Learning Action-oriented Grasping for Manipulation

Muhayy Ud Din[1], M Usman Sarwar[2], Imran Zahoor[2], Wajahat M Qazi[2] and Jan Rosell[1]
[1]Institute of Industrial and Control Engineering (IOC)
Universitat Politècnica de Catalunya (UPC) – Barcelona Tech, Spain.
[2]Department of Computer Science
COMSATS University Islamabad – Lahore Campus, Pakistan.

***Abstract*— Complex manipulation tasks require grasping strategies that simultaneously satisfy the stability and the semantic constraints that have to be satisfied for an action to be feasible, referred as action-oriented semantic grasp strategies. This study develops a framework using machine learning techniques to compute action-oriented semantic grasps. It takes a 3D model of the object and the action to be performed as input and provides a vector of action-oriented semantic grasps. We evaluate the performance of machine learning (particularly classification techniques) to determine which approaches perform better for this problem. Using the best approaches, a multi-model classification technique is developed. The proposed approach is evaluated in simulation to grasp different kitchen objects using a parallel gripper. The results show that multi-model classification approach enhances the prediction accuracy. The implemented system can be used as to automate the data labeling process required for deep learning approaches.**

I. INTRODUCTION

Robotic grasping is an essential component in the performance of manipulation tasks. It should take into account the way to hold an object in order to execute a given manipulation action. Classical approaches to the grasp planning consider robotic grasping as a pure geometric problem, focusing on optimizing the grasp quality based on *form closure* or *force closure* [1] [2] [3] [4]. These quality measures provide a well established mathematical framework for grasp analysis. The main aim of these approaches is to hold the objects with the robot gripper in such a way that it should not drop. However, in practical scenarios, the goal of the grasp is to perform a particular task such as *pick a can to pour the drink in a glass* as shown in Fig 1-a. In this context, a grasp planner that just satisfies the stability constraints may not be able to satisfy the task related requirements. This process of acquiring task related constraints can be incorporated through learning techniques.

Recently, deep learning approaches have gained popularity due to their effectiveness and accuracy. For grasping applications, these approaches usually take depth images as input and return a set of grasp points such that if fingers are placed there, a stable grasp is guaranteed. Deep learning-based approaches are data hungry [5] and require a huge amount of labeled data for an effective learning during the training process. In case of learning stable grasps, the data could be generated in a relatively easy way by parameterizing stable grasps. On the contrary, for the action-oriented grasping, the requirements to satisfy the action related constraints depend on the type of the action and the object geometry and are therefore difficult to generalize. A grasp may be valid for performing the particular action *pick to clean* and the same grasp may be not valid for the action *pick to pour* (as shown in Fig. 1-b) due to the action constraints. It is therefore challenging to find a way to prepare data in an automatic way to apply deep learning based approaches.

Fig. 1: Shows the *valid* and *invalid* grasping pose for *pick-to-pour* action computed by the proposed framework: a) shows the *valid* grasping pose. b) shows the *invalid* grasping pose.

This study uses learning techniques to develop an action-oriented semantic grasp generation system, i.e., a system that learns how to grasp an object to perform a particular action. It can be used to automate the process of *dataset* generation for vision-based task-oriented deep learning approaches. In addition to this, it can also be used as a stand alone semantic grasp generation system for model-based approaches, as it is presented in this paper.

The main contribution of this study is to use machine learning techniques to learn how to generate action-oriented semantic grasps. We perform a benchmarking to analyze which machine learning approaches are best fit for this particular problem. In order to increase the prediction accuracy, we choose the set of best techniques (those that have highest prediction accuracy in benchmarking) and develop a multi-model prediction approach, which applies a voting scheme to decide the feasibility of grasps according to the action to be performed. Finally, to make the approach complete, a motion planner is called to execute the grasp with the best score.

Rest of the paper is structured as follow. Sec. II formulates the problem and provides the overview of the proposed

This work was partially supported by the Spanish Government through the project DPI2016-80077-R.

978-1-7281-0302-0/19/$31.00 ©2019 IEEE

solution. Sec. III explains the proposed approach and Sec. IV explain the results. Finally, Sec. V concludes the work.

