Comparing Kinematics-Based and Learning-Based Approaches to Robotic Arm Tasking – Using Pouring as an Example

Tzu-Chieh Chen
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
Email: jessica990805@gmail.com

Chung-Ta King
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
Email: king@cs.nthu.edu.tw

***Abstract*—The vast advances of machine learning in recent years have encouraged researchers to try learning-based end-to-end neural models for performing robotics operations. On the other hand, traditional approaches that leverage known knowledge as rules also have their merits. In this paper, we focus on robotic arm tasking and compare the learning-based end-to-end approach with a kinematics-based approach in terms of their capabilities in trajectory planning, using pouring as an example. In kinematics-based approach, object detection is obtained from a deep neural network, and arm trajectory is calculated with traditional Inverse Kinematics (IK). In the learning-based end-to-end approach, a single neural network is developed that takes RGBD images as input and outputs joint parameters of the robot arm to move the arm forward. We compare these two approaches with two scenarios, static and dynamic, in terms of their time usage and memory usage. Our experimental results show that the kinematics-based approach is more suitable for static scenarios as it uses less processing time and memory, while the learning-based approach is more suitable for complicated and dynamic scenarios.**

Keywords-machine learning; neural network; Inverse Kinematics; robot tasking; trajectory planning.

I. INTRODUCTION

The vast advances of machine learning in recent years have spurred its widespread adoption across almost all research fields. In robotics, many researchers have tried learning-based end-to-end neural models for performing robot operations. Unlike traditional approaches to robot systems that require extensive programming and human knowledge, learning-based approaches use techniques such as human demonstration and reinforcement learning to train a policy for the robot to follow to accomplish prescribed tasks. Such policies are often realized with end-to-end neural models that take raw sensory inputs and generate control outputs directly.

For complex problems, the end-to-end approach can develop well performed models without deep knowledge of the problems [1]. In addition, it uses a single neural network to replace the many functional modules in traditional robot systems, enhancing the systems by using a single optimization criterion.

On the other hand, the models developed by the end-to-end approach may be difficult to improve or modify. Any structural change, e.g., changing the input dimension, often requires re-training, which may lead to a completely different model [1]. It is also difficult to tell why the model does not work well. This is quite different from the traditional approaches, in which one can check and identify which function modules may cause an error or inefficiency. It is interesting to compare the two approaches for a deeper understanding of their respective merits.

In this paper, we compare these two approaches using trajectory planning of a pouring task as an example. Pouring is a common task used in robotics research. Prior works approach trajectory planning of robot arms from a geometric perspective and kinematics. They normally involve object detection, trajectory planning and object manipulation. Object detection determines where the target object is, including its coordinates and orientation in the space. Recent works mostly detect objects with RGB or RGBD cameras. Object detection then becomes a visual recognition problem, which can be solved very well with deep neural networks [9].

Given the location of the target object, trajectory planning then determines a path for the robot arm to reach the object. Traditionally, the problem is solved by *Inverse Kinematics* (IK), which determines the joint parameters to move the arm to the given location [7]. Once the end effector of the arm reaches the location, it can then manipulate the target object. On the other hand, the learning-based approach trains a neural model to determine the joint parameters [2]. For example, Staffan et al. used Receptive Field Cooccurrence Histograms (RFCH) [3][4] for object recognition and pose estimation. When the human moves the target object, the magnetic trackers placed on the human hand help to recognize the grasp type [2]. They designed and evaluated an automatic grasp generation and planning system, which facilitates grasping of new objects based on the shape similarity with the objects that have already been learned [2].

Sulabh et al. focused on detecting a 'good grasp' from RGBD images. They introduced a robotic grasp detection system for parallel plate grippers to detect good robotic grasps for the robot to grasp [5]. Xinchen et al. focused on parallel jaw gripper's grasping learning in simulation. The system can reconstruct the 3D shape of the target from RGBD inputs by a shape generation network. It then predicts the grasping outcome from an outcome prediction network [6].

