Semantic Segmentation of Modular Furniture

Tobias Pohlen Ishrat Badami Markus Mathias Bastian Leibe
Visual Computing Institute, Computer Vision Group
RWTH Aachen University
tobias.pohlen@rwth-aachen.de {badami, mathias, leibe}@vision.rwth-aachen.de

Abstract

This paper proposes an approach for the semantic segmentation and structural parsing of modular furniture items, such as cabinets, wardrobes, and bookshelves, into so called interaction elements. Such a segmentation into functional units is challenging not only due to the visual similarity of the different elements but also because of their often uniformly colored and low-texture appearance. Our method addresses these challenges by merging structural and appearance likelihoods of each element and jointly optimizing over shape, relative location, and class labels using Markov Chain Monte Carlo (MCMC) sampling. We propose a novel concept called rectangle coverings *which provides a tight bound on the number of structural elements and hence narrows down the search space. We evaluate our approach's performance on a novel dataset of furniture items and demonstrate its applicability in practice.*

1. Introduction

Visual understanding of indoor scenes is a crucial task for many applications in robotics. Most notably, the interaction of autonomous robots with complex indoor scenes requires an accurate semantic labeling. While there have been many approaches for a coarse labeling of entire indoor scenes (*e.g*. [6, 15, 29, 12]), only very little work goes beyond an analysis on the object level [22].

In this paper, we want to address the logical next step and provide a more detailed analysis of object semantics (see Figure 2). We take advantage of the fact that many furniture items exhibit a modular internal structure: They are composed of a set of common *interaction elements* (doors, drawers, or shelves) in a variable yet constrained (modular) spatial configuration.

We present a first approach for the semantic segmentation of such modular furniture that is applicable to bookshelves, wardrobes, office cabinets, *etc*. Given the front face of a furniture item, our goal is to find and label the individual interaction elements (Figure 1). Here we focus on rectangular front faces and the three most common interaction elements: *doors*, *drawers*, and *shelves*.

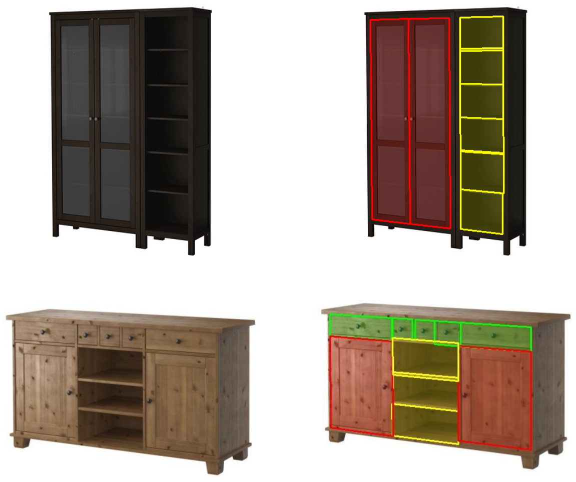
Figure 1: Our approach performs semantic segmentation of modular furniture into doors (red), drawers (green), and shelves (yellow).

Comparing the problem of furniture segmentation to the problem of segmenting entire indoor scenes reveals several new challenges: First, we are interested in a structured segmentation that clearly defines the boundaries of the individual interaction elements. A noisy pixel-wise segmentation would not be sufficient in order to infer the structural information that is required for an interaction. Second, as also noted by [34], traditional visual cues such as color and texture are not particularly useful for labeling furniture as their surfaces are often uniformly colored and have a similar textural appearance. Third, furniture items often include additional decorative elements prohibiting simple rectangle detection from being able to locate or even determine the number of parts (See Figure 1).

We therefore propose a two-stage segmentation approach. In the first stage (Section 3), we generate an *over-complete* set of *interaction element proposals*. Here,

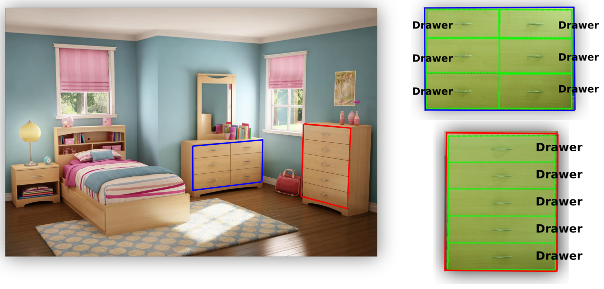

Figure 2: For a given indoor scene (left) the bounding box of the furnitures' front face can be estimated *e.g*. by [16]. We rectify this bounding box which then serves as the input to our pipeline (right).

over-complete means that for all interaction elements there should be at least one matching proposal. Under the assumption that all interaction elements are rectangular, each proposal is a labeled and weighted rectangle. The weight is proportional to the probability of the candidate being an interaction element of a certain kind (*i.e*., a door, a drawer, or a shelf). In the second stage of the approach (Section 4) we select the set of proposals that forms the best *modular* semantic segmentation of the furniture, *i.e*. the separation into interaction elements. We formulate this proposal selection as an energy minimization problem.

Contributions. 1) To the best of our knowledge, we propose the first approach devised for detailed segmentation of modular furniture from single images. 2) One important aspect when representing an object composed of a set of parts is the size of the part set. We provide a novel approach to estimate tight bounds on this quantity. To that end we estimate minimum and maximum coverings by elements through solving a sequence of quadratic integer programming problems. 3) We propose a new classification approach based on a generative codebook. It is able to classify furniture elements that are weakly textured but still share class-specific structural traits. 4) We present a new furniture dataset and corresponding ground truth annotations[1].