II. Problem Statement and Solution Overview

A. Problem statement

Let $\mathcal{G}$ be the grasp space representing the set of all possible force closure grasps for a robotic-gripper $\mathcal{R}$, and let the graspable objects in the robot workspace be represented as $\mathcal{O} = \{o_1, \ldots, o_n\}$. A grasp $g \in \mathcal{G}$ is parameterized as $g = (x, y, z, q_x, q_y, q_z.q_w)$, representing the position and orientation (quaternion) of the gripper w.r.t the object frame in 3D space. It is computed in such a way that if the robot closes the fingers from this pose, it guarantees force closure grasp.

Let also the action space $\mathcal{A} = \{a_1, \ldots, a_n\}$ contain the set of all possible actions that a robot can perform with $\mathcal{R}$. The grasp feasibility evaluation function $\mathcal{F}$, represented in Eq.(1), evaluates the feasibility of the given grasp $g \in \mathcal{G}$ against the given action $a \in \mathcal{A}$ for the particular object $o \in \mathcal{O}$ and returns valid or invalid, depending on whether the grasp is feasible or infeasible (which includes the cases when it is not applicable to the object).

$$\mathcal{F} : \mathcal{G} \times \mathcal{A} \times \mathcal{O} \rightarrow \{\text{valid, invalid}\} \quad (1)$$

In this work we will consider that the robot is performing manipulation tasks in a kitchen environment with a set of kitchen objects $\mathcal{O}_{\text{kitchen}}$ (e.g., cup, plate, and bottle) such that $\mathcal{O}_{\text{kitchen}} \subseteq \mathcal{O}$. The manipulation task involves the set of actions $\mathcal{A}_{\text{kitchen}} = \{$*grasp_to_clean, grasp_to_serve, grasp_to_pour, grasp_to_push, grasp_to_pull*$\}$ such that $\mathcal{A}_{\text{kitchen}} \subseteq \mathcal{A}$.

The, the problem is to take as input an action $a \in \mathcal{A}_{\text{kitchen}}$, the 3D model of the of the robot gripper $\mathcal{R}$, and an object $o \in \mathcal{O}_{\text{kitchen}}$, and generate a set of force-closure grasps, use the grasp the feasibility evaluator $\mathcal{F}$ to evaluate their feasibility for the given action a, returning the best ones, in such a way that if the robot executes one of these grasps, the corresponding manipulation action will be able to be successfully executed.

B. Solution overview

The solution to above stated problem is divided in two phases: a *learning phase* and an *application phase*, as shown in Fig. 2:

1) *Learning phase:* It generates a set of force closure grasps $\{g_o\} \in \mathcal{G}$ for each object $o \in \mathcal{O}_{\text{kitchen}}$ using a grasping simulator. In order to generate the *training dataset*, these grasps are labeled as valid or invalid as a function of the actions in $\mathcal{A}_{\text{kitchen}}$. The *training dataset* is then used by the machine learning techniques (such as decision trees classifiers) to learn the behavior of the function $\mathcal{F}$.
2) *Application phase:* During this phase a set of arbitrary grasps is generated for the objects that are in the workspace using the grasping simulator, and their feasibility is evaluated using the learned function $\mathcal{F}$. Finally, the set of feasible grasps are forwarded to the motion planner to compute the robot path.

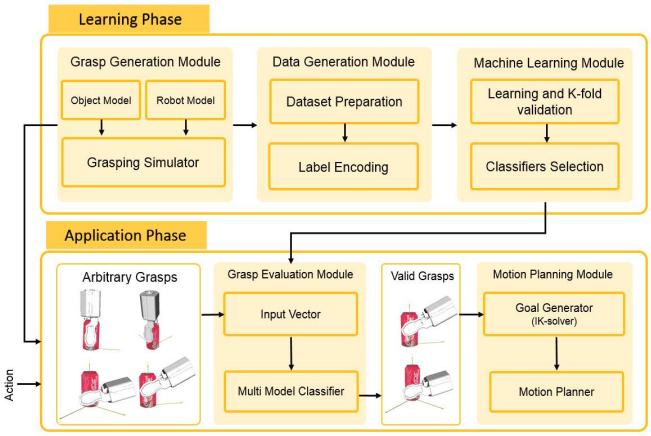

Fig. 2: Framework for computing action-oriented semantic grasps. Application phase shows the set of computed grasps to perform *grasp_to_pour* action.

III. Learning Action-Oriented Grasping

This section provides a detailed description of the methods used during the *learning* and *application* phases.

