2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Knowledge-Based Grasp Planning Using Dynamic Self-Organizing Network

Shiyi Yang and Soo Jeon

Abstract— **Category-based methods for task-specified grasp planning have recently been proposed in the literature. Such methods, however, are normally time consuming in both training and grasp determination process and lack capabilities to improve grasping skills due to the fixed training data set. This paper presents an improved approach for knowledge-based grasp planning by developing a multi-layer network using self-organizing map. A number of grasp candidates are learned in the experiments and the information that is associated with these grasp candidates is clustered based on different criteria on each network layer. A codebook which is composed of a small number of generalized models and the corresponding task-oriented grasps is generated from the network. In addition, the proposed network is capable of automatically adjusting its size so that the codebook can be continuously updated from each interaction with the novel objects. In order to increase the accuracy and convergence rate of the clustering process, a new initialization method is also proposed. Simulation results present the advantages of the proposed initialization method and the auto-growing algorithm in terms of accuracy and efficiency over some conventional methods. Experimental results demonstrate that novel objects can be successfully grasped in accordance with desired tasks using the proposed approach.**

Index Terms— **grasp planning, clustering algorithm, self-organizing map, initialization method, growing network**

I. Introduction

Grasp planning with robotic hands for performing daily tasks in human environments is a major challenge even for the best artificial robotic manipulators available today. Given the basic information (e.g. size and shape) on the everyday objects, the most conventional way to generate a good stable grasp in the sense of force-closure is to determine the contact locations of each fingertip [1], [2]. Finding a good stable grasp, however, is only a necessary but not a sufficient condition for efficient grasp planning, especially when some manipulations are expected after grasping [3]. In other words, a good grasp should be made in accordance with the desired task. And yet the desired tasks are normally not considered in the conventional approaches due to the difficulty in modeling tasks in the process of grasp planning.

Most of everyday tools that humans use consist of a particular component, as known as affordance part, which is designed specially to fulfill their functionalities and to make their grasps easier. For instance, the handle parts of mugs and teapots and the handles of knives and hammers. Thus, by learning object affordances from a deep neural network model [4], [5], a task-oriented grasp can be achieved through determining a proper grasp on the affordance parts [6], [7]. Some research studies expand on this idea in a more general concept that one task-oriented grasp can be generalized to objects that are similar in functionality and shape so as to facilitate the grasp planning procedure [8], [9]. By constructing categories to contain similar training objects and learning every possible task-oriented grasps with these training objects, a probabilistic approach that selects a learned grasp which is most likely applied to the target object to accomplish the task is proposed in [10]. The selected grasp can also be used to grasp any novel objects which are not in the training categories but similar to the training objects.

With this type of approach, a more suitable grasp for a novel object can be determined, as more training objects are contained in the category and more grasp candidates are learned from training. Consequently, the computational demand for analyzing the grasp data and selecting an appropriate grasp will be significantly increased when the training data set gets large. To address this technical issue, a number of clustering techniques have been utilized to condense the total grasp candidates into a codebook/dictionary which contains a small set of representative grasps [11], [12]. One of the most efficient methods is the *k*-means algorithm. However, it suffers from its own sensitivity to the noise or outliers. Compared to *k*-means algorithm, *k*-medoids algorithm [13] is more robust and has been used to form a codebook in [14]. Besides these two algorithms, the self-organizing map (SOM), also known as the Kohonen feature network [15], has also been applied to many applications, such as data exploration [16], image processing [17] and robotic grasp planning [18], [19]. It is a competitive feed-forward neural network and an efficient algorithm for feature extraction while maintaining the topological structure of input data. In particular, SOM forms a topological map which preserves the same distribution as the input data while projecting data form a high dimensional space onto a low dimensional output space. This feature of SOM makes it more suitable for clustering grasp candidates than *k*-means and *k*-medoids in our case, as information learned from grasp candidates using similar objects in the input space corresponds to close spatial locations on a topological map.

Like most of other unsupervised learning algorithms, SOM still suffers from two technical issues during implementation: initialization of the weights/locations for the neurons/centroids and determination of the number of neurons (i.e., clusters). In general, the key idea of initializing centroids is to search for such locations that are close to the majority of data and also as far as possible away from each other [20], [21]. From the viewpoint of statistics, selecting outliers as initial centroids is not acceptable, since outliers

S. Yang and S. Jeon are with the Department of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, ON, N2L 3G1, Canada e-mail: shiyi.yang, soojeon@uwaterloo.ca.

are normally referred to the data corrupted by noise and differ significantly from most of the training data. The situation we consider in this paper is quite different from those of the existing studies on the treatment of outliers. In our case, an outlier typically contains the dimension information of a particular model which is drastically different from other models in the same group. And because a set of corresponding grasp candidates are also contained in this outlier, it can not be neglected. Hence, the initialization process needs to focus on capturing the density of the training data while taking the outliers into account. Another issue from that SOM and other clustering algorithms suffer is that it is impracticable to predefine an optimal number of clusters due to the lack of prior information on the details of network. In addition, in robotic manipulation, the manipulators should keep improving the grasping skill as they continuously interact with more and more objects. In other words, the training set should constantly adapt and update with the related information from each grasp execution. Such self-updating capability, however, is not provided by the conventional category-based approaches due to employing fixed size of training data set. In data analytics, a number of approaches have been proposed to make networks capable of automatically adjusting the size to achieve the optimal clustering, such as data density-based approach [22] and hierarchical approach [23].