2. Related Work

Segmentation approaches. Many segmentation approaches rely on pixel grouping based on feature similarities [7, 17, 26, 9]. These basic segmentation approaches do not take semantic information into account. Most work done in order to incorporate semantic information, can be assigned to one of two categories. The first category of approaches groups neighboring pixels and then classifies them [10]. The second category tries to incorporate semantic information directly into the segmentation itself [28, 2, 19, 20]. Unfortunately, the aforementioned methods just provide a noisy pixel-wise segmentation and do not model the inherent structural properties of modular furniture items.

Indoor scene parsing approaches. [14] and [33] exploit the fact that man-made objects are mostly composed of rectangular elements. Their approach is closely related to ours in the sense that we also generate an initial over-complete set of rectangles and then arrange the current selection of rectangles in the inference stage. However, their work does not consider semantic information.

Facade parsing approaches. The problem of parsing building facades into the architectural elements is similar to our problem of furniture segmentation. Both – facades and modular furniture items – show rectangular, grid-like, recursive structures. One class of methods directly performs a bottom-up analysis, either starting from a noisy segmentation [23] or by inferring repetitive structures [25]. Another class of methods uses shape grammars [30] in order to combine top-down semantic grammar rules with the bottom-up shape cues derived from the image. In most cases, a suitable grammar is manually designed to fit one particular style of architecture [32, 24, 31, 27]. This keeps the parsing technique from generalizing well to similar problems. These grammar based approaches are most powerful if the grammar represents an underlying architectural style, where each instance follows a relatively similar derivation. This is not the case in our problem setting.

Furniture parsing. [21] addresses the problem of furniture detection and pose estimation using an exact 3D CAD model. [11] proposes joint 3D object and layout inference by explicitly modeling occlusion visibility and physical constraints. The inference of an 3D object heavily depends on the 3D CAD model. In contrast to those methods, our approach does not rely on any pretrained object models.

3. Interaction Element (IE) Proposals

In the following, we assume that we are given a single image of a modular furniture from an uncalibrated camera. Moreover, we assume that the furniture item has a rectangular front face whose bounding box is known. For this we rely on a preprocessing step (such as the method from [16]).

Using the bounding box of the furniture item's front face, we first approximately rectify the region of interest by computing a homography that maps the front face to a rectangle of approximately the same aspect ratio (Figure 2). The rectified region of interest serves as the input to our pipeline. This allows us to constrain the search to rectangular axis-aligned interaction elements.

In the first stage of the algorithm we generate an over-complete set of proposals with the goal of generating at least one matching proposal for each interaction element of the furniture item. Having an over-complete set of proposals allows us to compute the semantic segmentation by performing subset selection.

[1]`www.vision.rwth-aachen.de/furniture`

We generate the proposals in three steps: *detection*, *pruning*, and *weighting*. We first perform a supervised approach to generate a *semantic edge map*, a binary image that labels rectangle border pixels. Based on this semantic edge map, we exhaustively search for rectangle candidates. Then, we perform an unsupervised pruning step that removes rectangles that are unlikely to correspond to IEs of the furniture. Finally, we compute weights for all candidates and their corresponding labels, resulting in the final pool of rectangle proposals.

3.1. IE Candidate Generation

Semantic edge map. Following the idea of Dollár *et al.* [8] we predict edge pixels using a *random forest* [4]. In contrast to performing general edge detection, the goal of our semantic edge map is to identify only those edges that belong to interaction element boundaries. We train an ensemble of *binary decision trees* based on feature vectors $\mathbf{x} = (x_1, ..., x_d)^T \in \mathbb{R}^d$. Two different kinds of randomness are used in the tree training process: Each tree is computed on a randomly sampled subset of the training data and the parameters of the tree nodes are optimized over a randomly sampled subset of features. Each node v of the trees is a simple decision stump comparing one entry of the feature vector x_{d_v} to a threshold θ_v. The leaf nodes store the posterior probabilities for each class label. We stop growing the tree when the number of samples in a node falls below a threshold. As feature vector we use image patches of size 25×25 pixels defined over four channels: Intensity, derivatives in x and y direction, and gradient magnitude. For predictions, the output of the different trees are combined via weighted majority voting.

Candidate generation. Based on the semantic edge map, we detect horizontal and vertical lines using the Hough transform. By iteratively sampling two horizontal and two vertical lines, we form rectangle hypotheses defined by the convex hull of the four intersection points of the respective lines. A hypothesis is accepted (*i.e.*, a rectangle is detected) if the maximum distance from any boundary pixel to the closest edge pixel is small. This can be efficiently implemented using a distance transform of the edge map. If the number of iterations is sufficiently high, the candidate set contains all IEs with high probability.

3.2. IE candidate Pruning

The candidate generation step yields a large set of rectangles, most of which do not correspond to actual interaction elements. We perform an unsupervised pruning step to greatly reduce the number of irrelevant rectangles.

As a side product of this candidate pruning we also obtain lower and upper bounds on the possible number of interaction elements of a furniture item. The following algorithm is based on the idea that the correct set of non-overlapping interaction elements should cover almost the entire front face of the furniture item.