A. Training data generation

1) Grasp synthesis: To generate the *training dataset* for learning how to compute action-oriented semantic grasps, a stable grasp generation system is required. The GraspIt! (a simulation tool for grasp planning and visualization) is used for this purpose. The training dataset is prepared using 20 kitchen objects. The meshes of these objects are loaded into the GraspIt! (one by one) along with the model of the parallel gripper of the TIAGo robot. Since the actions in $\mathcal{A}_{\text{kitchen}}$ are usually applied to objects that are placed on a stable support, bottom grasps are discarded due to collisions with the supporting surface. In order to avoid such grasps, a stable supporting surface, such as a table, is introduced in GraspIt! in order to be considered as an obstacle during grasp planning process.

The grasps are generated using the eigen grasp planner [6]. It considers the gripper as a free-flying robot and randomly samples the poses of the gripper around the object to obtain the valid grasps. In order to ensure that the resultant grasps are force closure, it uses an energy function [6] as a quality measure (the lesser the energy the better the grasp). Only those grasps that have the energy score below a given threshold are kept. The planner runs for 10 seconds for each object and generates around 70,000 grasps. Among them the first 20 grasps (according to the energy function) are chosen for the *training dataset*.

2) Data labeling: The generated grasps are then manually labeled to prepare the *training dataset*. Each generated grasp for the given object $o \in \mathcal{O}_{\text{kitchen}}$ is evaluated corresponding to all actions in $\mathcal{A}_{\text{kitchen}}$ and labelled as *valid* or *invalid* for each action. The feature vector (shown in Fig. 3) consists of: 1) a

Feature Vector									
x	y	z	q_x	q_y	q_z	q_w	object	action	class label

Fig. 3: Feature Vector

grasp, represented as $g = (x, y, z, q_x, q_y, q_z, q_w)$ representing the pose of the gripper in the object frame, 2) the name of the object family such as cup, glass or jug, 3) the name of the action in $\mathcal{A}_{\text{kitchen}}$ such as *grasp-to-pour*, 4) the class label such as valid or invalid.

The non-numeric attributes of the *dataset* are transformed to numerical values. For instance, if object class contains 20 number of objects, the label encoder transforms these 20 non-numeric values to a set of numeric values $\{0, 1, \ldots, 19\}$, each value corresponding to a particular object. This label encoding scheme is applied to all the non-numeric attributes of feature vector.

B. Learning phase

The supervised learning is applied using the classification techniques over the generated *dataset*. The *dataset* is splitted into the *training dataset* (containing 80% of the data) and *test dataset* (containing 20% of the data). The learning is perform using the *scikit-learn*-`https://scikit-learn.org/stable/`, a Python-based library for machine learning algorithms. To determine which classification techniques perform better for this particular type of data, a benchmarking is performed using the most commonly used classification techniques. These techniques are: Gradient Boosting Classifier (GBC), Gaussian Naive Bayes Classifier (GNC), Linear Discriminant Analysis (LDA), K-Neighbors Classifier (KNC), Decision Tree Classifier (DTC), Bernoulli Naive Bayes Classifier (BNC) Linear Support Vector Classifier (LSC), Random Forest Classifier (RFC), and Logistic Regression Classifier (LRC), the detailed description of these algorithms can be found in [7]. The prediction accuracy of these techniques over the *test dataset* is compared. The results of the comparison show that GBC, DTC, and RFC have the highest accuracy score (above 90%) as shown in Fig. 4:

- The *Decision Tree Classifier* (DTC) uses a tree representation to solve the problem. Attributes correspond to internal nodes of the tree and leaf nodes correspond to the labels of the classes. DTC creates a model that predicts the value of the target variable by learning simple decision rules extracted form the data features.
- The *Random Forest Classifier* (RFC) develops multiple decision trees. These trees are then merged together to obtain stable an accurate prediction.
- The *Gradient Boosting Classifier* (GBC) develops a prediction model in terms of an ensemble of weak prediction models (decision trees are used as weak prediction models). The model is build in a stage-wise fashion and generalizes by optimizing an arbitrary cost function.

