RGB-D Object Recognition and Grasp Detection using Hierarchical Cascaded Forests

Umar Asif, Mohammed Bennamoun, and Ferdous Sohel

***Abstract*—This paper presents an efficient framework to perform recognition and grasp-detection of objects from RGB-D images of real scenes. The framework uses a novel architecture of hierarchical cascaded forests, where object-class and grasp-pose probabilities are computed at different levels of an image hierarchy (e.g., patch- and object-levels), and fused to infer the class, and the grasp of unseen objects. We introduce a novel training objective function which minimizes the uncertainties of the class labels and the grasp ground truths at the leaves of the forests, thereby enabling the framework to perform the recognition and grasp detection of objects. Our objective function is learnt on features which are extracted from RGB-D point clouds of the objects. For that, we propose a novel method to encode an RGB-D point cloud into a representation which facilitates the use of large Convolution Neural Networks (CNNs) to extract discriminative features from RGB-D images. We evaluate our framework on challenging object datasets where we demonstrate that our framework outperforms the state-of-the-art methods in terms of object recognition and grasp detection accuracies. We also show experiments using live video streams from a Kinect mounted on our in-house robotic platform.**

***Index Terms*—RGB-D object recognition, grasp detection, robotic grasping.**

I. Introduction

The recognition and grasp-detection of objects are crucial capabilities of an autonomous robot for visual perception and interaction with the real world. Object recognition and grasp detection in complex environments (e.g., in the presence of occlusion and clutter) are considered to be highly challenging tasks due to various factors such as: the segmentation of the visual information into object-hypotheses, the recognition of these hypotheses, disambiguation between visually/structurally similar objects, and the need for robustness to deal with real-world noise (e.g., variable lighting conditions, shadows, and noisy measurements from low cost sensors e.g., Kinect). Humans see novel objects and almost instinctively understand how to pick them up. Current robotic object recognition and grasp-detection systems, on the other hand, lag far behind the human performance levels. Given a noisy RGB-D view of an object, this paper presents an approach to recognizing the object and to find a valid grasp for robotic grasping. In the following, we briefly explain the challenges, and describe the state-of-the-art work and our proposed contributions for object recognition and grasp-detection. **Object recognition** deals broadly with two different problems: instance recognition and category recognition. An instance (e.g., "coffee mug") represents a unique object, whereas a category (e.g., "mug") represents instances which share similar features (e.g., shape or structure). **In instance recognition**, the task is to recognize the exact physical instance of an object which was previously learnt during the training phase. During testing, the object may then appear in a different setting (e.g., a different pose, in a different scene, under different lighting or different viewpoint) compared to the training instance. **In category recognition**, a set of training images with corresponding labels is presented offline. The labels group all object instances into a set of categories (e.g. all instances of mugs are assigned to the label "mug"). During testing, unseen instances of a given object category are presented, and the task of the algorithm is to assign the correct label to the object in the image.

Object recognition is affected by three main challenges: **i)** intra-class variance due to the difference in appearance of objects from the same category (e.g., red vs. green mug), **ii)** inter-class similarity due to the similar shape of objects from different categories (e.g., lemon vs. lime), and **iii)** the recognition time and computational complexity, which generally scale linearly with respect to the number of learned object-classes. Methods (e.g., [1], [2]) extract feature descriptors from the training models and the test scene and use a descriptor matching technique to determine model-to-scene point correspondences which are then used to generate a specific object hypothesis in the scene. In this context, **local descriptors** (e.g., [3]) perform well on objects with locally rich geometries or with high textures, they do not, however, efficiently handle simple objects with repetitive structures or objects with low/no textures. **Global descriptors** (e.g., [1]), on the other hand, are beneficial in terms of computational efficiency and memory footprint compared to the local descriptors (since each object is characterized by one single descriptor). However, global descriptors are less accurate in the presence of partial occlusions and require an accurate prior segmentation of the scene, which restricts their use in scenarios where objects highly occlude each others or in situations where an accurate prior segmentation is not available. Other methods (e.g., [4]–[6]) use the popular Bag-of-Words (BOW) approach to quantize local feature descriptors into visual words in a pre-defined visual vocabulary. The generated visual words are then used for classification. However, the feature quantization step in most of the BOW-based methods is computationally expensive and restricts their use in real-world robotic applications. Furthermore, the loss of information during the quantization step results in visual words which are often not discriminative enough for large scale image classification applications [7].

Robotic grasping in real-world environments is challenging

Umar Asif is with IBM Research Australia, Mohammed Bennamoun is with the School of CSSE, The University of Western Australia, Australia. Ferdous Sohel is with Murdoch University, Australia. Tel.: +61-86488-3455, email:umarasif@au1.ibm.com, This work was supported by Australian Research Council grants (DP150100294, DE120102960).

because it depends on the pose of the robotic gripper as well as the structural properties of the object to be grasped. Many recent works (e.g., [8]–[10]) treat the perception aspect of robotic grasping as a detection/recognition problem whereby, given a partial view of the object, the goal is to predict a grasp rectangle that defines the pose of the gripper to grasp the object. Specifically, the approaches in [8]–[10] extract rich features from the image data and learn a mapping from the features-set to grasp quality using a set of training objects. This knowledge is then used to predict grasp rectangles for resembling, or unknown objects. Although these approaches have produced impressive performances in recognizing and grasping known and familiar objects, they still struggle with challenges including: **i)** the requirement to design input features that are able to generalize to unknown objects, and **ii)** overall fast runtimes. In this paper, we propose to learn rich and highly discriminative features through transfer learning [11] in order to improve the recognition accuracies, and introduce a unified framework for object recognition and grasp detection to achieve real-time runtime. Earlier versions of this work appeared in [12] and [13], where we introduced a Cascaded Architecture of Random Forests (CaRFs) to recognize object categories in RGB-D point clouds. In this paper, we present an enhanced and extended version of CaRFs to handle other high-level tasks such as grasp detection for objects in real-world scenes. In addition to our work in [12] and [13], the specific contributions of this paper are as follows:

1) We present a unified framework (see Sec. III) for the recognition and grasp detection of RGB-D objects in real-world scenes. For that, we propose a novel training objective function (see Sec. VI-C), where we learn discrete and continuous predictors based on discrete object-class labels and continuous grasp-pose ground-truths at different levels of the image hierarchy. The proposed objective function decreases the uncertainties of object-class and grasp-pose probability distributions at the leaves of the hierarchical forests, thereby enabling the predictions of object classes and grasp poses.
2) We introduce a novel representation of RGB-D data to extract highly discriminative features using pre-trained CNNs. For that, we propose a method (see Sec. V-A), which captures the appearance and structural information of an RGB-D point cloud through multiple feature maps (RGB color, LAB color gradients, local surface normals, local orientations of surface normals, and projected distances of the points of the point cloud). We term this representation as "STructural EMbedding" (STEM) of RGB-D data. We show that, CNN-features extracted using STEM feature maps yield substantial improvements over the naive use of raw RGB-D images for the tasks of object recognition, and grasp detection (see Sec. IX-C).
3) We present an extensive evaluation of the proposed framework on two challenging datasets: **i)** the Washington RGB-D object dataset [4] and Washington Scene dataset [14] for object recognition, and **ii)** the Cornell Grasping object dataset [15] for object recognition and grasp detection. In experiments, we show that the proposed framework produces substantial improvements in the recognition and grasp-detection accuracies compared with the state-of-the-art methods (see Sec. IX).
4) We present real-world robotic experiments using live video streams from a Kinect (mounted on our in-house robotic platform named "AIPAR"), and demonstrate the capabilities of the proposed framework to successfully generalize to unknown objects with success rates of up to 93% for object recognition and 92% to execute successful grasps (see Sec. IX-E, and our video in the supplementary material).

II. Related Work

In this section we review the related work for object recognition and grasp detection.

A. Object Recognition

State-of-the-art methods (e.g., [4], [5], [16]) generalise the "Bag-of-Words" (BOW) approach and combine several local RGB and depth features for object recognition. For instance, Lai et al. [4] used a combination of several hand-crafted features (e.g., SIFT, color histograms, Spin Images, 3d bounding boxes) in a BOW model to perform object recognition on an RGBD object dataset [4]. In [17], kernel descriptors were proposed, which showed higher recognition accuracies compared with the hand-designed feature sets on the WRGBD object dataset [4]. Recently, Bo et al. [16] proposed a multi-layer sparse coding approach to unsupervised feature learning and demonstrated state-of-the-art performances in object recognition. Their approach uses Hierarchical Matching Pursuit (HMP) to learn hierarchical feature representations from RGB-D images of the objects in a top-down learning manner. Although, these sparse-coding based methods are highly efficient in learning discriminative features, they are computationally expensive which restricts their use in real-time robotic applications. Our work is related to these methods in the sense that, our framework also learns a hierarchical image representation to recognize objects. However, instead of using a BOW approach, we compute object-class probabilities at different levels of the image hierarchy and fuse the probabilities into a cumulative probabilistic output which is used for the final inferencing.

B. Grasp Detection

Recently, learning-based approaches (e.g., [8], [9], [18]) have shown their ability to produce an effective performance to grasp unknown objects [9]. These approaches extract rich features from the image data and learn a mapping from the features-set to grasp-quality using a set of training objects. This knowledge is then used to predict grasps for resembling, or unknown objects. For instance, in [9], grasp hypotheses were learned based on a set of 2D image features (edge and color) using synthetic data to grasp new objects. Koostra et al. [19] presented an early cognitive system to learn the grasping of unknown objects using edge and texture features. Ekvall et. al. [20] used shape-based approximations to

learn approach vectors which were used as potential grasps. Although, learning-based approaches have shown impressive performance in grasping known and familiar objects, these approaches still require a significant hand-engineering effort to design effective input features that are able to generalize to unknown objects. Lenz et al. [8] successfully addressed this challenge with the use of multiple CNNs for feature representation and final classification in a sliding window detection pipeline for grasp detection. However, their method operates at 13.5 seconds per frame with an overall accuracy of around 75 percent [8], [10]. This translates to a delay of 13.5 seconds between the recognition and grasp-detection of the object. Therefore, the high computational cost of their method restricts its use in real-time industrial applications, where a fast response time is highly important. Recently, Redmon et al. [21] applied a large CNN for detecting grasp candidates using the Cornell grasp dataset [15]. Due to the small amount of training data in the Cornell dataset, the authors pre-trained the CNN on ImageNet (which requires several days of training) and performed extensive augmentation (3000 times per image) to fine-tune on the Cornell dataset. We address the same problem as in [8], [21] but use a different pipeline for the pre-processing of the RGB-D data and a different framework for grasp-detection, which is capable of higher grasp detection accuracy.

III. Overview of the Proposed Framework

Fig. 1 shows the main steps of our proposed framework. Given an input RGB and a depth image (Fig. 1-A), the **first step** is to perform a segmentation of the scene (Fig. 1-B). This is achieved with the use of our previous segmentation algorithms [12], [22]–[24] (selected for their efficiency and accuracy, though any RGB-D based superpixel algorithm can be used). Specifically, we use the Hierarchical Pointcloud Decomposition (HPD) algorithm in [12] (see Sec. IV) to generate a hierarchical segmentation of the scene into mid-level segments (termed *surfels*), and high-level object hypotheses (termed *objects*). In the **second step**, for each object-hypothesis (of the segmented scene), the framework generates a STEM representation (see Sec. V-A) and extracts CNN-based features (see Sec. V-B). In the **third step**, the framework uses hierarchical cascaded forests (Fig. 1-D) to compute object class-label and grasp-pose probabilities at the surfel- and object-levels. These probabilities are fused into a cumulative probabilistic output (see Sec. VI), which is used to infer the class label and grasp-pose of the target object (Fig. 1-E). **Finally**, the framework uses a grasp synthesis algorithm (see Sec. VII) to perform robotic grasping of the target object (Fig. 1-E).

IV. Hierarchical Pointcloud Decomposition

Our goal here is to generate mid-level surfels $\mathcal{S}$, and high-level object hypotheses $\mathcal{O}$ from a given point cloud. To achieve this, we follow a two step procedure. **In the first step**, we group structurally similar points (i.e., points with close 3D proximities, and similar local- and global-orientations of their surface normals) into distinct clusters. This is achieved with the use of a clustering algorithm [24], where initial surfel centers ($s_k, k = 1, ..., N_s$ instantiated on a grid space of N_s cells as shown in Fig. 1-G), are iteratively grown into patches $\mathcal{S} = \{s_k\}_{k=1}^{N_s}$ (see Fig. 1-H). **In the second step**, we combine the surfels based on their perceptual grouping relationships (i.e., their local shape convexity and co-planarity) [12], to generate high-level object hypotheses $\mathcal{O}$ (see Fig. 1-I).

V. Feature Representation

Recently, deep convolutional neural networks (CNNs) [25] have gained a huge interest in image classification by demonstrating state-of-the-art performance in various computer-vision related tasks such object detection and semantic segmentation [26]. With the recent widespread availability of low-cost RGB-D sensors (e.g., Kinect), the use of RGB-D data, as opposed to simple RGB data, has demonstrated a substantial improvement in object recognition [27], and grasp detection [8] performances. However, the limited amount of currently available RGB-D-based labeled data is insufficient to train a large CNN from scratch. To overcome this problem, we propose a method to represent RGB-D data of a given point cloud by multiple feature channels (termed STEM representation), which facilitate transfer learning in order to learn rich and highly discriminative multi-modal features using large CNN models even when the available training data is scarce (e.g., Washington RGB-D dataset [4], and Cornell Grasping dataset [15]). In the following, we describe the computation of the proposed STEM representation and the extraction of STEM-based features at the surfel- and object-levels.