A "good" rectangle covering. Let $\Omega \subset \mathbb{R}^2$ be the region of interest (*i.e.* the rectangular front face of the furniture item). A selection of K rectangles $r_1, ..., r_K \subset \Omega$ is an (ϵ, δ)*-rectangle covering of* Ω if

$$\frac{1}{|\Omega|}\left|\bigcup_{k=1}^{K} r_k\right| \geq \epsilon, \quad \frac{|r_k \cap r_l|}{\min(|r_k|, |r_l|)} \leq \delta, \quad \forall k \neq l \tag{1}$$

where $|r_k|$ denotes the area (Lebesgue measure) of the rectangle r_k. These two properties state that a "good" rectangle covering must span at least a portion ϵ of the region of interest Ω while the intersection area between all two pairs of rectangles is smaller than a portion δ of the smaller rectangle. While we can learn the parameter ϵ using the coverage statistics of the training data, we set $\delta = 0.1$ in order to accept a selection of rectangles if each pair of rectangles does not overlap by more than 10%. Ideally, we would like to set $\delta = 0$. However, this might conflict with the sometimes noisy rectangle detection.

Calculation of the maximal (minimal) covering. Inferring the true number of parts can be challenging for furniture items with ambiguous rectangle structures. Because we will try to find an optimal selection of parts in the second step of the pipeline (Section 4), an estimate of the number of parts helps to restrict the size of the search space. In order to obtain these covering numbers, we successively try to fit a rectangle covering for a fixed number K of IE candidates.

This can be achieved by solving quadratic integer programs for different values of K. Let $r_1, ..., r_N$ be the IE candidates. We set up two matrices $A, O \in \mathbb{R}^{N \times N}$ for the two defining properties of an (ϵ, δ)-rectangle cover. Set

$$A_{n,m} = \frac{1}{|\Omega|}|r_n \cap r_m|, \tag{2}$$

$$O_{n,m} = \mathbb{I}\left[\frac{|r_n \cap r_m|}{\min\{|r_n|, |r_m|\}} > \delta\right], \tag{3}$$

for $1 \leq n, m \leq N$, where $\mathbb{I}$ is the *indicator function* that is 1 if the argument is true and 0 otherwise. Furthermore, let $q \in \mathbb{R}^N$ with $q_n = |r_n|/|\Omega|$ for $1 \leq n \leq N$. If for a fixed $K \in \mathbb{N}$, the optimal objective of the following quadratic integer program is not less than ϵ, then the optimal value x^* of the selector variables, defines an (ϵ, δ)-rectangle cover of size K:

$$\max_{x \in \{0,1\}^N} \; q^T x - \frac{1}{2}x^T A x - 2x^T O x \tag{4}$$

$$\text{subject to} \quad \sum_{n=1}^{N} x_n = K$$

We optimize this program using the Gurobi software [13].

Using the rectangle covering for pruning. We finally use the maximum (ϵ, δ)*-rectangle covering* to reduce the set of IE candidate rectangles. This pruning procedure is based on the intuition that the semantic segmentation should cor-

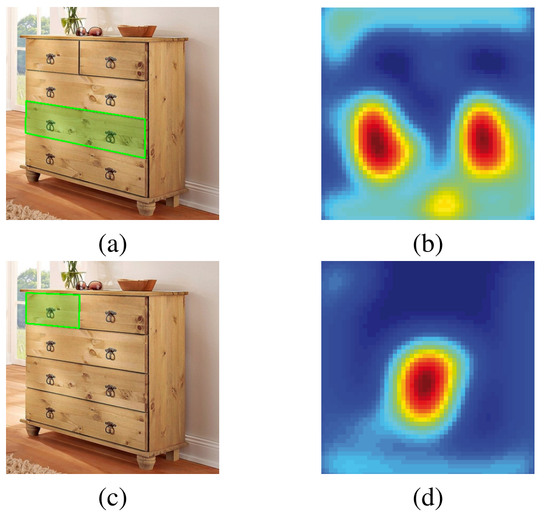

Figure 3: The images (b) and (d) show the distributions of gradients within the marked rectangles in the images (a) and (c) respectively. The door handles are clearly visible as small regions with large gradients.

respond to a maximal covering (*i.e.*, a rectangle covering with the largest possible number of rectangles). While this is not always the case, rectangles that are similar in size and location to the ones from a maximal covering usually form good proposals for interaction elements. Therefore, we prune the initial set of IE candidates of all the rectangles that are dissimilar to the ones in a computed maximal covering. To be specific, if $r_1, ..., r_N$ are the initial IE candidates and $r_{n_1}, ..., r_{n_K}$ form a maximal (ϵ, δ)-rectangle covering, then we prune the rectangle r_n from the set if $\max_{k=1,...,K} \frac{|r_n \cap r_{n_k}|}{|r_n \cup r_{n_k}|} < \theta$ where $\theta \in [0,1]$ is the *pruning threshold*.

3.3. IE Candidate Weighting

We assign each IE candidate a weight and a class label. The weight quantifies the likelihood of an IE candidate belonging to a certain class. We model the weights in terms of the conditional probability

$$p(l \mid r, I) \propto \underbrace{p(I \mid r, l)}_{\substack{\text{Appearance}\\ \text{likelihood}}} \underbrace{p(r \mid l)}_{\substack{\text{Shape}\\ \text{prior}}} \underbrace{p(l)}_{\substack{\text{Label}\\ \text{prior}}}, \tag{5}$$

where I is the image, r is a rectangle, and $l \in \{\text{door, drawer, shelf}\}$ is a class label. The shape and label priors can easily be modeled using standard machine learning techniques. For the shape prior, we extract the width and height of the rectangle relative to the size of the region of interest as well as its aspect ratio and then use kernel density estimation with a Gaussian kernel. Modeling the appearance likelihood, however, is more challenging.

