Segmentation and Unsupervised Part-based Discovery of Repetitive Objects

Rudolph Triebel Jiwon Shin Roland Siegwart
Autonomous Systems Lab, ETH Zurich
Tannenstrasse 3, 8092 Zurich, Switzerland
email: {rudolph.triebel,jiwon.shin}@mavt.ethz.ch, rsiegwart@ethz.ch

***Abstract*— In this paper, we present an unsupervised technique to segment and detect objects in indoor environments. The main idea of this work is to identify object instances whenever there is evidence for at least one other occurence of an object of the same kind. In contrast to former approaches, we do not assume any given segmentation of the data, but instead estimate the segmentation and the existence of object instances concurrently. We apply graph-based clustering in feature and in geometric space to presegmented input data. Each segment is treated as a potential object part, and the inter-dependence of object labels assigned to part clusters are modeled using a Conditional Random Field (CRF) named the "parts graph". Another CRF is then applied to the scene graph to smooth the class labels using the distributions obtained from the parts graph. First results on indoor 3D laser range data are evaluated and presented.**

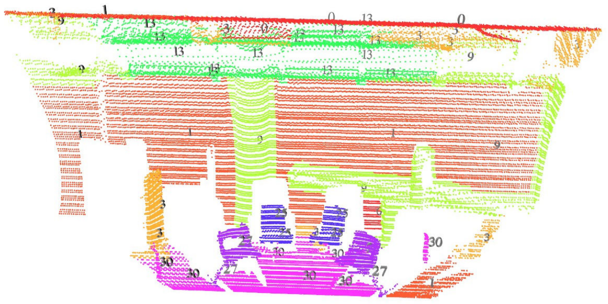

Fig. 1: 3D range scan of an indoor scene. Two different types of chairs are detected by exploiting the fact that particular constellations between back rests and seats occur more than once in the scene.

I. INTRODUCTION

The ability for a robot to learn and discover objects without any human guidance enhances its autonomy and makes it more independent. Such a robot requires no prior training and can more easily adapt to new, unknown environments. It is also able to autonomously draw conclusions about the structure of its environment. This functionality is useful when robots operate fully autonomously without human interaction. But also when robots live with humans, a high-level semantic analysis of the environment helps the robot to communicate with a human. As an example, a robot which can detect similarities of objects encountered in the environment, no longer requires a human to first label all occurences of objects in a previously acquired data set. Instead, it may ask the human, "I discovered several instances of something that looks like an interesting object. What is the name of the object?" In this paper, we take a first step in this direction.

We propose an approach to segment and discover objects of multiple occurrences without supervision, where an object is defined as a constellation of object parts. We segment input point clouds and treat each segment as an instance of a potential object part. In our work, object parts are determined by grouping similar segments together using clustering in a predefined feature space. In addition, the segments are clustered in the geometric space and the number of resulting connected components is used as an upper bound on the number of potential object classes. Then, two major reasonings are used to determine a class label for each segment: First, different object parts that often occur close to each other are more likely to correspond to the same object class. For example, the fact that a back rest of a chair and a chair seat are frequently observed in close vicinity to each other rises the evidence that there is an object class for which more than one instance appears in the scene (see Fig. 1). Second, an instance of an object part may appear to correspond to some object class, but given its physical context it is more likely to be part of another class. For example, a segment may appear to be a chair leg, but if surrounded by table parts it is more likely to be a part of a table. Both ideas are implemented using probabilistic reasoning based on Conditional Random Fields (CRFs) [1].

II. RELATED WORK

Most work on repetition detection has been in the field of image analysis. Detection of regularly repeating patterns has been the focus of many researchers [2, 3] with some recent work by Loy and Eklundh [4] on grouping of features based on symmetry, and by Wenzel *et al.* [5] on using symmetry to detect repetitive structures in facade images. Zeng and van Gool [6] employ point-wise repetition to improve segmentation results using mutual information. In 3D, discovery and utilization of repetition has been adressed in computer aided design and other synthetic models [7, 8, 9]. The work of Bokeloh *et al.* [10] is more closely related to this work. The authors proposed an algorithm for detecting structural redundancy by matching symmetric constellations of feature lines. In terms of repetition detection, the main challenge of our work lies in the lack of a repetition unit as we do not assume any regularity or symmetry of the repetition pattern.

In this work, we employ clustering to group similar segments in feature and geometric space. Clustering has received a considerable amount of attention by machine learning and pattern recognition communities. Some classic methods such as the expectation-maximization algorithm and k-means clustering assume that data can be modeled by a simple distribution, while other methods such as agglomerative clustering are sensitive to noise and outliers. To overcome these challenges, alternative approaches have been proposed. Ng, Jordan, and Weiss [11] presented a spectral clustering algorithm, which uses eigenvectors of the data matrix to group points together, and demonstrate how well the algorithm clusters even challenging data. Another work of interest is affinity propogation proposed by Frey and Dueck [12]. Affinity propagation clusters data by finding a subset of exemplars, which are cluster centers selected from data. This method avoids the pitfalls of bad initialization and does not require the number of clusters to be prespecified. In this work, we use affinity propagation to cluster segments in feature space.

Conditional Random Fields (CRFs) [1] are discriminative models, that have also been applied to object recognition problems [13, 14]. Notably, Quattoni, Collins, and Darrell [15] presented a part-based approach for object class recognition using a CRF. We also take a parts-based approach for objects. Ma and Grimson [16] proposed a coupled CRF to allow for interaction between contour and texture in image data. While our work does not explicitly use coupled CRFs, the interaction between part labels and class labels play a critical role in the success. The main idea in our work which has not been adressed previously is the use of Conditional Random Fields without any training set. Instead, we infer from clustering results the possible object labels.