In this paper, to efficiently analyze the grasp candidates, a multi-layer network is developed based on SOM. A task-oriented codebook which consists of a small number of generalized models and the corresponding representative grasps is generated from the network, and each representative grasp can be employed for grasping novel objects which are similar to the the generalized models in the codebook. In order to improve the accuracy and convergence rate of the clustering algorithm, a novel initialization method is also proposed by using Gaussian kernel to measure the similarity between the training data. Compared to other distance metrics (e.g., Euclidean distance and Manhattan distance), the Gaussian kernel can produce a large response when two data points are close to each other and exponentially fall off when two data points are apart. In addition, as a nonlinear function of Euclidean distance, Gaussian kernel can also provide some useful properties such as smoothness (i.e., the Gaussian function is infinitely differentiable). As a consequence, every training data including any outliers is considered as candidates for the initial neurons. Moreover, we incorporate the concept of growing network such that the proposed network can automatically determine the number of clusters based on the training data. In particular, the codebook can be adapted with new training data so that the generalized models can be refined and the grasping skills can be improved through every grasp execution with novel objects.

The reminder of this paper is organized as follow. The general framework of the proposed grasp planning approach is introduced in Section II. After reviewing the concept of the batch learning SOM in Section III, the proposed multi-layer network as well as the novel initialization method and auto-growing algorithm are presented in Section IV. Simulation and experimental results are provided to evaluate the performance of the proposed initialization method and auto-growing algorithm in comparison to some conventional methods and validate the effectiveness of the proposed grasp planning approach in Section VI. Finally, our concluding remarks are presented in Section VII.

II. General Framework

In this section, the basic procedure of the proposed grasp planning is introduced. To focus on the grasp planning and analyzing the training data, we assume that both the dimensions and pose (i.e., position and orientation) of novel objects are available and the category to which the novel objects belong is known. In this paper, the dimensions of an object are defined as the dimensions of the bounding box of the object. In addition, the novel object is assumed to be unseen by the robot but similar to the objects with that robot has interacted.

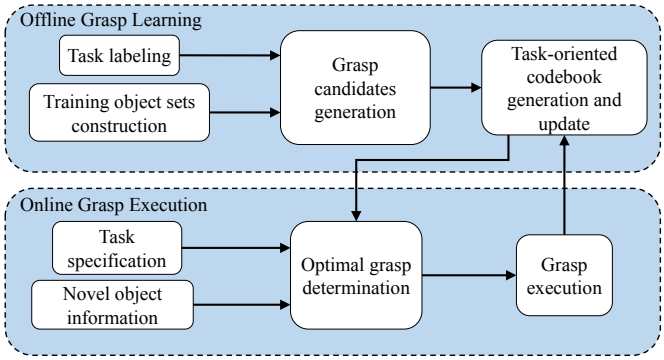


Fig. 1: General framework of the proposed grasp planning approach.

As shown in Fig. 1, the proposed approach is executed in two phases: offline learning phase and online execution phase. In the offline learning phase, a number of object categories are first built up and each of them contains a few objects (e.g., mugs or knives) that share similarities in shape and functionality. Then, every training object is grasped by a robotic manipulator to accomplish some certain tasks (e.g., transportation and tool use) in physical experiments. In doing so, the information related to each grasp candidate is recorded and saved into a training data set for each object category. In the end of offline phase, a number of codebooks are individually created and each codebook is composed of a number of generalized models and representative grasps for the corresponding object category. During the online operation, given the category to which the target object belongs, comparing the dimensions of the target object to that of the generalized models stored in the corresponding codebook, a suitable representative grasp is selected from the codebook and applied to the target object. Meanwhile, the codebook incorporates the related information associated with this grasp and updates automatically.

III. Batch Learning Self-Organizing Map

There are two different learning techniques for SOM: the online/sequential learning and the batch learning [15]. Using the online learning, the weighting vector (i.e., location) of the winning neuron (i.e., the neuron which is the most similar to the inputs) and its neighboring neurons are updated immediately after the determination of the winning neuron. This process is repeatedly performed for each input data. This technique, however, suffers from large computational load as it constantly updates the weights and a high dependence on the order of the input data. As to the batch learning [24], the weights remain the same while processing the input data and are updated after examining every input data in the entire training set. Hence, the batch learning technique is slightly more efficient in terms of computational cost and independent of the order of input data. In this paper, the batch learning SOM (BLSOM) forms the basic structure of the proposed multi-layer network.

Let $\Psi = \{\psi_i | \psi_i \in \mathbb{R}^d, i = 1, \ldots, I\}$ represent the training data set for one particular object category and each training data ψ_i represents a grasp candidate. A finite set $\Omega = \{\omega_j | \omega_j \in \mathbb{R}^d, j = 1, \ldots, J\}$ denotes locations for all the neurons in the competitive layer. In general, there are four steps to perform the batch learning algorithm: initialization, competition, cooperation and adaptation.