Many traditional appearance-based classification methods make use of strong visual cues such as color and/or texture. However, due to the mostly uniform appearance of furniture items, those features often are not sufficiently discriminative. Instead, we build our appearance likelihood based on the observation that interaction elements of a certain class tend to exhibit particular traits such as handles at distinct positions (*e.g.*, many drawers have a handle at the center). These traits are visible in a gradient magnitude image as regions with strong gradients (Figure 3).

We design the appearance likelihood in terms of a codebook of such traits. For this, we resize the appearance of an interaction element in the gradient magnitude image to a uniform size of $M \times M$ pixels and use this as a feature vector of dimensionality M^2. Let $v_{r,I} \in \mathbb{R}^{M^2}$ be the feature vector for the rectangle r in the image I. The idea is to represent $v_{r,I}$ in terms of a linear combination of codebook vectors called *codewords*. We learn one codebook per class. Given the codewords $p^{(1,l)}, ..., p^{(J,l)} \in \mathbb{R}^{M^2}$ for class l, we express the appearance likelihood as

$$p(I \mid r, l) \propto \max_{\substack{\pi \in [0,1]^J \\ \|\pi\|_1 \le 1}} \exp\left(- \left\| v_{r,I} - \sum_{j=1}^{J} \pi_j p^{(j,l)} \right\|_2^2 \right). \tag{6}$$

Hence, the likelihood is large if there exist coefficients π such that the feature vector $v_{r,I}$ can be approximated well (in the l^2-norm) by the linear combination $\sum_{j=1}^{J} \pi_j p^{(j,l)}$. In order to evaluate this expression, the optimal coefficients have to be determined. Taking the negative logarithm yields the quadratic program

$$\min_{\pi} \left\| v_{r,I} - \sum_{j=1}^{J} \pi_j p^{(j,l)} \right\|_2^2 \tag{7}$$

$$\text{subject to } \pi_j \ge 0, j = 1, ..., J$$

$$\sum_{j=1}^{J} \pi_j \le 1$$

whose solution is the optimal coefficient vector.

Codebook learning. Let $v^{(1)}, ..., v^{(N_l)}$ be the feature vectors corresponding to the groundtruth rectangles of class l. We learn the codebook for each class independently by executing the following algorithm.

1. Initialize the codewords by assigning $p^{(1,l)}, ..., p^{(J,l)}$ randomly to some feature vectors
2. While the decrease in the objective function (8) is significant
   (a) For each $n \in \{1, ..., N\}$ determine $\pi^{(n)}$ by solving the quadratic program (7) with $v_{r,I} \equiv v^{(n)}$.
   (b) Update the codewords by solving the quadratic

program

$$\min_{p^{(1,l)},\ldots,p^{(J,l)}} \sum_{n=1}^{N} \left\| v^{(n)} - \sum_{j=1}^{J} \pi_j^{(n)} p^{(j,l)} \right\|_2^2 \quad (8)$$

subject to $p_m^{(j,l)} \geq 0, j \in [J], m \in [M]$

where $[J] = \{1, ..., J\}$ and $[M] = \{1, ..., M\}$.

The iteration over step 2 is required due to the alternating optimization of $\pi^{(n)}$ and $p^{(1,l)}, ..., p^{(J,l)}$. This concept of learning a codebook is closely related to the field of sparse coding where one usually minimizes an l^1-regularized squared loss. Because our feature vectors are discrete probability distributions, we included additional constraints to reflect this setup. While intuition might suggest that constraining the problem to guarantee that the learned codewords form probability distribution themselves should be the best design choice, the opposite is the case. We want the codewords to reflect class specific visual traits. However, usually only a fraction of the entire probability mass is concentrated on these traits. Thus, they do not form probability distributions themselves. Therefore, we only constrain the codewords to be non-negative.

4. Proposal Selection by Energy Minimization

In the second step of the system, we compute the semantic segmentation by selecting the *most modular* subset of proposals. To this end, we define an energy function that scores the modularity of a selection of IE proposals by the number of *forced merges* in the *modularity tree* that we introduce in the next section. We minimize the energy function using an MCMC-based optimization technique called *simulated annealing* [5, 18, 3].

4.1. Modularity Tree

In order to score the modularity of a fixed selection of IE proposals, we build a *modularity tree*. The modularity tree inductively finds and merges constellations of similarly sized rectangles in a bottom-up fashion. Each node in the tree is labeled with a rectangle that tightly encloses the rectangles of its child nodes and a flag that indicates whether or not the node is a result of a *forced merge*. A forced merge always occurs if there are no two similar rectangles that can be merged, see Figure 4. Algorithm 1 shows the main loop of the tree building algorithm.

The tree is initialized using all rectangles as leaf nodes (InitializeTree). Each iteration of the while loop adds a new layer to the *modularity tree*.

The FindClusters function. It takes as argument a set of rectangles and returns a set of clusters. The clusters are found by maximizing the number of rectangles that are contained in clusters and minimizing the total number of clusters. From this objective it follows that some rectangles might not be merged in an iteration. These rectangles will not be added to $\mathcal{C}$. In the concrete example of Figure 4 the first iteration (level l_1) will only contain the red and the blue cluster.