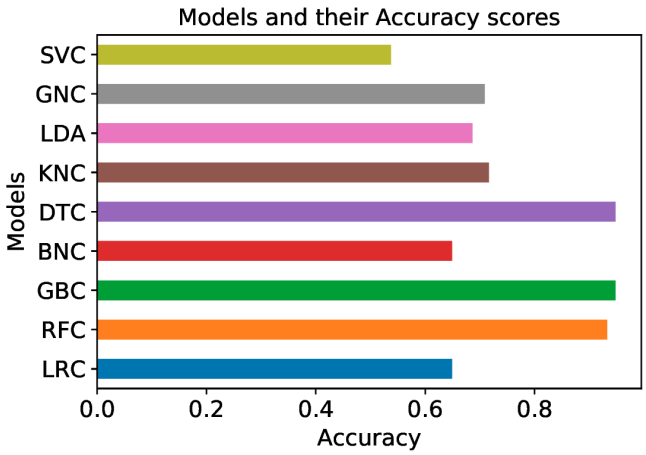

Fig. 4: Prediction accuracy comparison on *testdata* for benchmarking.

These three models are saved to build the multi-model classifier that is used during the application phase.

C. Application phase

1) Grasp feasibility evaluation: The application phase can be seen as a grasp filtration module that takes a vector of arbitrary grasps, generated by the GraspIt! for a particular object $o \in \mathcal{O}$. For each generated grasp, the grasp evaluation module generates an input vector that includes a grasp, name of the object class and action to be performed. These input vectors are passed to the multi-model classifier that classifies each input vector using the three best classifiers that are selected according to the process explained in Sec. III-B. To determine the set of feasible grasps for the particular action, the voting scheme is applied to the outputs of these classifiers. It evaluates the result of each classifier internally and labels the grasp as valid if at least two classifiers classify it as valid and invalid otherwise.

2) Motion planning: Once the set of feasible grasps is selected, motion planning module is responsible for computing the collision-free path to move the gripper from the given initial state to the goal state (i.e., computed feasible grasping pose). To compute the goal configurations for motion planner, the transfer function $\mathcal{T}(g) \rightarrow \mathbf{T}_{\mathbf{o}}^{\mathbf{h}}$ transforms each grasp g to an homogeneous transformation in the object reference frame. To set a motion planning query, the gripper pose is first computed into the word frame using Eq.(2),

$$\mathbf{T}_{\mathbf{h}}^{\mathbf{w}} = \mathbf{T}_{\mathbf{o}}^{\mathbf{w}} \cdot (\mathbf{T}_{\mathbf{o}}^{\mathbf{h}})^{-\mathbf{1}}, \tag{2}$$

where, $\mathbf{T}_{\mathbf{o}}^{\mathbf{w}}$ and $\mathbf{T}_{\mathbf{h}}^{\mathbf{w}}$ represent the object and gripper poses in the world reference frame, respectively. The arm configuration of the manipulator for each grasp is computed by solving the inverse kinematics for the corresponding $\mathbf{T}_{\mathbf{h}}^{\mathbf{w}}$. These configurations are then passed to the collision checker to obtain the valid goal configurations.

Motion planning is performed using *The Kautham Project* [8], a C++-based tool for motion planning. It

provides the sampling based motion planners to plan under geometric, differential and physics-based constraints. Open Motion Planning Library (OMPL-`https://ompl.kavrakilab.org/`) is used as a core set of sampling-based motion planners. RRTConnect [9], a sampling-based motion planner is used to compute the collision-free path to move the robot for performing the required action.

IV. Evaluation

A. Simulation scenarios

The proposed approach is tested by grasping several kitchen objects to perform different actions. For instance, an example scenario consists of the TIAGo robot operating in a kitchen environment, depicted in Fig. 1. The goal is to execute *grasp-to-pour*, *grasp-to-clean* actions for coke-can i.e., grasp the coke-can for pouring the drink into the glass and grasp the coke-can to through it into the bin, respectively.

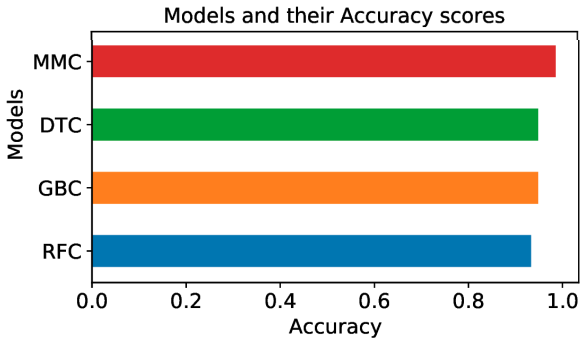

Fig. 5: Comparison of prediction accuracy in the application phase.