 ISBN: 978-1-61208-838-9

The comparison in this paper focuses mainly on robotic arm tasking by comparing the learning-based end-to-end approach with a kinematics-based approach in terms of their capabilities in trajectory planning, using pouring as an example. For the kinematics-based approach, we use a deep neural network (YOLO) [9] for object detection and IK for trajectory planning. For the end-to-end approach, we train a deep neural network using RGBD images as input and joint parameters as output. The training data are collected using the kinematics-based approach to ensure data consistency and fairness in comparison.

We compare these two approaches in performing a pouring task, which is to grab a red cup and pour the contents into a blue cup. The evaluation consists of two scenarios. (1) Static scenario: the cups are at the same locations throughout the task. (2) Dynamic scenario: the red cup will be moved during the task. We evaluate their time usage, memory usage and actions under different scenarios. In the static scenario, both approaches can complete all the tasks. However, in the dynamic scenario, the kinematics-based approach fails to finish some tasks while the end-to-end learning-based approach can complete all of them.

The contributions of our work are as follows: (1) We developed kinematics-based and learning-based end-to-end approaches to pouring task by a robotic arm. (2) To train the end-to-end model, we developed a data collection system based on the kinematics-based method. (3) We evaluate and compare these two approaches, analyzing their performance under static and dynamic scenarios. As far as we know, there is no other work providing such a comparison between these two approaches to robot tasking.

The rest of the paper is organized as follows. Sec. II presents the two approaches for comparison. Sec. III shows the experiments and discusses the results and Sec. IV concludes the paper with possible future extensions.

II. APPROACHES FOR COMPARISON

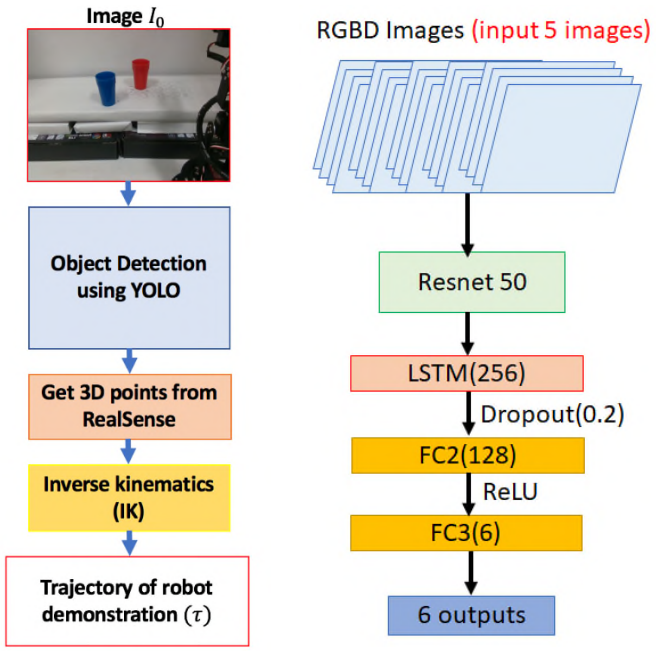
To compare the kinematics-based approach and learning-based end-to-end approach, we place two cups in the view. The goal is to pour the contents of the red cup into the blue cup. We first introduce our kinematics-based approach, discuss our methodology for training the end-to-end model, and then propose a deep neural network architecture for learning the pouring task.

A. Kinematics-Based Approach

Our kinematics-based model, shown in Figure 1(a), consists of an object detection module and a trajectory planning module determined by IK. The object detection module was based on YOLO [9] and output the 2D coordinates of the center of the cup. The whole pouring task is divided into a number of steps, 64 in our experiments. In each step, the camera takes one image and the model then decides how to move the arm accordingly. The trajectory of the robot in performing the pouring task can be represented as $\tau = (A_0, A_1, .. A_t)$, where $A_0, A_1, .. A_t$ are the joint angles of the robot at each time step.