A. Proposed STructural EMbedding (STEM)

STEM encodes the appearance and structural characteristics of an RGB-D point cloud in terms of five feature maps as shown in Fig. 2-Left. The first feature map $\boldsymbol{f}_{rgb}$ captures the original RGB color values of the points of the point cloud. The second feature map $\boldsymbol{f}_{lab}$ captures the most dominant gradients of the points in CIELab-colorspace. The third feature map $\boldsymbol{f}_{normals}$ captures the local surface normals of the points of the point cloud. The fourth feature map $\boldsymbol{f}_{angles}$ captures the local pose-invariant orientations of the surface normals of the points of the point cloud. The fifth feature map $\boldsymbol{f}_{dist}$ captures the projected distances of the points (of the point cloud) with respect to the centroid of the point cloud. Specifically, the feature map $\boldsymbol{f}_{lab}$ is composed of three channels represented by gradients ($\mathcal{G}$), gradient-directions (Γ), and raw intensity values of the L image-channel of the CIELab color image I_{lab}, respectively. To build $\mathcal{G}$ and Γ, the algorithm first computes the most dominant oriented gradients (∇) and gradient directions (τ), independently from the L, A, and B channels of image I_{LAB}. For a pixel i, ∇ and τ are given by:

$$\begin{aligned} \nabla_i(I_f, \sigma, \theta) &= (\boldsymbol{v}_a \times \boldsymbol{v}_b)/||\boldsymbol{v}_a \times \boldsymbol{v}_b||, \\ \tau_i(I_f, \sigma, \theta) &= atan2(||\boldsymbol{v}_a \times \boldsymbol{v}_b||, \boldsymbol{v}_a \cdot \mathrm{v}_b), \end{aligned} \tag{1}$$

where $I_f \in \{L, A, B\}$, the vectors $\boldsymbol{v}_a$ (defined between the top and bottom pixels to the query pixel) and $\boldsymbol{v}_b$ (defined between the left and right pixels of the query pixel) are defined in a 3D space where their first two components correspond to the 2D

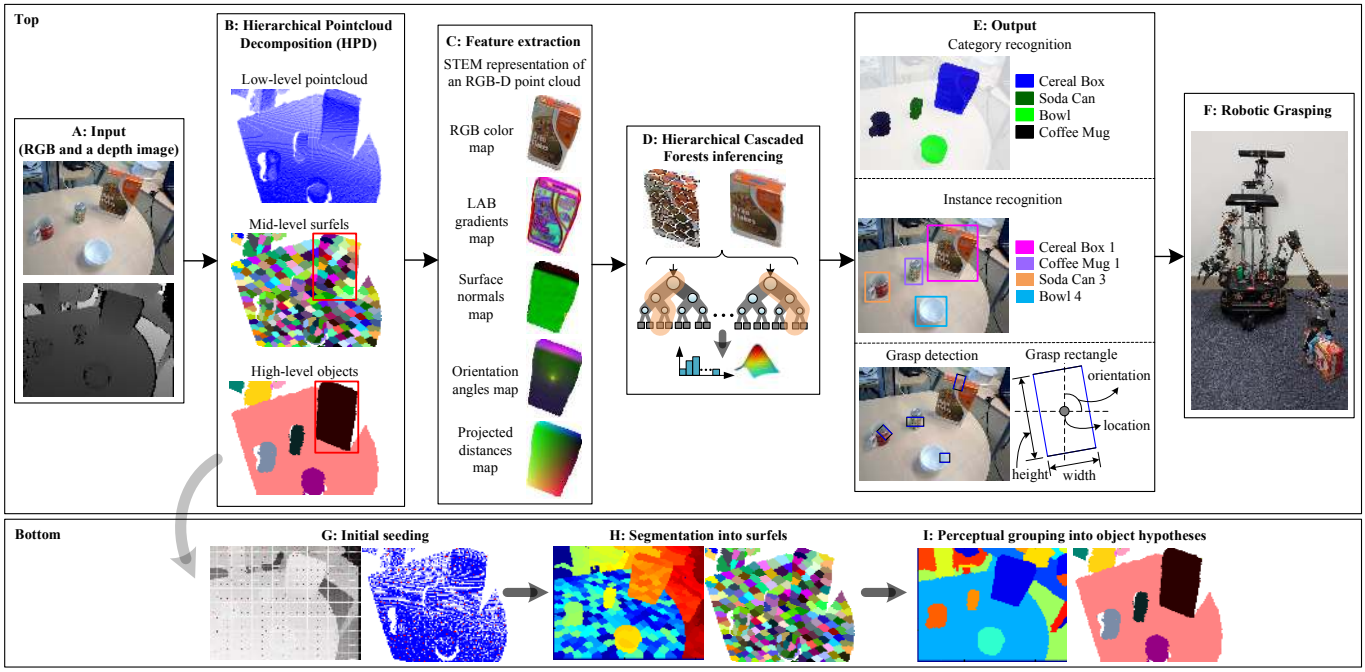

Fig. 1. **Top:** An overview of the proposed framework. Given an input RGB and a depth image of the scene (A), the proposed framework first segments the scene into surfels and objects (B). Next, for a query object (e.g., cereal box highlighted in red rectangle), the framework extracts features (C) and uses hierarchical cascaded Random Forests (D) to perform object recognition and grasp detection (E). The inferred grasp pose is used to perform robotic grasping (F). **Bottom:** Hierarchical Point cloud Decomposition (HPD) of a scene into surfels (H) and objects (I). Figure best viewed in color.

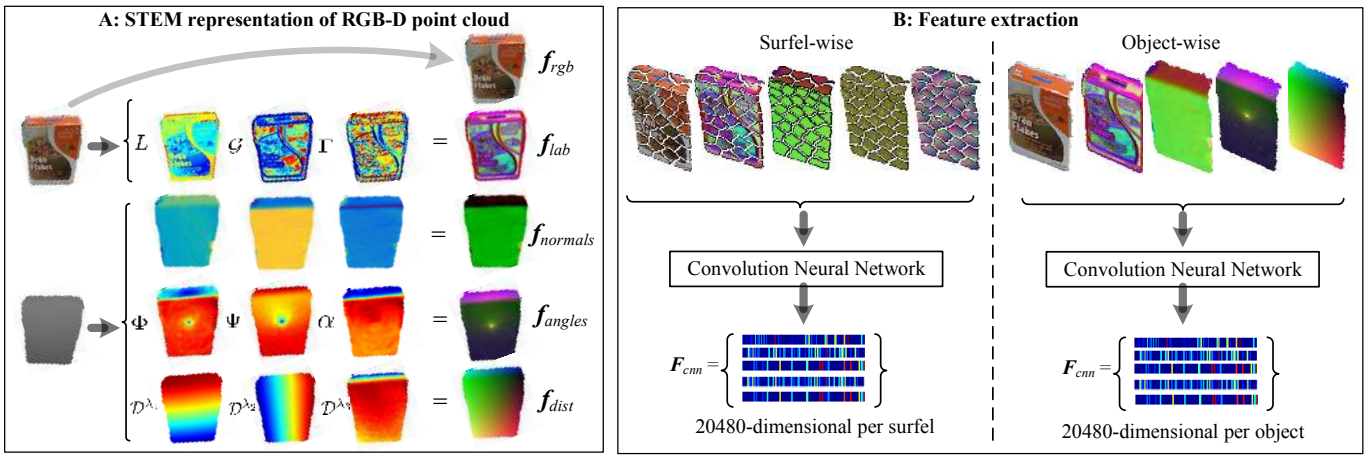

Fig. 2. **A:** Our STEM encoding of a point cloud. The feature maps $\boldsymbol{f}_{lab}$ and $\boldsymbol{f}_{normals}$ are computed for each point with respect to its local neighbourhood, whereas the feature maps $\boldsymbol{f}_{angles}$ and $\boldsymbol{f}_{dist}$ are computed with respect to the centroid of the point cloud. **B:** Feature extraction at different image level. At the surfel-level, CNN-features are computed for each surfel. At the object-level, CNN-features are computed for the entire object.

locations of the pixels and their third component corresponds to the intensity values (at the corresponding pixel locations) in the color channels. The term σ represents the scale and θ represents the orientation at which the gradient is computed. Next, the algorithm linearly combines the gradient magnitudes and gradient directions from the three color channels into multi-scale oriented gradient responses $\mathcal{G}$ and Γ, as follows:

$$\begin{aligned} \mathcal{G}_i &= \textstyle\sum_{\sigma} \sum_{I_f} \operatorname{argmax}_{\theta} \nabla_i(I_f, \sigma, \theta), \\ \Gamma_i &= \textstyle\sum_{\sigma} \sum_{I_f} \operatorname{argmax}_{\theta} \tau_i(I_f, \sigma, \theta). \end{aligned} \tag{2}$$

Finally, the feature map $\boldsymbol{f}_{lab}$ is constructed by normalizing the $\mathcal{G}$, Γ, and L values between 0 and 255.

$$\boldsymbol{f}_{lab} = \{||\mathcal{G}_i||, ||\Gamma_i||, ||L_i||\}, \forall i \in I_{LAB}. \tag{3}$$

The feature map $\boldsymbol{f}_{angles}$, encodes the orientations (Φ, Ψ, and α) of the local surface normals of the points in a point cloud $\mathcal{P}$ with respect to its centroid $\boldsymbol{c}_{\mathcal{P}}$. For a point $\boldsymbol{p}_i \in \mathcal{P}$, the orientations Φ, Ψ, and α are computed with respect to the centroid of the point cloud as in [1], but without the viewpoint component of [1] to retain rotational invariance (with respect to the viewpoint) which is required to build a general representation of an object when viewed from different viewpoints. Specifically, the orientations are computed as:

$$\begin{aligned} \Phi_{\boldsymbol{p}_i} = acos(\boldsymbol{n}_i \cdot \boldsymbol{v}_n), \Psi_{\boldsymbol{p}_i} = acos(\boldsymbol{n}_{\mathcal{P}} \cdot \boldsymbol{v}_n), \\ \alpha_{\boldsymbol{p}_i} = atan2(\rho, \nu), \rho = (\boldsymbol{n}_{\mathcal{P}} \times (\boldsymbol{v}_n \times \boldsymbol{n}_i)) \cdot \boldsymbol{n}_i, \nu = \boldsymbol{n}_{\mathcal{P}} \cdot \boldsymbol{n}_i, \end{aligned} \tag{4}$$

where $\boldsymbol{n}_i$ represents the surface normal of the point $\boldsymbol{p}_i$, $\boldsymbol{v}_n$ is a normalized vector between $\boldsymbol{p}_i$ and the centroid $\boldsymbol{c}_{\mathcal{P}}$, and $\boldsymbol{n}_{\mathcal{P}}$

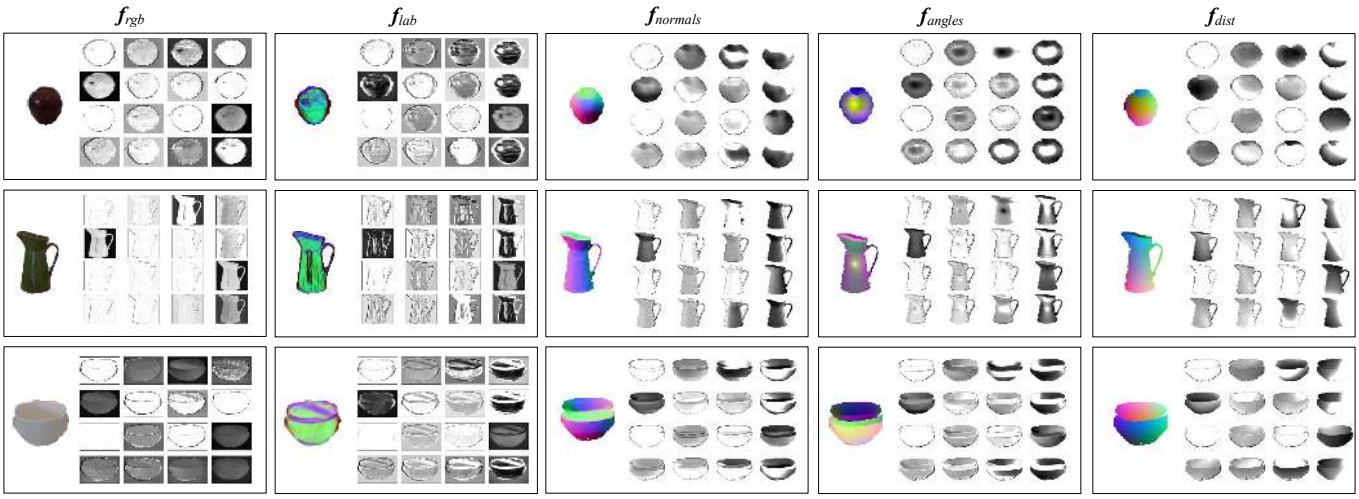

Fig. 3. CNN activations for sample images. From left to right, the columns show the CNN responses from the first convolutional layer for the $\boldsymbol{f}_{rgb}$, $\boldsymbol{f}_{lab}$, $\boldsymbol{f}_{normals}$, $\boldsymbol{f}_{angles}$, and $\boldsymbol{f}_{dist}$ input feature maps respectively. Note that the CNN responses are considerably different for different feature maps and therefore provide complimentary information.

represents the surface normal of the point cloud (estimated by taking the mean of the surface normals of all points in the point cloud). Finally, the feature map $\boldsymbol{f}_{angles}$ is constructed by normalizing the Φ, Ψ, and α values between 0 and 255.

$$\boldsymbol{f}_{angles} = \{||\Phi_{\boldsymbol{p}_i}||, ||\Psi_{\boldsymbol{p}_i}||, ||\alpha_{\boldsymbol{p}_i}||\}, \forall \boldsymbol{p}_i \in \mathcal{P}. \tag{5}$$

The feature map $\boldsymbol{f}_{dist}$ captures the 3D geometry of a point cloud. It is composed of three feature channels $\mathcal{D}^{\lambda_1}$, $\mathcal{D}^{\lambda_2}$ and $\mathcal{D}^{\lambda_3}$. These channels represent the signed distances of the points of a point cloud with respect to its centroid, along each of the three eigenvectors ($\boldsymbol{v}_{\lambda_1}$, $\boldsymbol{v}_{\lambda_2}$ and $\boldsymbol{v}_{\lambda_3}$) of the scatter matrix of the points of the point cloud. Mathematically, $\mathcal{D}^{\lambda_1}$, $\mathcal{D}^{\lambda_2}$ and $\mathcal{D}^{\lambda_3}$ of a point $\boldsymbol{p}_i \in \mathcal{P}$ are computed as:

$$\begin{gathered}\mathcal{D}^{\lambda_1}_{\boldsymbol{p}_i} = (\boldsymbol{p}_i - \boldsymbol{c}_{\mathcal{P}}) \cdot \boldsymbol{v}_{\lambda_1}, \mathcal{D}^{\lambda_2}_{\boldsymbol{p}_i} = (\boldsymbol{p}_i - \boldsymbol{c}_{\mathcal{P}}) \cdot \boldsymbol{v}_{\lambda_2}, \\ \mathcal{D}^{\lambda_3}_{\boldsymbol{p}_i} = (\boldsymbol{p}_i - \boldsymbol{c}_{\mathcal{P}}) \cdot \boldsymbol{v}_{\lambda_3},\end{gathered} \tag{6}$$

Finally, the feature map $\boldsymbol{f}_{dist}$ is constructed by normalizing the $\mathcal{D}^{\lambda_1}_{\boldsymbol{p}_i}$, $\mathcal{D}^{\lambda_2}_{\boldsymbol{p}_i}$, and $\mathcal{D}^{\lambda_3}_{\boldsymbol{p}_i}$ values between 0 and 255.

$$\boldsymbol{f}_{dist} = \left\{||\mathcal{D}^{\lambda_1}_{\boldsymbol{p}_i}||, ||\mathcal{D}^{\lambda_2}_{\boldsymbol{p}_i}||, ||\mathcal{D}^{\lambda_3}_{\boldsymbol{p}_i}||\right\}, \forall \boldsymbol{p}_i \in \mathcal{P}. \tag{7}$$

Our proposed STEM representation offers an effective and computationally inexpensive encoding of RGB-D images. Our hypothesis, to be borne out in experiments, is that the CNN-based features extracted from the feature maps of the proposed STEM representation of RGB-D data provide more discriminative multi-modal information for large-scale classification compared to the features extracted from raw RGB and depth images. To elaborate on this, we visualize the responses learnt by the first convolutional layer of a CNN for the output feature maps of the proposed STEM encoding in Fig. 3. From the figure, one can conclude that the features which qualitatively appear in the CNN responses are considerably different for different feature maps and therefore provide complimentary multi-modal information during the final classification.