1) **Initialization**: In this step, the locations of each neuron, Ω, are initialized with certain values (the details of the initialization method will be introduced in Section IV-B).
2) **Competition**: Present the entire training data set, Ψ, to the network. For each training data, the closest neuron (i.e., the best matching unit) is determined based on the Euclidean distance by

$$\alpha_i = \underset{j}{\operatorname{argmin}} \|\psi_i - \omega_j\|_2, \tag{1}$$

where α_i denotes the index of the winning neuron corresponding to the training data, ψ_i. This winning neuron represents the spatial location of an "excited" neuron, thereby determining the basis of cooperation for a topological neighborhood around it.
3) **Cooperation**: The most essential feature of SOM is that it utilizes the idea of topological neighborhood to define the relation between each neuron. There are some important properties of a topological neighborhood. The maximum of such neighborhood is at the the central neuron. In addition, it is symmetric about the associated central neuron and monotonically decreases as it gets far away from the central neuron. Specifically, the topological function of two neurons is defined as

$$h_{\alpha_i,j} = \exp\left(-\frac{\|\omega_j - \omega_{\alpha_i}\|_2^2}{2\sigma_h(n)^2}\right), \tag{2}$$

where σ_h (referred to the learning rate) controls the variation rate of the correlation between two neurons. In general, σ_h decreases exponentially with time as

$$\sigma_h(n) = \sigma_h(0) \exp\left(-\frac{n-1}{n_{max}}\right), n = 1, ..., n_{max} \tag{3}$$

where n indicates the current iteration number, $\sigma_h(0)$ is the initial value of σ_h and n_{max} denotes the maximum number of iterations.
4) **Adaptation**: In this step, each excited neuron is updated through suitable adjustment made to its location so as to increase the resemblance between the excited neurons and the training data in terms of data distribution. The update of the location is expressed as

$$\omega_j^{new} = \frac{\sum_{i=1}^{I} h_{\alpha_i,j} \psi_i}{\sum_{i=1}^{I} h_{\alpha_i,j}}. \tag{4}$$

IV. Grasp Candidates Analysis

In this section, we present a multi-layer network to analyze the grasp candidates and a novel initialization method to determine the locations of initial neurons. The proposed network can also automatically adjust the size (i.e., the number of neurons) based on the training data set.

A. Grasp Training Set

Each training data vector contained in Ψ is defined as

$$\psi_i = \left(\psi_i^{\mathcal{T}}, \psi_i^{\mathcal{D}}, \psi_i^{\mathcal{O}}, \psi_i^{\mathcal{G}}\right)'. \tag{5}$$

$\psi_i^{\mathcal{T}} \in \mathbb{R}^T$ denotes the tasks that are learned from the experiments and are represented by a series of binary numbers. The value of T is determined based on the number of tasks. For instance, if there are two tasks (i.e., delivery and tool use), T is assigned with value 1 (i.e., 0 for delivery task and 1 for tool use task). Note that, although some objects can be used to fulfill the same task (both bottles and mugs can be used to pour water from them), such task needs to be encoded individually for each object category. $\psi_i^{\mathcal{D}} \in \mathbb{R}^3$ denotes the dimensions of the bounding box of the object. $\psi_i^{\mathcal{O}} \in \mathbb{R}^4$ denotes the orientation of the object with respect to the base of the robotic manipulator in quaternion. $\psi_i^{\mathcal{G}} \in \mathbb{R}^{16}$ denotes a data vector which consists of all the information related to the grasp execution, such as initial joint position ($\mathbb{R}^3$) and joint velocity ($\mathbb{R}^3$) of each finger, position ($\mathbb{R}^3$) and orientation (in quaternion $\mathbb{R}^4$) of the hand with respect to the center of the object (i.e., the center of bounding box) and the strain-gauge data ($\mathbb{R}^3$) from each finger. The information contained in $\psi_i^{\mathcal{G}}$ will be applied to the target object to accomplish a desired task.

B. Network Initialization

As mentioned in Section I, most of the existing initialization methods eliminate the outliers from the training data set in the process of initialization to prevent problems happening in the data analysis. In our case, however, each individual training data (i.e., grasp candidate), even the outliers, need to be considered as candidates for initial neurons.

In this paper, a set of representative training data are selected to form the initial neurons for the network. To achieve this, the Gaussian kernel is utilized to measure the

similarities between the data and is defined as the classical squared exponential function with one constant hyperparameter, σ_k,

$$k(\psi_i, \psi_j) = \exp\left(-\frac{\|\psi_i - \psi_j\|_2^2}{2\sigma_k^2}\right). \quad (6)$$

Then, an objective function is constructed based on Eq. (6) as

$$\mathbb{T}_i = \frac{\frac{1}{J}\sum_{j=1}^{J} k(\psi_i, \omega_{0,j})}{\frac{1}{I-J-1}\sum_{l=1}^{I-J-1} k(\psi_i, \psi_l)}, \quad (7)$$

where $\omega_{0,j}$ represents the locations of the initial neurons. The numerator describes the relation between the current training data and the selected initial neurons, and the denominator characterizes the relation between the current training data and the rest of unselected training data. The proposed initialization method then greedily selects a serious of optimal weighting factor in the sense that they minimize Eq. (7). In other words, the desired initial neurons should be most dissimilar to each other and most similar to unselected training data. Specifically, after determining J initial neurons,

$$\omega_{0,J+1} = \psi_i^*, \quad (8)$$

where

$$\psi_i^* = \underset{\psi_i \in \Psi \backslash \Omega_0}{\operatorname{argmin}}\ \mathbb{T}_i \quad (9)$$

where Ω_0 represents the set of all selected initial neurons, and $\Psi \backslash \Omega_0$ denotes all unselected data in the training set. ψ_i^* will not be considered as a candidate for the initial neuron in the rest of initialization process. The details of the implementation of the proposed initialization method to select a total of J initial neurons are shown in Table I.