Let I_0 be the first image taken by the camera. It is used as the input of the object detection module. The object detection module outputs $c = (c_x, c_y)$, which is the 2D coordinates of the target's center in image I_0. With a mapping between the 3D world-space coordinates and the RGB image, output c can be used to get the 3D coordinate of the center of the target cup, $T = (T_x, T_y, T_z)$, from an RGBD camera with the origin of the coordinates being the robot arm.

The trajectory planning module then uses IK to calculate the joint parameters at each time step t to construct a path to reach T. The output of the trajectory planning module $\tau = (A_0, A_1, .. A_t)$ is then sent to the robot arm in order to control the robot to finish the task.



(a) Kinematics-based (b) Learning-based

Figure 1. Appraches for comparison.

B. Learning-Based End-to-End Approach

In the learning-based approach, the pouring task is viewed as consisting of a series of states and actions. The robot arm is controlled by a neural network model with an architecture as shown in Figure 1(b). Let I_t denote the RGBD image seen by the camera at time step t. Our model will take $S_t = [I_{t-4}, I_{t-3}, I_{t-2}, I_{t-1}, I_t]$ as the input state of the neural network, which corresponds to the current image plus four previous images seen by the camera before this time step t.

Our end-to-end model takes the sequence of images at time t as the input and uses a deep residual network (ResNet-50) to extract features from the input images. The last fully connected layers of ResNet-50 is replaced by Long Short-Term Memory (LSTM) [12] with an extra dropout layer and two fully connected layers. The first fully connected layer takes Rectified Linear Unit (ReLU) as the activation function. LSTM is used to deal with the time series data for sequencing the actions, e.g., deciding when to close the gripper. The last fully connected layer is the output layer of the end-to-end model. It predicts the action

 ISBN: 978-1-61208-838-9

$A_t = [a_1, a_2, a_3, a_4, a_5, a_6]$ of the robot arm, where $a_1, ..a_6$ correspond to the settings of the joint motors of the robot arm. We use Adam [13] as an optimizer and mean square error as our loss function.

III. EXPERIMENTS

A. Environments

The experiments were conducted on a 6DOF robot arm with an Intel RealSense Depth Camera, D415. The camera was set behind the robot arm to the left with a high angle of around 30 degrees (see Figure 2(a)). The task was to pour the contents of the red cup into the blue cup. Figure 2(b) shows the view of this task from the perspective of the camera.

We used MATLAB R2016B for programming IK and Python deep learning library Keras for training. Our experiments were performed on a NVIDIA GeForce GTX 1070 GPU with AMD Ryzen 5 2600 Six-Core 3.4GHz Processor.

(a) Setup (b) View from robot's camera

Figure 2. Experimental environment.

B. Data Collection

The network of the learning-based approach was trained on a dataset that was collected from the data collection system based on the kinematics-based method. The data collection system recorded the states and actions during each demonstration by controlling the robot arm using the kinematics-based approach. In this way, trajectory data were consistent in comparing both approaches. The dataset is composed of 40 pouring demonstrations. Each demonstration contains 64 RGBD images (steps) and 64 corresponding actions. In total, there are 2560 RGBD images and 2560 actions of the six motors of the robot arm. Figure 3 shows a part of our dataset with the states in sequence with the corresponding actions.

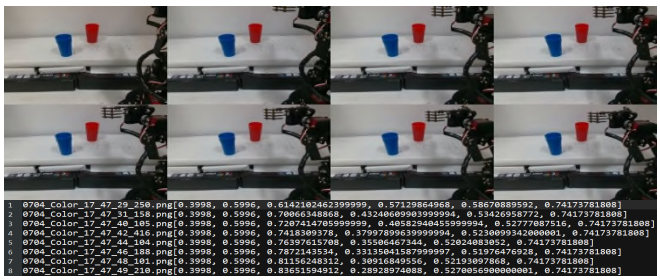


Figure 3. Sample states and actions in our dataset.

