point clouds and parse an entire city.

Disciplines
Computer Sciences

Comments
Toshev, A.; Mordohai, P.; Taskar, B.; , "Detecting and parsing architecture at city scale from range data," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* , vol., no., pp.398-405, 13-18 June 2010 doi: 10.1109/CVPR.2010.5540187

Detecting and Parsing Architecture at City Scale from Range Data

Alexander Toshev
GRASP Laboratory
University of Pennsylvania
Philadelphia, PA, USA
toshev@cis.upenn.edu

Philippos Mordohai
Department of Computer Science
Stevens Institute of Technology
Hoboken, NJ, USA
mordohai@cs.stevens.edu

Ben Taskar
GRASP Laboratory
University of Pennsylvania
Philadelphia, PA, USA
taskar@cis.upenn.edu

Abstract

We present a method for detecting and parsing buildings from unorganized 3D point clouds into a compact, hierarchical representation that is useful for high-level tasks. The input is a set of range measurements that cover large-scale urban environment. The desired output is a set of parse trees, such that each tree represents a semantic decomposition of a building – the nodes are roof surfaces as well as volumetric parts inferred from the observable surfaces. We model the above problem using a simple and generic grammar and use an efficient dependency parsing algorithm to generate the desired semantic description. We show how to learn the parameters of this simple grammar in order to produce correct parses of complex structures. We are able to apply our model on large point clouds and parse an entire city.

1. Introduction

We address a central scene understanding problem: the inference of semantic information from raw, unorganized data and the encoding of this information in a representation that is suitable for higher level tasks. Specifically, we are interested in interpreting architecture in terms of parts, which are either observable surfaces, such as roofs and facades, or unobserved *volumetric parts* that are enclosed by these surfaces.

The main contribution of our work is a framework for simultaneous building detection and parsing into a compact, symbolic representation. The input is an unorganized point cloud and the end result is a parse tree whose nodes are volumetric parts and surfaces (see Fig. 1). Unlike previous methods [9, 15, 18] that operate on the observed surfaces, we also consider the volumetric parts that can be inferred from subsets of their bounding surfaces. This dramatically increases robustness to occlusion and allows us to infer consistent parsings based on the volumetric parts, even when some of the surfaces are missing. The intuition is that a building can be viewed as a tree, whose nodes are volumetric parts that lie on or next to each other and are covered by planar patches such as roofs and roof parts, which we consider as children of the volumes (see Fig. 2).

To formalize the above representation, we introduce a simple grammar, which captures generic geometric properties between planar patches and volumes. Additionally, the grammar contains two supernodes called "building" and "non-building" which serve as ascendants of all other nodes except for the root node. In this way, we can perform detection while parsing – all parse trees rooted at supernode "building" are considered to represent buildings.

The simplicity of the above grammar allows us to use dependency parsing, which is an efficient parsing technique. This makes it possible to parse a whole city into a single tree. Additionally, we can use labeled data to estimate optimal parameters of the grammar by employing structured learning. In this way, we do not have to specify complex rules, but learn to parse in a data-driven way. As we show in Sec. 4, we can obtain semantical parses of complex architectural structures using our generic grammar.

2. Related Work

In this section, we review related work on building detection and on parsing architectural structures. Building detection and description from airborne imagery has been the focus of long-term efforts of several research groups in computer vision. The resulting systems [23, 16, 7, 11, 26, 13, 4, 17] detect buildings among clutter from aerial imagery by reasoning on simple primitives.

Range data have been extensively used as input for building detection in photogrammetry [20]. Techniques based on generic primitives, parametric models and rule-based reasoning have been published recently by Verma et al. [27] and Brédif et al. [3]. Matei et al. [21] and later Poullis and You [24] presented building segmentation results on very large scale range datasets with an emphasis on robustness.

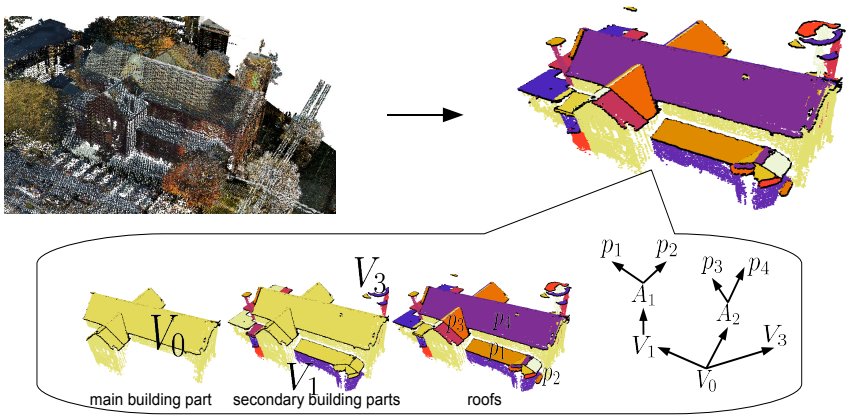

Figure 1. We extract from a raw unorganized 3D point clouds buildings and parse them into geometric and semantically meaningful parts, which are structured in a natural way and represent the topology of the building.

