Robot learning by Single Shot Imitation for Manipulation Tasks

Mohit Vohra[1] and Laxmidhar Behera[1,2], *Senior Member, IEEE*

***Abstract*— In this work, we present a Programming by imitation for a robotic manipulation system, which can be programmed for various tasks from only a single demonstration. The system is primarily based on the three components: *i*) scene parsing, *ii*) action classification, and *iii*) dynamic primitive shape fitting. All the above modules are developed by leveraging state-of-the-art techniques in 2D and 3D visual perception. The primary contribution of this system paper is an imitation-based robotic system that can replicate highly complex tasks by executing elementary task-specific program templates, thus avoiding extensive and exhaustive manual coding. In addition, we contribute by introducing a primitive shape fitting module by which it becomes easier to grasp objects of various shapes and sizes. To evaluate the system performance, the proposed robotic system has been tested on the task of multiple object sorting and reports 91.8% accuracy in human demonstrated action detection, 76.1% accuracy in action execution, and overall accuracy of 80%. We also examine the proposed system's component-wise performance to demonstrate the efficacy and deployability in industrial and household scenarios.**

I. Introduction

Applications of robotics are widespread: ranging from warehouse automation [1][2], construction autonomous [3], table tennis [4] to domestic tasks [5]. With the advancement in machine learning based techniques, state-of-the-art robots are getting enabled with more and more artificial general intelligence.

However, developing a general-purpose robotics system capable of performing a diverse range of tasks is still challenging. One of the primary challenges is to program a robot for each new task. Machine-level programming of a robotic system for each task can be quite difficult because it is not practically feasible to program the robotic system for all the cases, including every combination of the end-user requirements, environment and task. Therefore, alternative techniques such as Learning by Demonstration (LbD) or imitation learning are preferred. These techniques aim to enable end-users to program a general-purpose robot for their specific requirements by demonstrating the desired behavior.

Consider, for example, a domestic service robot that an owner wishes to sort the objects as per his requirement .i.e. either push some objects to one corner of the table or place (stack) some objects over the container, as shown in Fig. 1. To perform the task by the robotic system, it should estimate which object has to be pushed or which object has to be stacked on the top of which object. In addition, each time the task is performed, the robotic system should adapt to the user's requirements.

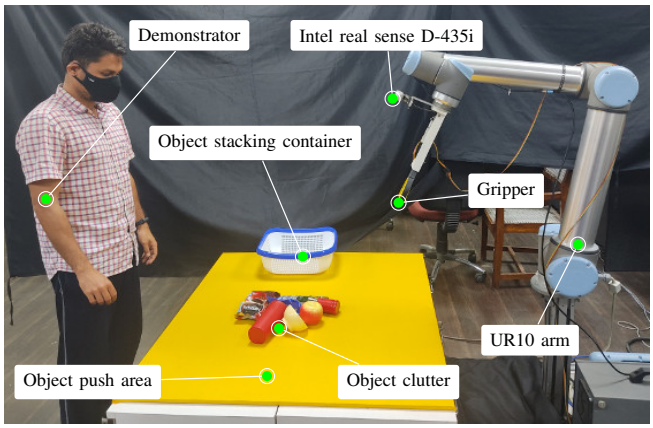


Fig. 1: The experimental Setup for the proposed system.

To perform according to the user requirement, we present an imitating robotic manipulation system that can be programmed according to the user requirement by a single human demonstration. The proposed system in this paper is pillared by the idea that the complete human demonstration can be decomposed into elementary tasks like single object pushing, object stacking, and object rotation. In order to replicate the complete task, the robotic system is equipped with the program templates corresponding to each elementary task, and the sequence of program templates to be executed, and arguments for each template like object ID or the final position, can be extracted from human demonstration. The major contributions of this system paper are as follows:

- Imitable robotic system capable of performing complex tasks programmable by single shot demonstrations.
- A scene parsing and an action classification module for complex visual demonstration analysis.
- A dynamic primitive shape fitting module to estimate the object's grasping pose.
- The extensive experiments to evaluate the robotic system's performance.