In unsupervised object detection, several authors have proposed adaptation of text analysis methods in image analysis. For example, Liu and Chen [17] proposed a modified probablilistic latent semantic analysis (pLSA) method to detect foreground objects from images. Sivic *et al.* [18] compare pLSA and Latent Dirichlet Allocation (LDA) to discover object categories from sets of images. Also, Endres *et al.* [19], use LDA to discover object classes from range data without supervision. While this approach can classify objects of multiple classes, it assumes that a ground plane and walls are extracted a priori and the objects are spatially disconnected. In our work, we do not make such assumptions. We consider every segment as a potential object part and test them to determine if they belong to an object. Lastly, this work is similar to our previous work [20], which also discovers objects without supervision, but it does not explicitly label the segments as we do in this work.

III. SEGMENTATION AND CLUSTERING

Object detection using unsupervised learning is significantly different from using supervised learning. In fact, objects can not be *detected*; that is, it is not possible to identify some part of the input data by matching it against an instance of a previously known object class as there is no such known object class. Instead, objects can only be *discovered* by hypothesizing the existence of an object based on some sort of repetition or pattern found in the data. Thus, to discover objects, we need to find similarities in the data. We do this by extracting features (see Sec. III-B) and comparing them by a clustering algorithm in the feature space, which is described in Sec. III-C.

```
Algorithm: SSCGF
data  : Point Cloud $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$
input :
  • Segmentation parameters $\kappa$ and $\tau$
  • Cluster parameters $\vartheta_f$ and $\vartheta_g$
output:
  • low-level segmentation $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_M\}$, $\mathbf{s}_i \subset \mathcal{P}$
  • feature-space clusters of segments $\mathcal{F}_1, \ldots, \mathcal{F}_C$
  • geometric clusters of segments $\mathcal{G}_1, \ldots, \mathcal{G}_K$
  • class label distributions $\mathbf{d}_1, \ldots, \mathbf{d}_M$, $\mathbf{d}_i \in [0,1]^K$
procedure:
$\mathcal{S} \leftarrow$ SuperPixelSegmentation($\mathcal{P}, \kappa, \tau$)
$\mathbf{f}_1, \ldots, \mathbf{f}_M \leftarrow$ FeatureExtraction($\mathcal{S}$)
$\mathcal{F}_1, \ldots, \mathcal{F}_C \leftarrow$ ClusterInFeatureSpace($\mathcal{S}, \vartheta_f, \{\mathbf{f}_i\}$)
$\mathcal{G}_1, \ldots, \mathcal{G}_K \leftarrow$ ClusterInGeometricSpace($\mathcal{S}, \vartheta_g$)
$\mathfrak{P} \leftarrow$ MakePartsGraph($\{\mathcal{F}_i\}, \{\mathcal{G}_i\}, \mathcal{S}$)
$\mathfrak{P} \leftarrow$ SmoothPartsGraph($\mathfrak{P}, K$)
$\mathfrak{S} \leftarrow$ MakeSceneGraph($\mathfrak{P}, \{\mathcal{G}_i\}, \mathcal{S}$)
$\mathfrak{S} \leftarrow$ SmoothSceneGraph($\mathfrak{S}, K$)
$\mathbf{d}_1, \ldots, \mathbf{d}_M \leftarrow$ ReadFromGraphNodes($\mathfrak{S}$)
```

Alg. 1: Segmentation and smoothed clustering in geometric and in feature space (SSCGF). Note that the number of segments M, as well as the numbers C and K of clusters are computed inside the particular subroutines (see text).

Another problem which arises here is that the *segmentation* of the data is unknown, i.e. we do not know where the boundaries of the objects are. The segmentation problem is tightly bound to the detection problem because a perfect segmentation would make object detection very easy – a simple comparison of the segmented object instances would suffice. To tackle this problem we perform three steps in our algorithm: first, we apply a low-level segmentation as described in Sec. III-A. Then, we obtain a coarser data segmentation by clustering in the geometric space as presented in Sec. III-C. Finally, we obtain a further improved segmentation by reasoning on parts of objects that occur in a similar constellation in different instances of the scene. The details of this are described in Sec. IV and Sec. V. An overview of the entire algorithm is shown in Alg. 1.

A. Low-level Segmentation

The first step in our algorithm is the segmentation of the data using the graph-based segmentation algorithm of Felzenszwalb and Huttenlocher [21], adapted to range image data. In the modified algorithm, we create a graph $\mathfrak{G} = \{\mathcal{V}, \mathcal{E}\}$ of vertices $\mathcal{V}$ and edges $\mathcal{E}$, where each point $\mathbf{p}$ in a given point cloud $\mathcal{P}$

corresponds to a vertex and an edge connects adjacent points. Here, adjacency of two points is determined from a triangular mesh built on the point cloud. Every edge $(\mathbf{p}_i, \mathbf{p}_j)$ has an associated weight w_{ij}, which is equal to the dissimilarity Δ of the connected points $\mathbf{p}_i$ and $\mathbf{p}_j$. In the case of a camera image, this can be the difference of the pixel intensities; in our case, we define $\Delta(\mathbf{p}_i, \mathbf{p}_j)$ as the dot product of the normal vectors $\mathbf{n}_i$ and $\mathbf{n}_j$ computed at $\mathbf{p}_i$ and $\mathbf{p}_j$. This yields for smooth surfaces, e.g. a plane or a sphere, being grouped as one segment, while surfaces with sharp edges, e.g. between two sides of a box, are grouped into two segments.

The algorithm begins with each vertex in its own segment. The edges are processed by increasing weights, and the two segments $\mathbf{s}_i$ and $\mathbf{s}_j$ connected by a given edge are merged whenever

$$w_{ij} \leq \min\left(d(\mathbf{s}_i) + \frac{\kappa}{|\mathbf{s}_i|}, d(\mathbf{s}_j) + \frac{\kappa}{|\mathbf{s}_j|}\right),$$

where $d(\mathbf{s})$ is the *internal difference* function defined by the maximal edge weight of all edges in the minimum spanning tree of the segment $\mathbf{s} \subseteq \mathcal{V}$, and κ is a consistency parameter that influences the granularity of the segmentation: a low value of κ requires segments to be more consistent and thus produces more but smaller segments.