Input:	(1) Original training set, $\Psi = \{\psi_1, ..., \psi_I\}$ (2) Number of initial neurons, J
Output:	K optimal initial neurons, $\Omega_0 = \{\omega_{0,1}, ..., \omega_{0,K}\}$
1:	**if** $j = 1$ (i.e., the first neuron) **do**
2:	**for** every training data, ψ_i, **do**
3:	**Calculate** $\frac{1}{I-1}\sum_{l=1}^{I-1} k(\psi_i, \psi_l)$
4:	**end for**
5:	**Determine** the optimal neuron, ψ_i^*, as
6:	$\psi_i^* = \text{argmin}_{\psi_i \in \Psi} \frac{1}{\frac{1}{I-1}\sum_{l=1}^{I-1} k(\psi_i, \psi_l)}$
7:	**else**
8:	**for** every unselected training data, ψ_i, **do**
9:	**Calculate** $\mathbb{T}_i$ by Eq. (7)
10:	**end for**
11:	**Determine** the optimal neuron, ψ_i^*, by Eq. (9)
12:	**end if**
13:	**Update** $\Omega_0 = \Omega_0 \cup \omega_{0,j}$, where $\omega_{0,j} = \psi_i^*$
14:	**Stop**, if J neurons are selected. Otherwise, back to Step 1.

TABLE I: Proposed initialization method

C. Multi-layer Network

There are four major factors that characterize human grasp choices [25]:

- **Desired Tasks**: The tasks that have to be accomplished, such as delivery and tool use.
- **Object type**: The geometric information on the target object.
- **Object pose**: The position and the orientation of the target object.
- **Grasp type**: An appropriate task-oriented hand pose and the corresponding motion of the hand.

Based on these criteria, we develop a multi-layer network which is comprised of four connected layers. The structure (i.e., number of neurons/groups) of each layer is different, due to the different criteria employing on each layer.

The training data set, Ψ, is first clustered based on the task criterion on the first layer. The neurons on the first layer, $\omega_{j_t}^{\mathcal{T}}$ $(j_t = 1, ..., J_t)$, have the same data structure as $\psi^{\mathcal{T}}$ and are used to cluster Ψ into J_t ($J_t = 2^T$ or $J_t = 2^T - 1$) groups. Specifically, only the first portion of ψ_i (i.e., $\psi_i^{\mathcal{T}}$) is considered in the clustering process on this layer.

On the second layer, each group from the previous layer is further clustered into $J_d^{j_t}$ groups based on the dimensions of the training objects. The value of $J_d^{j_t}$ depends on the number of data centered around $\omega_{j_t}^{\mathcal{T}}$. Specifically, it is initially determined by the 2^K rule which is given by

$$J_d^{j_t} = \lceil \log K_{j_t} \rceil, \quad (10)$$

where K_{j_t} denotes the total number of training data that are assigned to the neuron $\omega_{j_t}^{\mathcal{T}}$. Hence, the number of neurons on the second layer may be different for each group. The neurons on the second layer are represented by $\omega_{j_t,j_d}^{\mathcal{D}}\left(j_d = 1, ..., J_d^{j_t}\right)$ which has the same structure as $\psi^{\mathcal{D}}$. The initial locations of $\omega_{j_t,j_d}^{\mathcal{D}}$ in each group are determined by applying the proposed initialization method and further adjusted to satisfy the conditions of the network dynamics (the details of which will be introduced in the next subsection). Eventually, the locations of neurons are formed the same pattern as the data vectors from the connected parent group on the previous layer.

Following the same procedure operated on the second layer, each group from the second layer is clustered into $J_o^{j_t,j_d}$ groups based on the orientations of objects, $\psi_i^{\mathcal{O}}$, on the third layer and continuously clustered into $J_g^{j_t,j_d,j_o}$ groups based on the grasp data, $\psi_i^{\mathcal{G}}$, on the forth layer. The neurons on the third and forth layers are represented by $\omega_{j_t,j_d,j_o}^{\mathcal{O}}\left(j_o = 1, ..., J_o^{j_t,j_d}\right)$ and $\omega_{j_t,j_d,j_o,j_g}^{\mathcal{G}}\left(j_g = 1, ..., J_g^{j_t,j_d,j_o}\right)$, respectively. Eventually, the network generates a codebook and each code in the codebook consists of a series of connected neurons

$$\Omega^c = \left\{\omega_j^c | \omega_j^c = \left(\omega_{j_t}^{\mathcal{T}}, \omega_{j_t,j_d}^{\mathcal{D}}, \omega_{j_t,j_d,j_o}^{\mathcal{O}}, \omega_{j_t,j_d,j_o,j_g}^{\mathcal{G}}\right)'\right\}.$$

Ω^c contains J^c representative grasps and will be used for the online grasp generation and execution. The structure of the proposed multi-layer network is illustrated in Fig. 2.