Remainder of this paper is organized as follows. Sec. II provides the overview of the related work in the context to this paper. Sec. III provides the detailed overview of our method. Sec. IV demonstrates the algorithmic flow of the system. Sec. V provides experimental evaluation and finally, the Sec. VI concludes the paper.

II. Related Work

Learning by Demonstration (LbD) and imitation learning (IL) are popular approaches to enable robots with skills

[1]Authors are with Indian Institute of Technology Kanpur, India, e-mail:{mvohra, lbehera}@iitk.ac.in.
[2]Author is with TCS Innovation Labs, Noida, India

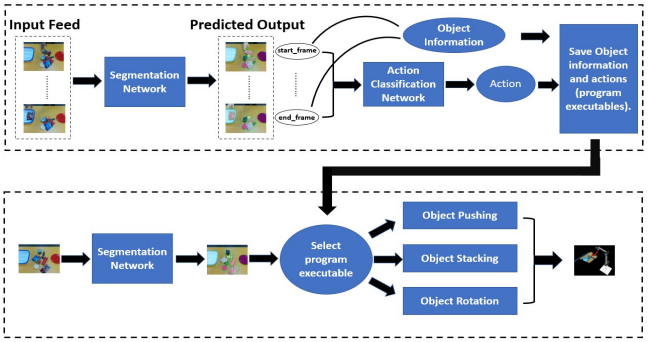


Fig. 2: Overview of imitating robotic system, the top block is demonstration phase and the bottom is the execution phase.

without having to program them by hand [6]. These techniques are used in a number of robotic applications such as robotic surgery [7], helicopter maneuvers [8], grasping of objects [9], carrot grating [10], and table cleaning [11]. Typically, these frameworks rely on human demonstrations in clutter-free environments. Hence, in [12], the authors propose the importance weighted skill learning technique for skill representation and reproduction. Further, to reduce the burden on human experts to provide demonstrations, the author [13] proposes to use a hierarchy of supervisors for learning from demonstrations. Interestingly, most LbD methods focus on low-level primitive actions such as manipulator trajectories. However, learning only motion primitives is not enough to solve high-level complex multi-step goal-oriented tasks. Hence, in [14], the authors propose the behavior trees as an alternative representation of the high-level task and present a method of learning behavior trees from the demonstration. In [15], the author focuses on programming the industrial robot for assembly tasks by parsing the human demonstration into a series of assembly skills and compiling the skill as robot executables. Similarly, in [16], the author presents a system for programming a robot to perform complex object manipulation tasks by observing a single RGBD video demonstration. While their work highly inspires our work, a requirement of accurate 3D models of the object can limit its practical feasibility. In [17], the author uses the simulated data to provide the demonstrations for training and successfully tested on real-world data for the task of an object placing and pushing.

III. Methodology

The workflow of our robotic system is shown in Fig. 2. It consists of two phases: a demonstration phase and an execution phase. In the demonstration phase, the system decomposes the demonstrated task into a sequence of elementary tasks (action primitives). While in the execution phase, the robotic system performs the demonstrated task in real-time. In both phases, raw visual information is the only source of knowledge to extract the crucial information. In the demonstration phase, the action performed by the demonstrator and corresponding manipulated object ID is extracted using raw visual information. Similarly, the object localization and grasp/push pose estimation (depending on the action performed by the demonstrator) in the execution phase is done using the raw visual feed. The following section will briefly explain the modules that allow the robotic system to extract the information in an autonomous mode.

Fig. 3: Output of `SegCNN`

A. Scene parsing

We utilize the visual information in order to obtain knowledge, such as:

1) Detecting the start and end of the action primitive. The frames at which these events occur are referred as `start_frame` (f_s) and `end_frame` (f_e).
2) Object IDs which are manipulated by the demonstrator.
3) The manipulated object's information i.e. final location or the stack object ID over which the manipulated object is placed.

In previous works [16], the demonstration consists of a single task only. Whereas we are interested in a complex human demonstration that may contain multiple action primitives. Therefore, we extract a pair of f_s and f_e for each primitive. In order to achieve that we take advantage of pixel-wise semantic segmentation using CNNs (`SegCNN`). The `SegCNN` plays a major role in parsing the visual scenes from the live feed. It enables the proposed system to autonomously and precisely reason about the object attributes such as its location, boundaries, and additionally, its state of occlusion.