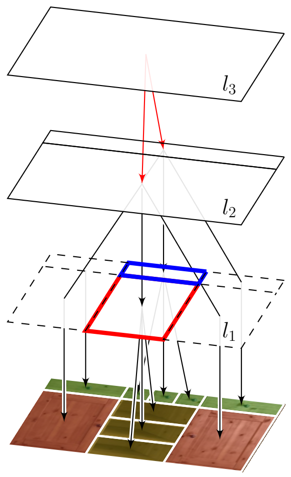

Figure 4: The modularity tree for the ground truth segmentation of the cabinet shown in figure 1 (d). There is a forced merge from level l_2 to level l_3 of the tree.

Algorithm 1 Build the modularity tree $\mathcal{T}$ from a selection of non-intersecting rectangles. Here, $d(r,p)$ is the *merge distance* between the rectangles r and p. If the union rectangle of r and p does not intersect with any other rectangle, then $d(r,p)$ is the sum of the width and height differences. Otherwise, it is infinity.

```
function BuildModularityTree($r_1, ..., r_K$)
    $\mathcal{T} \leftarrow$ InitializeTree($r_1, ..., r_K$)
    $N \leftarrow \{r_1, ..., r_K\}$
    while $|N| > 1$ do
        $\mathcal{C} \leftarrow$ FindClusters($N$)
        $f \leftarrow false$
        if $|\mathcal{C}| = 0$ then
            $(r,p) \leftarrow \arg\min_{r,p \in N, r \neq p} d(r,p)$
            if $d(r,p) = \infty$ then
                return Error
            $\mathcal{C} \leftarrow \{\{r,p\}\}$
            $f \leftarrow true$
        for $C \in \mathcal{C}$ do
            $N \leftarrow N \backslash C$
            $m \leftarrow$ GetUnionRectangle($C$)
            $\mathcal{T} \leftarrow$ AddParentNode($\mathcal{T}, C, m, f$)
            $N \leftarrow N \cup \{m\}$
    return $\mathcal{T}$
```

FindClusters first exhaustively lists all *valid* clusters and then selects the best subset of non-intersecting (set intersection) clusters. A cluster $\mathcal{C} = \{r_1, ..., r_L\}$ is *valid* if

1. the contained rectangles are of approximately the same size, and
2. the union (bounding) rectangle *only* significantly over-

laps with the elements of the cluster.

As a consequence, the second requirement *e.g.* ensures that two rectangles of the same size that are separated by a third rectangle of a different size cannot form a valid cluster.

Runtime. While the theoretical runtime of the algorithm is exponential in the number of rectangles, it is not a problem in practice. This is due to the small number of rectangles (Usually, $K < 15$) and a search strategy where we examine the most promising clusters first in the exhaustive search.

4.2. Energy function

Our objective is to find the most modular (*i.e.*, least number of forced merges) set of IE proposals that forms a rectangle cover. We can formalize this using a multi-objective optimization approach. Let $(r_1, l_1), ..., (r_K, l_K)$ be a set of IE proposals. Then, the energy function is given by

$$\begin{aligned} &E((r_1, l_1), ..., (r_K, l_K)) \\ &= \underbrace{\lambda_1 \sum_{n \in \mathcal{T}} \mathbb{I}[f(n)]}_{\substack{\text{Modularity} \\ \text{score}}} - \underbrace{\sum_{k=1}^{K} p(l_k \mid I, r_k)}_{\substack{\text{Label} \\ \text{energy}}} \\ &+ \underbrace{\lambda_2 \sum_{n \in \mathcal{T}} \sum_{\substack{c_1, c_2 \in \text{child}(n) \\ c_1, c_2 \text{leaf nodes}}} \mathbb{I}[l(c_1) \neq l(c_2)]}_{\substack{\text{Label} \\ \text{smoothing}}} \end{aligned} \tag{9}$$

where $f(n)$ indicates whether node n is the result of a forced merge and $l(c_i)$ is the label that is assigned to the rectangle corresponding to the leaf node c_i. By choosing λ_1 sufficiently large, we can prioritize modularity. This results in the optimum being the most modular selection of candidate rectangles with the best labels.

4.3. MCMC-based Optimization

We optimize the problem (9) using simulated annealing. Simulated annealing is an MCMC-based stochastic optimization method. It is particularly suitable for our problem because it works well for combinatorial optimization problems with arbitrary energy functions. Simulated annealing works by constructing a Markov Chain with a stationary distribution that concentrates all probability mass on the global minima of E. The Markov Chain can be understood as a random walk through the state space. If P is the current state, then we sample a new state $\tilde{P}$ with conditional probability $p(\tilde{P} \mid P)$. If $E(\tilde{P}) \leq E(P)$, then we accept the new state and set $P := \tilde{P}$. Otherwise we only accept the new state with a probability proportional to $\exp\left(-(E(\tilde{P}) - E(P))/t_n\right)$ where $n \in \mathbb{N}$ is the current iteration and t_n is a monotonically decreasing *cooling schedule* with $t_n \to 0$. By slowly decreasing the *temperature* t_n over time, accepting a worse state becomes less likely. In the limit $t_n \to 0$, we only accept globally optimal states.