In addition to introducing a 3D extension, we make other modifications to the original algorithm. First, as the normal at points with an insufficient number of neighboring points is ill-defined, we do not force every point to be in a segment. No vertices are generated in the graph for these points, and thus no segments are created containing them. Second, as a post-processing step, we remove segments that contain fewer points than a given minimal value τ, which are often caused by sensor imperfections or occlusions. In our experiments, a good choice of the segmentation parameters turned out to be $\kappa = 9$ and $\tau = 100$.

B. Feature Extraction

As shape descriptors, we use spin images [22], shape distributions [23], and shape factors [24]. A *spin image* for a given point $\mathbf{p}$ with normal vector $\mathbf{n}$ is defined as a 2D histogram H oriented along the line l through $\mathbf{p}$ with direction $\mathbf{n}$. Each bin of H counts the points with a certain distance to l and the plane through $\mathbf{p}$ with normal vector $\mathbf{n}$. For the spin image descriptor of a segment $\mathbf{s}$, we form vectors $\mathbf{h}_i$ of stacked lines of the histograms H_i for all points $\mathbf{p}_i \in \mathbf{s}$ and compute the average $\mathbf{h}^s$ over all $\mathbf{h}_i$.

A *shape distribution* is defined as a histogram of values of a predefined function $f : \mathcal{P}^r \to \mathbb{R}$, where r is the arity of f and is usually a value between 1 and 4. In our implementation, we use two binary functions $f_d(\mathbf{p}_i, \mathbf{p}_j)$ and $f_a(\mathbf{p}_i, \mathbf{p}_j)$, namely the Euclidean distance between $\mathbf{p}_i$ and $\mathbf{p}_j$ and the dissimilarity $\Delta(\mathbf{p}_i, \mathbf{p}_j)$ as defined above. The resulting histogram vectors $\mathbf{h}^d$ and $\mathbf{h}^a$ are computed by evaluating f_d and f_a on all pairs of points in a segment $\mathbf{s}$. To make the feature vectors invariant with respect to the sample density, we normalize the histograms $\mathbf{h}^d$ and $\mathbf{h}^a$ by the total number of bin entries. As we consider scale as a feature of an object, we do not perform normalization with respect to the maximum distance encountered in a segment.

Lastly, we compute *shape factors* per segment, i.e. the normalized eigenvalues of the covariance matrix C_i of all points in segment $\mathbf{s}_i$, collected in a vector $\mathbf{h}^f$. For each individual descriptor, we compute a PCA to reduce the dimensionality, and the results are combined into a feature vector $\mathbf{f}$.

C. Clustering in Feature Space

To find similar segments, we apply a clustering algorithm in the feature space. The number of existing clustering algorithms is large, and they include agglomerative clustering, k-means clustering [25], mean-shift estimation [26], spectral clustering [27, 11], and, more recently, affinity propagation (AP) [12]. We explored some of these clustering methods and decided for AP clustering because of its robustness and its ability to estimate the number of clusters implicitly. The basic principle is to determine *exemplars* out of all given points that are well-suited to explain the remaining data points. The application of the algorithm to our case is sketched as follows:

First a similarity value ς_{ij} is computed for all pairs $(\mathbf{f}_i, \mathbf{f}_j)$ of feature vectors. In our implementation, we use the negative squared Euclidean distance between $\mathbf{f}_i$ and $\mathbf{f}_j$. Then, in an iterative manner, two functions, namely the *responsibility* $r(i, j)$ and the *availability* $a(i, j)$ are computed for each vector pair, where r expresses how well-suited $\mathbf{f}_j$ is to serve as an exemplar for $\mathbf{f}_i$ and a expresses how appropriate it would be for $\mathbf{f}_i$ if $\mathbf{f}_j$ were its exemplar. These functions are defined as

$$r(i, j) = \varsigma_{ij} - \max_{j' \, s.t. \, j' \neq j} \left\{a(i, j') + \varsigma_{ij'}\right\} \quad (1)$$

$$a(i, j) = \min\left\{0, r(j, j) + \sum_{i' \, s.t. \, i' \notin \{i, j\}} \max\{0, r(i', j)\}\right\}, \quad (2)$$

where Eq. (2) is applied only if $i \neq j$. Initially, $a(i, j)$ is set to zero for all i and j. For the special case of "self-availability" the rule

$$a(i, i) = \sum_{i' \, s.t. \, i' \neq i} \max\{0, r(i', i)\} \quad (3)$$

is used. In each iteration, responsibilities and availabilities are computed and then, for each $\mathbf{f}_i$, an $\mathbf{f}_j$ is determined so that the sum $a(i, j) + r(i, j)$ is maximized. If the resulting j equals i, then $\mathbf{f}_i$ is identified as an exemplar, otherwise $\mathbf{f}_j$ serves as an exemplar for $\mathbf{f}_i$. The stopping criterion of the iteration is met when the assignments of points to exemplars do not change over a fixed number of iterations or a given number of maximum iterations is reached. The only parameter ϑ_f of the algorithm is the *self-similarity*, which can be specified either for each data point individually or commonly for all data points. This value influences the number of resulting clusters: a high value results in more clusters, a low value in fewer clusters. In our experiments, a good value turned out to be -0.2, specified equally for all data points.

As a result, we obtain C clusters $\mathcal{F}_1, \ldots, \mathcal{F}_C$ of similar segment instances, where each cluster defines a potential object part.

D. Clustering in Geometric Space

From the clustering in feature space we obtain a grouping of segments into object parts. However, we also want to reason on the object level, where objects are considered to consist of several parts. To accomplish this, we also perform a clustering in the geometric space where segments are represented by their center of gravity (COG). Unfortunately, affinity propagation (AP) is not a good choice to perform the clustering in the geometric space because it produces "star-like" clusters, i.e. all points in a cluster are connected directly to the cluster exemplar. This restricts the type of objects that can be detected, as for many objects such an exemplar part can not be found. Also, in AP clustering, the distances between points are not explicitly bounded, which often results in counter-intuitive clustering results.

Therefore, we apply a different, much simpler strategy to cluster the segments. We define a distance threshold ϑ_g and connect only those pairs of segments $(\mathbf{s}_i, \mathbf{s}_j)$ with an edge, for which the COGs are closer to each other than ϑ_g. As a result, all connected components will be farther away from each other than ϑ_g, which rises the evidence that they correspond to different object instances. This will be of importance later on.

