2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Robotic Object Manipulation with Multilevel Part-based Model in RGB-D Data

Kun Li[1] and Max Meng[2]

Abstract— **The performance of robotic object manipulation relies heavily on the selection of object model. In this article, we develop a multilevel part-based object model by applying latent support vector machine to training a hierarchical object structure. We implement our method with a robot arm and a depth sensor in Robot Operating System, and then we compare the recognition performance of this model with established methods on a point cloud data set and show the manipulation performance of our model on three practical tasks. The result demonstrates that our robot recognizes and manipulates objects more accurately with this multilevel part-based object model.**

I. Introduction

An autonomous robot is expected to recognize and manipulate objects with minimal human intervention and mainly based on sensory information, especially visual information, so the embedded object representation and recognition method is essential for the robot to detect target objects accurately and manipulate them dexterously. Existing vision-guided manipulation methods mainly represent each object as a cluster in feature space, and the robot arm may pick up and move the cluster to other locations. For example, in [1], the robot represented the door handle as a feature cluster in depth data and RGB data, and then it rotated the detected region to open the door. In other studies, the researchers may alter the number of visual sensors [2], or the number of collected images [3], but the representation scheme remains unchanged. As a result, the robot can only perform simple object manipulations, like moving the feature cluster to another location, or rotating the feature region by a certain angle. For more complicated manipulations, such as opening a paper box, the robot must recognize the paper box first, and then locate the lid of the box, in order to rotate the lid and open the box. By treating the paper box as two independent feature clusters, the robot has to learn, detect and manipulate them separately, which discards the relation between the object parts, and leads to degraded performance on object recognition and manipulation.

Part based object model and deep belief network try to solve this problem by representing each object as a collection of small parts and organizing them according to their relationships. For example, the generative part-based object model in [4] represented each object with part descriptors and part locations, and then learned the distribution over all the objects parts. Object recognition by conditional random field [5] also relied on detection of object parts to identify the whole object. Although these methods recognize object parts accurately, it's hard to extend them to complicated objects because of the inflexible model structure. Deformable part based object model [6] solved this problem by representing each object with a two layer structure, where the root template described the whole object and each part template described one component of the object. The root template and part templates were related by a latent variable, denoting their displacements, which was eventually trained with latent support vector machine. Deformable part-based model, when applying to robotic manipulation, faces problems that the detected local templates may be non-manipulable object parts, and the two layer structure is inflexible in many tasks. Considering this problem, deep belief network [7] represented each target as a multi-layer structure, where each layer took the output of the lower layer as input. The relation between the output and input in each layer was firstly learned by restricted Boltzmann machine (RBM), and then fine-tuned in a supervised way. The nodes in this multi-layer structure may still not be practical object parts, but one extension [8] solved this problem by introducing convolutional RBM and probabilistic max-pooling. However, convolutional deep belief network requires massive amount of training data, making it inefficient in many practical tasks.

In our work, we propose a multilevel part-based object model, incorporating the hierarchical structure from deep belief network and the discriminative training of each layer from deformable part-based model. Our method represents objects more flexibly with the extendable hierarchical structure, and requires less training data with latent support vector machine.

II. Related Work

Part-based object models mainly adopt two-layer structure: object parts and the whole object. For example, deformable part based model [6] represents each object as a root template, with several part templates anchored to it, and all of the templates are defined as weight vectors on a feature pyramid. This model allows the small parts to move within the root template, and encodes the movements in a latent variable. Then latent support vector machine is developed to learn the templates and their latent relations. An extension of deformable part-based model [9] uses semantically labelled training data to detect physically meaningful object parts and find new objects from the voting of learned objects and their

*This work was partially supported by RGC GRF CUHK415611 awarded to Max Meng

[1]Kun Li is with Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong kli@ee.cuhk.edu.hk

[2]Max Meng is with Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong max@ee.cuhk.edu.hk

taxonomies. However, this two-layer structure is too frigid to represent many practical objects with physically manipulable parts.