Several authors have used ground-level images as inputs. Werner and Zisserman [28] presented a multi-view method that reconstructs facades and estimates features, such as windows and balconies, by sweeping polygonal primitives in directions parallel to the facades. Berg et al. [2] addressed facade detection and segmentation from a single image using a conditional random field.

Haala et al. [14] detect buildings from merged airborne and terrestrial LIDAR data by decomposing the space into 3D cells using planes fitted to the data. The cells are classified as building or non-building, and buildings are extracted as connected components.

A multi-view method for parsing buildings using a grammar was present by Dick et al. [9] who represent buildings, imaged from the ground level, as collections of planes and parameterized primitives. MCMC is used to optimize structure taking into account global properties, such as symmetry. Han and Zhu [15] tackled the single image case in a combined bottom-up top-down framework that employs a grammar with rectangular primitives. Recently, Koutsourakis et al. [18] used a parametric grammar and an MRF for each rule to delineate floors and windows in a single view. Ripperda and Brenner [25] also use a grammar and reversible jump MCMC for inferring facade descriptions from LIDAR data in the form of a derivation tree. Lafarge et al. [19] use very simple parametric models to infer building descriptions from a digital surface model. A Bayesian formulation enables the propagation of structure between adjacent primitives and evaluates the likelihood of larger assemblies.

A key difference between the above methods and ours is that the former employ a domain dependent grammar, while we use a more generic grammar which is based on simple geomertic relations. Moreover, our grammar formulation allows for exact inference in polynomial time using dependency parsing, while the above methods resort to approximate inference, mainly using MCMC.

3. Generic Tree Representation for Buildings

A common way to represent buildings is to use planes endowed with geometric features [20, 28, 15, 27] or use more complex primitives manually designed to represent building parts [9, 3, 19]. The former approaches are generic and hence have the advantage of broad applicability, while the latter are potentially more robust to missing parts and noise but require more domain knowledge.

In order to combine the best of both worlds – generic representation which reasons over larger building parts – we choose to make *volumes*, which are implicit in plane-based representations, explicit and use them as primitives. At a coarse level, a building can be viewed as a tree of volumetric parts that lie on or next to each other and are covered by planar patches such as roofs and roof parts (Fig. 2). Such hierarchical description is a natural way to describe architecture, because a building has usually a main body to which smaller building parts are attached. Thus, the building parse tree can be formed by making each volume a child of either the ground or of a neighboring volume of larger prominence in the building description. The children of the volumes in this representation are roofs and roof parts of the buildings which are described as planar patches.

While parsing the point cloud of a city, we aim to extract a set of building trees, as described above, designating all buildings in the city. To combine all the building parses in a single tree we introduce two supernodes designating "building" and "non-building". Then, all building trees would be rooted at the "building" node, while all remaining primitives, extracted from the input point cloud, should be attached to the "non-building" node. Thus, a single parse tree of a city represents parses of individual trees as well as classification of the point cloud into "buildings" and "non-buildings" – all primitives descending from the "building" supernode should represent city architecture.