D. Network Dynamics

As mentioned in the precious section, the numbers of the initial neurons on the second, third and forth layers are determined based on the 2^K rule. Although it is easy to

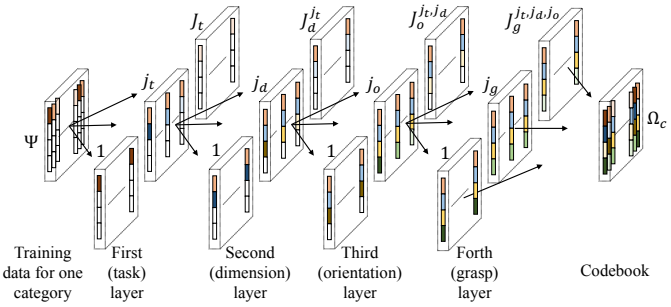
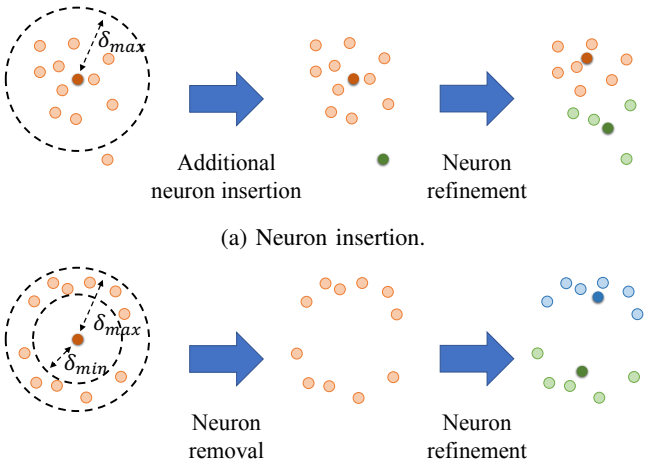


Fig. 2: The structure of the proposed multi-layer network. The training data represent the grasp candidates collected from the experiments. Neurons on different layer are associated with different portions of the training data. Eventually, each output data in the codebook consists of one particular neuron from each layer.



(a) Neuron insertion.

(b) Neuron removal.

Fig. 3: Neuron insertion (a) and removal (b).

implement, the number of the neurons may not be optimal for analyzing the data in the sense of accuracy of clustering. Moreover, the manipulators should keep improving the grasping skill as they interact with more and more objects. In other words, the codebook should constantly update through changing the sizes of layers in the network.

To address the issue of identifying a proper number of neurons, we propose an auto-growing algorithm to adjust the size of the network through adding or removing neurons. In order to do so, a distortion function is used to quantify the quality of clustering for each group and given by

$$\mathbf{J}_j = \sum_{i}^{I} \mathbf{1}\{\alpha_i = j\}\|\psi_i - \omega_j\|_2, j = 1, ..., J, \qquad (11)$$

where α_i is defined by Eq (1), and $\mathbf{1}\{\bullet\}$ represents an indicator function. Specifically, $\mathbf{1}\{\bullet\} = 1$ only when the argument inside is true, and 0 otherwise, i.e., $\mathbf{1}\{True\} = 1$ and $\mathbf{1}\{False\} = 0$. For instance, $\mathbf{1}\{2 = 2\} = 1$ and $\mathbf{1}\{5 = 3\} = 0$. Equation (11) characterizes the correlation between the neuron and the input data surrounding it. We then define a minimum threshold value, δ_{min}, and a maximum threshold value, δ_{max}, to control the compactness of each cluster (i.e., range of variation within each cluster). In particular, if the Euclidean distance between some input data and the closest neuron to which they have been assigned is greater than δ_{max}, then the input data which is the furthest from the neuron will be considered as an additional neuron and inserted into the network. This process is illustrated in Fig. 3a. On the other hand, if the distance between a neuron and every data surrounding it is less than δ_{max}, but the average value of distortion function, $\bar{\mathbf{J}}_j$, of that neuron is greater than δ_{min}, then such neuron will be removed from the network, as shown in Fig. 3b. Once the process of insertion and removal is completed, the locations of neurons are refined by the BLSOM. This procedure is repeatedly executed until the conditions of the network dynamics are satisfied. The details of this procedure is shown in Table II.

Conditions:	(1) $\|\psi_i - \omega_j\|_2 < \delta_{max}, \forall \alpha_i = j$
	(2) $\bar{\mathbf{J}}_j < \delta_{min}$
1:	**for** every group, $j = 1$ to J, **do**
2:	**while** either condition 1 or condition 2 is true **do**
3:	**if** condition 1 is false **do**
4:	**Determine** the index of the additional neuron by $\alpha_{insert} = \text{argmax}_{i \in I}\, \mathbf{1}\{\beta_i = j\}\|\psi_i - \omega_j\|_2^2$
5:	**Insert** $\psi_{\alpha_{insert}}$ as an addition neuron to the network
6:	**elseif** condition 2 is false **do**
7:	**Remove** ω_j from the network
8:	**end if**
9:	**Refine** positions of neurons by BLSOM
10:	**end while**
11:	**end for**

TABLE II: Dynamics of the proposed network.

V. Grasp Generation and Codebook Update

Using the codebook produced from the proposed multi-layer network, the process of generating a task-oriented grasp for a novel object and that of updating the codebook are presented in this section.