Some other methods introduce more layers in the object model. Hierarchical part-based object model [10] represents each object as multiple layers of parts and subparts, and each part is assigned to a parent part from higher layer. But the object part in this model has no physical meaning and only helps to improve the object recognition accuracy. Convolutional deep belief network by Lee et al. [8] extends deep belief network by including convolution and probabilistic pooling to learn low level basic object components, intermediate object part representations, and high level object concepts, but it requires enormous training data, making it inapplicable to practical robotic manipulation tasks.

Our model adopts the hierarchical object structure to describe more object details, and requires the composing parts to be physically meaningful so that the robot may use them in manipulation tasks. To train this model for practical tasks with limited number of labeled data, we use latent support vector machine developed in [6], which is readily extendable to our multi-layer object structure.

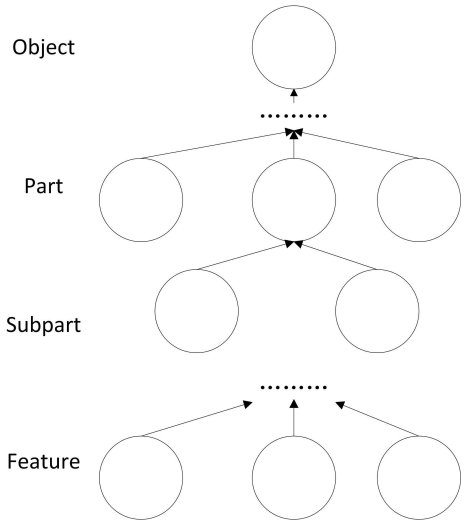

Fig. 1: Multi-level object structure: each object is composed of multiple parts, and each part also has its own "child" part. The bottom layer is the raw sensory data. The core of this structure is a function that aggregates each layer of object parts into larger object parts.

III. The Method

A. Object Model

Our object model, like many other hierarchical models, represents each object as a hierarchy of object parts, as shown in Figure 1. Our model differs from others in that all the object parts are physically meaningful and manipulable for the robot. One example of our object model is shown in Figure 2.

The whole object model can be described as a distribution:

$$P(x,h^1,\cdots,h^l)=\prod_{k=1}^{l}P(h^k|h^{k-1})$$

where x is the raw feature data, and h^i denotes the i_{th} layer. These layers are described in detail below:

1) Local Feature Clusters: The bottom layer, h_0, is composed of feature vectors directly extracted from the visual data. For each sample, we firstly split it into $N*N$ regions, and the variable N depends on applications. For each region, we adopt the bag-of-words method to construct a feature word based on multiple types of features, including color histogram, speeded up robust features [11], histogram of oriented gradient [12], and fast point feature histogram [13]. These feature words, along with the locations of all regions, are used to learn the layers of the object parts from h_1 to h_l. For example, in Figure 2, we may split the whole image into $20*20$ squares, and then extract from each region a feature vector that concatenates the features above.

2) Object Parts: After we get the local feature words, we construct the high level object part layers with the following mapping function:

$$f_{ji}=f(w_{ji},b_{ji},z_{ji},h_{j-1}),j=1,\cdots,l$$

where f_{ij} indicates the i_{th} object part in layer j, w represents the weight, b represents the offset, and z represents the spatial configuration, describing each part's location (x_i,y_i) in its parent object window. This function describes the relation that every object part is composed of basic components from lower layer with a weight vector w and in certain spacial configuration z, which indicates the location of each basic component relative to the parent object part.

Layer h_1 is composed of the most fundamental object parts and directly built upon the feature words in h_0. With the mapping function, layer h_1 outputs a binary vector, indicating the presence of object parts, and a position vector, describing the locations of objects parts. Layers h_2 to h_{l-1} have the same structure. They all take the part presence vector and location vector of lower layer as input, and use the mapping function to learn larger object parts in current layer.

For the purpose of robotic object manipulation, all object parts, from h_1 to h_{l-1}, are learned only when they are physically meaningful. In Figure 2, the basic object parts, like the bumper and the tag, are in red bounding boxes, and they form larger object parts-the box and the handle-in green bounding boxes.