C. Evaluations

The goal of our experimental evaluation is to compare the kinematics-based approach with the learning-based end-to-end approach. We let both perform the same pouring task. The experiments can be divided into two parts:

- Static scenario: Cups were fixed at the same locations until the task was completed.
- Dynamic scenario: The red cup may be moved dynamically while the task was in progress.

Static Scenario

We placed the red cup in 30 different positions in the area, which the robot arm can reach. The positions spread evenly across the workspace to ensure that the entire area could be tested. Figure 4 shows the 30 positions. Our experiments show that both the learning-based end-to-end approach and the kinematics-based approach can complete all the tasks successfully.

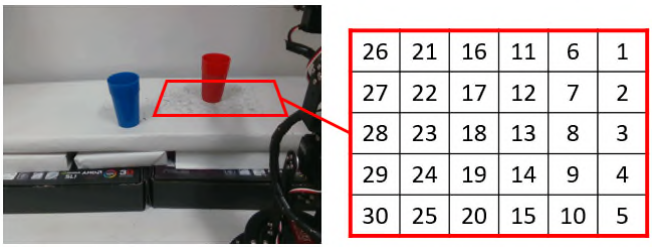


Figure 4. The 30 positions to place the read cup.

We show the computation time and the memory usage of the kinematics-based and the learning-based end-to-end approaches in Table 1. The computation time is the CPU time to execute the model, not counting communication. For the kinematics-based approach, the computation time includes the time spent on object detection and trajectory planning. For the learning-based approach, the computation time is the time to run through the neural model. During the experiments, the number of moving steps of the kinematics-based approach was always 64. On the other hand, although the learning-based approach was trained with the training data that completed the pouring task in a fixed 64 steps, the number of steps taken by the learning-based approach in inference was fewer than 64 steps, with an average of 48. This is because the learning-based approach could identify states similar to those trained and took corresponding actions, thereby skipping steps.

TABLE I. EXPERIMENTAL RESULTS OF STATIC SCENARIO

	Ave. CPU time	*Ave. # steps*	*Ave. time*	*Memory usage*	*Size*
Kinematics	7.83 s	64	119.4 s	2081 MB	236 MB
Learning	10.27 s	48	84.1 s	2574 MB	278 MB

Both approaches used most memory space on model loading. The memory usage of the learning-based approach is higher, because it must process a series of images instead of just one in the kinematics-based approach. For

computation time, the learning-based approach spends only 10.27 seconds on model prediction, meaning that our model can output an action in 0.21 seconds. Both approaches spent a lot of time on serial communication. However, since the learning-based approach may skip steps, the computation time can be shortened.

The total computation time of the kinematics-based method remains almost the same, because its step count is fixed and the calculation time of IK does not change a lot. The total time of the learning-based approach depends on the number of steps in moving the robot arm. If the model completes the task with fewer steps, the total time will be less.

We can see from Figure 5 that there is an uptrend of moving steps in the learning-based approach. The red dots show the positions in which more steps are needed to finish the task. The reason is that when the distance between the robot arm and the target is far apart, it may be possible to choose similar states that will finish the task faster by skipping some steps. However, when the robot arm is close to the target, it could only do it step by step.

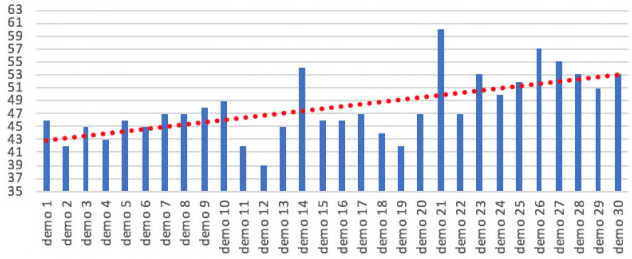


Figure 5. Uptrend of the moving steps of the end-to-end model.