IV. Scene Graph and Parts Graph

One important aspect of the work presented here is the reasoning about object *instances* solely based on the extraction of potential object *parts*, represented as segments. The challenge here is to find a proper definition of an object class as we do not know of how many parts an object consists and whether or not all of its parts are visible. In addition, the number of observed objects is unknown – we only know the number of object parts. Two intuitions and one assumption help us to reason on parts to discover objects: First, we exploit the fact that segments which occur physically close to each other are more likely to correspond to the same object. Second, segments of one type which occur often in the vicinity to segments of another type give evidence that several instances from the same object class exist. Referring back to the introductory example, we can say: if we find many backrests of a chair that are all close to chair seats, then there is probably an object class which consists of at least these two parts. Furthermore, we work under the assumption that the number of possible object classes is bounded by the number of connected components in the graph which results from clustering the geometric space using the distance threshold ϑ_g. We call this the *scene graph* and give details in Sec. IV-A. Our assumption implicitly states that two objects are always supposed to be farther away from each other than ϑ_g. This may seem very restrictive, but in fact, it limits the applicability of the algorithm not as much as it appears. The reason is that the number of connected components only bounds the number of object *classes*, not the number of actual object *instances*. Thus, even if two different objects are closer to each other than ϑ_g, there are usually enough other connected components with at least two objects of the same class, so that the number of connected components exceeds the number of object classes.

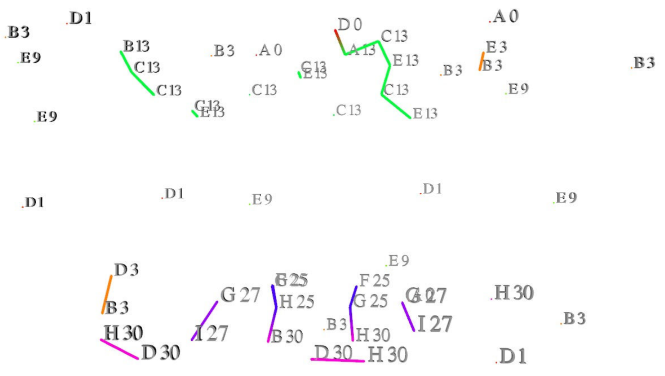

Fig. 2: Example of the scene graph obtained from the range scan shown in Fig. 1. Notice that segments with the label G are given the class labels 25 and 27, depending on their neighboring segments. The figure shows the scene graph after smoothing as described in Sec. V-B.

To analyse the multiple occurence of segments representing the same object part in constellation with segments that represent a different part, we define another, simpler graph structure named the *parts graph*. The nodes in the parts graph correspond to clusters in feature space as described in Sec. III-C and the edges represent connections in the scene graph. Details on the parts graph are given in Sec. IV-B.

A. The Scene Graph

After clustering the segments in the geometric space, we obtain K subsets of $\mathcal{S}$, namely $\mathcal{G}_1, \ldots, \mathcal{G}_K$. From our assumption, the number of potential object classes is bounded by K. To encode the fact that neighboring segments are more likely to correspond to the same object class, we define the scene graph $\mathfrak{S}$, which consists of the node set $\mathcal{V}_s = \mathcal{S}$ and the edge set $\mathcal{E}_s = \{(\mathbf{s}_i, \mathbf{s}_j) \mid \exists \mathcal{G}_i : (\mathbf{s}_i, \mathbf{s}_j) \subset \mathcal{G}_i\}$. Thus, $\mathcal{E}_s$ consists of all connections between segments that are closer to each other than ϑ_g. Furthermore, to each node in scene graph we assign a *class label* $y \in \{1, \ldots, K\}$. And finally, as each node $\mathbf{s}$ of $\mathfrak{S}$ corresponds to an instance of an object part, we can associate it with a *part label* $x \in \{1, \ldots, C\}$. At first sight, x should be fixed to the index of the feature space cluster $\mathcal{F}$ to which $\mathbf{s}$ belongs. However, as we formulate our problem in a probabilistic framework, we say that this index is only the *most likely* part label for $\mathbf{s}$ and all others are still possible. Details of this are explained in Sec. V.

An example of a scene graph is shown in Fig. 2. Here, part labels are represented as capital letters and class labels as numbers. We can see that many of the connected components only consist of one segment, as the segments are mostly far apart from each other. Also observe that there are segments with the same part label x, but with different class labels y. This stems from the fact that they occur in different contexts.

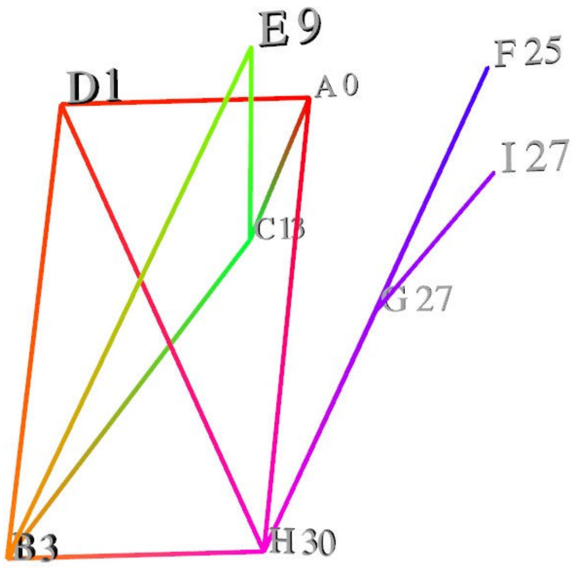

Fig. 3: Example of the parts graph obtained from the scan shown in Fig. 1. The positions of the nodes are determined at random, as only the topology of the graph is important. Again, part labels are represented as capital letters and class labels as numbers.