An appropriate formalism to describe the above notion

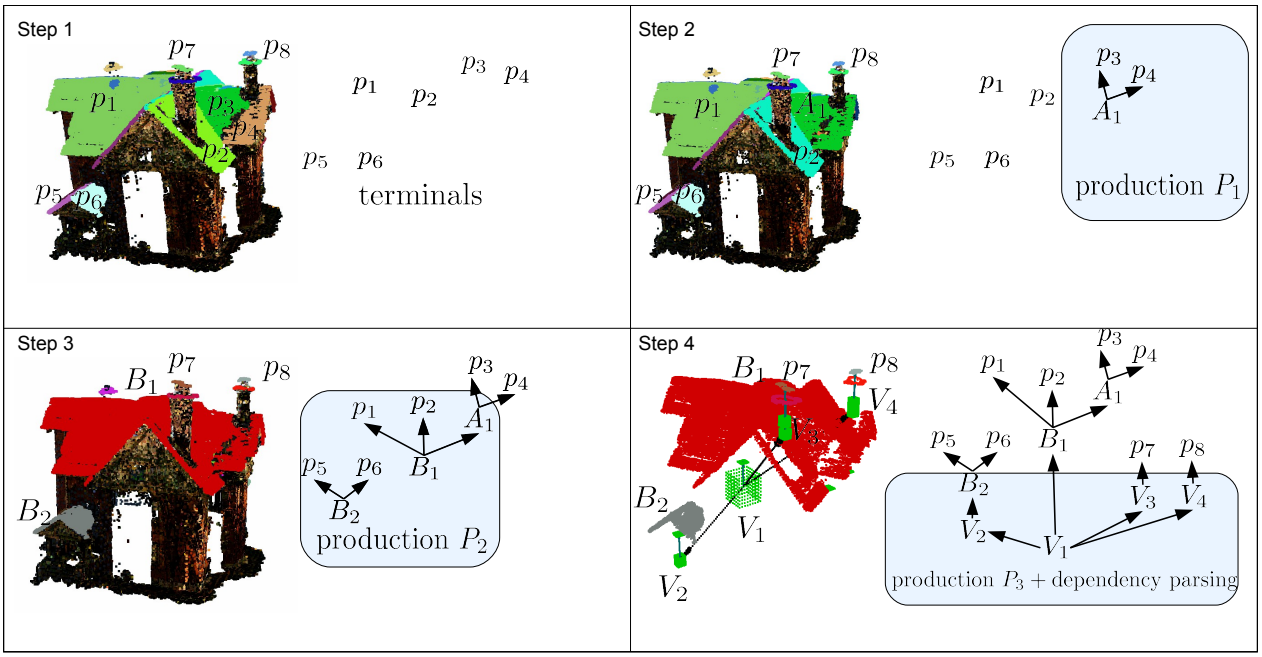

Figure 2. Parsing of a building and the resulting tree. Step 1: the non-vertical planar patches of the input building are terminal nodes. Step 2: merging of two neighboring co-planar roof patches through production P_1. Step 3: merging of several touching planar patches through production P_2. Step 4: each roof has as a parent a volume (green boxes). Using dependency parsing we obtain a full parse tree of the building representing its natural structure – the chimneys and the smaller building being children of the main building.

is a grammar $G = (\mathcal{V}_T, \mathcal{V}_N, \mathcal{P}, \mathcal{S}, w)$ consisting of a set of terminals $\mathcal{V}_T$, a set of non-terminals $\mathcal{V}_N$, a production set $\mathcal{P}$, two supernodes $\mathcal{S} = \{S_B, S_{NB}\}$, designating "building" and "non-building", and a scoring function $w(\cdot)$ for the instantiation of each production. Next, we provide a description of the grammar.

Grammar Symbols The terminal nodes $\mathcal{V}_T = \{p_1 \dots p_n\}$ are planar patches extracted from the point cloud of the detected building. The non-terminals $\mathcal{V}_N = \mathcal{A} \cup \mathcal{K} \cup \mathcal{S}$ are of two types. $\mathcal{A}$ designates roof components constructed as groups of planar patches (see nodes A_1 and B_1 in Fig. 2). Further, the volumes enclosed by the roofs construct the last set $\mathcal{K}$, the non-terminals. Finally, we have two supernodes $\mathcal{S} = \{S_B, S_{NB}\}$.

In order to instantiate $\mathcal{A}$ and $\mathcal{B}$, we need to detect planar patches in a point cloud. This choice is motivated by the fact that man-made structures, and buildings in particular, are well described using polyhedral representations, whose building blocks are planar patches. These primitives are extracted using RANSAC-based [12] plane detection in the point cloud. The points on each detected plane are split into connected components and the largest component is retained as a planar patch. Two points are connected if they lie within 0.5 m. We sample plane hypotheses from the point cloud until we either explain 80% of the points or we have reached a limit on the number of sampled planes, which in our experiments is set to 2000.

P_1: co-planar patch collection inference $p_1 \dots p_k \rightarrow A$, $A \in \mathcal{A}$, $p_i \in \mathcal{V}_T$ if the set $\{p_1 \dots p_k\}$ is a maximal connected component in the graph with vertices $\mathcal{V}_T$ and edges $\{(p_i, p_j) \mid touching(p_i, p_j) \wedge coplanar(p_i, p_j)\}$
P_2: connected patch collection inference $A_1 \dots A_k \rightarrow B$ for $A_i, B \in \mathcal{A} \cup \mathcal{V}_T$ if the set $\{A_1 \dots A_k\}$ is a maximal connected component in the graph with vertices $\mathcal{A} \cup \mathcal{V}_T$ and edges $\{(A_i, A_j) \mid touching(A_i, A_j) \wedge \neg coplanar(p_i, p_j)\}$
P_3: volume inference $B \rightarrow V$ for $B \in \mathcal{A} \cup \mathcal{V}_T$ and $V \in \mathcal{V}$
P_4: volume inference $V_1 \dots V_k \rightarrow V$ for $V, V_i \in \mathcal{V}$
P_5: class inference $V \rightarrow S$ for $V_i \in \mathcal{V}$ and $S \in \mathcal{S}$ one of two supernodes

Table 1. Productions for building parsing (see Sec. 3.1 for details).

3.1. Productions

The design of the productions $\mathcal{P}$ should enable the inference of a hierarchy among roof parts and a set of volumes on which the roofs and facades are anchored as well as a set of child-parent relations between the volumes and roof parts. This set of relations should be hierarchical in nature and represent a natural topological structure of the building in 3D. For this purpose we introduce a set of productions $\mathcal{P} = \{P_1, \dots P_5\}$ as defined in Table 1.