Dynamic Scenario

In the dynamic scenario, the red cup was moved while the robot arm was reaching the cup. To allow the kinematics-based approach to handle moving objects properly, we used the following procedure:

1. **While** the distance between the robot arm and the target object d > a threshold h **do**
2. Detect the target object
3. **If** the target object is moved since last detection and the moving distance m > 1.5mm **then**
4. Re-plan the trajectory
5. Divide the trajectory into n points (n is based on d)
6. Move the robot arm to the next point in the trajectory
7. Grab the target object

We tested both the kinematics-based approach and the learning-based end-to-end approach under 10 different cases. The tests were split into four parts according to the way the target red cup is moved:

- Right-to-left:
  The target red cup was placed at the upper right, middle right and lower right, respectively, in the workspace of the robot arm. We then moved the cup to the left side of the workspace, as shown in Figure 6. It was found that both approaches could complete all the right-to-left tasks. The most important factor for the kinematics-based approach to success was the target cup's final position -- the whole cup could be seen by the camera and not be blocked by the robot arm.
- Left-to-right: The target red cup was placed at the upper left, middle left and lower left, respectively, in the workspace. Then, we moved the cup to the right side of the workspace. Our experiments showed that the kinematics-based approach failed in upper-left and middle-left positions, due to a failure in object detection. This is because the moving robot arm blocked the view of the camera, and, consequently, the target object could not be detected. It follows that the trajectory planning module mistook the blue cup for the target. The lower-left task was successful because the final position of the red cup was in front of others and the view of the cup was not obstructed by the robot arm.
- Top-to-bottom: We placed the red cup at the top of the workspace and then moved the cup to the bottom side of the workspace. Both the kinematics-based approach and the learning-based approach can complete the tasks. It seems that if the object detection module could recognize the target correctly, the kinematics-based approach can complete the task successfully.
- Bottom-to-top: We placed the red cup at the bottom of the workspace and moved the cup to the top side of the workspace. The experimental results were the same as in the top-to-bottom case.

From the experiments, we can see that the learning-based approach succeeded in all the tasks tested, while the kinematics-based approach failed in some cases. For the kinematics-based approach to succeed, it is critical that the target object be visible and identifiable so that its location can be determined. If the location cannot be determined or is detected incorrectly, IK will calculate a wrong path and the robot arm cannot reach the target object.

Furthermore, the kinematics-based method consists of several independent modules, which are optimized separately under different criteria. Even if we add rules to let it handle dynamic scenarios in which objects may be moved, there are always unexpected situations. Figure 7 shows a conflict of the rules. If the distance between the robot arm and the target object is less than the threshold h, the trajectory planning module will not re-plan the path. This may happen when the robot arm is very close to the target object, but it obstructs the view of the camera. The model will mistake the blue cup as the target. Choosing a suitable threshold value is also a difficult problem.

By observing the trajectories generated by the kinematics-based approach, we further find that it could only handle the change of the position of the target cup early in the process. This is perhaps because the joints of the robot arm require sufficient lead time to adapt to new positions. From the above observations and discussions, we find that the kinematics-based approach has limitations in handling complex and dynamic tasks.