The architecture of the `SegCNN` network is based on an encoder-decoder mechanism and is a modified form of [18]. Table I depicts the network architecture including the parameters and spatial dimensions. The input to the `SegCNN` is an RGB image of size 640×480. The image is passed through a series of convolutional layers followed by ReLU and Batch-Norm layers. The spatial size of the feature maps is gradually reduced to $1/16^{th}$ of the original image size. Before the final prediction, the feature maps from the encoding stages are upsampled to one-half of the original image size by using deconvolutional layers. The upsampled feature maps are then concatenated and passed through a convolutional layer to combine multiscale information and to obtain final predictions at the same resolution of the input image size. Fig. 3 shows some of the outputs of the network.

B. Action classification

The action classification module classifies the actions performed by the demonstrator onto the objects, namely object pushing, stacking, and rotating. All such actions require knowledge of temporal information about the visual scenes. Therefore, we employ a CNN in conjunction with LSTM as explained below.

The CNN-LSTM architecture consists of an encoder followed by one LSTM unit. The architecture of the CNN encoder is the same as that of `SegCNN`. The encoder transforms

	Layer	k	Output Tensor Dim
#0	Input	-	H×W
	Encoder		
#1	Conv	3	32 × H/2 × W/2
#2	Conv	3	64 × H/2 × W/2
#3	Conv	3	64 × H/2 × W/2
#4	Conv	3	128 × H/4 × W/4
#5	Conv	3	128 × H/4 × W/4
#6	Conv	3	128 × H/4 × W/4
#7	Conv	3	256 × H/8 × W/8
#8	Conv	3	256 × H/8 × W/8
#9	Conv	3	256 × H/8 × W/8
#10	Conv	3	512 × H/16 × W/16
#11	Conv	3	512 × H/16 × W/16
#12	Conv	3	512 × H/16 × W/16
	Decoder		
#13	DConv (#12)	3	512 × H/8 × W/8
#14	DConv	3	256 × H/4 × W/4
#15	DConv	3	128 × H/2 × W/2
#16	Conv	3	64 × H/2 × W/2
#17	Dconv (#9)	3	256 × H/4 × W/4
#18	Dconv	3	128 × H/2 × W/2
#19	Conv	3	64 × H/2 × W/2
#20	Dconv (#6)	3	128 × H/4 × W/4
#21	Conv	3	64 × H/2 × W/2
#22	DConv (#16 ⊕ #19 ⊕ #21)	3	144 × H×W
#23	Conv	3	128 × H×W
#24	Conv	3	96 × H×W
#25	Conv	3	12 × H×W

Table I. `SegCNN` architecture.'⊕' denote concatenation. H, W denotes image dimensions, and k is the kernel size.

the input image frame into embeddings of 1024 and passes to the LSTM unit with 1024 as the number of input features, 384 as the number of hidden units. The output of the LSTM unit is then passed through fully connected layers followed by a softmax layer to obtain final predictions.

C. Program Executables

The goal of this paper is to replicate the complex tasks using a set of primitive actions. In this work we adhere to three action primitives: object pushing, object stacking, and object rotation. In order to replicate these actions, the robotic system is equipped with corresponding program executables which are discussed below.

Object Pushing: In this, the demonstrator pushes or displaces an object without lifting it significantly from the table surface. The robotic system is programmed for specific steps to replicate the task, including bringing the gripper closer to the target object, aligning the gripper towards the final position, and moving the gripper parallel to the table surface to the final location. Further, the target object ID and final position are extracted from the demonstration.

Object Stacking: In this, the demonstrator grasps an object and places it over another object. The robotic system is programmed for specific steps to replicate the task, for example, reaching near to the target object, grasping the object, lifting the object, moving the object over to the stacking object, and place (or drop) the object over the stacking object. The arguments for this executable are extracted from the human demonstration.