3) Object: After variable l layers of object parts, we construct the final object with the output from layer $l-1$. We use the mapping function corresponding to the whole object to identify its presence and location. In Figure 2, the object in blue bounding box is built from the two large object parts in green boxes.

B. Training

The mapping function $f=f(w,b,z,h)$ is the core of our object model, and we adopt latent support vector machine to learn the parameters $w,b,$ and z. Denoting the input of each layer as x, the output as y, the configuration as z, and the

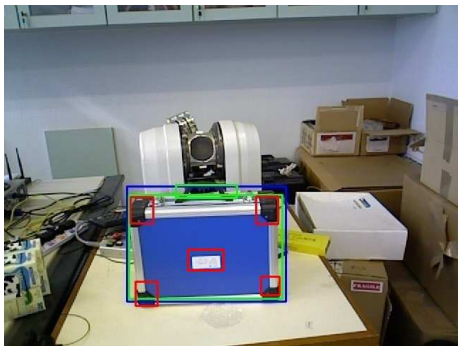

Fig. 2: An example of hierarchical part-based object model: we model the toolbox as a four-layer object. The first layer is the raw sensory data; the second layer is small object parts; the third layer are main object parts; and the final layer is the whole object.

weight and bias parameter as β, the output is defined as:

$$y_{\beta,Z}(x) = \max_{z \in Z(x)} \beta \cdot \phi(x,z)$$

The objective function is a non-convex function:

$$\lambda ||\beta||^2 + \Sigma_{i=1}^{n} \max(0, 1 - y_i \cdot f_{\beta,Z}(x_i))$$

In the original work of Felzenszwalb [6], coordinate descent method is developed to optimize such a function, which iterates the two steps below until convergence.

- Given β, find the optimized configuration z for each sample,

$$\forall i : z_i \leftarrow argmax_{z \in Z(x)} \beta \cdot \phi(x_i, z)$$

- Construct the positive train vector with z, and learn β with normal support vector machine that minimizes the objective function:

$$\lambda ||\beta||^2 + \Sigma_{i=1}^{n} \max(0, 1 - y_i \cdot f_{\beta,Z}(x_i))$$

In our work, we use latent support vector machine to train the mapping functions $f = f(w,b,z,h)$ layer by layer. For layer h_1, the feature words x from layer h_0 and their locations are used to construct training vectors in a format as follows:

$$feature = [x_0, x_1, \cdots, x_n, d_{P1}, \cdots, d_{Pn}, 1]$$

where x_0 is the descriptor of current object parts, $x_i, i = 1, \cdots, n$ are the descriptors of object parts inside x_0, $d_{Pi} = (x_i, y_i, x_i^2, y_i^2), i = 1, \cdots, n$ describe the locations of object parts relative to the object, and the final 1 is for bias. For layers from h_2 to h_l, the descriptors are binary vectors indicating the presences of object parts from the lower layer.

With the training vectors, we use SVM light library [14] to learn the model's parameters, β:

$$\beta = [w_0, w_1, \cdots, w_n, d_1, \cdots, d_n, b]$$

where $w_i, i = 0, \cdots, n$ are the weight vectors, $d_i = (d1, d2, d3, d4), i = 1, \cdots, n$ are the parameters for the quadratic deformation loss functions, and b is the bias term.

With the learned parameter β and feature descriptor x, we update the spatial location parameter z according to

$$f_\beta(x) = max_{z \in Z(x)} \beta \cdot \phi(x_i, z)$$

where z represents the locations of the object parts in its parent object window, and ϕ is the feature vector under current z. Normally, the searching space for z is really large for exhaustive searching. For example, the first layer of our model has about 45000! possibilities. We reduce the searching space by assuming that all points in one object part have the same relative location z.

We repeat the process above until the changes of β and z are smaller than preselected thresholds. After the training, we get a set of parameters β corresponding to each node of our multilevel object structure.