B. Feature Extraction

We used the CNN network of [11] to extract CNN-based feature vectors $\boldsymbol{F}_{cnn}$ at the surfel- and object-levels. At the surfel-level, $\boldsymbol{F}_{cnn}$ is computed for each surfel using the STEM feature values of the points within the query surfel. At the object-level, $\boldsymbol{F}_{cnn}$ is computed using the STEM feature maps for the entire object (see Fig. 2-Right). The network of [11] accepts a three-dimensional image of size 224×224 as input. Therefore, for a surfel or an object, the corresponding STEM feature maps are first resized to 224×224×3 dimensions and then independently fed into the CNN to extract $4096-$ dimensional feature vectors from the fully connected layer named *fc7* of the network. Finally, $\boldsymbol{F}_{cnn}$ is built by the concatenation of the 4096-dimensional feature vectors for all the feature maps of the STEM representation (see Fig. 2-Right).

VI. Hierarchical Cascaded Forests for Classification and Regression

A. Problem Description

We formulate the task of object grasping as an object classification and grasp regression problem, where the learning objective is to decrease object-class and grasp-pose uncertainties at the leaves of the forests. For this, we compute a posterior distribution $p(\boldsymbol{u}|\boldsymbol{x}) = p(y|\boldsymbol{x})p(\boldsymbol{r}|\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, ..., x_d) \in \mathbb{R}^d$ represents our $d-$dimensional input *feature vector*, and $\boldsymbol{u} \in \mathbb{R}^{1+n}$ is the $(1+n)-$dimensional output or *prediction* variable which has two components: **i)** a one-dimensional discrete classification component $y_k \in \mathcal{Y}$ (which represents the class label), and **ii)** an $n-$dimensional continuous regression component $\boldsymbol{r}_k \in \mathcal{R}$ which represents the grasp-pose of the data sample. We use the five dimensional representation for the grasp-pose proposed by Lenz et al. [8]. This representation gives the location and orientation of a two-finger gripper before it closes on an object. It is given by:

$$\boldsymbol{r}_k = (r^x_k, r^y_k, r^\phi_k, r^h_k, r^w_k), \tag{8}$$

where, (r^x_k, r^y_k) denotes the center, r^ϕ_k the orientation relative to the horizontal axis, r^h_k the height and r^w_k the width of the rectangle (see Fig. 1-E for an example). Learning the distribution $p(\boldsymbol{u}|\boldsymbol{x})$ allows us to make predictions about the class and grasp-pose for previously unseen test objects.

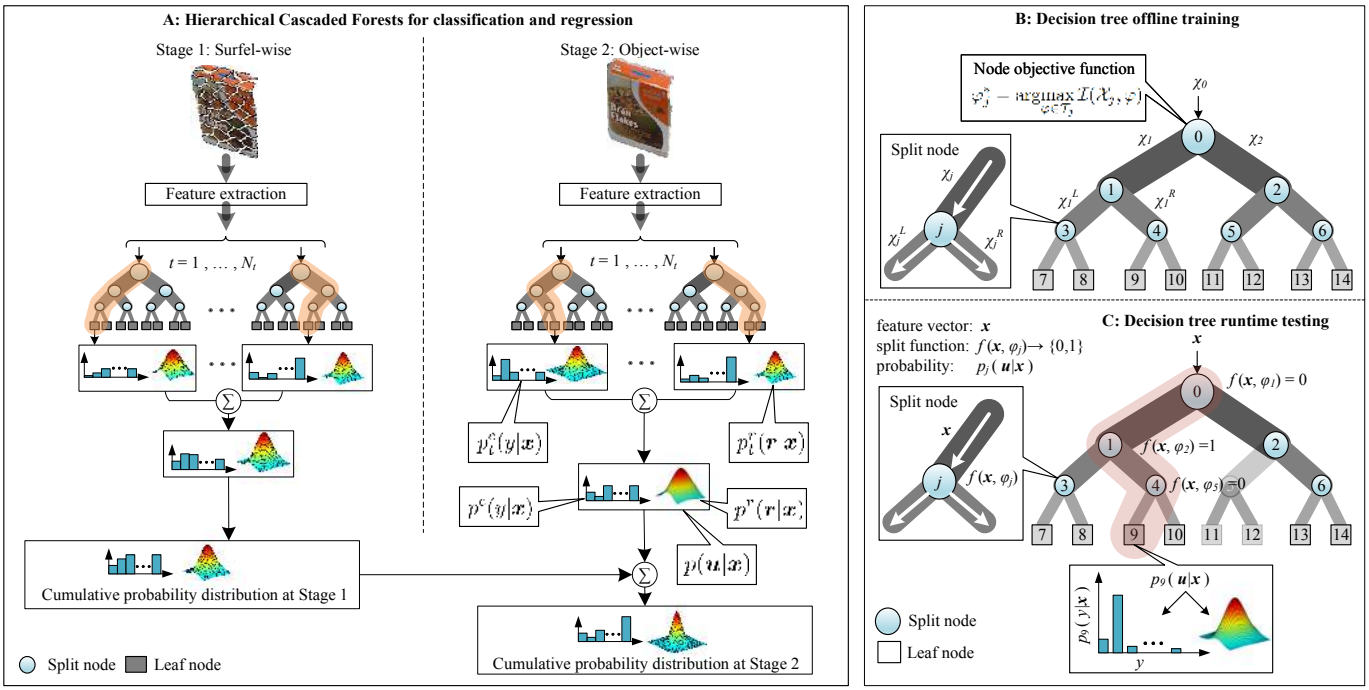

Fig. 4. **A:** An overview of the proposed hierarchical cascaded forests for object classification and grasp regression. For a given point cloud, the framework computes surfel- and object-wise probabilistic outputs and fuses them to infer the class label and grasp-pose of the target object. **B:** Training a decision tree involves sending the entire training subset $\mathcal{X}_0$ into the tree and optimizing the parameter φ of the split nodes. The edge thickness is proportional to the amount of training data traversing through it. **C:** During testing, at each split node, a binary test is applied and the feature sample $\boldsymbol{x}$ is sent to the appropriate child node. The process is repeated until a leaf node is reached (highlighted path) which stores the posterior probability distributions of object-class labels and grasp poses. Figure best viewed in color.

B. Proposed Architecture of Hierarchical Cascaded Forests

Fig. 4-A shows the architecture of the proposed framework which is comprised of two stages. **Stage 1**: We compute object-class and grasp-pose probabilistic distributions at the surfel-level using features extracted from the surfels. **Stage 2:** We compute probabilistic distributions at the object-level using features extracted from the entire object, and combine these probabilities with the probabilistic output (obtained from Stage 1) for all surfels which belong to the query object. This yields the final probabilistic output of the cascaded forests in terms of discrete object-class label distributions and continuous grasp-pose probability distributions. Specifically, we learn a probabilistic function E (Eq. 9), which is the combination of the output probabilities from two cascaded Random Forests (termed surfel- and object forests). For an object $o \in \mathcal{O}$, E_o is given by the sum of the probabilistic output of the object forest (i.e., $\wp_o$) and the mean of the probabilities of the surfels ($\mathcal{S}_o = \{s_k\}_{\forall k \in o}$) which belong to the query object o. The probability of an individual surfel s_k is given by the output of the surfel forest (i.e., $\wp_s$). The general form of our probabilistic function E can be written as:

$$E_o = \beta \wp_o + (1-\beta)\frac{1}{|\mathcal{S}_o|}\sum_{\forall s \in \mathcal{S}_o}(\wp_s), \tag{9}$$

where, $\wp$ for a sample $\boldsymbol{x}$ represents the probabilistic output of a forest (i.e., $p(\boldsymbol{u}|\boldsymbol{x})$), given by the average of the tree probabilities of all trees in the forest as shown in Fig. 4-A. The probabilities from the two classifiers are combined with $0 \leq \beta \leq 1$, being the trade-off parameter for balancing their relative influence. We empirically found that a value of $\beta = 0.6$ produced the best balance between the surfel-level and the object-level classifications. In the following, we describe in detail: our learning algorithm (see Sec. VI-C) and our inferencing algorithm (see Sec. VI-D).

C. Randomized Learning

An ensemble of randomized decision trees (termed Random Forest) is an ensemble of weak learners which are binary decision trees [28]. We train one Random Forest (RF) for the surfel-level, and one RF for the object-level. Our learning process is the same for each of the two RFs.

Let a forest $\mathcal{F} = \{T_t\}_{t=1}^{N_t}$ be a set of trees T_t, each one trained on a random subset $\mathcal{X}_0 \subseteq \{\boldsymbol{x}_i\}_{i=1}^{N_k}$ of the training data $\mathcal{X} = \{(\boldsymbol{x}_k, \boldsymbol{u}_k)\}_{k=1}^{N_k}$, where N_k is the number of training samples. For a given tree T_t, our learning procedure starts at the root node (i.e., $j = 0$ as illustrated in Fig. 4-B), where we learn a binary split function $f(\boldsymbol{x}, \varphi_j)$ which "best" splits the incoming training data $\mathcal{X}_j$ into left-subtree data and right-subtree data $\mathcal{X}_j^L$ and right-subtree data $\mathcal{X}_j^R$:

$$\begin{aligned} \mathcal{X}_j^L(\mathcal{X}_j, \varphi) &= \{(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X}_j | f(\boldsymbol{x}, \varphi_j) = 0\}, \\ \mathcal{X}_j^R(\mathcal{X}_j, \varphi) &= \{(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X}_j | f(\boldsymbol{x}, \varphi_j) = 1\}, \end{aligned} \tag{10}$$

where,

$$\begin{aligned} \mathcal{X}_j = \mathcal{X}_j^L \cup \mathcal{X}_j^R, \mathcal{X}_j^L \cap \mathcal{X}_j^R = \emptyset, \\ \mathcal{X}_j^L = \mathcal{X}_{2j+1}, \mathcal{X}_j^R = \mathcal{X}_{2j+2}. \end{aligned} \tag{11}$$

At each node j, the best split function parameter φ_j is chosen out of a set of randomly generated split parameter values $\mathcal{T} = \{\varphi\}$, which maximizes an objective function $\mathcal{I}$:

$$\varphi_j = \underset{\varphi \in \mathcal{T}_j}{\operatorname{argmax}}\, \mathcal{I}(\mathcal{X}_j, \varphi), \tag{12}$$

The splitting procedure is applied recursively to all the newly constructed nodes in a breadth-first training manner and the learning continues until a termination criterion is met. In this work, we used two termination conditions: **i)** the number of samples reaching the leaf node is smaller than a threshold (set to be equal to 2), or **ii)** a maximum tree depth is reached. The last created nodes are termed "leaf nodes" which store the probabilistic distributions (i.e., $p(\boldsymbol{u}|\boldsymbol{x})$) of the samples which reached these nodes. We also considered a hybrid training strategy, where we trained the first few tree levels with breadth-first and then switched to depth-first for the lower tree levels. However, the hybrid strategy did not produce any noticeable improvements in terms of inference speed/accuracy compared to a pure breadth-first training, except that the breadth-first uses a very large amount of memory when the tree gets deep. Since, we trained shallow trees using a limited amount of training data, we found the breadth-wise training to be equally performant compared to the hybrid/depth-first training strategies. The split functions that we used in our current setup are called axis-aligned functions. They are defined by: $f(\boldsymbol{x}, \varphi_j) = (v \cdot \boldsymbol{x} \geq \varphi_j)$, where v is a d-dimensional binary (random) vector and $\varphi_j \in \mathbb{R}$ is a threshold. Note that v has only one non-zero entry and thereby selects one dimension from the input feature space. In order to learn the probability $p(\boldsymbol{u}|\boldsymbol{x})$ at the split nodes, we propose an integrated quality measure $\mathcal{I}(\cdot)$, which is a linear weighted combination of a discrete information gain $\mathcal{I}_y$ and a continuous information gain $\mathcal{I}_r$. It is given by:

$$\mathcal{I}(\mathcal{X}_j, \varphi) = (1-\omega)\cdot\mathcal{I}_y(\mathcal{X}_j, \varphi)+\omega\cdot(1-e^{-d_p})\cdot\mathcal{I}_r(\mathcal{X}_j, \varphi), \quad (13)$$

where, the parameter $\omega \in \mathbb{R}$ controls the relative influence of the classification objective ($\mathcal{I}_y$) and the regression objective ($\mathcal{I}_r$) on the overall integrated quality measure $\mathcal{I}$. From Eq. 13, we observe that when ω is small, classification is preferred over regression and vice versa. In experiments, use ω to tune the proposed learning objective for optimal classification performance (i.e., object recognition), for optimal regression performance (i.e., grasp detection), and for optimal joint classification-regression performance (see Sec. IX-D). The classification objective $\mathcal{I}_y$ in Eq. 13 optimises the performance of object classification. It is given by:

$$\mathcal{I}_y(\mathcal{X}_j, \varphi) = H_y(\mathcal{X}_j) - \sum_{i\in\{L,R\}} \frac{|\mathcal{X}_j^i|}{|\mathcal{X}_j|} H_y(\mathcal{X}_j^i), \quad (14)$$

where $H_y(\cdot)$ is the Shannon entropy of the distribution of the training class labels in the set $\mathcal{X}_j$. It is given by:

$$H_y(\mathcal{X}_j) = -\sum_{y\in\mathcal{Y}} p(y|\mathcal{X}_j)\log(p(y|\mathcal{X}_j)), \quad (15)$$

where $p(y|\mathcal{X}_j)$ represents the empirical class distribution extracted from the samples, within the set $\mathcal{X}_j$, as a normalized histogram of the class labels. The regression objective $\mathcal{I}_r$ in Eq. 13 learns the regression aspect of the decision trees by measuring a continuous information gain, which optimises the regression performance of grasp-poses within a node. It is given by:

$$\mathcal{I}_r(\mathcal{X}_j, \varphi) = H_r(\mathcal{X}_j) - \sum_{i\in\{L,R\}} \frac{|\mathcal{X}_j^i|}{|\mathcal{X}_j|} H_r(\mathcal{X}_j^i), \quad (16)$$

where $H_r(\cdot)$ is the differential entropy of the set of training samples in the set $\mathcal{X}_j$. It is given by:

$$H_r(\mathcal{X}_j) = -\frac{1}{|\mathcal{X}_j|}\sum_{\boldsymbol{x}\in\mathcal{X}_j}\int_{\boldsymbol{r}} p(\boldsymbol{r}|\boldsymbol{x})\log p(\boldsymbol{r}|\boldsymbol{x})d\boldsymbol{r}, \quad (17)$$

where, $p(\boldsymbol{r}|\boldsymbol{x})$ is the conditional probability distribution of grasp-poses of the training data which arrive at the node. We model this conditional probability by an n-dimensional multivariate Gaussian distribution which can efficiently be stored in terms of their means and covariance matrices. It is given by:

$$p(\boldsymbol{r}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{r}; \bar{\boldsymbol{\mu}}_{\boldsymbol{r}}(\boldsymbol{x}), \Lambda_{\boldsymbol{r}}(\boldsymbol{x})), \quad (18)$$

where, $\bar{\boldsymbol{\mu}}_{\boldsymbol{r}}$ is the mean and $\Lambda_{\boldsymbol{r}}$ is the conditional $n \times n$ covariance matrix obtained from probabilistic Gaussian fitting to the training data arriving at the node:

$$\begin{aligned} \bar{\boldsymbol{\mu}}_{\boldsymbol{r}} &= \textstyle\int \boldsymbol{r}\cdot p(\boldsymbol{r}|\boldsymbol{x})d\boldsymbol{r}, \\ \Lambda_{\boldsymbol{r}} &= \textstyle\int(\boldsymbol{r}-\bar{\boldsymbol{\mu}}_{\boldsymbol{r}})(\boldsymbol{r}-\bar{\boldsymbol{\mu}}_{\boldsymbol{r}})^T\cdot p(\boldsymbol{r}|\boldsymbol{x})d\boldsymbol{r}. \end{aligned} \quad (19)$$

By substituting Eq. 18 in Eq. 17 we can re-write the differential entropy as:

$$H_r(\mathcal{X}_j) = \frac{1}{2}\log((2\pi e)^n|\Lambda_{\boldsymbol{r}}(\mathcal{X}_j)|), \quad (20)$$

where, $|\cdot|$ denotes the determinant of a matrix. Finally, substituting Eq. 20 in Eq. 16 yields the information gain for a multivariate continuous probabilistic distribution as:

$$\mathcal{I}_r(\mathcal{X}_j, \varphi) = \log|\Lambda_{\boldsymbol{r}}(\mathcal{X}_j)| - \sum_{i\in\{L,R\}} \frac{|\mathcal{X}_j^i|}{|\mathcal{X}_j|}\log|\Lambda_{\boldsymbol{r}}(\mathcal{X}_j^i)|. \quad (21)$$

At the end of the learning process we obtain: **i)** the optimum weak learners (split functions) associated with each node, **ii)** a learned tree structure, **iii)** a discrete class distribution $p(y|\boldsymbol{x})$ (in the form of normalized histograms of the class labels), and **iv)** a continuous probabilistic distribution $p(\boldsymbol{r}|\boldsymbol{x})$ (as the learned means $\bar{\boldsymbol{\mu}}_{\boldsymbol{r}}$ and the covariance matrices $\Lambda_{\boldsymbol{r}}$ of the regression targets $\boldsymbol{r}$) of the training data stored at each leaf.

D. Inferencing

During testing, a test sample $\boldsymbol{x}$ is traversed through each tree in the forest starting from the root node until it reaches a leaf node (see Fig. 4-C for an illustration). Since the split nodes act on features, the input test sample is likely to end up in a leaf associated with training samples which are all similar to the test sample. Thus, it is reasonable to assume that the associated label and the pose must also be similar to that of the training points in that leaf. This justifies the use of the label and pose statistics which are stored in that leaf to predict the label and the pose associated with the input test sample. The classification output for each tree T_t is captured using the conditional distribution $p^c_{T_t}(y|\boldsymbol{x})$, where y represents the categorical label. The probabilistic classification output of

Algorithm 1 : Inferencing of hierarchical cascaded forests.

Input: $\mathfrak{C} = \{\mathcal{F}_s, \mathcal{F}_o\}$, surfel- and object-wise trained Random Forests.
Output: $y^*, \boldsymbol{r}^*$, object class and grasp-pose predictions respectively.
$o \in \mathcal{O}$, set of object hypotheses of the segmented scene.
Function Inference($\mathfrak{C}, o$)
  $\mathcal{S}_o \leftarrow \{s_k\}_{\forall k \in o}$, set of surfels which belong to o
  $\wp_o \leftarrow$ ensembleProb($\mathcal{F}_o, o$) ▷ Eq. 22 and Eq. 25
  $\wp_s \leftarrow \varnothing$, initialize surfel-wise probability matrix
  for each $s_k \in \mathcal{S}_o$ **do**
    $\wp_s \leftarrow \wp_s +$ ensembleProb($\mathcal{F}_s, s_k$)
  $\wp_s \leftarrow \wp_s / |\mathcal{S}_o|$
  $E_o \leftarrow \wp_o + \wp_s$ ▷ Eq. 9
  $y^*, \boldsymbol{r}^* \leftarrow$ predict label and regression targets using E_o
Function ensembleProb($\mathcal{F}, \boldsymbol{x}$)
  $N_t \leftarrow |\mathcal{F}|$, initialize number of trees
  $P \leftarrow \varnothing$, initialize ensemble probability matrix
  for each tree $T_t \in \mathcal{F}$ **do**
    $p \leftarrow$ treeProb($T_t, \boldsymbol{x}$), $P \leftarrow P + p$
  return $P \leftarrow P / N_t$
Function treeProb($T, \boldsymbol{x}$)
  for each node $n_j \in T$ **do**
    if n_j is a split node **then**
      if $f(\boldsymbol{x}, \varphi_j) = 1$ **then**
        treeProb($T_R, \boldsymbol{x}$), send sample to right-subtree
      else
        treeProb($T_L, \boldsymbol{x}$), send sample to left-subtree
    else
      return $< p^c_T(y|\boldsymbol{x}), p^r_T(\boldsymbol{r}|\boldsymbol{x}) >$

the forest (i.e., $p^c(y|\boldsymbol{x})$) can then be defined as the average of the posterior class probabilities of all trees in the forest:

$$p^c(y|\boldsymbol{x}) = \frac{1}{N_t} \sum_{t=1}^{N_t} p^c_{T_t}(y|\boldsymbol{x}). \tag{22}$$

Finally, the class label y^* for a query object $o \in \mathcal{O}$ is obtained by substituting the object- and surfel-wise posterior probabilities ($p^c(y|\boldsymbol{x})$) in Eq. 9 and taking the Maximum A-Posteriori (MAP) estimate of the cumulative probabilistic output E (i.e., the class with the maximum score):

$$y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \left(E_o\right). \tag{23}$$

The regression output $p^r_{T_t}(\boldsymbol{r}|\boldsymbol{x})$ of the t^{th} tree is given by the multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{r}; \bar{\boldsymbol{\mu}}_{\boldsymbol{r}}(\boldsymbol{x}), \Lambda_{\boldsymbol{r}}(\boldsymbol{x}))$. It is given by:

$$p^r_{T_t}(\boldsymbol{r}|\boldsymbol{x}) = \frac{|\mathcal{X}_l|}{|\mathcal{X}_0|} \mathcal{N}(\boldsymbol{r}; \bar{\boldsymbol{\mu}}^l_{\boldsymbol{r}}(\boldsymbol{x}), \Lambda^l_{\boldsymbol{r}}(\boldsymbol{x})), \tag{24}$$

where, the vector $\bar{\boldsymbol{\mu}}^l_{\boldsymbol{r}}$ denotes the mean of all samples reaching the leaf l (i.e., $\mathcal{X}_l$), and $\Lambda^l_{\boldsymbol{r}}$ denotes the associated covariance matrix. The probabilistic output of the forest (i.e., $p^r(\boldsymbol{r}|\boldsymbol{x})$) is then given by the average of all trees in the forest:

$$p^r(\boldsymbol{r}|\boldsymbol{x}) = \frac{1}{N_t} \sum_{t=1}^{N_t} p^r_{T_t}(\boldsymbol{r}|\boldsymbol{x}). \tag{25}$$

The grasp inference $\boldsymbol{r}^*$ is then obtained by substituting the surfel- and object-wise posterior estimates ($p^r(\boldsymbol{r}|\boldsymbol{x})$) in Eq. 9 and taking the mean of the predicted regression targets. The inferencing algorithm is summarized in Algorithm 1.

VII. Grasp Synthesis

Here, we present our algorithm to perform robotic grasps using the inferred grasp-rectangle. Our algorithm proceeds in two steps: **i)** grasp generation, and **ii)** grasp execution. **For grasp generation**, the algorithm generates a grasp-configuration for a 2-finger gripper (in terms of two grasping points, gripper position and orientation). **For grasp execution**, the algorithm uses 7 degrees-of-freedom inverse kinematics to drive a robotic arm to grasp the target object.

A. Grasp Generation

Our general grasp notation $\boldsymbol{G}_{\boldsymbol{r}}$ for a grasp-rectangle $\boldsymbol{r}$ is defined as:

$$\boldsymbol{G}_{\boldsymbol{r}} = \{\boldsymbol{p}_g, \boldsymbol{q}_g, L^r, \boldsymbol{p}_{ap}\} \tag{26}$$

As illustrated in Fig. 5-B, $\boldsymbol{p}_g$ and $\boldsymbol{q}_g$ are the two grasping points associated with the grasp-rectangle. L^r is a local reference frame whose origin is the midpoint $\boldsymbol{p}_m$ of the grasp-rectangle. The z axis of L^r corresponds to the surface normal $\boldsymbol{n}_{p_m}$ of the point located at the centroid (r^x, r^y) of the grasp-rectangle. The x and y axes of L^r correspond to the unit direction vector $\hat{\boldsymbol{p}_w}$ along the width, and the unit direction vector $\hat{\boldsymbol{p}_h}$ along the height of the grasp-rectangle, respectively. $\boldsymbol{p}_{ap}$ is a point located at a distance d_{al} from L^r along $+\boldsymbol{n}_{p_m}$ (positive direction of the normal) and is used as the approach point to align the gripper for the corresponding grasp (see Fig. 5-F). To compute the grasping points $\boldsymbol{p}_g$ and $\boldsymbol{q}_g$, the algorithm finds a pair of grasping points which satisfy the reflection symmetry criteria in both the 2D image plane and the 3D Cartesian space (i.e., their surface normals minimize the angles θ_1 and θ_2 as shown in Fig. 5-A and Fig. 5-B). These grasping points are extracted from the set of points which lie within an enclosed search volume $\boldsymbol{V}^{\boldsymbol{r}}_g$ defined by the grasp-rectangle (see Fig. 5-B). The width and height of $\boldsymbol{V}^{\boldsymbol{r}}_g$ are defined by the width and height of the grasp-rectangle, respectively. The depth of $\boldsymbol{V}_g$ was set to 1cm in our experiments. For the case where $\boldsymbol{V}^{\boldsymbol{r}}_g$ does not contain a pair of points which satisfies the reflection symmetry criterion, the midpoints along the height of the grasp-rectangle are selected as the grasping points (see Fig. 5-C for an example).

B. Grasp Execution

To drive the gripper to the grasp location, our grasp-execution represents the following mapping:

$$\boldsymbol{G}_e = (\boldsymbol{p}_g, \boldsymbol{q}_g, L^r, \boldsymbol{p}_{ap}) \rightarrow (L^g, d_g). \tag{27}$$

As illustrated in Fig. 5-F, L^g denotes the gripper reference frame calibrated with respect to the camera. The orientation of L^g is such that $\boldsymbol{z}^{L^g}$(z-axis of L^g frame) is parallel to the grippers fingers, $\boldsymbol{x}^{L^g}$ connects the fingers, $\boldsymbol{y}^{L^g} = \boldsymbol{z}^{L^g} \times \boldsymbol{x}^{L^g}$, and the origin $\boldsymbol{p}^{L^g}$ is placed between the two fingers. The term d_g corresponds to the distance between the fingers (i.e.,