Object Rotation: Pouring is one of the essential tasks used frequently in households. It relies on rotating or tilting a bottle that holds some liquid [19]. So instead of the exact pouring task, we focus on the object rotating task, which can be extended to a more complicated task of liquid pouring in the future. Object rotating task is when the demonstrator first grasps a bottle and rotates it over a cup. Further, the demonstrator places the bottle back to its original configuration. The steps required to execute the task are grasping the bottle, bringing the bottle near the cup, rotating the bottle over it, and finally, placing it back to its original position.

(a) (b) (c) (d)

Fig. 4: Failure of PCA to obtain min-area rectangle axis. PCA axis, and the proposed method axis.

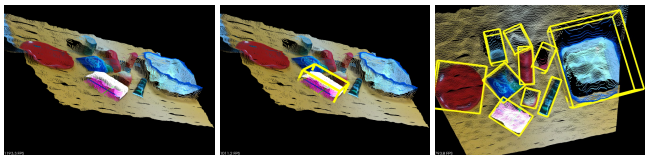
D. Dynamic model fitting

Grasping of the object is an unsolved problem and various solutions have been proposed in literature. For example, the author [20] proposes a model-free grasp pose estimation method that generates multiple grasp hypotheses by processing the visible regions of the object. One of the main limitations of the above method [20] is that it considers only the case of vertical grasping. In vertical grasping, the robotic manipulator will approach the object along the surface normal direction, where the surface is the base over which the objects will be placed. On the other hand, in a more general case, where the robotic system is not constrain for specific grasping direction, the author [21] uses the CAD model for estimating grasp poses for the target objects. Since most of the objects used in this work are general household objects, hence generating the CAD model for grasp pose estimation is not feasible.

Instead of estimating the exact CAD model of the target objects, we aim to approximate the target objects with simple shape primitives like cuboidal or cylindrical shapes. One of the main advantages of approximating the objects with shape primitives is that the final grasp pose can be calculated analytically. Further, to cover the wide range of the objects, the dimensions and orientations of the shape primitives should be estimated at runtime. The detailed procedure for approximating the objects with shape primitives is detailed below.

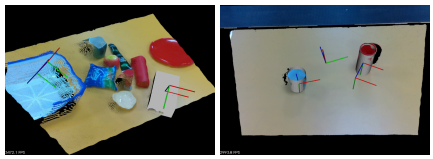
Let R denote the target object's point cloud, and the table's surface is approximated by a plane T. The R is projected onto plane T, and the projected cloud is represented by P, which will be further processed as explained below:

1) Cuboidal primitive: To approximate the objects with cuboidal primitive, we first estimate a minimum area rectangle corresponding to the projected cloud P. In general, principal component analysis (PCA) is used for obtaining minimum area rectangle by computing principal-axis (min and max variance) from a set of data points. However, the resulted axis does not guarantee a minimum area rectangle as shown in Fig. 4d. This encourages us to develop a



(a) (b) (c)

Fig. 5: Primitive cuboidal shape fitting.(a) lines/rectangle, (b) single object model, and (c) multiple object model.



(a) (b)

Fig. 6: Grasp Pose. (a) Cuboidal, (b) Cylindrical.

technique to compute a pair of orthogonal vectors, such that the rectangle formed by the pair should have a minimum area.

For estimating the pair of orthogonal vectors, we first compute the centroid C of P. Then, we randomly select a point $p \in P$ and calculate a unit vector $\hat{p}$ from C to p. Further, we compute a unit vector $\hat{n}$ orthogonal to $\hat{p}$. It must be noted that both the $\hat{p}$ and $\hat{n}$ lies in the same plane T (Fig. 5a). Next, we check whether a point (query) $q \in P$ is closer to any of the $\hat{p}$ and $\hat{n}$. To examine, we generate a unit vector $\hat{q}$ from C to q and compute the dot products $\hat{q}.\hat{p}$ and $\hat{q}.\hat{n}$. If any of the absolute of the dot products exceed a threshold $th(0.99)$, the point q is marked closer to the respective unit vector ($\hat{p}$ or $\hat{n}$). In this way, the points in P can be divided into three sets S_1, S_2, S_3, where the sets S_1 and S_2 consists of points closer to the $\hat{p}$ and $\hat{n}$ respectively and the set S_3 consists of the remaining points.