A. Task-Oriented Grasp Generation

Assuming the dimension and orientation of the target object is known, a data vector associated with the desired task and the target object is defined as

$$\Psi_{novel} = \left(\psi^{\mathcal{T}}_{novel}, \psi^{\mathcal{D}}_{novel}, \psi^{\mathcal{O}}_{novel}\right)'. \qquad (12)$$

Thus, a suitable grasp can be selected from the codebook by

$$\beta^* = \underset{j \in J^c}{\text{argmin}} \|\Psi_{novel} - \left(\omega^{c,\mathcal{T}}_j, \omega^{c,\mathcal{D}}_j, \omega^{c,\mathcal{O}}_j\right)'\|_2, \qquad (13)$$

where β^* denotes the index number of the selected representative grasp. The data related to the grasp execution, $\omega^{c,\mathcal{G}}_{\beta^*}$, is retrieved from $\omega^c_{\beta^*}$ and applied to the target object to accomplish the desired task.

B. Codebook Update

As mentioned in subsection IV-D, robots need to improve their grasping skills though constantly updating the codebook. Once the desired task is successfully executed by the selected representative grasp, the original codebook is updated with the data contained in Ψ_{novel} along with $\psi^{\mathcal{G}}_{novel}$ which is recorded during the grasp operation. Consequently, the relation between each representative grasp and the corresponding generalized models is enhanced.

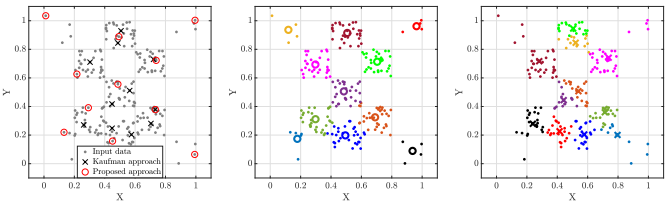
VI. Results

In this section, the performance of the proposed initialization method and the auto-growing algorithm is first evaluated in simulation, and then the proposed knowledge-based grasp planning method is validated through a number of experiments conducted with different knives.

A. Simulation Results

The proposed initialization method is applied to localizing the initial neurons on an artificial two-dimensional (2D) data set. The data set consists of 7 regions in which 30 data points are uniformly distributed and 4 regions located at each corner where some sparse data points appear. The data points are indicated as the dot ('•') markers in Fig. 4a. We also compare the performance of the proposed method with a popular approach, the Kaufman approach [13], in terms of the accuracy of the clustering. The Kaufman approach is one popular method in the initialization problems and its performance has been proved to be better than other approaches in [20]. The basic idea of the Kaufman approach is to select the first initial neuron at the most central location in the data set and iteratively determine the rest so that the resulting neurons can represent the dense distribution of the data set.

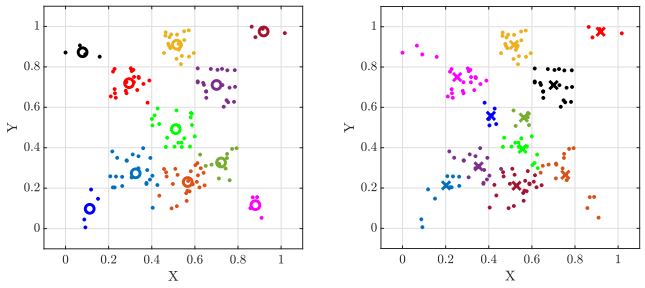
Following the procedure of the proposed initialization method introduced in subsection IV-B with $\sigma_k = 0.1$, 11 initial neurons are sequentially determined and indicated as the circle ('○') markers in Fig. 4a. Because we take the outliers into account in the process of localizing the initial neurons, the selected initial neurons do not only cover the majority part of the data set, but also indicate both the existence and the locations of the outliers. As a comparison, the selected initial neurons based on the Kaufman approach are indicated as the cross ('×') markers in Fig. 4a. The pattern of the data set can be represented by these neurons appropriately. The outliers (i.e., sparse data points close to the corners), however, are clearly neglected in the initialization process as expected. In order to further compare the accuracy of the proposed method with the Kaufman method, we refine the selected initial neurons indicated in Fig. 4a by the BLSOM. Given the maximum number of iterations $n = 2$ and the initial value of learning rate $\sigma_h(0) = 0.1$, the clustering result using the initial neurons selected by the proposed method is illustrated in Fig. 4b, where all the regions including the ones at corners are precisely located. In Fig. 4c, those corner regions, by contrast, are merged into the closest data points. As a result, the accuracy of the clustering result is low due to the effect of the outliers.



(a) Initial neurons (b) Proposed method (c) Kaufman method

Fig. 4: Initial positions of 11 neurons selected by the proposed method and the Kaufman method (a). Refined positions through BLSOM using the proposed method (b) and the Kaufman method (c).

To evaluate the performance of the proposed auto-growing algorithm, the optimal number of clusters for the same 2D data set is automatically determined by the proposed method. To demonstrate the advantage of the proposed method, a classic density-based method [22] (i.e., insert a neuron when there is a cluster far away from others and remove a neuron when two clusters are too close to each other) is also applied to the same data set.



(a) Proposed method (b) Density-based method

Fig. 5: Progressively select 11 neurons using the proposed method (a) and the density-based method (b).