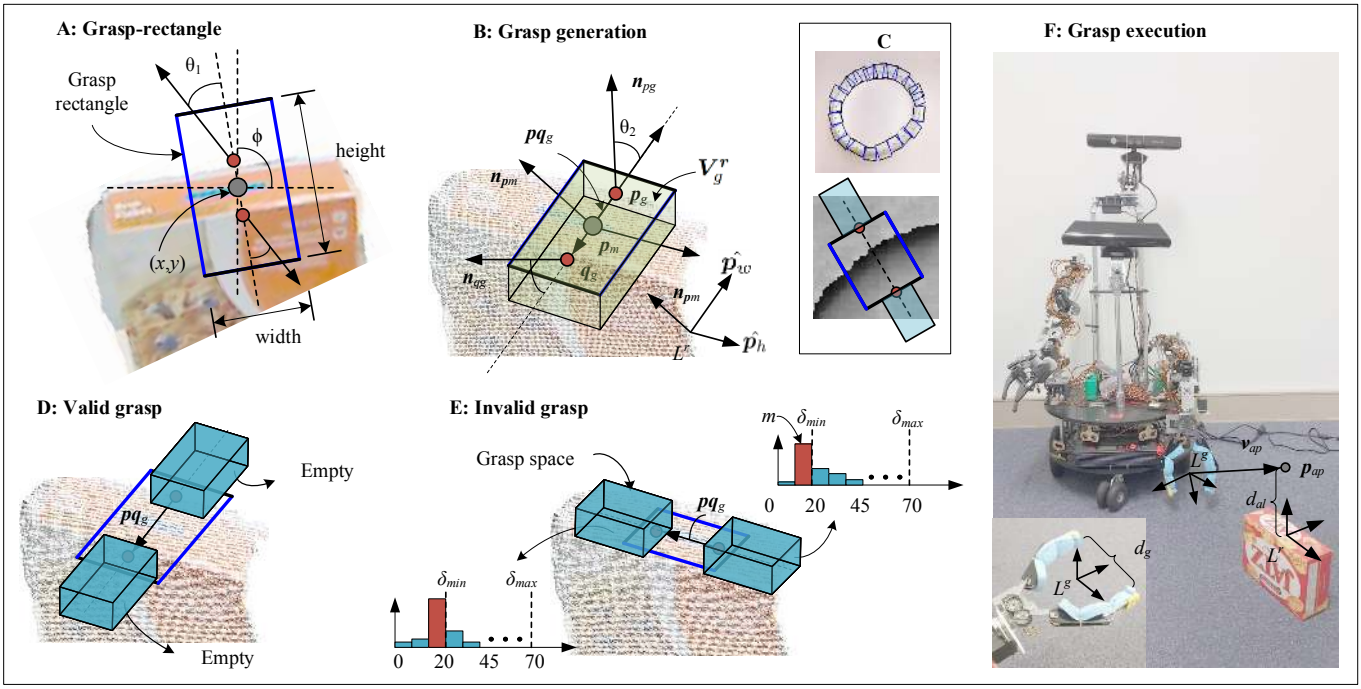

Fig. 5. Grasp synthesis using the grasp rectangle. **A**: The grasp rectangle shown in 2D image plane, where its size is specified by the blue and black lines and the orientation of the rectangle is represented by the angle ϕ. **B**: An illustration of our grasp generation in terms of two grasping points $\boldsymbol{p}_g$ and $\boldsymbol{q}_g$ (which satisfy the reflection symmetry criterion) and the search volume $\boldsymbol{V}_g^{\boldsymbol{r}}$. If there exist no point-pair which satisfies the reflection symmetry criterion, the midpoints along the height of the grasp-rectangle are selected as the grasping points (see **C**). **D**: Illustration of a valid grasp, where the grasp space around each grasping point is empty. **E**: Illustration of an invalid grasp, where the grasp space contains colliding points (i.e., the mode values of their corresponding distance histograms are less than the threshold δ_{min}. **F**: Illustration of the reference frames L^r and L^g, the approach point $\boldsymbol{p}_{ap}$, and the approach vector $\boldsymbol{v}_{ap}$.

derived from the width of the inferred grasp rectangle). The grasp execution algorithm proceeds as follows: the gripper is set to a pre-grasp configuration (i.e., $d_g = d_{g(max)}$), and is moved to the approach point, $\boldsymbol{p}_{ap}$ (i.e., $\boldsymbol{p}^{L^g} := \boldsymbol{p}_{ap}$), along the shortest directed path $\boldsymbol{v}_{ap}$. Next, the gripper orientation is set to align with the object surface (i.e., $\boldsymbol{z}^{L^g} := \boldsymbol{n}_{\boldsymbol{p}_m}$), and $\boldsymbol{x}^{L^g} := \boldsymbol{p}\hat{\boldsymbol{q}}_g$, and the gripper is translated along the direction $\boldsymbol{n}_{\boldsymbol{p}_m}$ until it reaches the grasp-position (i.e., $\boldsymbol{p}^{L^g} > \boldsymbol{p}_m$). Finally, the fingers move from the pre-grasp configuration to the grasp-configuration (i.e., $d_g \leq ||\boldsymbol{pq}_g||$) and the grasp-execution concludes when the joints settle in a static configuration.

VIII. Selective Grasp

A grasp is considered to be valid if there is enough empty space around its associated grasping points for the gripper to place its fingers before closing onto the object. Otherwise, the grasp is considered invalid (see Fig. 5 (D, E) for an illustration). To facilitate the inferencing of valid grasps (i.e., grasps with grasping points which have sufficient empty space for the gripper to fit its fingers without collision with the neighboring surfaces), we present a variant of STEM-CaRFs termed "Selective Grasp" model which produces an additional output r^{κ} (termed confidence score) for every grasp predicted in the hierarchical inferencing and selects the grasp with the highest r^{κ} value as the final grasp for the target object. The confidence score r^{κ} represents the likelihood of a valid grasp. Given a grasp rectangle $\boldsymbol{r}$, the computation of its associated r^{κ} proceeds as follows: For each grasping-point ($\boldsymbol{p}_g$ and $\boldsymbol{q}_g$), a volumetric space (termed "grasp space") is constructed in the point cloud such that its length and width are co-linear and perpendicular to the line $\boldsymbol{pq}_g$ respectively (see Fig. 5-E). The height of the grasp space is considered along the direction opposite to $\boldsymbol{n}_m$. Next, for a grasping point (e.g., $\boldsymbol{p}_g$), the algorithm extracts the set of points $\boldsymbol{\chi}_p$ within its corresponding grasp space and computes a normalized histogram of the projected distances between the points $\boldsymbol{\chi}_p$ and the grasping point. Finally, the mode value ($m_{\boldsymbol{p}}$) of the corresponding histogram is used to compute r^{κ}:

$$r^{\kappa} = \begin{cases} 0, & \text{if } (m_{\boldsymbol{p}} \leq \delta_{min} \text{ OR } m_{\boldsymbol{q}} \leq \delta_{min}) \\ 1, & \text{if } (\boldsymbol{\chi}_p = \emptyset \text{ AND } \boldsymbol{\chi}_q = \emptyset) \\ \frac{m_{\boldsymbol{p}}/m_{\boldsymbol{q}} - \delta_{min}}{\delta_{max} - \delta_{min}} & \text{otherwise} \end{cases} \tag{28}$$

where, δ_{min} and δ_{max} are two distance thresholds, empirically found to be 20mm and 70mm respectively. A small value of m (i.e., $\leq \delta_{min}$) corresponds to the case where grasp space contains colliding points, thereby indicating an insecure grasp and assigning a value of $r^{\kappa} = 0$ to the grasp-rectangle. On the other hand, if the grasp space around each grasping point is empty, r^{κ} is set to 1, thereby assigning the highest probability of a valid grasp to the grasp-rectangle. For all other cases, the value of r^{κ} is scaled between 0 and 1, corresponding to grasps with low to high confidence.

The Selective Grasp model is trained in a similar way as the STEM-CaRFs model. During testing, for an object $o \in \mathcal{O}$, the Selective Grasp model produces a 6-dimensional output for each of the hierarchically inferred grasps, where the first five values are the grasp coordinates and the sixth value is

the confidence score r^{κ} of the grasp rectangle. Specifically, the model generates a regression estimate at the object level (i.e., $\wp_o$), a regression estimate for each surfel at the surfel-level (i.e., $\{\wp_s\}_{\forall s \in \mathcal{S}_o}$), an average regression estimate at the surfel-level (i.e., $\frac{1}{|\mathcal{S}_o|}\sum_{\forall s \in \mathcal{S}_o}(\wp_s)$), and a regression estimate averaged at both the surfel- and object levels (i.e., E_o). The final grasp estimate $\boldsymbol{r}^*$ is then determined by taking the Maximum A-Posteriori (MAP) estimate of the predicted regression targets with respect to the confidence score r^{κ}.

$$\boldsymbol{r}^* = \underset{r^{\kappa}}{\operatorname{argmax}} \left(\wp_o, \{\wp_s\}_{\forall s \in \mathcal{S}_o}, \frac{1}{|\mathcal{S}_o|} \sum_{\forall s \in \mathcal{S}_o} (\wp_s), E_o \right). \quad (29)$$

In experiments, we show that the Selective Grasp model produces more accurate grasps compared to the baseline STEM-CaRFs model.

IX. Experiments and evaluation

We evaluated our framework for the tasks of object recognition, and grasp detection using popular object datasets (Washington RGB-D object dataset (WRGBD) [4] and Cornell Grasp object dataset [15]) and through robotic experimentation. For feature extraction, we used the deep network of [11]. The open-source VLFeat [29] library provides a pre-trained version of this network through its MatConvNet toolbox [30]. The configuration "E" of the VGG model [11] which we used in our setup has nearly 144 million parameters. These parameters cannot be learnt by using just the few thousand training images of the WRGBD dataset and the 885 images of the Cornell grasp dataset [31]. Therefore, to adapt the pre-trained CNN to the new tasks (RGB-D object recognition and grasp detection) and with the new input domain (STEM representation), we finetuned the CNN network (that was pre-trained on ImageNet dataset) for each feature map of the proposed STEM representation. For fine-tuning, the learning rate was initialized to 0.0001 and was decreased by a factor of 10 every 20k iterations. The batch size was set to 128 and the momentum parameter was set to 0.9.

Unless stated otherwise, we used the same fixed set of parameters for training and testing. Specifically, we trained forests with 50 trees and a maximum tree depth of 6 each. These parameters were optimized on subsets of the training datasets (see Sec. IX-D for a discussion on the selection of these parameters). We used bagging-without-replacement during the training process and therefore trained each tree on a random subset of the training samples. Specifically, at the surfel level, we extracted surfel-wise features from a fixed number of randomly selected surfels per training image. Using subsets of surfels reduced the training time and ensured a roughly even contribution from each training image. At the object-level, we extracted one object-wise feature per training image.

A. Evaluation on Washington RGB-D Object Dataset

This dataset contains 300 objects grouped into 51 categories. For each object, there are three turntable sequences captured from different camera elevation angles $(30^\circ, 45^\circ, 60^\circ)$. The dataset is challenging because it contains a large variety of textured objects (e.g., food bags or cereal boxes) as well as texture-less objects (e.g., bowls, fruits, or vegetables) captured from different viewpoints. For training and testing, we used the experimental setup described in [4], and tuned the proposed framework for object recognition. Specifically, for category recognition, we report the cross-validation accuracy for ten predefined folds over the objects (i.e. in each fold, the test instances are completely unknown to the system). For instance recognition, we employ the Leave-Sequence-Out scheme of Bo et al. [16], where training is performed on the 30° and 60° sequences, while testing is performed on the 45° sequence of every instance.

TABLE I
Comparisons with the state-of-the-art methods on the WRGBD object dataset for object category and instance recognition.

Method	Category accuracy (%)			Instance accuracy (%)		
	Depth	RGB	RGB-D	Depth	RGB	RGB-D
[4] Random Forest	66.8±2.5	74.7±3.6	79.6±4.0			
[32] HMP	70.3±2.2	74.7±2.5	82.1±3.3	39.8	75.8	78.9
[4] kSVM	64.7±2.2	74.5±3.1	83.8±3.5			
[33] HKD			84.1±2.2			
[34] IDL	70.2	78.6	85.4±3.2			
[35] RICA	79.7±3.1	84.1±2.9	86.7±2.7	49.6	88.3	89.7
[36] CKM descriptor			86.4±2.3		82.9	90.4
[17] Kernel descriptor	80.3±2.9	80.7±2.1	86.5±2.1	54.7	90.8	91.2
[37] CNN-RNN	78.9±3.8	80.8±4.2	86.8±3.3			
[16] SP+HMP	81.2±2.3	82.4±3.1	87.5±2.9	51.7	92.1	92.8
[12] CaRFs			88.1±2.4			
[38] CNN features		83.1±2.0	89.4±1.3		92.0	94.1
STEM-CaRFs (ours)	**80.8±2.1**	**88.8±2.0**	**92.2±1.3**	**56.3**	**97.0**	**97.6**

TABLE II
Recognition accuracies (%) of the proposed STEM-based features on the WRGBD Scenes dataset [14].

Category	Depth	$\boldsymbol{f}_{normals}$	$\boldsymbol{f}_{angles}$	$\boldsymbol{f}_{dist}$	**STEM-D**
Bowl	51.46	64.95	64.85	50.94	63.91
Cap	44.03	75.35	77.53	68.54	74.06
Cereal box	93.40	78.92	94.15	74.65	94.42
Coffee mug	60.77	59.66	74.58	67.67	71.54
Flashlight	93.44	90.70	80.57	91.41	91.17
Soda can	61.20	92.97	64.98	70.83	89.81
Average	68.26	78.08	76.12	70.52	**81.79**

Table I shows our object recognition results on the WRGBD object dataset and a comparison with the state-of-the-art methods. Table I shows that the proposed framework produces superior accuracies compared to the state-of-the-art methods for both the category and instance recognition tasks. For instance, we achieved an improvement of around 2.8% in the overall RGB-D categorization accuracy, compared with the best results of 89.4% reported in [38], and an improvement of around 3.5% in the overall RGB-D instance-recognition accuracy, compared with the best results of 94.1% reported in [38]. Table I also shows that we have significantly improved over our previous categorization results in [12]. These results provide strong evidence for the advantage of using STEM representation for feature learning (Sec. V-A), where the CNN-based features extracted from the feature maps of the proposed STEM representation are highly discriminative and provide more rich and complimentary information for object recognition compared to the features learnt from only RGB and depth images (e.g., [12], [38]).

In order to understand the significance of the proposed

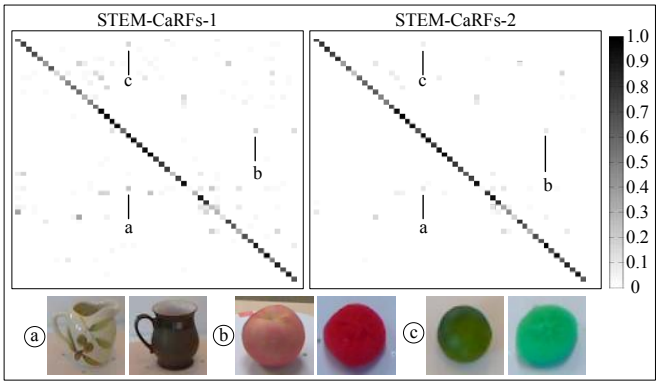

Fig. 6. Confusion matrices for WRGBD object dataset. Average classification rates are shown along the main diagonal. Selected outliers: **a)** pitcher recognized as coffee mug, **b)** peach as sponge, and **c)** lime as sponge.

cascaded architecture for object recognition, we performed experiments using two variants of our classification model: **i)** STEM-CaRFs-1, where we perform inferencing at the surfel-level, and **ii)** STEM-CaRFs-2, where we cascade the surfel-wise and object-wise forests and perform inferencing using their cumulative probabilistic output (i.e., after stage 2). Fig. 6 shows the results for these experiments in terms of the confusion matrices of the 51 test categories, where the reported results show that the confusions significantly decrease as we go from STEM-CaRFs-1 to STEM-CaRFs-2 configurations. Fig. 6 also reveals the advantage of the cascaded inference architecture for classes with high inter-class similarities in their structural or appearance characteristics (e.g., pitcher v/s coffee mug, peaches v/s similarly coloured sponges), where the confusions consistently decrease as we go from stage-1 to stage-2.