Algorithm 2 The MCMC-based optimization algorithm for selecting the best segmentation for a fixed $K \in \mathbb{N}$. Parameters and notation: $T_{start} > T_{end} > 0$ are the start and end temperatures, respectively, $\alpha > 0$ controls how quickly the temperature decreases over time, and $\mathcal{U}_A$ is the uniform distribution over the set A.

```
function SELECTPROPOSALS($(r_1, l_1), ..., (r_N, l_N)$)
    $P \leftarrow ((r_1, l_1), ..., (r_K, r_K))$
    $t \leftarrow T_{start}$
    while $t > T_{end}$ do
        // Decrease the temperature
        $t \leftarrow \alpha t$
        // Sample a state in the neighborhood
        Sample $k \sim \mathcal{U}_{\{1,...,K\}}$
        Sample $p \sim \mathcal{U}_{\{(r_1,l_1),...,(r_N,l_N)\}}$
        $\tilde{P} \leftarrow P$
        $\tilde{P}(k) \leftarrow p$
        // Accept the state with a certain probability
        Sample $u \sim \mathcal{U}_{[0,1]}$
        if $\exp\left(-\frac{1}{t}(E(\tilde{P}) - E(P))\right) \geq u$ then
            $P \leftarrow \tilde{P}$
```

For our application, we chose a *geometric cooling schedule* of the form $t_n = \alpha t_{n-1}$ with $\alpha \in (0, 1)$. Further information about simulated annealing can be found in [3].

Let $\mathcal{D} = \{r_1, ..., r_N\}$ be the set of candidate rectangles. For a fixed $K \in \mathbb{N}$, the state space for our problem is given by the set of all sets of proposals of size K. We sample a new state $\tilde{P}$ from the neighborhood of a given state P by uniformly sampling a proposal in P that we replace with a uniformly sampled proposal in the state space. Hence, P and $\tilde{P}$ only differ in a single proposal. In order to guarantee that we only choose the optimal state among the possible (ϵ, δ)-rectangle covers of size K, we add a penalty term to the objective function that is large if two rectangles intersect more than δ or less than a portion ϵ of the region of interest is covered by rectangles. This procedure is motivated by the use of barrier functions in interior point methods. Algorithm 2 depicts the entire optimization procedure.

We run the optimization algorithm for each $K \in \{K_{min}, ..., K_{max}\}$ where K_{min} and K_{max} are the sizes of the minimal and maximal rectangle covering, respectively (Section 3.2). As the final segmentation, we choose the selection with the smallest number of forced merges. In the case that the minimum number of forced merges is achieved for more than one $K \in \{K_{min}, ..., K_{max}\}$, we choose the segmentation with the largest number of interaction elements.

5. Evaluation

We evaluate our approach on a novel dataset of 140 images from the IKEA online furniture catalog [1] showing cabinets for which we provide ground truth annotations. All

Edge detector	**Precision**	**Recall**
Random forest	**65.6%**	**97.3%**
Canny	56.2%	78.9%

Table 1: The table shows the pixel-wise classification accuracy of the random forest-based edge detector and the Canny edge detector.

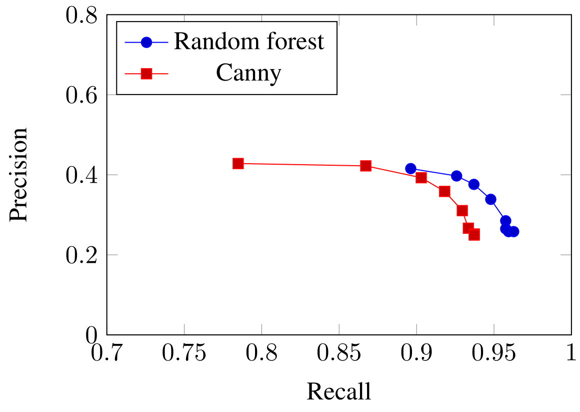

Figure 5: IE candidate generation results for varying acceptance thresholds. Note that we generate an over complete set of rectangles. Therefore, a low precision is to be expected.

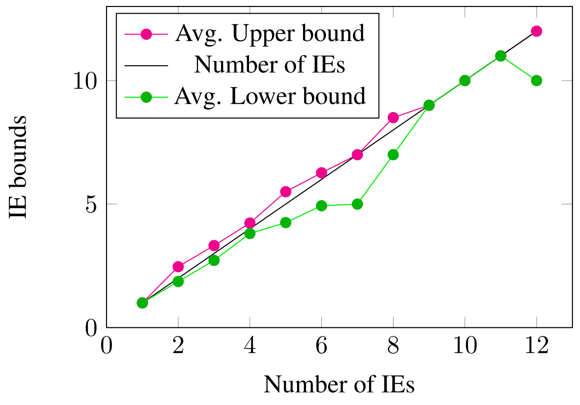

Figure 6: The tightness of the obtained bound on the true number of interaction elements.

reported results are averages obtained from a 5-fold cross-validation. We count an interaction element as detected if its intersection over union score (IoU score) with a selected IE candidate is greater than 0.65.

5.1. IE Candidate Evaluation

In the first stage of our approach (Section 3) we generate an over-complete set of IE proposals from rectangles detected in a semantic edge map. The semantic edge map is generated by a random forest (Section 3.1). We compare the pixel-wise classification performance of the random forest-based approach to the Canny edge detector. To this end, we accumulate class votes for each pixel and then apply hysteresis thresholding on the votes image in order to obtain a comparable edge map. Table 1 reports the classification accuracy. We see that the random forest-based edge detector finds more semantic edges (higher recall) while at the same time producing less irrelevant edges (higher precision).