In the simulation, we let $\delta_{max} = 0.2$ and $\delta_{min} = 0.1$. Given $n = 3$ and $\sigma_h(0) = 0.1$, the clustering result using the proposed method is demonstrated in Fig. 5a, where the data set is clustered into 11 groups including every regions in the center as well as the ones at corners. By contrast, the result of running the density-based approach to select 11 neurons is shown in Fig. 5b. Apparently, more neurons are needed to satisfy the conditions introduced in Table II due to the poor clustering result. As a consequence, the process of determining the number of clusters using density-based approach gets computationally demanding and inefficient.

B. Experimental Results

In this subsection, the proposed knowledge-based grasp planning is applied to the task-oriented grasp execution for a categories of knives.

1) Experimental Setup: The proposed approach is tested on a Barrett Whole Arm Manipulator (WAM) equipped with a three-fingered BarrettHand, as shown in Fig. 6. The test category consists of six different kitchen knives which are shown in Fig. 7. The dimensions of the bounding box of each knife are listed in Table III. The knives are placed on a foam base with inserting a small part of cutting edge into the base to increase the stability of the knives, as shown in Fig. 6.

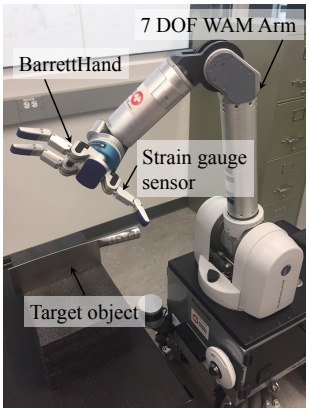


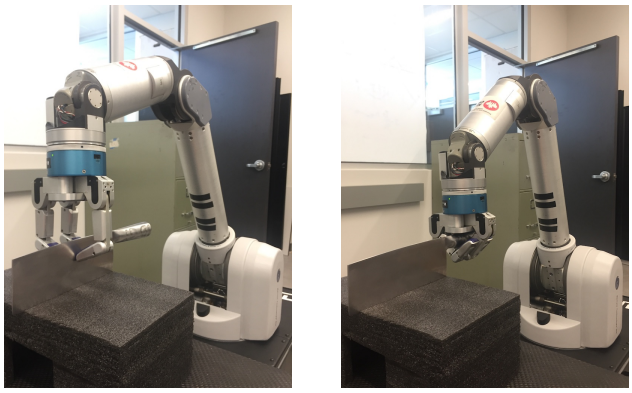
Fig. 6: Experimental setup.

Fig. 7: Training objects (knives) used in experiments.

Object #	Dimensions of bounding box (cm)		
	Length	Width	Thickness
1	29.6	8.4	2.8
2	30.7	8.3	1.8
3	29.6	5.1	2.1
4	29.5	3.4	1.7
5	25.8	2.4	1.6
6	22.1	2.4	1.6

TABLE III: Dimensions of the training knives.

2) Grasp Candidates Collection: In order to increase the reliability of each grasp candidate and efficiently eliminate the inappropriate grasps, the grasp candidates used to build up the training data set are collected in the experiments instead of simulations. Two tasks (delivery and tool use) are performed with each training knife, as shown in Fig. 8. We encode the delivery task as $(0, 0)'$ and the tool use task as $(0, 1)'$. The blade of each knife is grasped from the top to perform the delivery task as shown in Fig. 8a, and the handle of each knife is grasped to accomplish the tool use task as shown in Fig. 10c. The stain-gauge data can be acquired from the corresponding sensors equipped on BarrettHand and is used to control the amount of grasping force. Each knife is placed in 36 different orientations with respect to the base of WAM, and two grasp candidates associates with each orientation (one for each task) for each knife are executed. Eventually, 432 grasp candidates are collected from the experiments in total.



(a) Delivery task (b) Tool use task

Fig. 8: Collecting task-oriented grasp candidates through using training knives to accomplish desired tasks: delivery (a) and tool use (b).

3) Online Grasp Execution: Once the training data set is built up, we evaluate the proposed approach by performing task-oriented grasp on a number of novel objects (similar to the training objects but apparently different in both shape and size), as shown in Fig. 9. The dimensions of the bounding box of each test knife are listed in Table IV. We let $n = 2$, $\sigma_h(0) = 0.1$ and $\sigma_k = 1$ through out the entire network. Given $\delta_{max} = 0.5$ and $\delta_{min} = 0.25$ for the second layer, $\delta_{max} = 3$ and $\delta_{min} = 2$ for the third layer and $\delta_{max} = 0.15$ and $\delta_{min} = 0.1$ for the third layer, 36 representative grasps are generated to form a codebook for the category of knives. Figure 10 demonstrates that each test knife can be successfully grasped in accordance with the desired tasks. The information contained in the codebook for the knife category is updated with the new data after each grasping execution so as to improve the grasping skills.