Roofs and roof parts The first production P_1 finds large planar patches by merging neighboring co-planar patches. This production rectifies mistakes of plane detection – it groups patches which lie on the same plane but were grouped in different planes in the plane detection phase. The application of this production is exemplified in Fig. 2 by the grouping of patches p_3 and p_4 into A_1.

The second production P_2 further groups planar patches, which are neighboring but not coplanar, in connected planar components; this step usually detects roofs. In Fig. 2 this results in grouping patches p_1, p_2, and A_1 into tree node B_1.

To define these productions we use two predicates: $touching(p_i, p_j)$ is true if the two patches have common boundary; $coplanar(p_i, p_j)$ holds if the angle between the planes of p_i and p_j is smaller than 10 degrees. Then, to assess the applicability of a production $r : p_1 \dots p_k \rightarrow A$, we assign a score

$$P_1 : \text{score}(r) = \begin{cases} 0 & \text{if } touching(p_i, p_j) \quad \forall i \neq j \\ -\infty & \text{otherwise} \end{cases} \tag{1}$$

$$P_2 : \text{score}(r) = \begin{cases} 0 & \text{if } coplanar(p_i, p_j) \quad \forall i \neq j \\ -\infty & \text{otherwise} \end{cases} \tag{2}$$

Volumes The reasoning about volumes is performed in the next two productions. Volumes are generated by applying production P_3, which attaches volumes to roofs. The production P_4 is used to establish a hierarchy among the volumes. There are two challenges – the hierarchy among volumes is ambiguous and one needs to ensure that two connected volumes are of the same class.

The ambiguity of volume hierarchy is exemplified in Fig. 2, where the building has two main roofs and hence two main volumes. The one is V_1 below the large building roof B_1, while the second V_2 is smaller and positioned in the front under roof B_2 (see resulting tree in step 4 of the figure). Hence, the aim of this production would be to infer that the smaller volume V_2 is a child of the larger V_1. We expect a child volume to be a volume neighboring its parent whose size is smaller than the size of the parent node. Hence, in case nodes C and C' are touching, $C, C' \in \mathcal{V}_\mathcal{T} \cup \mathcal{V}_\mathcal{N}$, we can capture the above notion using a normalized difference of the areas of the nodes:

$$a(C, C') = \frac{\text{area}(C) - \text{area}(C')}{\max(\text{area}(C), \text{area}(C'))} \tag{3}$$

where area$(\cdot)$ is the area of the projection of the node onto the ground plane.

Moreover, one needs to ensure that two volumes are connected only if they are of the same class. In other words, a volume of a building can be a parent of another volume of a building but not of a volume designating a non-building structure, e. g. car, tree, power line, etc. To define this, we use a volume classification score $c(V)$, defined in eq. (5) in the next subsection. This score is positive if V is part of a building and negative otherwise (see next section for definition). Then, the score of this criterion for a production $r : V_1 \dots V_k \rightarrow V$ can be formalized as:

$$P_4 : \text{score}(r) = \begin{cases} \sum_i \left(\theta_1 a(V_i, V) + \theta_2 c(V_i) c(V)\right) \\ \qquad \text{if } touching(V_i, V) \text{ for all } i \\ -\infty \quad \text{otherwise} \end{cases} \tag{4}$$

where θ_1 and θ_2 are parameters weighting the contribution of the individual terms.

Classification of volumes In order to perform building detection while parsing, we use the two supernodes $\mathcal{S} = \{S_B, S_{NB}\}$ which are ancestors of all nodes in the city parse tree except for the root and designate the class of all their descendants. This is implemented using production P_5 which should ensure that building volumes are descendants of S_B, while all non-building nodes are descendants of S_{NB}. This is achieved by scoring each instantiation of P_5 by how likely a volume is a building or non-building. This score is based on a set of features, which can be extracted for a volume V from the set of all non-vertical planar patches $P_V = \{p_1, \dots, p_k\}$, which enclose this volume. Such planar patches usually represent the upper surface of the volume and in the case of buildings are roofs. If we define by $\hat{p}_i$ the projection of p_i onto the ground plane, $\hat{P}_V = \{\hat{p}_1, \dots, \hat{p}_k\}$, then the features are defined as follows:

f_1 : *Elevation* computed as the mean value of the difference of the elevation of each point included in the planar patches in P_V and the mean ground plane elevation. This feature captures the fact that buildings tend to be quite elevated with respect to the ground.

f_2 : *Distance to the nearest ground point*, computed as the median distance of the centroid of the points contained in $\hat{P}_V$ to the 30 closest ground points. Most of the buildings have centroids farther away from the ground, while the centroids of other objects such as cars and trees are much closer to the ground.

f_3 : *Convexity of the upper volume surface*, defined as the mean convexity of all $\hat{p}_i$. The convexity of a single planar patch $\hat{p}_i$ is the ratio between the area of the patch and the area of its convex hull. This feature helps to discriminate trees from man-made objects, since the planar patches detected on trees tend to be non-convex.