Now the points in S_1 and S_2 are parameterized by line segments l_1 and l_2. The pair l_1, l_2 represents the one rectangle candidate with area $|l_1| \times |l_2|$. The whole procedure of finding orthogonal pairs is repeated for the remaining points in S_3 till 99% of the points in P are represented by the line segments.

Among all the rectangle candidates, we select the rectangle which has a minimum area (Fig. 5a). To compute the height, we calculate the point-wise distance between the points $\in R$ and their corresponding projected points $\in P$. The largest among the distances represents the height of the cuboidal primitive. An example of a cuboidal primitive fitting is shown in Fig. 5b. Further, the dimensions and the orientation of the cuboidal primitive are computed online. Hence the proposed method can approximate any objects with cuboidal primitive given the object cloud. The result of the method for various objects is shown in Fig. 5c.

2) Cylindrical primitive: In cylindrical primitive fitting, we estimate three parameters, i.e., radius, height, and the axis of the cylinder. In order to compute radius and height, we calculate the centroid C of P. Next, we compute the distance of each point $p \in P$ from C. The largest distance is taken as the radius of the cylindrical model. The height of the cylindrical primitive is computed as the maximum distance between a point $\in R$ and its projection $\in P$. Fig. 6b shows an example of the cylindrical model fitting. The steps to estimate the axis remains the same as the cuboidal primitive fitting.

E. Grasp pose estimation

As mentioned earlier, the grasp pose can be calculated analytically once the object is approximated by shape primitive. For the cuboidal primitive, the object is grasped from the center of the cuboidal's visible surface, which has a maximum surface area. The orientation of the grasp pose is selected so that the gripper always makes a fixed angle (30^o) with the surface. This is done in order to prevent the failure of the motion planning algorithms. Fig. 6a shows the grasp pose for the cuboidal model fitted on the target object.

Similarly, for the cylindrical primitive, the object is grasped from the lateral surface, such that the contact point or grasp point, center point of the cylinder primitive, and the camera origin, when projected onto the table surface, lie on the same line. We observe that choosing the grasp pose by satisfying the above criteria always leads to a valid motion plan. Fig. 6b shows the grasp pose for the cylindrical model fitted on the target objects.

IV. Algorithmic flow

The proposed system consists of two phases, i.e., the demonstration and execution phases, and the transition between two phases is done by user-generated signal. With the start signal, the system enters into the demonstration phase.

A. Demonstration phase

In this phase, the system starts parsing the demonstration from the live video feed. During the demonstration, the system continuously monitors the video feed for estimating the f_s and f_e. Each frame is passed through the `SegCNN`, which predicts a pixel-wise segmentation mask. From the mask, the bounding rectangle is estimated for each object.

At each frame, we keep on checking the intersection between the rectangles corresponding to the hand (R_h) and the objects (R_o). If an intersection between the R_h and any of the R_o is found, we mark that frame as the f_s. The object corresponding to the intersecting rectangle R_o is the target object. Once the f_s has been detected, we further keep on monitoring the intersection between the R_h and all of the R_o. If the intersection with all of the R_o is null, we mark the corresponding image frame as the f_e.

Now the frames between f_s and f_e, which is called a clip, will be used for action classification. Since the number of frames in the clip can be arbitrary, therefore, we represent the clip by a fixed number of frames (16), selected uniformly between the f_s and f_e. The fixed-length clip is fed to the action classification network, and based on the output following information will be saved:

Table II. Performance of SegCNN

Augmentation						mIoU
colour	scale	mirror	blur	rotate	synthetic scenes	
✗	✗	✗	✗	✗	✗	23.7
✓						46.8
✓	✓					48.3
✓	✓	✓				52.3
✓	✓	✓	✓			57.9
✓	✓	✓	✓	✓		60.4
✓	✓	✓	✓	✓	✓	85.9

Pushing task: For this, besides the action label and target object ID, the final location, which is the 3D centroid point of the target object at f_e will be saved.

Stacking task: In this, with the action label and the target object ID, the stack object ID needs to be saved. Stack Object ID represents the object whose rectangle has a maximum intersection with the target object's rectangle at f_e.