The simulation setup is generated using *The Kautham Project*. The communication between *Graspit!*, *scikit-learn*, and *The Kautham Project* is performed using ROS-(`http://wiki.ros.org/`.

B. Classification accuracy during application phase

The key contribution of this approach is the action-oriented classification of the grasping poses. We use multi-model classification approach that selects the best classifiers (having higher prediction accuracy on *test dataset*) and encapsulate them in a single algorithm which applies the voting scheme over the prediction of those models. Fig. 5 shows the accuracy comparison of the individual classifier and the multi-model classifier. The multi-model classifier enhances the overall prediction accuracy of the system. The true positive, true negative, false positive and false negative predictions of each approach are depicted in Fig. 6 in terms of confusion matrices.

The results show that the proposed approach (multi-model classifier) has high prediction accuracy during the application phase. Each grasp is evaluated as a function of the action to be performed and the type of the object, to label it as valid or invalid. The multi-model classifier can be used to automate the data labeling process for deep learning based approaches.

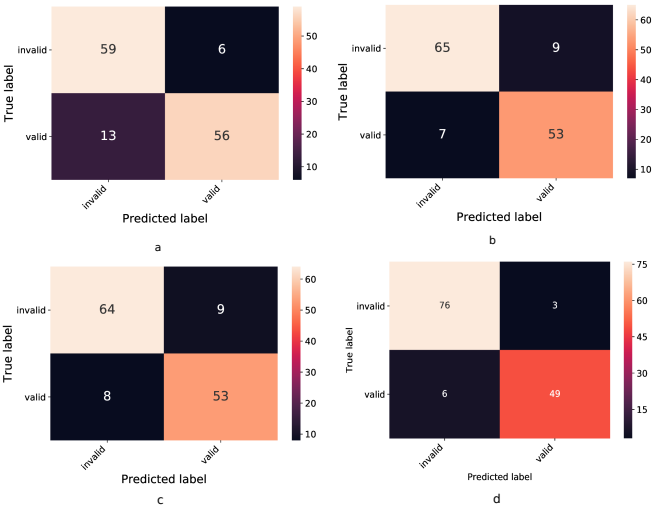

Fig. 6: Confusion matrices of the selected classifiers a) decision tree classifier, b) gradient boosting classifier, c) random forest classifier, d) multi model classifier.

V. Conclusions

This study proposed an approach to generate task-oriented semantic grasps using machine learning techniques. To analyze which machine learning approaches are suitable for this problem, a benchmarking study is performed. It shows that Decision Trees Calssifiers (DTC), Gradient Boosting Classifiers (GBC) and Random Forest Classifiers (RFC) have the highest accuracy to address this problem. A multi-model classification approach is proposed that uses these three classifiers and applies a voting scheme for classification. The proposed approach is tested with various scenarios, the multi-model classifier shows even a better accuracy.

As an ongoing work, this approach is being used to automate the data labeling process for deep learning approaches for manipulation planning, in such a way that from an image of an object and the type of action to be performed with it, the system returns the grasp to be performed.

References

[1] V.-D. Nguyen, "Constructing force-closure grasps," *The International Journal of Robotics Research*, vol. 7, no. 3, pp. 3–16, 1988.

[2] C. Ferrari and J. F. Canny, "Planning optimal grasps." in *ICRA*, vol. 3, 1992, pp. 2290–2295.

[3] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 886–900, 2012.

[4] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous Robots*, vol. 38, no. 1, pp. 65–88, Jan 2015.

[5] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning— the mit press," *Cambridge, Massachusetts*, 2016.

[6] M. Ciocarlie, C. Goldfeder, and P. Allen, "Dimensionality reduction for hand-independent dexterous robotic grasping," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3270–3275.

[7] M. Bowles, *Machine learning in Python: essential techniques for predictive analysis*. John Wiley & Sons, 2015.

[8] J. Rosell, A. Pérez, A. Aliakbar, Muhayyuddin, L. Palomo, and N. García, "The kautham project: A teaching and research tool for robot motion planning," in *Proc. of the IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2014.

[9] J. J. Kuffner Jr and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, 2000.