f_4 : *Scatter* is the mean value of the scatter of all points enclosed in the volume, where the scatter of a point is the ratio of the third to second eigenvalue of the scatter matrix computed at that point. This features is helpful in detecting trees, which tend to have high scatter value, while all planar structures have low scatter value.

f_5 : Mean *area* and *aspect ratio* of all $\hat{p}_i$. Buildings have much larger surfaces than trees, cars, etc. Additionally, other elevated structures such as power lines and cranes are thinner than buildings and hence have smaller aspect ratio.

f_6 : *Degree of enclosure by empty space*, defined as the portion of the boundary of p_i not touching other planar patches. This is motivated by the observation that many small individual objects such as cars, trees, people, poles are not adjacent to any other planar patches, while buildings parts tend to be tightly enclosed by planar patches.

f_7 : *Fitting error*, as the average fitting error of all points in p_i obtained from plane detection. This features discriminates natural structures, which are not always well described by planes, from man-made structures.

In order to compute the above features, we need to detect the ground plane. Assuming that the $z-$axis of the point cloud coordinate system is aligned with the world $z-$axis, we compute a histogram of the z coordinates of all points, where the bins have size $3m$. Since the ground plane contains large objects such as streets, sidewalks, etc., and all objects touch the ground, we can assume that the bin with the largest value contains the ground points. The ground plane is defined to have z coordinate equal to the elevation of that bin and is parallel to the x and y axes.

We use the above features for a volume V to define a feature vector $f(V) = (\cdots f_i \cdots f_i f_j \cdots)^T$, $i, j \in \{1, \dots 7\}$. Then we use a score $c(\cdot)$ for V being a building:

$$c(V) = b^T f(V) \tag{5}$$

We use not only the features but also their products in order to capture feature correlations. Note that this is the score we use in eq. (4).

Then we can score an instantiation $r : V \rightarrow S$ of P_5:

$$P_5 : \text{score}(r) = \begin{cases} \theta_3 c(V) & \text{if } S = S_B \\ -\theta_3 c(V) & \text{if } S = S_{NB} \end{cases} \tag{6}$$

where θ_3 is a weight for this score type. Learning of these parameters is described in the next section.

The proposed grammar is generic and simple – it consists of basic geometric entities such as planar patches and volumes, and geometric relations between them based on relative position and size. Moreover, as we will see below, it is also a semantic description since each node, being a volume or surface, can be easily interpreted as a meaningful building part. These descriptions are structured in a way that reflects a natural building interpretation into parts (see final result in Fig. 2).

3.2. Parsing

Inference The goal of the parsing is to infer a parse tree T of a building from a set of input planar patches $A = \{p_1 \dots p_m\}$ such that the overall score of the parse is maximized:

$$T^* = \arg\max_T \text{score}(A, T) \tag{7}$$

The parse tree is constructed by applying a sequence of productions $R = \{r_1 \dots r_n\}$ from Table 1. As a result, the parse score can be computed in terms of the scores of the applied productions R:

$$\text{score}(A, T) = \text{score}(A, R) = \sum_i \text{score}(r_i) \tag{8}$$

The parsing procedure is simplified by the observation that the first three productions are deterministic by design. Production P_1 requires the creation of the neighborhood graph containing all initially detected planar patches and removes from this graph edges connecting non-coplanar patches. Each maximal connected component in this graph leads to an application of this production with score 1. We proceed similarly for production P_2, where the neighborhood graph contains not only the initial planar patches but also the nodes generated from P_1. Finally, production P_3 attaches a volume to each obtained planar component. We will denote by $V = \{V_1 \dots V_k\}$ the set of the generated volumes.

The last two productions P_4 and P_5 cannot be applied deterministically. They create a hierarchy among the volumes and classify them as "buildings" or "non-buildings" by creating a tree T over $V \cup \mathcal{S}$, where $\mathcal{S}$ are the two supernodes. Note that each of those two productions is a sum of terms involving at most two nodes (see eq. (4) and eq. (6)). Hence, the parse T can be interpreted as a spanning tree in the graph with nodes $V \cup \mathcal{S}$ with a score defined using the production scores:

$$\begin{aligned} &\text{score}(A, R) = \text{score}(A, T; \theta) = \\ &\sum_{(V_i, V_j) \in T} (\theta_1 a(V_j, V_i) + \theta_2 c(V_i) c(V_j)) \\ &+ \sum_{(S_B, V_i) \in T} \theta_3 c(V_i) + \sum_{(S_{NB}, V_i) \in T} \theta_3 (-c(V_i)) \end{aligned} \tag{9}$$

where T is the set of edges in the parse tree and $\theta = (\theta_1, \theta_2, \theta_3)^T$.

Maximizing the above cost is equivalent to finding a Maximum Spanning Tree (MST) in a directed weighted graph over $V \cup \mathcal{S}$. The MST is computed using the Chi-Liu-Edmunds (CLE) algorithm [5]. It proceeds by greedily trying to construct a directed MST. If this results in a cycle, CLE contracts the cycle to a single new node and redefines the edges adjacent to this node such that the MST in the new graph has the same weight as the one in the original graph. After a MST has been computed in the contracted graph by recursively invoking the algorithm, the tree is expanded to a MST in the original graph (see Fig. 2).