 ISBN: 978-1-61208-838-9

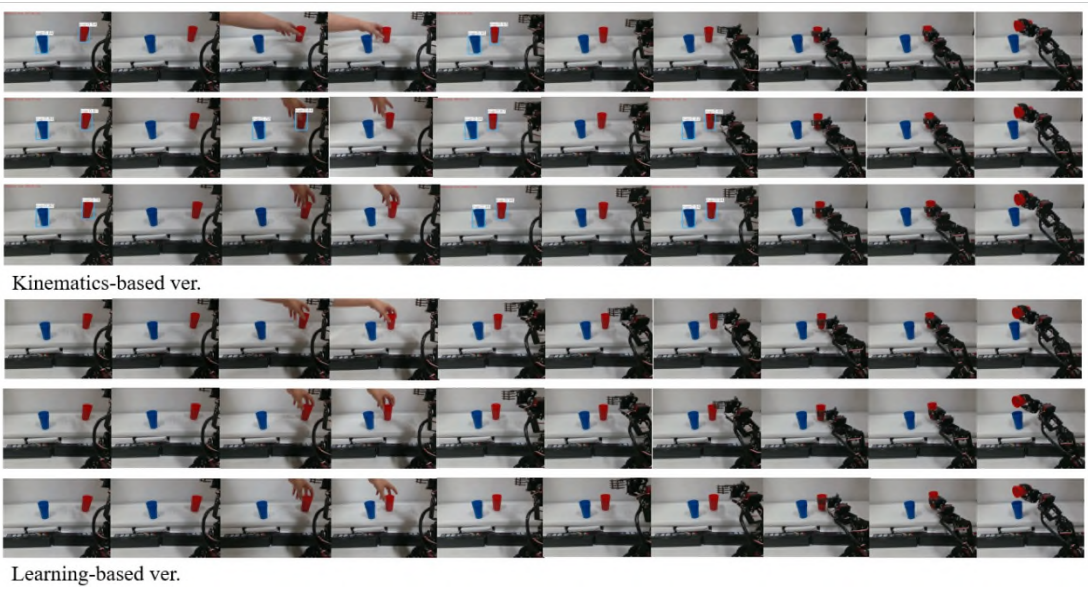


Figure 6. Right-to-left task.

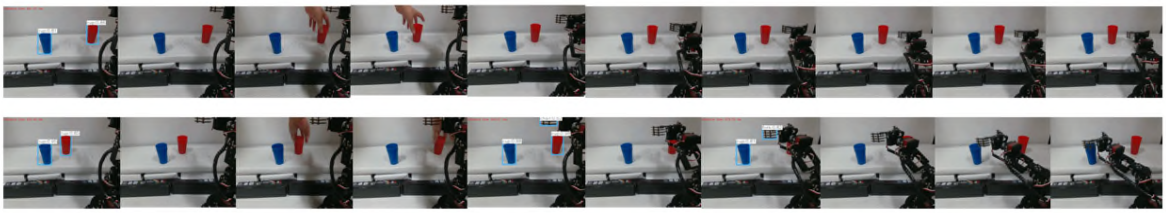


Figure 7. A conflict of the rules in the kinematics-based approach.

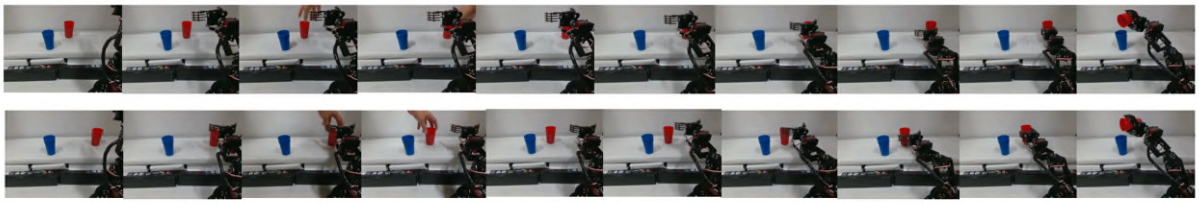


Figure 8. The learning-based approach stilll accomplishes the task.

On the other hand, the learning-based approach succeeds in cases when the target object is obstructed. This is because the learning-based end-to-end approach takes the entire view as the state to decide the action to perform. When it encounters an unseen state, it will try to find a similar known state and act according to that state. Therefore, even when the target is moved and obstructed, the model can still find a proper action to perform.