B. The Parts Graph

As mentioned before, apart from the class labels of specific segment instances, we also want to reason on the interaction among different object parts in general. We do this using the parts graph $\mathfrak{P}$. The node set $\mathcal{V}_p$ of $\mathfrak{P}$ corresponds to all clusters $\mathcal{F}_1, \ldots, \mathcal{F}_C$ of the feature space, and the set of edges $\mathcal{E}_p$ is defined as

$$\mathcal{E}_p = \{(\mathcal{F}_i, \mathcal{F}_j) \mid i \neq j, \exists (\mathbf{s}_k, \mathbf{s}_l) \in \mathcal{E}_s : \mathbf{s}_k \in \mathcal{F}_i \wedge \mathbf{s}_l \in \mathcal{F}_j\}.$$

This means, whenever there are two segments connected in the scene graph, there is a connection between the corresponding segment types in the parts graph. As in the scene graph, each node of the parts graph has an assigned part label x and a class label y, although with a slightly different interpretation: An assignment of a class label y to a node $\mathbf{s}$ in $\mathfrak{S}$ means that the particular segment instance $\mathbf{s}$ is most probably part of an object of class y, whereas the same assignment to a node $\mathcal{F}$ in $\mathfrak{P}$ means that in general segments that are elements of the cluster $\mathcal{F}$ are primarily more likely to be of class y. In contrast to the scene graph, each node of $\mathfrak{P}$ has a unique most-likely part label x, because these are directly determined by the feature space clustering.

In Fig. 3, we see an example of a parts graph. The positions of the nodes have been defined randomly as only the topology of the graph is important. Note that this concrete example of a parts graph is non-planar, although there is an equivalent planar representation. In general however, parts graphs may not be representable as planar graphs, for example in the case of a full connectivity.

V. SMOOTHING

So far, we described the construction of the two graph structures "scene graph" and "parts graph", but we did not specify how these are used to determine class labels for the segments. We do this using probabilistic reasoning: the nodes in both graphs are interpreted as random variables and the edges are used to model conditional dependencies between adjacent nodes. For the scene graph, this means that the class label y_i of a given segment $\mathbf{s}_i$ not only depends on the local evidence of the node i, i.e. the features $\mathbf{f}_i$ extracted from $\mathbf{s}_i$, but also on the class labels y_j of all neighbors $\mathbf{s}_j$ in the scene graph. Intuitively, a strong evidence for $\mathbf{s}_i$ to belong to a certain class may "outvote" the weaker evidence for $\mathbf{s}_j$ being of a different class. Similarly, a class label y_i for a given node $\mathcal{F}_i$ in the parts graph may be so strong that it *propagates* to the class labels of the neighbors of $\mathcal{F}_i$. The reasoning here is that if a certain type of segment $\mathcal{F}_i$ is very likely to be of a given class, then all segment types that occur frequently in the vicinity of $\mathcal{F}_i$ are also more likely to be of the same class. Applying this strategy to a graph leads to a *smoothed* graph because it tends to remove sudden changes of class labels between adjacent graph nodes.

In our implementation, we use Conditional Random Fields (CRFs) [1] to perform the smoothing. Our CRF models the conditional distribution

$$p(\mathbf{y} \mid \mathbf{f}) = \frac{1}{Z(\mathbf{f})} \prod_{\mathcal{V}} \varphi(\mathbf{f}_i, y_i) \prod_{(i,j)\in\mathcal{E}} \psi(\mathbf{f}_i, \mathbf{f}_j, y_i, y_j), \tag{4}$$

where $Z(\mathbf{f}) = \sum_{\mathbf{y}'} \prod_{i=1}^{N} \varphi(\mathbf{f}_i, y_i') \prod_{(i,j)\in\mathcal{E}} \psi(\mathbf{f}_i, \mathbf{f}_j, y_i', y_j')$ is the *partition function*, $\mathcal{V}$ is either $\mathcal{V}_s$ or $\mathcal{V}_p$ and $\mathcal{E}$ is $\mathcal{E}_s$ or $\mathcal{E}_p$. The dinstinction between the scene graph and the parts graph is made using different definitions for the *node potential* φ and the *edge potential* ψ. This will be described next.

However, before we proceed with the definition of the potentials, we mention some aspects in our formulation of the CRFs that are slightly different from others found in the literature. First, we recall that node and edge potentials are usually defined using the log-linear model so that, for the case of the node potential

$$\log \varphi(\mathbf{f}_i, y_i) = w_n \cdot f_n(\mathbf{f}_i, y_i), \tag{5}$$

where w_n is a weight vector and f_n a feature function, which is high if $\mathbf{f}_i$ and y_i in some sense match well. For example, it can be defined as the outcome of a local classification. For our case of an unsupervised setting, we let f_n be the conditional probability $p(y_i \mid \mathbf{f}_i)$, i.e. a scalar value that is high if y_i matches well to $\mathbf{f}_i$. The same is also done for the edge potentials. As a result, the feature functions range between 0 and 1. This simplifies the weighting between node and edge potentials and turns the weight vectors w_n and w_e into scalars as well. In contrast to supervised learning with CRFs, we can not learn the node and edge weights w_n and w_e, as there is no training data available. Instead, we have to determine them manually. We do this using an appropriate evaluation measure on a validation set. This is described in Sec. VI.

A. Smoothing the Parts Graph

After creating the parts graph, the next step in our algorithm is to run the inference step using the CRF that corresponds to $\mathfrak{P}$. First, we note that the node features of this CRF are not equal to the feature vectors $\mathbf{f}_1, \ldots, \mathbf{f}_M$ extracted from the segments, because a node in $\mathfrak{P}$ actually represents a cluster in feature space and not a single segment. Instead, we define the node features in $\mathfrak{P}$ to be the mean $\bar{\mathbf{f}}$ of all feature vectors inside the corresponding cluster.

Furthermore, we observe that the dependence between feature vectors $\mathbf{f}$ and labels y is only implicit as we cannot model it directly. We can, however, model conditional probabilities between segment labels x and class labels y, as well as between feature vectors $\mathbf{f}$ and segment labels x. Following the definition of the node feature of $\mathfrak{P}$ as a conditional probability, we have

$$p(y_i \mid \bar{\mathbf{f}}_i) = \sum_{x=1}^{C} p(y_i \mid x) p(x \mid \bar{\mathbf{f}}_i), \tag{6}$$

where we assume a conditional independence of the labels and the features given the segment types. To obtain the two terms in the sum, we proceed as follows.