C. Detection

In the detection phase, we combine the bottom-up and top-down process to make sure all meaningful object parts are correctly located. The bottom-up process builds a hierarchy of all learned object parts, and the top-down process retrieves the part-based structure of target object. One example is shown in Figure 3.

1) Bottom-up Detection: First, we split the visual input into $N * N$ regions and extract feature words from each one of them. Then we use sliding window method to extract all learned object parts in layer h_1 from the feature words and their locations based on the following relationship:

$$y_{\beta,Z}(x) = \max_{z \in Z(x)} \beta \cdot \phi(x,z)$$

The result includes a binary indicator y of object part presence, and the configuration z of feature words within the detected object part.

After that, we use the detected object parts in layer h_1 and their locations to find all learned high-level object parts, which form layer h_2 to h_{l-1}. After the bottom-up process, we have a large set of object parts A organized as a hierarchy, where each object part is composed of smaller object parts in special configurations. To speed up the detection process, this step is computed before the robot's actual object detection. In Figure 3, the object parts, like bottle lid, toolbox handle, spoon and paper board, are detected during the bottom-up process, based on the feature words extracted from the point cloud data.

2) Top-down Detection: When the robot receives command to detect certain object, it will retrieve the learned parameters, and use them to find the object from all of the detected object parts A. If the object exists, it will be identified with its composing parts $h_{l-1} \in A$. Then it retrieve the smaller composing parts h_{l-2} from the stored hierarchy, and repeat the process until reaching the bottom layer. After the top-down detection process, we get an object with its composing parts and their locations. In Figure 3, we detect the "toolbox" based on the "handle" and "box", and then retrieve the saved hierarchy of these parts to form a hierarchical toolbox structure.

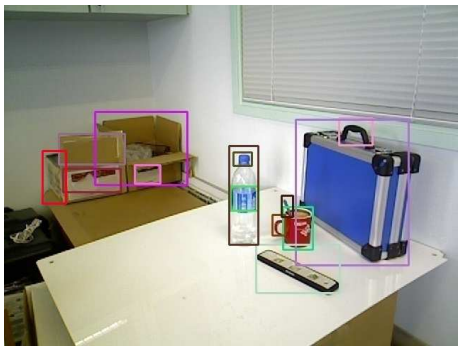

Fig. 3: The detection process: firstly we extract feature words from each of the $N*N$ image regions, and then detect basic object parts from it. These basic parts are used to construct larger object parts and form a hierarchy. When the robot detects an object, it only needs to find the large composing parts and retrieve smaller parts from the hierarchy.

D. Manipulation

Robotic object manipulation relies on complicated control strategy for different objects in general. To simplify the control problem in our application, we represent each manipulation command C with the following format:

$$C = (O_i, A_i), i = 0, \cdots, N$$

where O_i represents the i_{th} part of the target object, and A_i is the action to perform on the i_{th} part. We represent each action A_i as a set of predesigned joint angels Ω after the robot hand grasping the target object. For example, the command "rotate a handle", after that the robot hand detecting the object and grasping it, is simply the change of joint angle in the joint near the end-effector. We predesign a set of actions and store the sequences of joint angle changes for each action.

Before the real manipulation, the robot will firstly be calibrated to calculate the transformation matrix T between the image coordinate system and the robot base frame. During practical tasks, after the robot receives a command C from the operator, it will decompose the command into objects O and actions A, and follow the work flow shown in Figure 4. After the detection of object parts O_i, the robot will transform the coordinate P_o of its bounding box into robot workspace $P_r = T * P_o$. Then the robot hand can grasp the resultant cubic area to get the object and change its joint angles based on stored A_i. The manipulation is complete when all the (O_i, A_i) pairs are processed sequentially by the robot.

One example is given in Figure 5, where the robot is commanded to grab a toolbox. The robot firstly finds the location of the toolbox and its handle, and then approaches to the handle after transforming its location into robot workspace. After that, the robot grasps the handle and changes its joint angles to move it.