Rotating task: For rotating task, we store the type of action primitive performed.

B. Execution phase

Upon finishing, the demonstrator issues an end of demonstration signal. All the objects are now placed back onto the workspace with different configurations compared to the demonstration phase. With the user's signal, the proposed system enters into the execution phase. In this phase, the proposed system gets the live feed of the current scene, and the motion of the robotic system is controlled by the stored action sequences along with the additional information, which is explained below:

Pushing task: The target object will be localized in the current frame when the retrieve action is a push action. The corresponding program executable will generate the robotic commands to push the object to the desired final location.

Stacking task: In this case, the target object and the stack object are approximated with cuboidal primitives in the current frame, and the corresponding grasp pose and place pose is estimated. The appropriate program executable will allow the robot to complete the stacking task.

Rotating task: When the retrieved action is a rotating action, the system approximates the objects (`bottle` and `cup`) with cylindrical primitive in the current frame. Using the 6D pose of the `bottle` and `cup`, required robotic commands are generated to execute the desired operation.

V. Experiments

In this section, we conduct the performance analysis of the proposed system. The robotic system is tested on various tasks ranging from a single object pushing to a complicated task of sorting the multiple objects. The object set used for the experiment is the collection of general household objects: `milk-powder packet`, `coffee-powder packet`, `apple`, `sauce-tube`, `salt-dispenser`, `cup`, `bottle`, `basket`, `plate`, and `bowl`. Further, we also report the performance analysis of the scene-parsing and action classification modules as any misclassification in these modules will severely impact the system performance.

Table III. Action classification performance

	Pushing	Stacking	Rotating
Acc.(%)	100	100	100

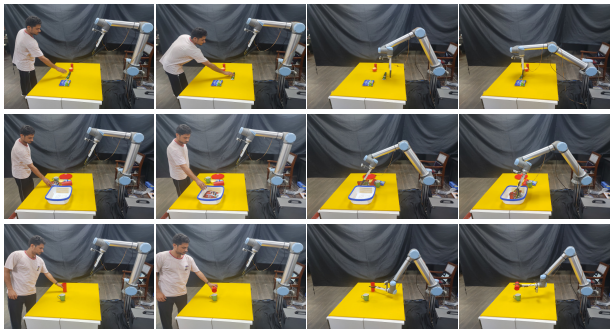


Fig. 7: (a) Pushing, (b) stacking, and (c) rotating experiments.

A. Hardware

The overall robotic platform is shown in Fig. 1 which consists of a 6-DoF, industry-grade robotic manipulator UR10 from universal robots, and a host desktop PC with 1×GPU RTX2060. We use Intel real-sense D435i as our image acquisition system and follow an eye-in-hand approach [22]. A two-finger gripper is used for grasping. The gripper has been tinkered in our lab by utilizing 3D printing and a few low-cost components available in the market.

B. Scene parsing

The scene parsing module is responsible for correctly identifying the objects in the scene. The module essentially performs semantic segmentation of input images, and in order to evaluate semantic segmentation approaches, a mean-intersection-over-union (mIoU) score is preferred. Therefore, we report the mIoU score as a performance metric for the `SegCNN`. Further, our dataset consists of roughly 30 images per class, i.e., 10 objects and 1 hand. We train on randomly selected 20 images per object and keep the 10 images for testing. Since the number of images is very small, we employ the data multiplication strategy [22]. In order to evaluate, we report the mIoU score for the test dataset while studying the effect of comprehensive data augmentation [22]. Table II shows the performance analysis of the `SegCNN` module.

C. Action classification

The manipulation action performed by the robot is directly governed by the prediction of the action classification module. In order to train the module, we collect 30 clips for each action. Randomly selected 25 sequences are used for training, while the remaining 5 are used for testing purposes. Table III depicts the performance of the action classification module. The score reported is simply the system's accuracy, which is defined as the ratio of the number of correctly classified clips to the total number of clips. Due to the small dataset and less complexity of the classification task, our system reports 100% accuracy on the test dataset.