The class label posterior $p(y_i \mid x)$ is computed using Bayes' rule with the assumption of a uniform prior of the labels:

$$p(y_i \mid x) = \frac{p(x \mid y_i) p(y_i)}{\sum_{y'} p(x \mid y'_i) p(y'_i)} = \frac{p(x \mid y_i)}{\sum_{y'} p(x \mid y'_i)} \tag{7}$$

The class likelihoods $p(x \mid y_i)$ are determined by counting the occurence of segments of type x inside the geometric cluster y_i and dividing by the number of all segments in y_i. The posteriors $p(y_i \mid x)$ can be computed at creation time of $\mathfrak{P}$ and collected in a $K \times C$ matrix, as they do not depend on the node features.

For the segment type posterior $p(x \mid \bar{\mathbf{f}}_i)$, we perform a nearest-neighbor search in feature space inside a sphere of radius ρ around $\bar{\mathbf{f}}_i$. Denoting the number of feature vectors in the sphere with type x as ν_x and the number of all elements in the sphere as ν, we approximate $p(x \mid \bar{\mathbf{f}}_i)$ with ν_x/ν.

Similar to Eqn. (6), we define the edge feature of $\mathfrak{P}$ as

$$p(y_i, y_j \mid \bar{\mathbf{f}}_i, \bar{\mathbf{f}}_j) = \sum_{x_i=1}^{C} \sum_{x_j=1}^{C} p(y_i, y_j \mid x_i, x_j) p(x_i, x_j \mid \bar{\mathbf{f}}_i, \bar{\mathbf{f}}_j). \tag{8}$$

As is common in literature related to CRFs, the edge features are designed to be zero whenever the labels y_i and y_j of the adjacent nodes are different ("generalized Potts model", see [28, 29]). The rationale of this is that only edges between equally labeled nodes should propagate the belief between the nodes. Thus, the first term in the sum of Eqn. (8) can be simplified to $p(y_{ij} \mid x_i, x_j)$, where y_{ij} is the common label of the nodes. We can compute this expression by counting the occurences of edges between x_i and x_j in cluster y_{ij} and applying Bayes rule as in Eq. (7).

For the second term in the sum of Eqn. (8), we apply the formulation

$$\begin{aligned} p(x_i, x_j \mid \bar{\mathbf{f}}_i, \bar{\mathbf{f}}_j) &= p(x_i \mid \bar{\mathbf{f}}_i, \bar{\mathbf{f}}_j) p(x_j \mid \bar{\mathbf{f}}_i, \bar{\mathbf{f}}_j) \\ &= p(x_i \mid \bar{\mathbf{f}}_i) p(x_j \mid \bar{\mathbf{f}}_j), \end{aligned} \tag{9}$$

which results from the conditional independence assumptions on x_i and x_j, and from those for x_i and $\bar{\mathbf{f}}_j$, as well as x_j and $\bar{\mathbf{f}}_i$. The resulting terms $p(x_i \mid \bar{\mathbf{f}}_i)$ and $p(x_j \mid \bar{\mathbf{f}}_j)$ are again computed using nearest-neighbor.

Once the edge and node potentials are defined, we can do inference on the parts graph. We do this using max-product loopy belief propagation. This is an approximate algorithm that returns labels $\mathbf{y}$ that maximize the conditional probability given in Eqn. (4). However, it also returns distributions over the class labels at each node. These will be used later.

B. Smoothing the Scene Graph

From the output of the inference step run on $\mathfrak{P}$, we obtain at each node of $\mathfrak{P}$ a distribution over class labels y. As the nodes of $\mathfrak{P}$ uniquely represent the segment types x, we can use that information to read the probability $p(y \mid x)$ directly from the parts graph. Thus, when creating the scene graph, we can again determine values for $p(y \mid x)$, as we did this for the parts graph, with the difference that now $p(y \mid x)$ also reveals the conditional dependencies between labels of segment types x, that have been observed in a close distance. This means, that we again compute a matrix for $p(y \mid x)$, but now we simply read the class label distributions off the nodes of $\mathfrak{P}$, as they result from belief propagation. In accordance to the node feature of $\mathfrak{P}$, we define the node feature of $\mathfrak{S}$ as the conditional probability

$$p(y_i \mid \mathbf{f}_i) = \sum_{x=1}^{C} p(y_i \mid x) p(x \mid \mathbf{f}_i). \tag{10}$$

Note that now the feature vectors $\mathbf{f}_i$ correspond to those that are actually extracted for each segment, i.e. at each node of $\mathfrak{S}$. As before, the term $p(x \mid \mathbf{f}_i)$ is computed with a nearest-neighbor search in feature space.

Finally, we define the edge feature of $\mathfrak{S}$ as

$$p(y_i, y_j \mid \mathbf{f}_i, \mathbf{f}_j) = \sum_{x_i=1}^{C} \sum_{x_j=1}^{C} p(y_{ij} \mid x_i, x_j) p(x_i, x_j \mid \mathbf{f}_i, \mathbf{f}_j), \tag{11}$$

where we denote the common class label with y_{ij} as above and the right term in the sum is computed according to Eqn. (9). Unfortunately, the computation of $p(y_{ij} \mid x_i, x_j)$ is not straightfoward, as we can not simply read this value from the edges of $\mathfrak{P}$. Instead, we use the following strategy: The only interesting case is when the class labels of x_i and x_j are equal. Thus, we can interpret the common label y_{ij} as an *edge label* that is determined by one of the adjacent node labels. In a sense, this expresses which of the nodes was responsible for the common edge label y_{ij}. We can formulate that using a binary variable c that is true if node x_i is responsible for y_{ij} and false otherwise. Using this, we can estimate $p(y_{ij} \mid x_i, x_j)$ by marginalizing out over c:

$$\begin{aligned} p(y_{ij} \mid x_i, x_j) &= \sum_{c} p(y_{ij} \mid x_i, x_j, c) p(c, \mid x_i, x_j) \\ &= p(y_{ij} \mid x_i, x_j, c) p(c) + p(y_{ij} \mid x_i, x_j, \neg c) p(\neg c) \\ &= 0.5 * p(y_{ij} \mid x_i) + 0.5 * p(y_{ij} \mid x_j) \end{aligned} \tag{12}$$

Here, we assumed that c is independent on x_i and x_j and that its prior probability is 0.5. Using Eqn. (12), we can compute $p(y_{ij} \mid x_i, x_j)$ by reading the values for $p(y_{ij} \mid x_i)$ and $p(y_{ij} \mid x_j)$ from the parts graph. As for the parts graph, we use max-product loopy belief propagation for the inference in the scene graph.