In order to further illustrate the performance contributions obtained from choosing the random forest-based approach over a standard edge detector, we investigate the performance of the subsequent rectangle detector. The rectangle detector directly uses the semantic edge map in order to find IE candidates. Figure 5 reports the rectangle detector performance by varying rectangle acceptance thresholds. Again, the random forest-based edge detector consistently outperforms the Canny edge detector.

In the final stage of the IE candidate generation step, we prune the initial detector output and obtain upper and lower bounds on the possible number of interaction elements. A significant increase in precision from 37.6% before pruning to 73.9% after pruning while only reducing the recall from 93.7% before pruning to 88.0% after pruning indicates that the pruning step removes a considerable number of irrelevant rectangles while keeping most of the relevant ones. Furthermore, the obtained bounds on the number of parts are correct for 95.2% of all images. Finally, Figure 6 shows the average upper and lower bounds for furniture items having different numbers of parts. We see the bounds enclose the true number of parts tightly.

IE class	**Detection rate**
Door	86.9%
Drawer	85.4%
Shelf	64.3%

Table 2: The table shows the percentage of interaction elements whose rectangles have correctly been identified during the proposal selection stage.

		Prediction		
		Door	Drawer	Shelf
Truth	Door	87.4%	3%	9.6%
	Drawer	1.9%	91.1%	7%
	Shelf	16.3%	32.7%	51%

Table 3: The confusion matrix for the labeling of the rectangles.

5.2. Proposal Selection Evaluation

In the second stage of the pipeline (Section 4), we select proposals by minimizing an energy function. We report two interesting measurements to assess the performance of this stage. First, in Table 2, we measure the accuracy of the selected rectangles regardless of their predicted labels. The results show that the correct number of rectangles have been

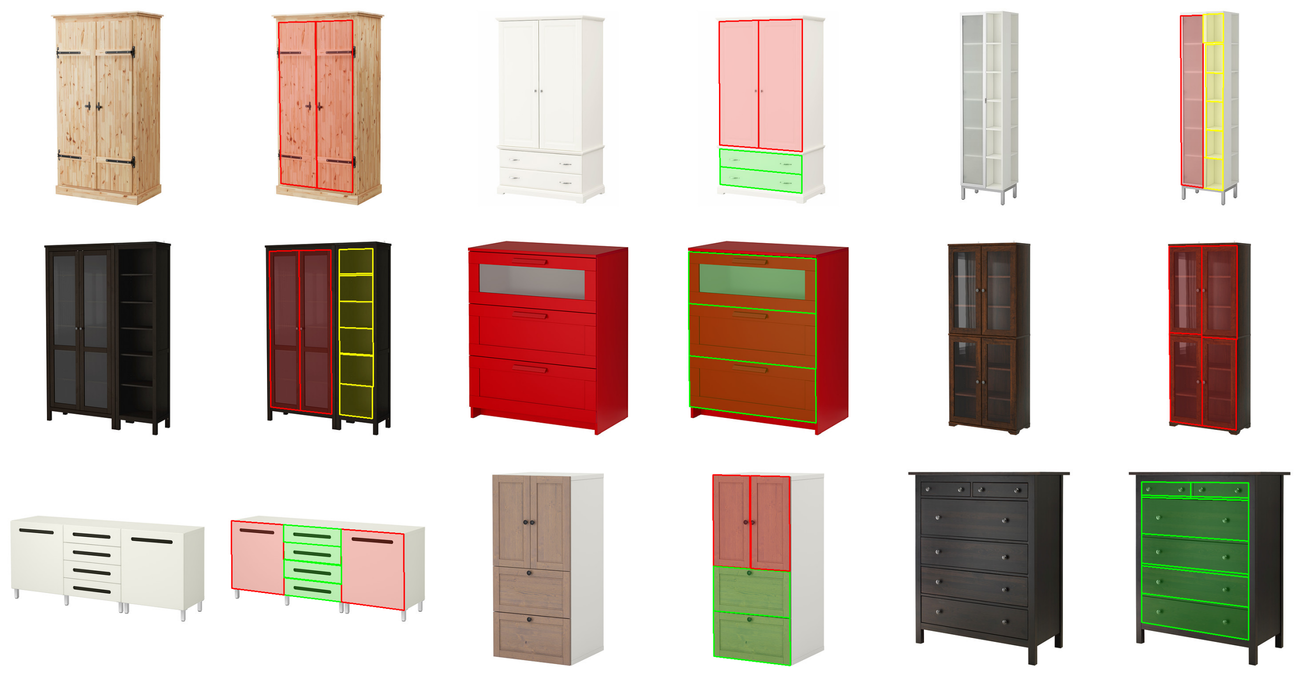

Figure 7: Some successfully segmented images. Doors are red, drawers are green, and shelves are yellow.