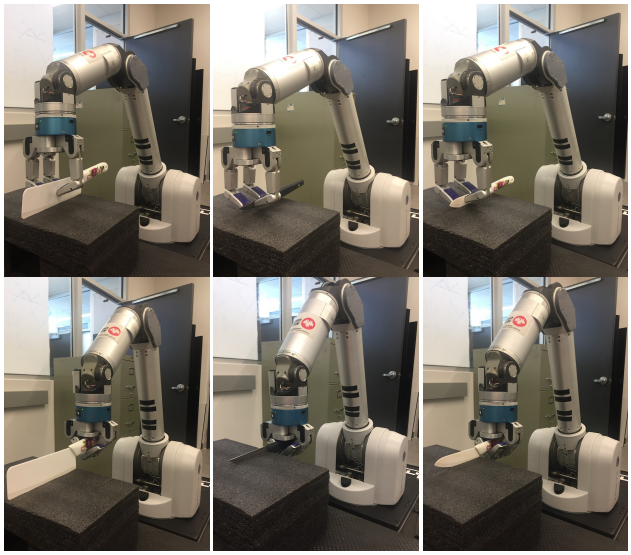
Fig. 9: Novel objects used to test the proposed approach.

VII. Conclusion

This paper presents a knowledge-based approach to the problem of task-oriented grasp planning. A multi-layer self-

Object #	Dimension of bounding box (cm)		
	Length	Width	Thickness
1	29.2	6.7	1.7
2	25.9	2.5	1.6
3	21.1	3.4	1.5

TABLE IV: Dimensions of the test objects (knives).



(a) Test knife 1 (b) Test knife 2 (c) Test knife 3

Fig. 10: Task-oriented grasp execution using test knives.

organized network is constructed to analyze the training data and generate a small number of representative grasps which can be applied to a group of similar objects. By using Gaussian kernel to measure the similarity between the training data, a novel method that can determine the locations of initial neurons while taking outliers into account is proposed. Simulation result shows that the proposed initialization method can accurately capture the outliers in the data set and render the distribution of input data. Moreover, an auto-growing algorithm is also developed so that an optimal number of clusters can be automatically and efficiently determined based on the structure of the data. Specifically, the codebooks are capable of constantly improving and adapting through each interaction with novel objects. The proposed knowledge-based approach is validated in a number of physical experiments in which a suitable grasp can be successfully generated for a novel object in accordance with a specific task.

Further work will include the incorporation with vision cameras to detect the pose of the target objects so as to enhance the practicality of the proposed grasp planning approach.

REFERENCES

[1] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2000, pp. 348–353.

[2] T. Yoshikawa, "Multifingered robot hands: Control for grasping and manipulation," *Annual Reviews in Control*, vol. 34, no. 2, pp. 199–208, 2010.

[3] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3d object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.

[4] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2765–2770.

[5] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[6] H. O. Song, M. Fritz, D. Goehring, and T. Darrell, "Learning to detect visual grasp affordance," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 798–809, 2015.

[7] L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater, and J. Santos-Victor, "Affordances in psychology, neuroscience, and robotics: A survey," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 1, pp. 4–25, 2016.

[8] M. Stark, P. Lies, M. Zillich, J. Wyatt, and B. Schiele, "Functional object class detection based on learned affordance cues," in *International conference on computer vision systems*. Springer, 2008, pp. 435–444.

[9] N. Curtis and J. Xiao, "Efficient and effective grasping of novel objects through learning and adapting a knowledge base," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 2252–2257.

[10] E. Nikandrova and V. Kyrki, "Category-based task specific grasping," *Robotics and Autonomous Systems*, vol. 70, pp. 25–35, 2015.

[11] C. Borst, M. Fischer, and G. Hirzinger, "Grasping the dice by dicing the grasp," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 4, 2003, pp. 3692–3697.

[12] R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragic, "Generalizing grasps across partly similar objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3791–3797.

[13] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[14] F. Sun, C. Liu, W. Huang, and J. Zhang, "Object classification and grasp planning using visual and tactile sensing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 7, pp. 969–979, 2016.

[15] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[16] M. Hagenbuchner, A. Sperduti, and A. C. Tsoi, "A self-organizing map for adaptive processing of structured data," *IEEE transactions on Neural Networks*, vol. 14, no. 3, pp. 491–505, 2003.

[17] T. Logeswari and M. Karnan, "Hybrid self organizing map for improved implementation of brain mri segmentation," in *International Conference on Signal Acquisition and Processing*, 2010, pp. 248–252.

[18] J. F. Steffen, R. Haschke, and H. Ritter, "Som-based experience representation for dextrous grasping," in *International Workshop on Self-Organizing Maps: Proceedings*, 2007.

[19] M. Johnsson and C. Balkenius, "Sense of touch in robots with self-organizing maps," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 498–507, 2011.

[20] J. M. Pena, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern recognition letters*, vol. 20, no. 10, pp. 1027–1040, 1999.

[21] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert systems with applications*, vol. 40, no. 1, pp. 200–210, 2013.

[22] Q. Zhang, C. Zhu, L. T. Yang, Z. Chen, L. Zhao, and P. Li, "An incremental cfs algorithm for clustering large data in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1193–1201, 2017.

[23] H. Liu and X.-j. Ban, "Clustering by growing incremental self-organizing neural network," *Expert Systems with Applications*, vol. 42, no. 11, pp. 4965–4981, 2015.

[24] T. Kohonen, "Essentials of the self-organizing map," *Neural networks*, vol. 37, pp. 52–65, 2013.

[25] T. Feix, I. M. Bullock, and A. M. Dollar, "Analysis of human grasping behavior: Correlating tasks, objects and grasps," *IEEE transactions on haptics*, vol. 7, no. 4, pp. 430–441, 2014.