C. Obtaining the Class Label Distributions

Once the inference step on the scene graph is performed, we read the class label distributions $\mathbf{d}_1, \ldots, \mathbf{d}_M$ from all nodes of $\mathfrak{S}$. This is possible, as mentioned before, because belief propagation stores these distributions for each node. For the final discovery result, we report the most likely class label at each node, but it is important to note that the distributions may be useful for later reference, for example in an online application, where several data sets are acquired subsequently and the discovery of similar objects is done across the data sets.

VI. EXPERIMENTAL RESULTS

We tested the algorithm on data acquired from real-world scenes using a nodding SICK laser scanner with a horizontal opening angle of 100 degrees and a nodding range of 90 degrees. Each set was captured at a horizontal resolution of 0.25 degrees and a vertical resolution of 0.2 degrees. We evaluated 50 data sets from four different rooms, each room containing some number of chairs, trash cans, flip charts, plants, etc. Objects were placed up to 90 degrees of rotation from each other. Most scenes contained two or three objects of the same type, but some scenes contained up to four objects of three different kinds.

A. Qualitative Evaluation

In addition to the result shown in Fig. 1, Fig. 4 shows some more results of our object dicovery algorithm. All points that belong to the same object are depicted with the same color, and the numbers represent the class label to which each segment belongs. For instance, the scene in Fig. 1 contains four chairs of two different kinds, and they are correctly labeled as 25 (blue) and 27 (violet). Furthermore, we see that also most of the background segments, e.g. on the floor, ceiling and walls, have plausible labels. For many of them, only the local class label evidence was relevant, as they are not connected in the scene graph.

B. Quantitative Evaluation

In contrast to supervised learning algorithms, the performance of an unsupervised object discovery method is difficult to evaluate, as there is no real ground truth. The major problem here is that humans tend to be focused and might miss similarities in the data that are irrelevant for them. However, a good way to still evaluate unsupervised object discovery methods has been recently proposed by Tuytelaars *et al.* [30]. This method uses the *conditional entropy* of the "ground truth" class labels $\mathbf{y}^*$ given the class labels $\mathbf{y}$ that resulted from the discovery algorithm. Applied to our case, the conditional entropy is computed as

$$H(Y^* \mid Y) = \sum_{\sigma(y)=1}^{K^*} p(\sigma(y)) \sum_{y^*=1}^{K^*} p(y^* \mid \sigma(y)) \log \frac{1}{p(y^* \mid \sigma(y))},$$

where K^* is the number of ground truth classes and σ is an *oracle mapping* from the set of discovered classes to the set of ground truth classes, which is determined from a *tuning set* (for details see [30]). Intuitively, $H(Y^* \mid Y)$ determines the number of ground truth labels a segment can have once its discovered object label is known. The smaller this number is, the better is the result of the discovery algorithm. In the best case it is zero, which means that each discovered class label directly implies a ground truth label.

We created hand labeled ground truth data consisting of the five classes "ceiling", "floor", "wall", "chair", and "other", and evaluated the performance of our algorithm for different values of the parameters ϑ_g, ρ, and the node and edge weights (w_n, w_e). The results are shown in Fig. 5. Two conclusions can be drawn from these graphs: First, with values around 1, the results are very good compared to the results of similar algorithms described in [30]. And second, our algorithm is relatvely robust against small changes in the choice of the parameters, especially for the distance threshold ϑ_g, which is responsible for the clustering in geometric space.

VII. CONCLUSION AND OUTLOOK

We presented a fully unsupervised approach to segment 3D range scan data and to discover objects of a similar type that occur more than once in the scene. Our approach uses the only assumption that the number of actually existing object classes is not higher than the number of connected components in the scene, which holds in most cases. We applied probablistic reasoning based on Conditional Random Fields to model conditional dependencies of object part labels that are close to each other. In experiments on real data we showed that our algorithm is able to discover objects such as chairs in an indoor environment. In the future, we plan to use this technique in an online framework where the evidence of existing object classes is accumulated over time and across different data sets.

ACKNOWLEDGMENTS

This work was partially supported by the EC in the project robots@home under contract number FP6-IST-045350, in the project BACS under contract number FP6-IST-027140, and in the project EUROPA under contract number EUROPA-FP7-231888.

References

[1] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of International Conference on Machine Learning*, 2001.

[2] T. Tuytelaars, A. Turina, and L. van Gool, "Noncombinatorial detection of regular repetitions under perspective skew," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 418–432, 2003.

[3] Y. Liu, R. T. Collins, and Y. Tsin, "A computational model for periodic pattern perception based on frieze and wallpaper groups," *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 26, pp. 354–371, 2004.