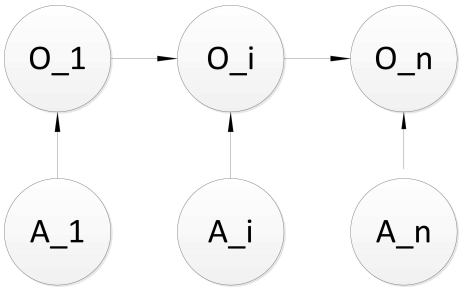

Fig. 4: Manipulation planning: each manipulation command is composed of a sequence of actions A_i on target object part O_i

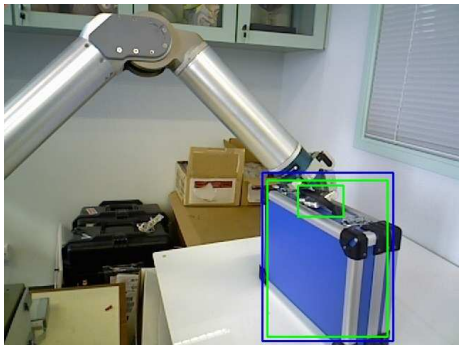

Fig. 5: The manipulation process: after the robot detects objects and object parts from the sensor data, it transforms their locations to coordinates in robot workspace. Then the robot hand approaches these locations to grasp target regions, and then performs predesigned actions like rotation and translation.

IV. Experiment

A. Recognition Setup

We tested our model for object recognition on a RGB-D data set [15] that included 51 categories of objects, and for each category, we randomly selected 400 samples for training and 100 samples for testing. We split each training sample into $10*10$ regions, and then labeled its meaningful object parts manually. To construct the feature words for each region, we concatenate color histogram and fast point gradient histogram. For SURF and HOG feature, we build feature vector with bag of words method. First, we extract SURF and HOG features from all the training samples, and build 64 clusters for each feature type with k-means clustering algorithm. After that, we construct two feature words, each with the length of 64. Concatenating all these features together, we get a feature vector with the length of 223. To compare our model with other methods, we also carry out experiments on RGB data and depth data, in addition to the RGB-D data. In the case of RGB data, we exclude fast point gradient histogram from the feature vector, because this feature is extracted from the depth data; in the case of depth data, we only use the fast point gradient histogram. All the other experiment settings are the same.

The feature words, together with part-based labeling, are

used to train our object model based on latent support vector machine. In the recognition phase, we extract feature words from the testing samples, apply our object model to recognizing the object parts, and then use the recognized object parts to identify the object categories.

B. Evaluation

We repeat our experiments for 10 times and calculate the average recognition accuracy. The result is shown in Figure 6.

Our model recognizes the object categories more accurately than established methods on the RGB-D data set. This shows the advantage of physically meaningful part-based object model over other methods. However, because some objects have no obvious part-based structure, our method has only slightly better performance over that part of them.

Our model has deficiency in that it requires too much labeling work, including labels for each object and all its composing parts. To make sure all learned object parts are meaningful for the robot, we have to manually label all those areas, as most automatically learned object parts are feature regions without physical meaning. We will try to reduce the amount of manual labeling by including robot-based active learning in our future work.

C. Manipulation Setup

We implemented our model with a WAM robot arm and Xtion Pro depth sensor in Robot Operating System, and tested this method on three manipulation tasks, including opening a paper box, grabbing a toolbox and pushing a chair. During the training stage, we collected 360 samples of point cloud data for each object from 4 view angles in 5 light conditions, and labeled all the object parts in these samples, like the legs and back of the chair, the handle and bumper of the tool box, and the paper boards of the paper box. Then we split these samples into $10 * 10$ regions and extracted feature words from everyone of them. These feature words were used to train our object models, and the learned parameters were used in manipulation tasks. When the robot received command C from the operator, it decomposed the command into target object and target action, and then the robot captured new data with Xtion Pro to detect designated objects. The detections were transformed into robot workspace. The robot grasps them and manipulates them based on the target actions.