Furthermore, from Figure 8, we can see that the learning-based approach can still finish the task even if we move the target cup when the robotic arm almost grabs it. The end-to-end model is less affected by the obstructions caused by the moving robotic arm as it knows the end goal of the task. From all our experiments, we conclude that the learning-based end-to-end approach is more suitable for handling complicated and dynamic scenes.

In Table 2, we show the time usage and the memory usage of the kinematics-based and the learning-based end-to-end approaches. The biggest difference between static scenario and dynamic scenario is the computation time. The kinematics-based method spends additional time for object detection and re-planning the trajectory when the target object is moved. On the other hand, the learning-based end-to-end approach can handle the dynamic scenario with the same trained neural model and thus incur similar amount of computation time as in the static scenario. By observing the number of moving steps of both approaches, the learning-based approach completes the task more efficiently in the dynamic scenario too.

 ISBN: 978-1-61208-838-9

TABLE II. EXPERIMENTAL RESULTS OF DYNAMIC SCENARIO

	Avg. CPU time	*Avg. # of steps*	*Avg. total time*	*Memory usage*	*Size*
Kinematics	21.53 s	66	137.2 s	2279 MB	236 MB
Learning	9.33 s	47	81.7 s	2572 MB	278 MB

IV. CONCLUSIONS AND FUTURE WORKS

In this work, we compare the kinematics-based approach and the learning-based end-to-end approach to robot tasking, using pouring as an example. The kinematics-based approach is composed of object detection module and trajectory planning module, in which the trajectory is calculated by IK. The end-to-end learning network was trained by the dataset that was collected by the data collection system based on the kinematics-based approach.

The two approaches are compared in static scenario and dynamic scenario by evaluating their time usage and memory usage, and observing how they complete the tasks. Our experimental results demonstrate that the kinematics-based approach is suitable for static scenario, because it requires less computation time and memory. The learning-based approach is more suitable for complicated and dynamic scenarios, because it can perform properly for unseen and dynamic states.

In the future, we plan to improve the kinematics-based approach by using different object detection methods, e.g., grasp detection model, to handle more dynamic scenarios. Furthermore, we found that it was too troublesome to collect test data, since we need to keep changing the position of the target object manually. It is necessary to develop an automated data collection system for collecting the testing data easier. Additionally, we would like to extend both the kinematics-based approach and the learning-based approach to more complicated scenarios, e.g., blue cup in different positions, complex scenes and background, and obstacles. It is interesting to study how the two approaches handle these complex states.

ACKNOWLEDGMENT

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 109-2218-E-007-023 and by Information and Communications Research Laboratories of Industrial Technology Research Institute, Taiwan.

REFERENCES

[1] "End-to-end learning, the (almost) every purpose ML method" [Online]. Available: https://towardsdatascience.com/e2e-the-every-purpose-ml-method-5d4f20dafee4 [retrieved: March, 2021].

[2] S. Ekvall and D. Kragic, "Learning and evaluation of the approach vector for automatic grasp generation and planning," in *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 4715-4720.

[3] S. Ekvall and D. Kragic, "Receptive field cooccurrence histograms for object detection," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 84-89.

[4] S. Ekvall, D.Kragic, and F. Hoffmann, "Object recognition and pose estimation using color cooccurrence histograms and geometric modeling," *Image and Vision Computing*, vol. 23, pp. 943-955, 2005.

[5] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 769-776.

[6] X. C. Yan et al., "Learning 6-DOF grasping interaction via deep geometry-aware 3D representations," in *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3766-3773.

[7] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modeling, Planning, and Control*, Springer-Verlag, 2009.

[8] "Program inverse kinematics algorithms with MATLAB" [Online]. Available: https://ww2.mathworks.cn/discovery/inverse-kinematics.html [retrieved: March, 2021].

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788.

[10] "An overview of ResNet and its variants" [Online]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035 [retrieved: March, 2021].

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735-1780, 1997.

[13] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of 2015 International Conference on Learning Representations*, (ICLR), 2015.

 ISBN: 978-1-61208-838-9