We also evaluated the effectiveness of the depth-derived feature maps of the proposed STEM representation compared to the use of raw depth images for depth-based object recognition. For this, we performed additional recognition experiments on the more challenging RGB-D scenes dataset [14] using only depth images. This dataset consists of eight video sequences of office, kitchen, and meeting room environments. Each sequence contains 1700 to 3000 RGB-D image frames containing objects placed on flat surfaces. The ground truth for this dataset is available in the form of object bounding boxes for six object classes which overlap with the classes of the WRGBD object dataset [4]. For these experiments, we trained the proposed framework for object category recognition using the WRGBD object dataset and used the ground truth bounding boxes of the WRGBD scenes dataset for testing. The results of these experiments are shown in Table II that reports the recognition accuracy averaged over all eight video sequences of the WRGBD scenes dataset. It is evident from the table that the model trained using the depth-dervied STEM-based features (STEM-D) outperforms the model trained using features extracted from raw depth images for all the test classes. Some classes (e.g., bowl, cap, coffee mug, and soda can) greatly benefited from the proposed STEM-based feature learning. This clearly indicates that the features learnt from the depth-derived feature maps of the proposed STEM representation provide more rich and complimentary information compared to the features learnt from raw depth images. This evaluation shows the importance of the proposed STEM-based feature learning for robust depth-based object recognition in real-world scenes.

TABLE III
OBJECT RECOGNITION RESULTS ON THE CORNELL GRASPING DATASET.

Method	Object recognition accuracy (%)		
	RGB	depth	RGB-D
Jian et al. [10]	-	-	84.7
Jian et al. [10]+FPFH [40]	-	-	89.6
Deep learning [8]	90.3	92.8	93.7
Surfel-wise only	72.6	79.5	80.8
Object-wise only	75.3	81.2	83.2
STEM-CaRFs	**91.2**	**93.5**	**94.1**

TABLE IV
GRASP DETECTION RESULTS ON THE CORNELL GRASPING DATASET.

Method	Grasp detection accuracies (%)	
	image-wise split	object-wise split
Jian et al. [10]	60.5	58.3
Deep learning [8]	73.9	75.6
CNN-Regression [21]	84.4	84.9
CNN-Multi [21]	88.0	87.1
Surfel-wise only	75.3	77.1
Object-wise only	84.7	84.8
STEM-CaRFs	86.8	86.2
STEM-CaRFs (Selective Grasp)	**88.2**	**87.5**

B. Evaluation on Cornell Grasping Object Dataset

The Cornell Grasping Object Dataset [15] contains 885 images of 240 distinct graspable objects and labeled ground-truth in terms of grasp-rectangles. During training, we performed data augmentation through random image rotations, flips, translations, and ZCA-whitening. For testing, we used the five-fold cross validation setup described in [8]. Specifically, the setup in [8] considers two different splits of the data: **i)** image-wise splitting which splits images randomly (i.e., the training set and the validation set do not share the same image), and **ii)** object-wise splitting which splits object instances randomly (i.e., the training set and the validation set do not share any images from the same object-class). Image-wise splitting criteria evaluates the performance of a classifier in terms of its generalization to new positions for known objects. On the other hand, the object-wise splitting criteria evaluates how well the classifier generalizes to novel/unknown objects [8]. We present our grasp-detection results using the "rectangle-metric" used in Jian et al. [10]. It considers a grasp to be correct if: **i)** the grasp angle is within 30° of the ground-truth grasp, and **ii)** the Jaccard index of the predicted grasp and the ground-truth is greater than 25%. The Jaccard index for a predicted rectangle $\boldsymbol{r}^*$ and a ground-truth rectangle $\boldsymbol{r}^g$ is given by: $J(\boldsymbol{r}^g, \boldsymbol{r}^*) = |\boldsymbol{r}^g \cap \boldsymbol{r}^*| / |\boldsymbol{r}^g \cup \boldsymbol{r}^*|$. For object recognition, we manually categorized the objects in the Cornell grasping dataset into 16 distinct categories, where objects with similar grasp ground truths were grouped into the same category. Table III shows our object recognition results on the Cornell dataset. It shows that our framework outperforms the state-of-the-art methods in object recognition accuracy. Specifically, our framework produces an improvement of 9.4% over the features in [10] and 4.5% over those features combined with FPFH features [40]. We also observe that the accuracies improve as we go from the RGB modality to the depth

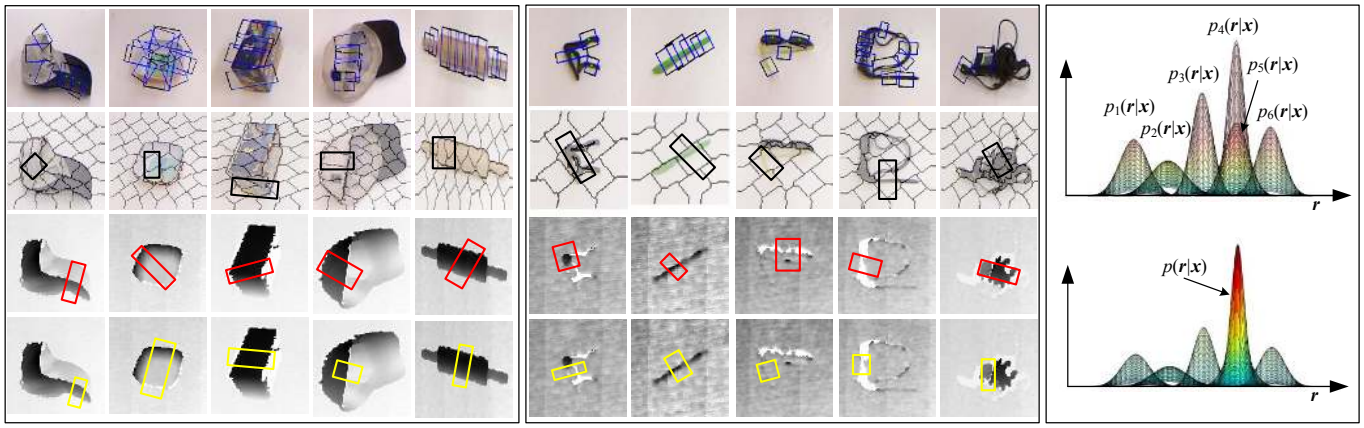

Fig. 7. **Left and Middle**: Examples of grasps produced by surfel-wise inference model (shown in black), object-wise inference model (shown in red), and the proposed Selective Grasp model (shown in yellow). The ground truth grasps are shown in blue. **Right**: Illustration of the regression posteriors of six different regression trees in an ensemble model. Some posteriors correspond to a higher confidence (sharp gaussian distributions) than the others. The ensemble posterior (obtained by averaging all tree posteriors) is more influenced by the informative trees.

modality. The highest accuracy is obtained when using all input modalities (i.e., RGB-D). This indicates that the color-derived and the depth-derived feature maps of the proposed STEM representation provide non-redundant information and their combination leads to a feature representation which is more robust than either modality alone.

Table IV shows our grasp detection results on the Cornell grasping dataset. It shows that the performance of our framework improves as we go from a single-level inferencing (i.e., surfel-wise or object-wise only) to a cascaded hierarchical inferencing (i.e., STEM-CaRFs). The surfel-wise model uses local information about the object to compute surfel-wise predictions which are then averaged to estimate the final grasp for the target object. Although, the surfel-wise model produces reasonable grasps for most of the objects, it is effected by sensor noise or missing depth information. For instance, the surfel-wise model fails to predict viable grasps for the case of incorrect segmentation or missing depth data (see Fig. 7-middle for some examples). On the other hand, the object-wise model makes much better predictions in these scenarios because it considers the entire object (and uses global information about the object) to make its prediction unlike the surfel-wise model which focuses only on local regions. Our cascaded architecture (STEM-CaRFs) combines the strongest aspects of both the local and global models. Within the cascaded forests, some trees produce more confident predictions than the others. The fusion operation yields combined regression estimates which are heavily influenced by the most confident trees (see Fig. 7-right for an example), thereby producing more accurate regression estimates compared to the estimates produced by the single-level models. Table IV also shows that the porposed STEM-CaRFs model outperforms the deep learning method in [8] and the CNN-based regression model of [21]. These improvements are credited to the use of the proposed STEM-based feature learning which learns more useful and highly discriminative multi-modal feature representations from RGB-D images compared to the feature learning approach of [21], where the authors used RGD image channels (the blue channel of the RGB image was replaced by the depth image) as an undifferentiated input to a CNN model for direct grasp regression. While the grasp rectangles predicted by the STEM-CaRFs model are valid for most of the objects, an output grasp can be infeasible for the actual robotic grasping if the space around a predicted grasp is not collision-free. This effect is mitigated by the proposed Selective Grasp model which ranks the hierarchically inferred grasps based on their confidence scores (r_κ) and selects the grasp with the highest confidence score as the final grasp for the target object unlike the STEM-CaRFs model which always output the average of the regression outputs of the cascaded forests. The proposed Selective Grasp model outperforms our baseline STEM-CaRFs model and the method of [21]. Overall, the grasps predicted by the Selective Grasp model clearly represent valid grasps for robotic grasping (see Fig. 7 for some examples).

TABLE V
RECOGNITION ACCURACIES (%) OF THE PROPOSED STEM BASED FEATURES ON THE WRGBD OBJECT DATASET [4].

Feature map	Category recognition	Instance recognition
$\boldsymbol{f}_{rgb}$	88.82	97.00
$\boldsymbol{f}_{lab}$	86.89	96.68
$\boldsymbol{f}_{normals}$	80.83	62.40
$\boldsymbol{f}_{angles}$	80.45	58.97
$\boldsymbol{f}_{dist}$	78.68	55.25
STEM-All feature maps	**92.25**	**97.63**

C. *Significance of the STEM-based features*

To highlight the significance of the proposed STEM representation for the extraction of highly discriminative features, we compare our STEM-based CNN-features with local features of [41] (extracted from RGB-D images), in terms of their capability to separate a large number of classes under different training set sizes. We use the t-SNE embedding [42] to visualize the results. As shown in Fig. 8 (A, B), the use of STEM-based features results in a good separation of the object-classes (Fig. 8-A) even when the training set size is small (e.g., 10% of the available training data used for learning). Fig. 9 (A1, A2) shows that object instances of a single class (*toothpaste*) also become well-separated in the feature space. In contrast, local features extracted from RGB-D images exhibit this property only to a very limited extent, i.e., classes and instances are less well-separated as shown in

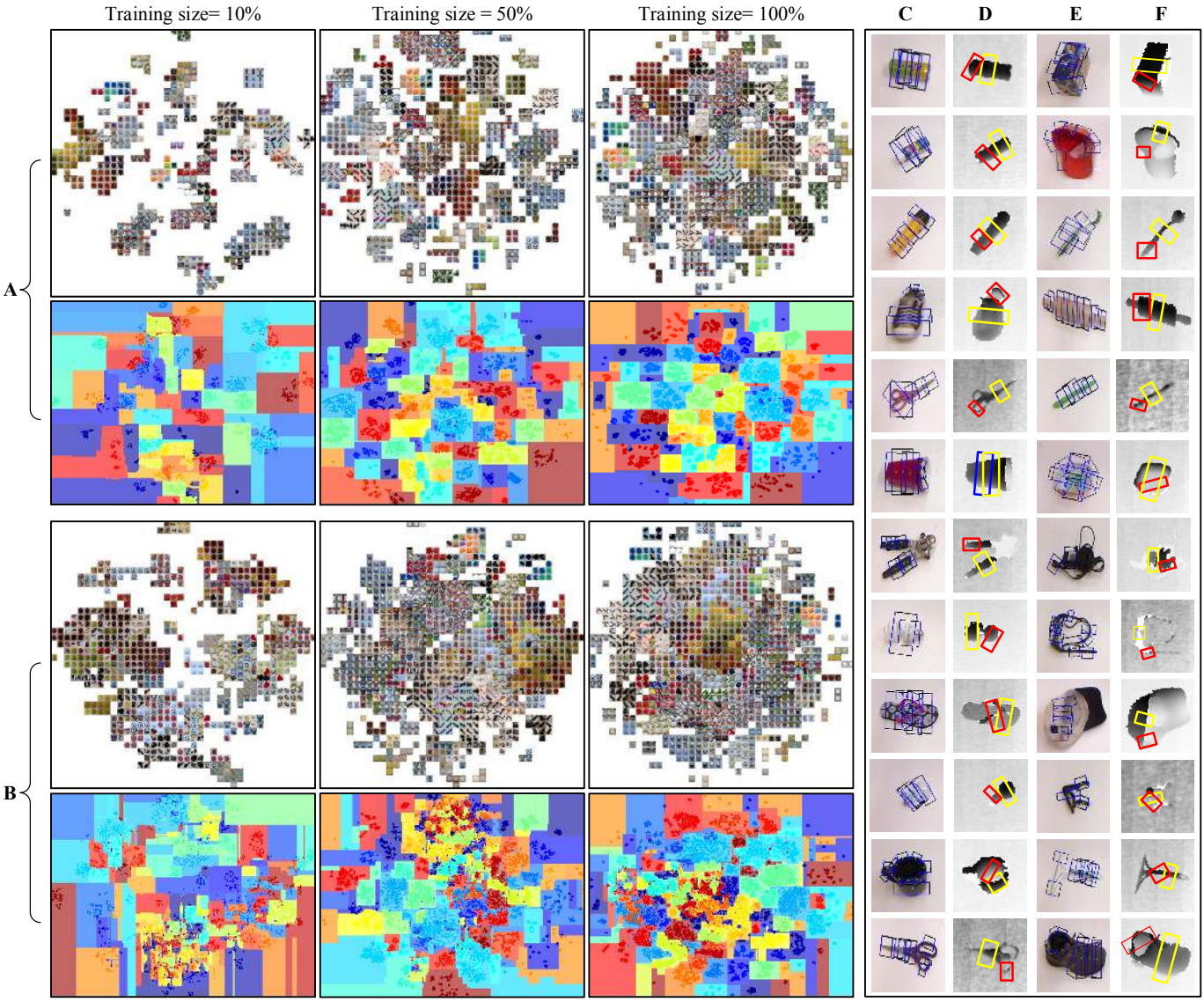

Fig. 8. **A-B:** Visualization of our STEM-based CNN features **(A)** compared to the local features [41] extracted from raw RGB-D images **(B)**, in terms of the t-SNE embeddings and the boundaries learnt by our object forest using axis-aligned weak learners w.r.t. different training set sizes. STEM-based CNN features separate the object classes better than the local features, especially when the training data is small. The training subsets are sampled randomly from the full training set. **C-F:** Qualitative comparison of grasp detection results based on depth derived STEM features (shown in yellow), and features extracted from raw depth images (shown in red) on the Cornell grasping dataset. The ground truths are shown in blue in columns **C** and **E**.