Input: Directed graph $G = (V, E, w)$ with
$w : E \rightarrow \mathbb{R}$ is an edge weight function.

$T =$ **CLE**(G, w)
For each node $v \in V$ select a predecessor $a(v)$ such that $(a(v), v)$ has the highest score among all arcs enetering v.
$G_a \leftarrow (V, E_a)$, $E_a \leftarrow \{(a(v), v) | v \in V\}$.
If G_a a tree **then return** G_a.
else find a cycle C in G_a.
Contract: Generate a graph G_c from G by contracting C on a new node c:
Remove C from G_c and add a new node c.
For each $v \in V \setminus C$
Add edge (c, v) to G_c with
$b(v) \leftarrow \arg\max_{v' \in C} w(v', v)$
$w(c, v) \leftarrow w(b(v), v)$
Add edge (v, c) to G_c with
$b(v) \leftarrow \arg\max_{v' \in C}(w(v, v') - w(a(v'), v')$
$w(v, c) \leftarrow w(v, b(v)) - w(a(b(v)), b(v)) + w(C)$
where $w(C)$ is the sum of all edges in cycle C
$T_c \leftarrow$ **CLE**(G_c, w);
Expand MST T_c over G_a to an MST T over G:
Add C to T_c.
Add all outgoing edges from C to $V \setminus C$.
Find predecessor v' of c in T_c and v in C with $b(v) = v'$.
Connect cycle to T_c by adding edge (v', v).
Break cycle by removing $(a(v), v)$.
return T

Table 2. Pseudocode for the Chi-Liu-Edmonds algorithm [5].

This type of parsing is called dependency parsing and has been extensively applied in Natural Language Processing, for example for relations extraction among others [8, 22]. A major advantage of the proposed parsing is its tractability. The complexity of deterministic plane parsing is in $O(|A|)$ and the CLE algorithm computes an optimal parse in $O(|V|^3)$. The overall time complexity is cubic in the number of input planes, which in our experiments is at most several hundred. Note, that there are more efficient algorithms for computing MST, which have complexity quadratic in terms of the graph size [22].

Learning The productions of the building grammar, as introduced in Sec. 3.1, are parametrized by paramters b and θ in eq. (5) and eq. (9).

The score from eq. (5) should be high if a volume belongs to a building and low otherwise. We choose to train a binary linear SVM on the feature vector $f(V)$ and use the implementation of [10].

The second set of parameters θ weights the contribution of the different production scores during the inference described in eq. (9). After estimating b, we learn θ using an averaged structured perceptron [6, 22]. The structured perceptron exploits the fact that the cost function is linear in the parameters: score$(A, T) = \theta^T \psi(A, T)$, where $\psi(A, T)$ is a vector containing the data terms for all edges in T from

Input: set of parses $\{A_i, T_i\}$, for $i = 1 \ldots N$
number of iterations L
Initialize: $\theta_0 \leftarrow 0$, $k \leftarrow 0$
For $j = 1 \ldots L$
For $i = 1 \ldots N$
$T^\star \leftarrow \arg\max_T$ score$(A_i, T; \theta_k)$
$\theta_{k+1} \leftarrow \theta_k + \psi(A_i, T_i) - \psi(A_i, T^\star)$
$k \leftarrow k + 1$
end
end
$\theta \leftarrow \frac{1}{LN} \sum_{s=1}^{LN} \theta_s$

Table 3. Pseudocode for the averaged structured perceptron learning for the dependency parsing.

eq. (9). Then, the learning algorithm sequentially visits each training example and updates the parameters such that the above loss for the example is locally optimized (see Table 3). Since we have few parameters, we need only a few epochs and set $L = 5$.

4. Experimental Results

We evaluate our approach by computing the building detection and parsing accuracies. We use a large scale dataset [1] covering downtown Ottawa and containing approximately 1 billion points collected by several passes of airborne and terrestrial range scanners. Due to its large size, the point cloud is partitioned into approximately 350 blocks and we run our algorithm per block. Note, that plane extraction is the bottleneck of the algorithm, since we need to sample plane candidates from the entire point cloud. After this step, however, we obtain between 50 and 300 planar patches per block. As a result, the dependency parsing can be computed in less than 1 sec per block on a 3.50 Ghz processor, which shows the efficiency of the proposed inference.

For building detection, we have manually labeled all buildings in 87 blocks which cover an area of approximately $3km^2$. These buildings include tall and complex structures from downtown Ottawa as well as low residential houses in areas covered with trees. In addition, we have manually parsed 18 buildings in 9 blocks.