Table IV. Single Action Single Step

Task	Object-ID										Avg.
	Milk-powder	Coffee-powder	Apple	Sauce-tube	Salt-dispenser	Cup	Bottle	Basket	Plate	Bowl	
Pushing	17	17	20	12	11	10	14	18	20	20	16
Stacking	13	12	20	2	5	12	17	-	-	14	12
Rotating	-	-	-	-	-	-	-	-	-	-	20

D. System evaluation

1) Single Action Single Step: In this case, the demonstrator is constrained to choose one of the three actions and performs the action only once. The experimental setup for the task of pushing, stacking, and rotating is shown in Fig. 7a, Fig. 7b, Fig. 7c respectively, and the evaluation criteria for the experiment are explained below.

Pushing task: The pushing task is considered a success if the robotic system pushes the correct object within the range of $0.1m$ of the target location otherwise not. The pushing task has been performed for 20 trails for each object. The performance is summarized in Table IV with the final tab shows the average performance for all objects.

Stacking task: The task is marked as a success *iff* the robotic system places the correct object over the correct stacking object. In our experiment, we have two stacking objects, naming a `basket` and a `plate`, and the choice of the stacking object, however, depends on the demonstrator and is not fixed. For this task, 20 trials has been performed for each object, except `basket` and `plate`, and Table IV reports the performance of the system.

Rotating task: The rotating task is marked as a success *iff* the robot successfully grasps the bottle, tilts it near the cup, and places it back to its initial position. Table IV represents the performance of the system for the rotating task. For this task, individual evaluation for each object does not make sense; hence, only the final tab is valid for this experiment.

2) Single Action Multiple Step: In this case, the demonstrator is constrained to choose any one of the actions, and it is allowed to repeat the chosen action on different objects. In this experiment, we evaluate the performance of the system for pushing tasks and stacking tasks, as the rotating tasks can not be used for multiple objects.

To evaluate the system performance, we define three variables: α represents the number of actions performed by the demonstrator, β depicts the number of actions (along with arguments for program executables) predicted correctly, and γ is a total number of successful actions executed by the robotic system. The fraction $\frac{\beta}{\alpha}$ denotes the accuracy of the system in the demonstration phase, while the fraction $\frac{\gamma}{\alpha}$ denotes the accuracy in the execution phase. The evaluation criteria for this task is explained below.

Pushing task: We randomly select 4 or 5 objects and place them on the table as shown in Fig. 7a. The demonstrator then pushes the objects one by one until the end-of-demonstration signal is issued. During the demonstration phase, α and β are calculated manually, while the γ is calculated in the execution phase. The task is repeated for 10 rounds with different object configurations. Table V shows the performance of the system in this task.

Table V. Single Action Multi Step

Task	Trials	α	β	γ
Pushing	10	44	44	36
Stacking	10	46	42	25

Stacking task: In this case, the stacking objects are placed at the two opposite ends of the table, and the object pile is placed at the center of the table (Fig. 7b). The object pile consists of 4 or 5 objects selected randomly. In the demonstration phase, the demonstrator grasps an object and places it on one of the stack object, and corresponding α and β values are calculated. During the execution phase, γ value is calculated and the system performance is reported in Table V.

3) Multiple Action Multiple Step: In this case, the demonstrator is unrestricted, and can take any action in any order. The experimental setup is shown in Fig. 1, which has a stacking object (`basket` or `plate`), pushing area, and the object pile (4-8 objects). Further, the demonstrator is asked to sort the objects as per his choice. For each round, during the demonstration phase, the score α and β is calculated and in the execution phase, the γ is calculated. Further, the experiment is repeated for 20 rounds. Besides the α, β, and γ, we also count the number of rounds in which the proposed system has successfully sorted all objects as per the demonstration. Table VI shows the performance of our system for this task. From the experiment, we observed that the proposed system has a success rate of 91.8% in predicting the user actions, 76.1% of success rate in replicating the user demonstrated actions, and the overall system has a success rate of 80% in the task of user-specific object sorting experiment. The link of the experiment is `https://youtu.be/w1iARSb_3P0`.