D. Evaluation

We repeated each manipulation task for 20 times, and computed the ratio of successful manipulations over total manipulation attempts. The success rate of object manipulation is shown in Figure 8. One example of the manipulation sequences is shown in Figure 7. In manipulation tasks, our model detects the object and its composing parts simultaneously, and the detections are used in manipulation planning. Each manipulation task is composed of a sequence of individual action on designated object part. For the chair and paper box, the target object parts have a simple structure and the robot can easily act on them, so these two manipulation tasks have a relatively high success rate. For the toolbox, the success rate is quite low, because the robot hand can't always grasp the hole area inside the toolbox handle based on our model's detections. In future work, we will include more types of object parts during learning, so that the robot can manipulate structured objects more accurately.

target object	chair	toolbox	paper box
success rate	0.80	0.55	0.75

Fig. 8: Manipulation success rate on three tasks: we repeat each task for 20 times, and then count the number of successful manipulations to compute the success rate.

V. Conclusions

In this article, we proposed a multilevel part-based object structure, and used latent support vector machine as a core learning machine for training. Our model features in that all composing parts are physically meaningful and manipulable for a robot, so this model can enhance the robotic object manipulation tasks significantly. We compared the object recognition accuracy of our model with the established methods on a point cloud data set, and then demonstrated the ability of our model on three practical manipulation tasks with a robot arm. The result shows that a real robot can work better on both object recognition and object manipulation with multilevel part-based object model.

The deficiency of this model is that it requires detailed labels of all object parts, which is labor-consuming for practical robot tasks. In the future, we will use robot-based active learning to automate the part-based labeling process. Besides, we will try novel unsupervised methods like deep belief network to train such a structure.

References

[1] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. Le, A. Wellman, and A. Y. Ng, "High-accuracy 3d sensing for mobile manipulation: Improving object detection and door opening," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, may 2009, pp. 2816 –2822.

[2] D. Kragic, M. Bjorkman, H. I. Christensen, and J.-O. Eklundh, "Vision for robotic object manipulation in domestic settings," *Robotics and Autonomous Systems*, vol. 52, no. 1, pp. 85 – 100, 2005.

[3] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ser. ICRA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3534–3541.

[4] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, 2003, pp. II–264–II–271 vol.2.

[5] A. Quattoni, M. Collins, and T. Darrell, "Conditional random fields for object recognition," in *In NIPS*. MIT Press, 2004, pp. 1097–1104.

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[7] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009, also published as a book. Now Publishers, 2009.

Methods	RGB	Depth	RGB-D
SIFT+Texton+Color Histogram+Spin Images+3D Bounding Boxes	74.5	64.7	83.8
Sparse Distance Learning	78.6	70.2	85.4
RGB-D Kernel Descriptors	80.7	80.3	86.5
Hierarchical Matching Pursuit	82.4	81.2	87.5
Multilevel Part-based Model	85.2	81.5	89.7

Fig. 6: Recognition accuracy on RGB-D data set: the accuracies of other methods are cited from [16]. We modify the feature word to simulate the case when only RGB data or depth data is available.

Fig. 7: Demonstration of the manipulation process: the robot detects objects and object parts first, and then grasps target object parts, followed by rotation and translation.

[8] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 609–616.

[9] A. Farhadi, I. Endres, and D. Hoiem, "Attribute-centric recognition for cross-category generalization," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, june 2010, pp. 2352 –2359.

[10] G. Bouchard and B. Triggs, "Hierarchical part-based visual object categorization," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 2005, pp. 710–715 vol. 1.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 2005, pp. 886–893 vol. 1.

[13] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *The IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 05/2009 2009.

[14] T. Joachims, *Making large-scale support vector machine learning practical*. MIT Press, Cambridge, MA, 1998.

[15] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 1817–1824.

[16] L. Bo, X. Ren, and D. Fox, "Unsupervised feature learning for rgb-d based object recognition," in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics, J. P. Desai, G. Dudek, O. Khatib, and V. Kumar, Eds. Springer International Publishing, 2013, vol. 88, pp. 387–402. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-00065-7_27