Fig. 8-B and Fig. 9 (B1, B2). Fig. 8 (C-F) shows a qualitative comparison of the grasps produced using depth-derived STEM features and the grasps produced using local features [41] extracted from raw depth images. From the figure we see that the grasps detected by the model trained using raw depth data appear invalid for most of the cases. On the other hand, the grasps detected using our depth-derived STEM features are clearly valid for almost all of the tested cases. These results signify the integration of several structural cues (e.g., pose invariant orientations of the surface normals and the projected distances of the points with respect to the centroid of the point cloud) in an RGB-like image structure (i.e., all feature values scaled between 0 and 255) yielding an image representation which is rich and more discriminative compared to the raw depth values.

We further evaluate the performance of the proposed STEM representation for large scale RGB-D object recognition by independently using the feature maps of STEM representation for feature learning. The results of these experiments are shown in Table V and Fig. 10. Table V shows that the CNN network generalizes surprisingly well to the individual feature maps of the proposed STEM representation for the case of category recognition, where we observe mean accuracy of 86.8%, 80.83%, 80.45%, and 78.68% for the cases of $\boldsymbol{f}_{lab}$, $\boldsymbol{f}_{normals}$, $\boldsymbol{f}_{angles}$, and $\boldsymbol{f}_{dist}$, respectively. Fig. 10 shows the categorization accuracy for each class for the color-derived features ($\boldsymbol{f}_{rgb}$+$\boldsymbol{f}_{lab}$) and the depth-derived features ($\boldsymbol{f}_{normals}$+$\boldsymbol{f}_{angles}$+$\boldsymbol{f}_{dist}$). From the figure, we observe that some object classes were classified with a higher accuracy using depth-derived features compared to the use of color-derived features. These objects have texture-less surfaces with no distinguishing visual features and hence the depth-derived features were able to utilize shape information to achieve a higher accuracy (e.g., see the case of *bowl* class

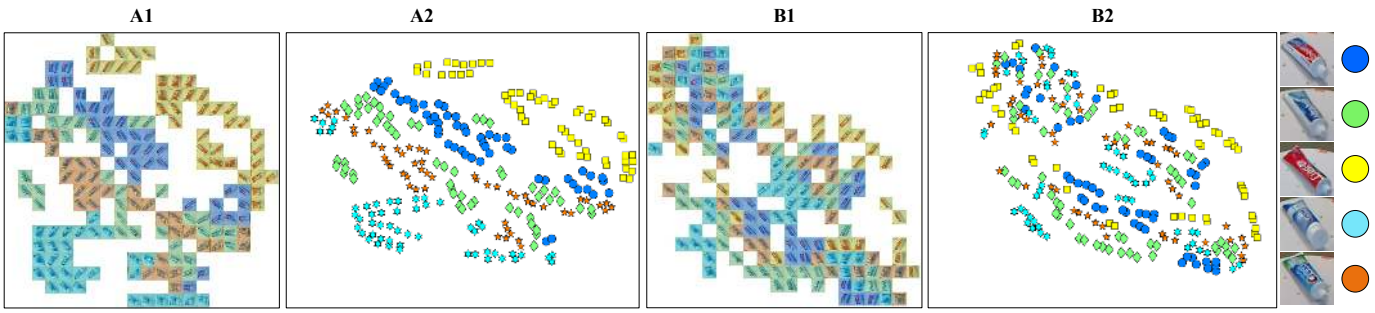

Fig. 9. Visualization of our STEM-based CNN-features **(A1, A2)** compared to the local features [41] extracted from raw RGB-D images **(B1, B2)**, in terms of the t-SNE embeddings of the *toothpaste* class, colored by instances (shown by large circles). STEM-based CNN features separate the object instances better than the local features extracted from raw RGB-D images.

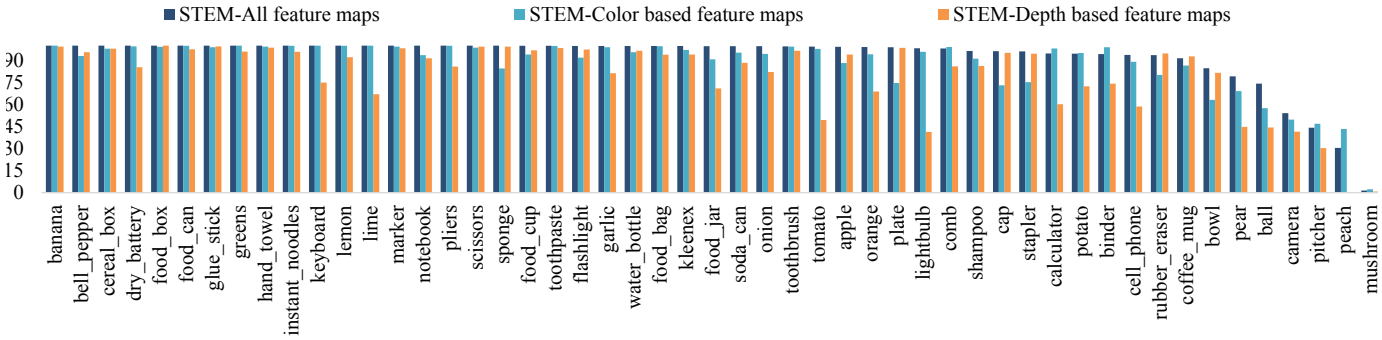

Fig. 10. Classification accuracy (measured per category) of our STEM-based CNN features on the WRGBD object dataset [4].

in Fig. 10). The best performance is achieved by using all the features maps of the proposed STEM representation for feature learning, yielding a mean accuracy of 92.2%. For instance recognition, we observe that the accuracies for the depth-derived features are considerably low compared to the accuracies for the color-derived features (as shown in Table V). This is attributed to the fact that color information provides better discrimination across intra-class instances while they share very similar structural characteristics (e.g., soda cans are cylindrical). Nonetheless, this problem can effectively be mitigated by the fusion of the color-derived and depth-derived feature maps in the proposed STEM representation yielding an average accuracy of 97.63%. These results validate our hypothesis that STEM feature maps contain complimentary information and it is appropriate to use STEM-encoding as an initialization for the fine tuning of a large CNN pre-trained on RGB images.

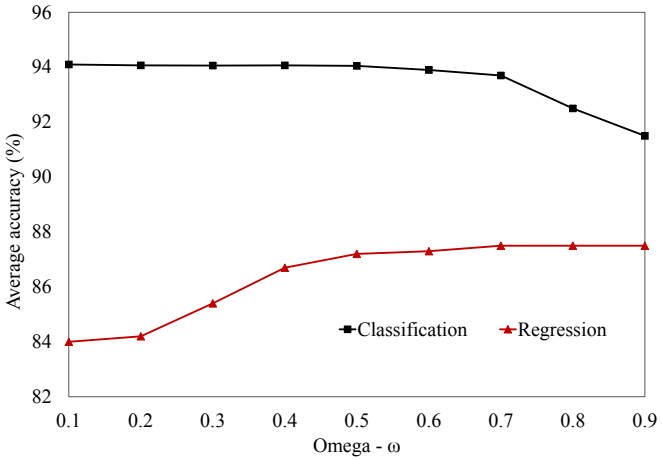

Fig. 11. Average accuracies for object recognition and grasp detection on the Cornell grasping dataset for different values of ω.

D. Parameter Selection

Here, we analyze the most relevant parameters of our proposed framework. **First**, we investigate the performance of the proposed framework by varying the parameter ω in Eq. 13, which controls the relative influence of classification and regression on the overall objective function. Fig. 11 shows the average accuracies for object recognition and grasp detection for different values of ω on the Cornell grasping dataset. From the figure we observe that, for large values of ω (e.g., $\omega > 0.65$), the classification accuracy starts decreasing. We manually inspected the cascaded ensemble and found that the tree-learning termination criterion was reached in parts of the trees during training before the object classes were properly clustered in the leaves. Since the influence of the regression component $\mathcal{I}_r$ in Eq. 13 on the overall objective function increases with an increase in the value of ω, for large values of ω, the objective function (see Eq. 13) favours regression over classification. This leads to a sub-optimal clustering of the class-labels at the leaves of the forests and results in a decrease in the overall classification accuracy (for values of $\omega > 0.65$). On the other hand, the grasp regression accuracy increases with the increase in the ω value (as shown in Fig. 11). We empirically found that, a value of $\omega = 0.6$ produced the best balance between the classification and the regression contributions to the overall training objective function. This analysis supports our observation that while the proposed framework can be tuned for optimal classification performance

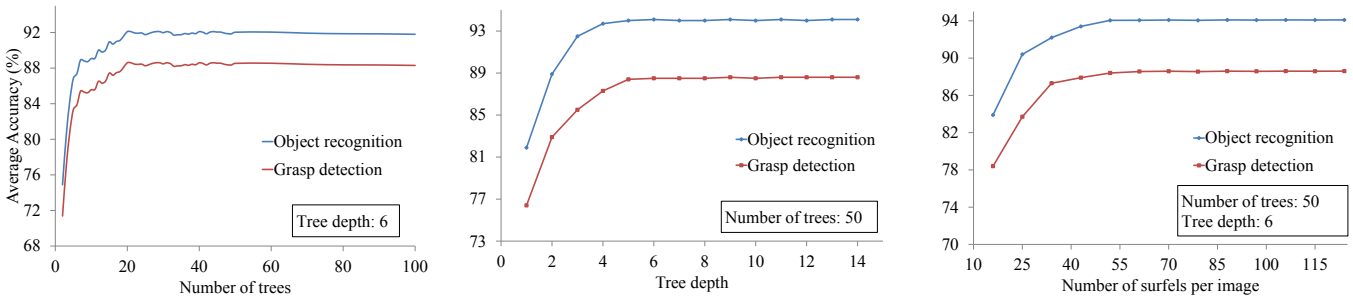

Fig. 12. Left to right: Evaluation of the parameters (number of trees, tree depth, and segmentation resolution) of the framework on the Cornell dataset.

TABLE VI
RESULTS FOR ROBOTIC EXPERIMENTS, SORTED BY OBJECT CATEGORY.

Category	Trials	Recognition (%)	Grasp success rate (%)
Tennis ball	10	91.6	84.2
Soda can	15	94.6	98.8
Stapler	8	91.2	93.7
Food box	19	92.3	98.4
Food can	21	95.5	98.5
Shampoo	16	95.6	97.5
Joystick	15	91.1	82.6
Coffee mug	10	93.4	85.2
Dish cleaner	20	92.5	88.3
Average	-	**93.0**	**91.9**

or optimal regression performance by simply adjusting the parameter ω, nevertheless, the proposed framework is robust for joint classification-regression. The joint minimization of the uncertainties of the object-class labels and the grasp ground truths at the leaves of the cascaded forests using the proposed quality measure $\mathcal{I}$ (see Eq. 13) produces accurate separation of both the object classes and the grasp-pose manifolds. This signifies the robustness of the proposed framework for the recognition and grasp detection of objects within a unified framework. **Next,** we evaluate the number of trees N_t while setting the maximum tree depth to 6 on the Cornell grasping dataset. The results of this experiment are shown in Fig. 12-left. As expected, we can see a clear trend of an increasing recognition accuracy with respect to the number of trees up to $N_t = 50$. For more trees, i.e., $N_t > 50$, the results are saturated. By setting $N_t = 50$, increasing the tree depth has little influence beyond 6 levels (see Fig. 12-middle). **Finally,** we evaluate the performance of the proposed framework in terms of resolution of the segmentation, where we set the number of trees to be $N_t = 50$ with a depth of 6 levels each, and report the results in Fig. 12-right. The figure shows that increasing the resolution of the segmentation from 55 to 125 surfels per image does not produce a significant difference in the overall accuracy. Overall, our framework is robust with respect to the variations of its hyper-parameters.

E. Robotic experiment

To evaluate the performance of our framework for robotic grasping of household objects, we ran an extensive series of robotic experiments using live video streams acquired by a Kinect that is mounted on our in-house robotic platform named "AIPAR". AIPAR is equipped with two arms with seven degrees of freedom each. We only used the left arm for these experiments. The end-effector for this arm is a two-finger gripper. We attached rubber strips to the gripper fingers to increase friction. For each experiment, we placed a number of different objects within a working square of 25cm×25cm square (with respect to the robot base) on the floor (see Fig. 13-left). This square was chosen to be well-contained within the robots workspace, allowing objects to be grasped from most approach vectors. Object positions and orientations were varied between trials, although objects were always placed in configurations in which at least one viable grasp was accessible to the robot. During testing, the robot is commanded by voice to grasp a specific object (e.g., "AIPAR pick food can"). Upon successful recognition and grasp detection of the object, the robot executes the grasp on the object and lifts it 30 cm upwards. A grasp is determined to be successful if it is sufficient to lift the object and hold it for three seconds.