Table VI. Multi Action Multi Step

Trials	α	β	γ	Success
20	134	123 (91.8%)	102 (76.1%)	16 (80.0%)

VI. Conclusion

We have proposed a programming by imitation-based robotic manipulation system, which can be programmed for complex tasks like object sorting from a single human demonstration. The proposed system decomposes the human demonstrated task into primary tasks like pushing, stacking, and rotating using the segmentation and action classification networks. In order to replicate the demonstrated task, the robotic system is equipped with elementary task-specific program executables whose sequence and arguments are extracted from the demonstration. Further to facilitate the grasping, we propose a dynamic model fitting module for approximating the objects with cuboidal or cylindrical primitives, whose parameters are calculated at runtime. We have

observed that PCA failed to estimate the optimal cuboidal primitive for symmetric objects, whereas the proposed module successfully handled such cases. The proposed system has been tested for various tasks ranging from single object pushing to the complicated task of multiple-object sorting. For the object sorting task, the proposed system had shown the accuracy of 91.8% in predicting the demonstrated actions, 76.1% in action replication, and the overall accuracy of 80.0%. In addition, the component-wise performance of the proposed system is also reported for demonstrating the efficacy and deployability in industrial and household scenarios.

REFERENCES

[1] S. V. Pharswan, M. Vohra, A. Kumar, and L. Behera, "Domain-independent unsupervised detection of grasp regions to grasp novel objects," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 640–645, IEEE, 2019.

[2] M. Vohra., R. Prakash., and L. Behera., "Edge and corner detection in unorganized point clouds for robotic pick and place applications," in *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics - ICINCO,*, pp. 245–253, INSTICC, SciTePress, 2021.

[3] A. Kumar, M. Vohra, R. Prakash, and L. Behera, "Towards deep learning assisted autonomous uavs for manipulation tasks in gps-denied environments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1613–1620, IEEE, 2021.

[4] R. Prakash, M. Vohra, and L. Behera, "Learning optimal parameterized policy for high level strategies in a game setting," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–6, IEEE, 2019.

[5] T. Mettler, M. Sprenger, and R. Winter, "Service robots in hospitals: new perspectives on niche evolution and technology affordances," *European Journal of Information Systems*, vol. 26, no. 5, pp. 451–468, 2017.

[6] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[7] J. Van Den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *2010 IEEE International Conference on Robotics and Automation*, pp. 2074–2081, IEEE, 2010.

[8] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[9] M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt, "One-shot learning and generation of dexterous grasps for novel objects," *The International Journal of Robotics Research*, vol. 35, no. 8, pp. 959–976, 2016.

[10] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1458–1471, 2015.

[11] J. Kim, N. Cauli, P. Vicente, B. Damas, F. Cavallo, and J. Santos-Victor, ""icub, clean the table!" a robot learning from demonstration approach using deep neural networks," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 3–9, IEEE, 2018.

[12] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Learning generalizable robot skills from demonstrations in cluttered environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4655–4660, IEEE, 2018.

[13] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 827–834, IEEE, 2016.

[14] K. French, S. Wu, T. Pan, Z. Zhou, and O. C. Jenkins, "Learning behavior trees from demonstration," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7791–7797, IEEE, 2019.

[15] Y. Wang, Y. Jiao, R. Xiong, H. Yu, J. Zhang, and Y. Liu, "Masd: A multimodal assembly skill decoding system for robot programming by demonstration," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1722–1734, 2018.

[16] J. Huang, D. Fox, and M. Cakmak, "Synthesizing robot manipulation programs from a single observed human demonstration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[17] A. Bonardi, S. James, and A. J. Davison, "Learning one-shot imitation from humans without humans," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3533–3539, 2020.

[18] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[19] Y. Huang and Y. Sun, "Learning to pour," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7005–7010, IEEE, 2017.

[20] M. Vohra, R. Prakash, and L. Behera, "Real-time grasp pose estimation for novel objects in densely cluttered environment," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–6, 2019.

[21] W. Wan, K. Harada, and F. Kanehiro, "Planning grasps with suction cups and parallel grippers using superimposed segmentation of object meshes," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 166–184, 2020.

[22] A. Kumar and L. Behera, "Semi supervised deep quick instance detection and segmentation," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8325–8331, IEEE, 2019.