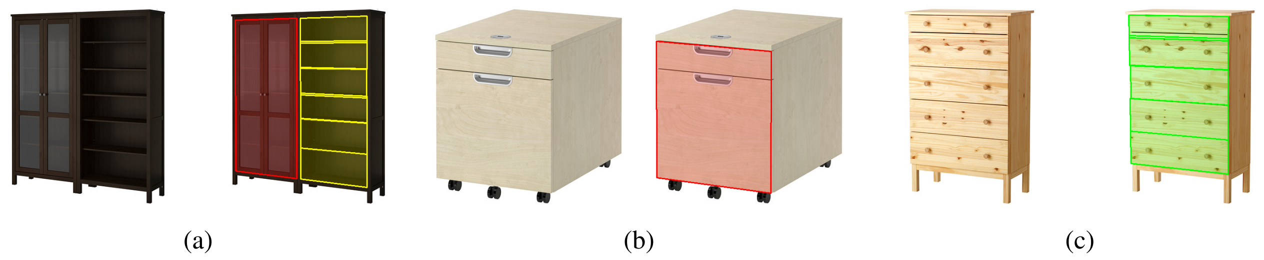

Figure 8: Failure cases. (a) The boundary of the two individual doors has not been found; (b) as selecting the two correct interaction elements would result in a forced merge, the wrong rectangle has been selected; (c) the top most drawer is misaligned. In this case, the strong wooden texture induces significant edges that led to the incorrect rectangle being detected.

detected in most test cases. Second, we measure the accuracy of the predicted labels for those interaction elements whose rectangles have correctly been found. Table 3 shows the label confusion matrix. Our overall accuracy is 84.6%.

While we obtain strong classification results for doors and drawers, the classification accuracy of shelves is lower. This is the result of several factors. First, shelves exhibit considerably fewer visual cues than most doors and drawers. Hence, they can easily be mistaken for drawers or doors without handles. Second, shelves are similar in size and aspect ratio to drawers. Therefore, the confusion between shelves and drawers is higher than the confusion between shelves and doors. Finally, the data set is imbalanced in the sense that the number of shelves is significantly smaller than the number of doors and drawers. This results in a lower prior probability for shelves. Figures 7 and 8 show some qualitative results and common failure cases.

6. Conclusion

In this paper, we have proposed a method for the semantic segmentation of modular furniture. In two stages, we first generate an over-complete set of proposals and then select a subset of proposals that minimize an energy function across multiple scales. We have demonstrated the performance of our approach on a novel data set with ground truth annotations which is publicly available[2]. This paper marks a first step in the direction of detailed semantic segmentation of indoor scenes. At this level of detail, semantic segmentations open up new possibilities to robustly use autonomous robots in indoor environments.

Acknowledgment. The work in this paper was funded by the EU project STRANDS (ICT-2011-600623).

[2]`www.vision.rwth-aachen.de/furniture`

References

[1] Ikea online furniture catalog. `http://www.ikea.com/us/en/`.

[2] A. Barbu and S. C. Zhu. Graph partition by swendsen-wang cuts. In *ICCV*, 2003.

[3] D. Bertsimas and J. Tsitsiklis. Simulated Annealing. *Statistical Science*, 8(1):10–15, 1993.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[6] W. Choi, Y. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *CVPR*, 2013.

[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5):603–619, 2002.

[8] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *PAMI*, 37(8):1558–1570, 2015.

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

[10] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, 2009.

[11] A. Geiger and C. Wang. Joint 3d object and layout inference from a single rgb-d image. In *GCPR*, 2015.

[12] S. Gupta, P. A. Arbeláez, R. B. Girshick, and J. Malik. Indoor scene understanding with RGB-D images: Bottom-up segmentation object detection and semantic segmentation. *IJCV*, 112(2):133–149, 2015.

[13] I. Gurobi Optimization. Gurobi optimizer reference manual, 2015.

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

[15] V. Hedau, D. Hoiem, and D. A. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[16] V. Hedau, D. Hoiem, and D. A. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*, 2010.

[17] M. Kass., A. P. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[19] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *ECCV*, 2002.

[20] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[21] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing IKEA objects: Fine pose estimation. In *ICCV*, 2013.

[22] D. Lin, S. Fidler, and R. Urtasun. Holistic Scene Understanding for 3D Object Detection with RGBD Cameras. In *ICCV*, 2013.

[23] M. Mathias, A. Martinović, J. Weissenberg, and L. Van Gool. Atlas: A three-layered approach to facade parsing.

[24] M. Mathias, A. Martinovic, J. Weissenberg, and L. J. Van Gool. Procedural 3d building reconstruction using shape grammars and detectors. In *3DIMPVT*, 2011.

[25] P. Müller, G. Zeng, P. Wonka, and L. J. Van Gool. Image-based procedural modeling of facades. In *SIGGRAPH*, 2007.

[26] S. Osher and N. Paragios. *Geometric Level Set Methods in Imaging,Vision,and Graphics*. Springer, 2003.

[27] N. Ripperda and C. Brenner. Reconstruction of faade structures using a formal grammar and rjmcmc. In *DAGM-Symposium*, 2006.

[28] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[29] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[30] G. Stiny. *Pictorial and formal aspects of shape and shape grammars*. Birkhuser, 1975.

[31] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Parsing facades with shape grammars and reinforcement learning. *PAMI*, 35(7):1744–1756, 2013.

[32] C. A. Vanegas, D. G. Aliaga, and B. Benes. Building reconstruction using manhattan-world grammars. In *CVPR*, 2010.

[33] Y. Zhao and S. C. Zhu. Image parsing with stochastic scene grammar. In *NIPS*, 2011.

[34] Y. Zhao and S. C. Zhu. Scene parsing by integrating function, geometry and appearance models. In *CVPR*, 2013.