Table VI shows the results of these experiments using a set of 35 household objects grouped in 9 different categories (most of these objects were not included in the training dataset, and thus were completely new to the framework), and a total of 134 trials. From Table VI, we see that the proposed framework correctly predicts the object category for more than 92% of the trials. Even with the added grasp-detection task, the joint classification-regression model maintains high accuracies for the case of unseen objects. Table VI also shows that AIPAR was able to successfully execute a grasp for more than 90% of the trials. Fig. 13-right shows AIPAR executing several of these grasps. A video of one of these experiments is available in the supplementary material. Among these experiments, the joystick-controller proved to be the most difficult object for the robot to grasp. From a top-down angle, there is only a small space of viable grasps with a span of less than 7cm. The grasps across the two "handles" tend to slip off. Therefore, all valid grasps are very close to the 7cm span and even a slight imprecision in positioning can lead to failure. Using the proposed selective-grasp model (Sec. VIII), AIPAR was able to execute successful grasps for more than 90% of the trials. Our framework was also able to successfully detect and execute grasps with non-vertical approach vectors. The grasps shown for the "food box", and "dish cleaner" (shown in Fig. 13-right) were all executed by aligning the gripper to a non-vertical approach vector. This shows that our framework is not restricted to the detection of top-down grasps, but can be applied to grasps from many orientations. Fig. 14 shows a qualitative evaluation of our framework on scenes with objects in stacked layouts. From the figure it is clear that our framework successfully handles the case of object classification and grasp detection under different object layouts

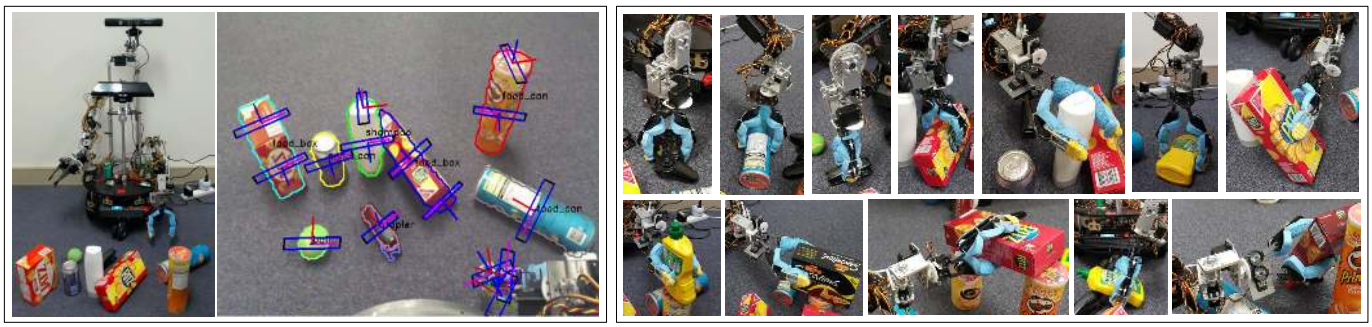

Fig. 13. **Left:** Visualization of our robotic experiment setup (where objects are placed in front of the robot in different orientations and locations, and the task of the robot is to pick a target object selected by the human operator), and the output of the proposed framework in terms of segmented objects, their predicted class labels and estimated grasp rectangles (each with a fixed height of 3 cm). **Right:** Examples of robotic grasping. Figure best viewed in color.

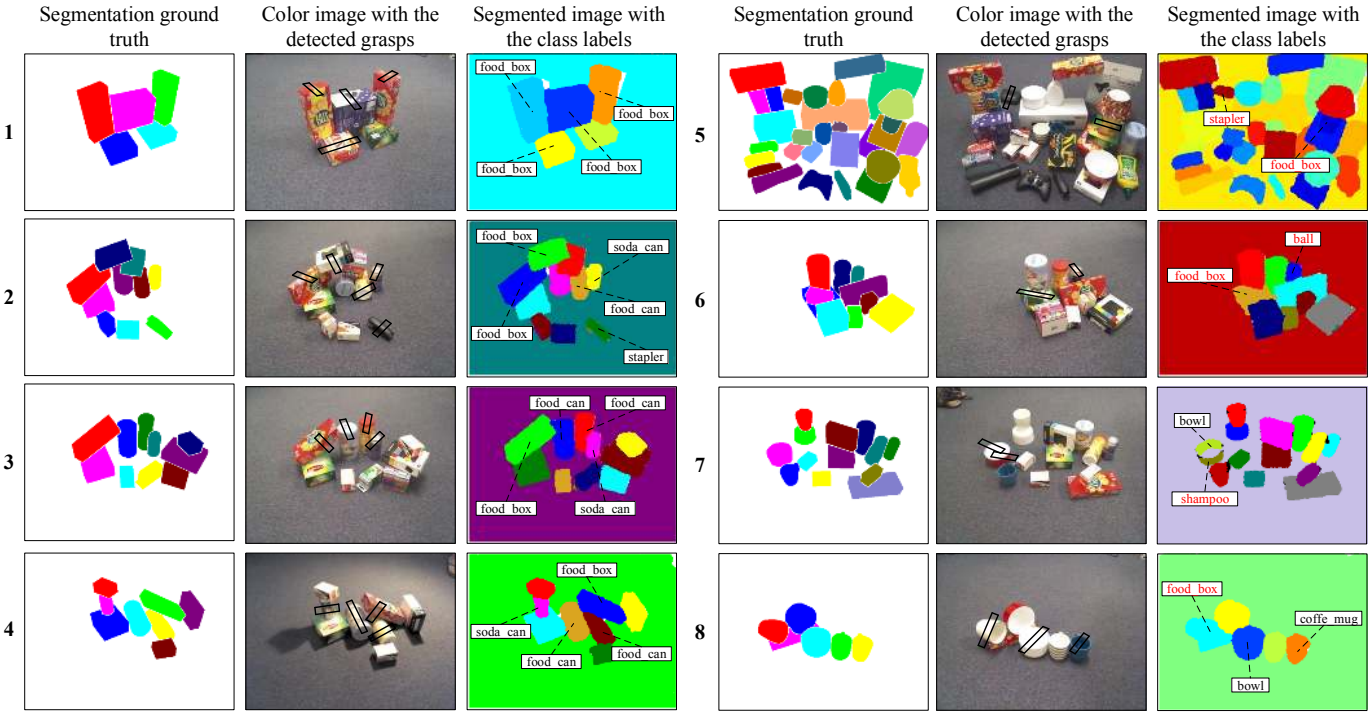

Fig. 14. Qualitative evaluation of the proposed framework for object classification and grasp detection on real-world scenes with objects in stacked layouts. Note that for clarity we only show the class labels of selected object hypotheses with their corresponding grasp rectangles. Figures 5 to 8 show cases of incorrect classification and incorrect segmentation (shown in red color). Figure best viewed in color.

and partial occlusions. Fig. 14-(5-8) show the cases where our framework produces acceptable/valid grasps even when the classification fails (e.g., soda can is classified as a ball as shown in 6) or for the cases of incorrect segmentation (e.g., the bowl splits into two parts as shown in 7, or the coffee mug merges with the food box as shown in 8).

X. Conclusions

We introduce a novel framework of hierarchical cascaded forests for the recognition and grasp detection of RGB-D objects in real-world scenes. To achieve this, we propose a novel image representation (termed STEM) to extract multi-modal features using large CNN models, and a cascaded architecture of Random Forests (STEM-CaRFs) to learn object-class and grasp-pose probability distributions at different levels of an image hierarchy (e.g., surfel and object-levels). The experiments show that, the proposed STEM-based features separate object classes in a highly discriminative manner compared to local features extracted from raw RGB-D images, and the proposed STEM-CaRFs framework yield superior object recognition and grasp detection accuracies compared to the state-of-the-art methods. Furthermore, the real-world experiments using our in-house robotic platform show that the proposed framework is robust to the real-world noise and it generalizes to new object instances with high predictive accuracies. The proposed framework is scalable in terms of training set size and in terms of parallelization. These characteristics make the proposed framework ideally suited for real-world robotic applications, where robustness to real-world noise and generalization to new object instances are the key requirements.

References

[1] Rusu, R.B., Bradski, G., Thibaux, R., Hsu, J.: Fast 3d recognition and pose using the viewpoint feature histogram. In: IROS. (2010) 2155–2162
[2] Asif, U., Bennamoun, M., Sohel, F.: Real-time pose estimation of rigid objects using rgb-d imagery. In: 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), IEEE (2013) 1692–1699

[3] Knopp, J., Prasad, M., Willems, G., Timofte, R., Van Gool, L.: Hough transform and 3d surf for robust three dimensional classification. In: ECCV. (2010) 589–602
[4] Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: ICRA. (2011) 1817–1824
[5] Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. Volume 2. (2006) 2169–2178
[6] Asif, U., Bennamoun, M., Sohel, F.: Discriminative feature learning for efficient rgb-d object recognition. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE (2015) 272–279
[7] Wu, Z., Ke, Q., Sun, J., Shum, H.Y.: A multi-sample, multi-tree approach to bag-of-words image representation for image retrieval. In: ICCV. (2009) 1992–1999
[8] Lenz, I., Lee, H., Saxena, A.: Deep learning for detecting robotic grasps. arXiv preprint arXiv:1301.3592 (2013)
[9] Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. IJRR **27**(2) (2008) 157–173
[10] Jiang, Y., Moseson, S., Saxena, A.: Efficient grasping from rgbd images: Learning using a new rectangle representation. In: ICRA. (2011) 3304–3311
[11] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
[12] Asif, U., Bennamoun, M., Sohel, F.: Efficient rgb-d object categorization using cascaded ensembles of randomized decision trees. In: ICRA. (2015)
[13] Asif, U., Bennamoun, M., Sohel, F.: Simultaneous dense scene reconstruction and object labeling. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, IEEE (2016) 2255–2262
[14] Lai, K., Bo, L., Ren, X., Fox, D.: Detection-based object labeling in 3d scenes. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE (2012) 1330–1337
[15] : Cornell grasping dataset. http://pr.cs.cornell.edu/grasping/rect_data/data.php,accessed:2013-09-01/
[16] Bo, L., Ren, X., Fox, D.: Unsupervised feature learning for rgb-d based object recognition. In: Experimental Robotics. (2013) 387–402
[17] Bo, L., Ren, X., Fox, D.: Depth kernel descriptors for object recognition. In: IROS. (2011) 821–826
[18] Bohg, J., Kragic, D.: Learning grasping points with shape context. Robotics and Autonomous Systems **58**(4) (2010) 362–377
[19] Popovic, M., Kootstra, G., Jorgensen, J.A., Kragic, D., Kruger, N.: Grasping unknown objects using an early cognitive vision system for general scene understanding. In: IROS. (2011) 987–994
[20] Ekvall, S., Kragic, D.: Learning and evaluation of the approach vector for automatic grasp generation and planning. In: ICRA. (2007) 4715–4720
[21] Redmon, J., Angelova, A.: Real-time grasp detection using convolutional neural networks. arXiv preprint arXiv:1412.3128 (2014)
[22] Asif, U., Bennamoun, M., Sohel, F.: Unsupervised segmentation of unknown objects in complex environments. Autonomous Robots **40**(5) (2016) 805–829
[23] Asif, U., Bennamoun, M., Sohel, F.: Model-free segmentation and grasp selection of unknown stacked objects. In: ECCV. (2014) 659–674
[24] Asif, U., Bennamoun, M., Sohel, F.: A model-free approach for the segmentation of unknown objects. In: IROS. (2014) 4914–4921
[25] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012) 1097–1105
[26] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014) 580–587
[27] Bo, L., Ren, X., Fox, D.: Learning hierarchical sparse features for rgb-(d) object recognition. IJRR **33**(4) (2014) 581–599
[28] Breiman, L.: Random forests. Machine learning **45**(1) (2001) 5–32
[29] Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/ (2008)
[30] Vedaldi, A., Lenc, K.: Matconvnet – convolutional neural networks for matlab. CoRR **abs/1412.4564** (2014)
[31] Silberman, N., Sontag, D., Fergus, R.: Instance segmentation of indoor scenes using a coverage loss. In: ECCV. Springer (2014) 616–631
[32] Bo, L., Ren, X., Fox, D.: Hierarchical matching pursuit for image classification: Architecture and fast algorithms. In: NIPS. (2011) 2115–2123
[33] Bo, L., Lai, K., Ren, X., Fox, D.: Object recognition with hierarchical kernel descriptors. In: CVPR. (2011) 1729–1736
[34] Lai, K., Bo, L., Ren, X., Fox, D.: Sparse distance learning for object recognition combining rgb and depth information. In: ICRA. (2011) 4007–4013
[35] Le, Q.V., Karpenko, A., Ngiam, J., Ng, A.Y.: Ica with reconstruction cost for efficient overcomplete feature learning. In: NIPS. (2011) 1017–1025
[36] Blum, M., Springenberg, J.T., Wulfing, J., Riedmiller, M.: A learned feature descriptor for object recognition in rgb-d data. In: ICRA. (2012) 1298–1303
[37] Socher, R., Huval, B., Bath, B., Manning, C.D., Ng, A.Y.: Convolutional-recursive deep learning for 3d object classification. In: NIPS. (2012) 665–673
[38] Schwarz, M., Schulz, H., Behnke, S.: Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2015) 1329–1335
[39] Eitel, A., Springenberg, J.T., Spinello, L., Riedmiller, M., Burgard, W.: Multimodal deep learning for robust rgb-d object recognition. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE (2015) 681–687
[40] Rusu, R.B., Blodow, N., Beetz, M.: Fast point feature histograms (fpfh) for 3d registration. In: ICRA, IEEE (2009) 3212–3217
[41] Bosch, A., Zisserman, A., Munoz, X.: Image classification using random forests and ferns. In: ICCV. (2007) 1–8
[42] Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(2579-2605) (2008) 85

Umar Asif received a Masters degree in Mechatronics Engineering from The University of New South Wales, Australia, and a PhD degree in Computer Vision from The University of Western Australia, Australia. He is currently a Research Scientist at IBM Research Australia. His research interests include, computer vision, machine learning, and robotics.

Mohammed Bennamoun received the MSc degree in control theory from Queens University, Canada, and the PhD degree in computer vision from QUT in Brisbane, Australia. He is currently a Winthrop Professor at the University of Western Australia, Australia. He is the coauthor of the book Object Recognition: Fundamentals and Case Studies (Springer-Verlag, 2001). He has published more than 300 journal and conference publications. His areas of interest include control theory, robotics, artificial neural networks, and computer vision. He served as a guest editor for a couple of special issues in International journals such as the International Journal of Pattern Recognition and Artificial Intelligence. He was selected to give tutorials at the European Conference on Computer Vision (ECCV 02), the International Conference on Acoustics Speech and Signal Processing (2003) and Interspeech (2014). He also contributed in the organization of many local and international conferences.

Ferdous A. Sohel received PhD degree from Monash University, Australia in 2009. He is currently a Senior Lecturer at Murdoch University, Australia. Prior to joining Murdoch University, he was a Research Assistant Professor at the School of Computer Science and Software Engineering, The University of Western Australia from January 2008 to mid-2015. His research interests include computer vision, multimodal biometrics, and robotics. He has published more than 70 scientific articles. He is a recipient of prestigious Discovery Early Career Research Award (DECRA) funded by the Australian Research Council. He is also a recipient of the Early Career Investigators award (UWA) and the best PhD thesis medal form Monash University. He is a Member of Australian Computer Society and a Senior Member of the IEEE.