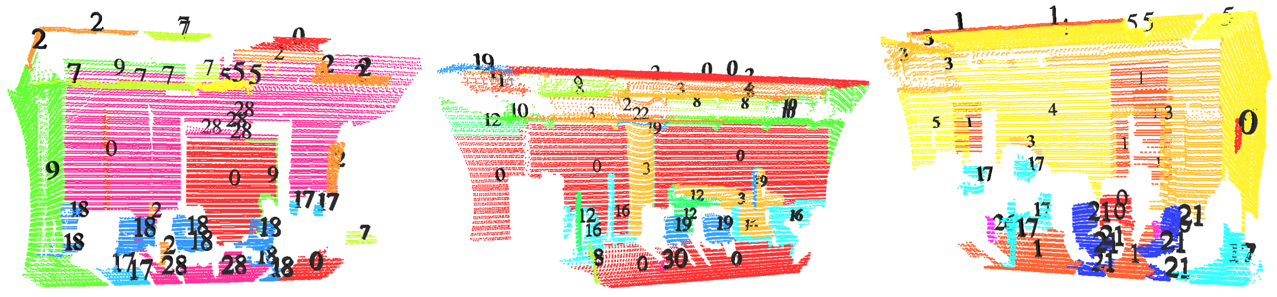

Fig. 4: Three example scenes observed with our nodding SICK laser scanner. Objects that are discovered through the algorithm are colored, where all points which belong to the same object class are assigned to one color. For visualization when viewed in greyscale, the discovered class labels are also shown.

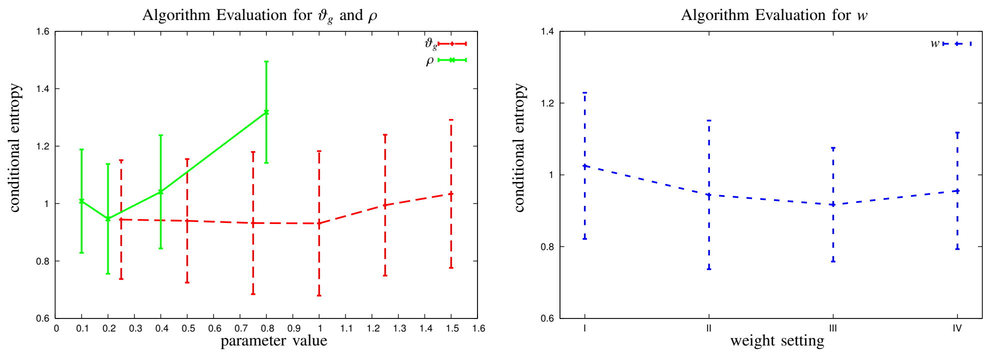

Fig. 5: Evaluation of our discovery algorithm using the conditional entropy with its standard deviation over 50 data sets. Left: Results for different values of ϑ_g and ρ. As it can be seen, ρ has a stronger influence on the results as ϑ_g. Right: Results for different settings of the node and edge weights (w_n, w_e). Here, 'I' corresponds to the pair $(0.5, 1.5)$, 'II' to $(1, 1)$, 'III' to $(1.5, 0.5)$ and 'IV' to $(2.0, 0)$. The best result is obtained for $w_n = 1.5$, $w_e = 0.5$.

[4] G. Loy and J.-O. Eklundh, "Detecting symmetry and symmetric constellations of features," in *ECCV*, 2006, pp. 508–521.

[5] S. Wenzel, M. Drauschke, and W. Förstner, "Detection of repeated structures in facade images," *Pattern Recognition and Image Analysis*, vol. 18, no. 3, pp. 406–411, 2008.

[6] G. Zeng and L. van Gool, "Multi-label image segmentation via point-wise repetition," in *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, Alaska, USA, June 2008.

[7] D. Shikhare, S. Bhakar, and S. Mudur, "Compression of large 3d engineering models using automatic discovery of repeating geometric features," in *Vision, Modeling, and Visualization*, 2001.

[8] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser, "A planar-reflective symmetry transform for 3d shapes," in *SIGGRAPH*. New York, NY, USA: ACM, 2006, pp. 549–559.

[9] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas, "Discovering structural regularity in 3d geometry," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–11, 2008.

[10] M. Bokeloh, A. Berner, M. Wand, H. Seidel, and A. Schilling, "Symmetry detection using feature lines," *Computer Graphics Forum (Proceedings of Eurographics)*, vol. 28, no. 2, pp. 697–706(10), April 2009.

[11] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Adv. in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., 2002.

[12] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, February 2007.

[13] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *IEEE Conf. on Computer Vision*, 2007.

[14] C. Galleguillos, A. Rabinovich, and S. Belongie, "Object categorization using co-occurrence, location and appearance," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[15] A. Quattoni, M. Collins, and T. Darrell, "Conditional random fields for object recognition," in *Adv. in Neural Inform. Proc. Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., 2005, pp. 1097–1104.

[16] X. Ma and W. E. L. Grimson, "Learning coupled conditional random field for image decomposition with application on object categorization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[17] D. Liu and T. Chen, "Semantic-shift for unsupervised object detection," in *Proc. of the Conf. on Comp. Vision and Pattern Rec. Workshop*, 2006.

[18] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman, "Discovering object categories in image collections," in *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005.

[19] F. Endres, C. Plagemann, C. Stachniss, and W. Burgard, "Unsupervised discovery of object classes from range data using latent dirichlet allocation," in *Proc. of Robotics: Science and Systems*, 2009.

[20] J. Shin, R. Triebel, and R. Siegwart, "Unsupervised discovery of repetitive objects," in *IEEE Int. Conf. Robotics and Automation*, 2010.

[21] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[22] A. Johnson, "Spin-images: A representation for 3-d surface matching," Ph.D. dissertation, Robotics Institute, Carnegie Mellon Univ., 1997.

[23] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Trans. on Graphics*, vol. 21, no. 4, pp. 807–832, 2002.

[24] C.-F. Westin, S. Peled, H. Gudbjartsson, R. Kikinis, and F. A. Jolesz, "Geometrical diffusion measures for MRI from tensor basis analysis," in *ISMRM '97*, Vancouver Canada, April 1997, p. 1742.

[25] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[26] D. Comaniciu, P. Meer, and S. Member, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[27] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[28] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of markov random fields for segmentation of 3d scan data," in *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005, pp. 169–176.

[29] R. B. Potts, "Some generalized order-disorder transformations," *Proc. Cambridge Phil Soc.*, vol. 48, 1952.

[30] T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine, "Unsupervised object discovery: A comparison," *Int. Journal of Computer Vision*, vol. 88, no. 2, pp. 284–302, 2009.