To evaluate the building detection performance, we trained the parsing algorithm over the 9 blocks. For comparison, we train a binary SVM over the patches in those 9 blocks as described in the previous section. The remaining 78 labeled blocks are used for testing. As evaluation metric, we use the percentage of planar patches which have been classified correctly as "building" or "non-building", here called accuracy. The accuracy of the parsing algorithm is 89.3%, while the SVM achieves 87.9%. To analyze the accuracy gains of the algorithm, in Fig. 3, we show the accuracy over small patches and its change if we add larger patches. Among the small patches, such as awnings, chim-

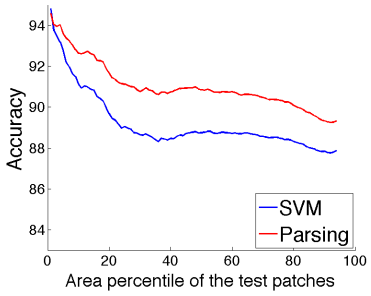

Figure 3. Accuracy of parsing and SVM patch classification. We sort the patches in increasing order according to their area. The graph shows classification accuracy for the p-first patches in the above order, where p is shown as the percentile of all patches.

neys, etc., the parsing performs better than the SVM, while for the large patches, both parsing and SVM perform comparably. Note that although such small structures cover relatively small area of the city, they represent important building parts, which can be best detected exploiting the context provided by the parsing.

For evaluation of the accuracy of recovering dependencies, we used 3 random blocks from the 9 labeled for training and the remaining for testing (we perform 3 such splits). We achieve accuracy of 76.2% in correctly detecting the parent of each planar patch. As shown in Fig. 4, the root of a building parse is naturally the main building part, while chimneys, awnings, etc., are leaves. Note that particularly complex roofs (examples 1 and 2) are decomposed into their constituent planar patches.

5. Conclusion

We propose a simple and generic hierarchical representation for building detection and parsing. The main advantage of our approach is efficiency: our parsing algorithm is polynomial in the number of extracted planar patches, allowing efficient parsing at a city scale. Using geometric and shape features of building parts, we show how to learn models to parse buildings in a way consistent with human interpretation and potentially useful for automated search and retrieval.

References

[1] The Wright State 100 dataset. Available after registration at http://www.daytaohio.com/Wright_State100.php.

[2] A. Berg, F. Grabler, and J. Malik. Parsing images of architectural scenes. In *ICCV*, 2007.

[3] M. Brédif, D. Boldo, M. Pierrot Deseilligny, and H. Maitre. 3D building reconstruction with parametric roof superstructures. In *Int. Conf. on Image Processing*, 2007.

[4] H. Cantzler, R. Fisher, and M. Devy. Quality enhancement of reconstructed 3d models using coplanarity and constraints. In *DAGM*, 2002.

[5] Y. Chu and T. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14, 1965.

[6] M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

[7] R. Collins, C. Jaynes, Y. Cheng, X. Wang, F. Stolle, E. Riseman, and A. Hanson. The ascender system: Automated site modeling from multiple aerial images. *CVIU*, 72(2), 1998.

[8] A. Culotta and J. Sorensen. Dependency tree kernels for relations extraction. In *ACL*, 2004.

[9] A. Dick, P. Torr, and R. Cipolla. Modelling and interpretation of architecture from several images. *IJCV*, 60(2), 2004.

[10] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9, 2008.

[11] A. Fischer, T. Kolbe, F. Lang, A. Cremers, W. Forstner, L. Pluemer, and V. Steinhage. Extracting buildings from aerial images using hierarchical aggregation in 2d and 3d. *CVIU*, 72(2), 1998.

[12] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6), 1981.

[13] Y. Guo, H. Sawhney, R. Kumar, and S. Hsu. Learning-based building outline detection from multiple aerial images. In *CVPR*, 2001.

[14] N. Haala, S. Becker, and M. Kada. Cell decomposition for the generation of building models at multiple scales. In *Photogrammetric Computer Vision*, 2006.

[15] F. Han and S. Zhu. Bottom-up/top-down image parsing by attribute graph grammar. In *ICCV*, 2005.

[16] A. Huertas and R. Nevatia. Detecting buildings in aerial images. *Computer Vision, Graphics, and Image Processing*, 41(2), 1988.

[17] Z. Kim and R. Nevatia. Automatic description of complex buildings from multiple images. *CVIU*, 96(1), 2004.

[18] P. Koutsourakis, L. Simon, O. Teboul, G. Tziritas, and N. Paragios. Single view reconstruction using shape grammars for urban environments. In *ICCV*, 2009.

[19] F. Lafarge, X. Descombes, J. Zerubia, and M. Pierrot Deseilligny. Structural approach for building reconstruction from a single DSM. *PAMI*, 2009.

[20] H. Maas and G. Vosselman. Two algorithms for extracting building models from raw laser altimetry data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3), 1999.

[21] B. Matei, H. Sawhney, S. Samarasekera, J. Kim, and R. Kumar. Building segmentation for densely built urban regions using aerial lidar data. In *CVPR*, 2008.

[22] R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *ACL*, 2005.

[23] D. McKeown, W. Harvey, and J. McDermott. Rule based interpretation of aerial imagery. *PAMI*, 7(5), 1985.

[24] C. Poullis and S. You. Automatic reconstruction of cities from remote sensor data. In *CVPR*, 2009.

[25] N. Ripperda and C. Brener. Reconstruction of Facade Structures Using a Formal Grammar and RjMCMC. In *DAGM*, 2006.

[26] J. Shufelt. Performance evaluation and analysis of monocular building extraction from aerial imagery. *PAMI*, 21(4), 1999.

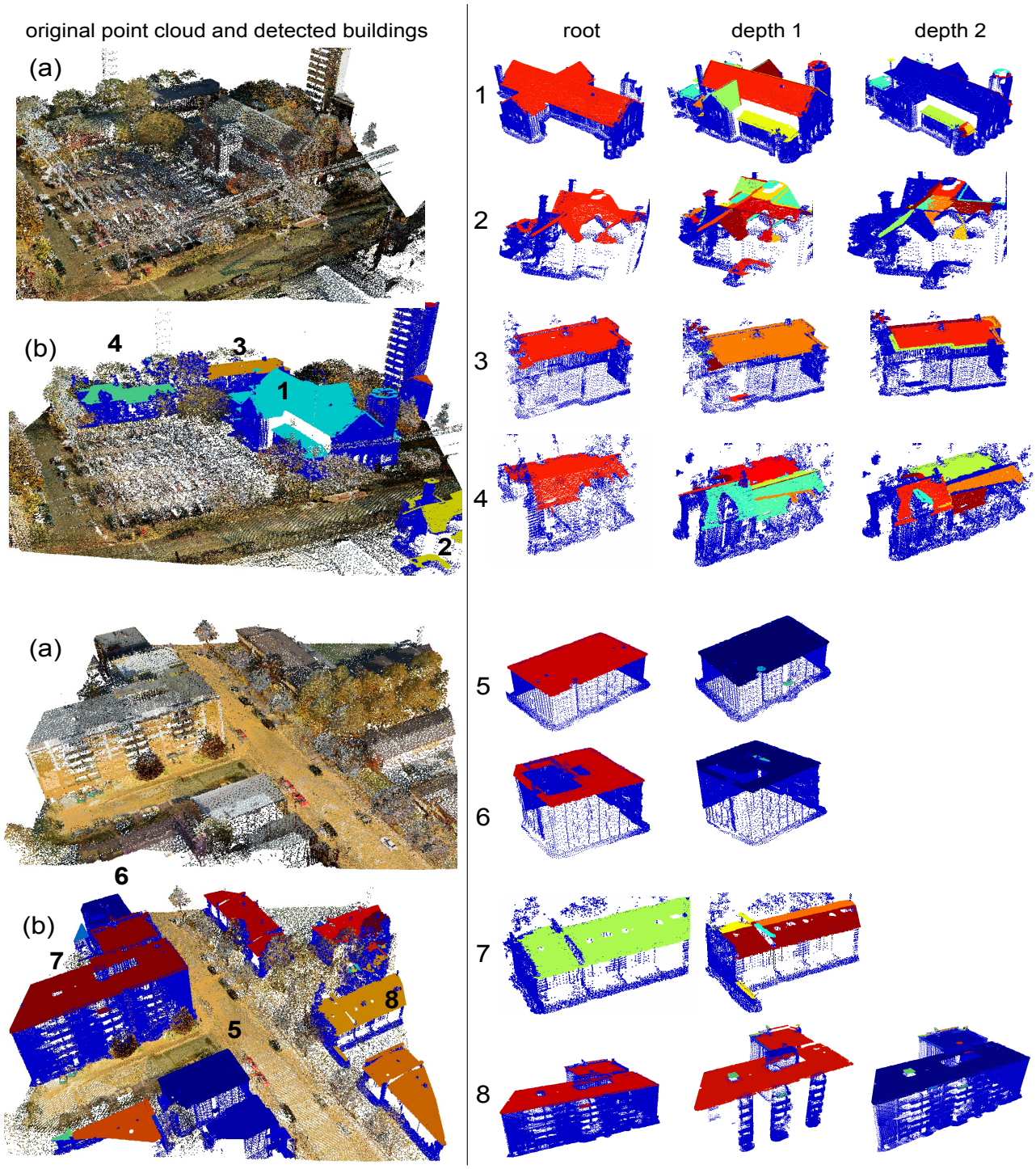

Figure 4. We present building detection and parsing results on two blocks of the point cloud. For each block, we show on the left side (a) the original point cloud and (b) the detected buildings overlaid on the point cloud, each individual building having a different color. For some of the detected buildings, we show the parses, where in each column we show all the nodes at a particular depth (nodes are colored randomly, remaining building patches and facades are colored dark blue).

[27] V. Verma, R. Kumar, and S. Hsu. 3D building detection and modeling from aerial lidar data. In *CVPR*, 2006.

[28] T. Werner and A. Zisserman. New techniques for automated architectural reconstruction from photographs. In *ECCV*, 2002.