2020 IEEE/ASME International Conference on
Advanced Intelligent Mechatronics (AIM), Boston, USA
(Virtual Conference), July 6-9, 2020

DeepClaw: A Robotic Hardware Benchmarking Platform for Learning Object Manipulation*

Fang Wan[1], Haokun Wang[2], Xiaobo Liu[3], Linhan Yang[4] and Chaoyang Song[5]

***Abstract*— We present DeepClaw as a reconfigurable benchmark of robotic hardware and task hierarchy for robot learning. The DeepClaw benchmark aims at a mechatronics perspective of the robot learning problem, which features a minimum design of robot cell that can be easily reconfigured to host robot hardware from various vendors, including manipulators, grippers, cameras, desks, and objects, aiming at a streamlined collection of physical manipulation data and evaluation of the learned skills for hardware benchmarking. We provide a detailed design of the robot cell with readily available parts to build the experiment environment that can host a wide range of robotic hardware commonly adopted for robot learning. We propose a hierarchical pipeline of software integration, including localization, recognition, grasp planning, and motion planning, to streamline learning-based robot control, data collection, and experiment validation towards shareability and reproducibility. We present benchmarking results of the DeepClaw system for a baseline Tic-Tac-Toe task, a bin-clearing task, and a jigsaw puzzle task using three sets of standard robotic hardware. Our results show that tasks defined in DeepClaw can be easily reproduced on three robot cells. Under the same task setup, the differences in robotic hardware used will present a non-negligible impact on the performance metrics of robot learning. All design layouts and codes are hosted on Github for open access (https://github.com/bionicdl-sustech/DeepClaw).**

***Index Terms*— Benchmark, Manipulation, Robot Learning**

I. Introduction

Robot Learning is an active field of research that adopts a data-driven approach by applying machine learning algorithms to advanced robots for autonomous control towards intelligent and physical interactions. In order to build up the database for model training, researchers usually adopt passive perception such as RGB or depth cameras to formulate an image-based robot control scheme as experiment setup [1], [2], [3], [4]. Such an image-based method provides a continuous collection and validation of the interaction data in manipulation tasks. However, active perception may be necessary to capture the localized interaction at the physical interface between the gripper and objects, as well as the functional performance of the multi-joint manipulators.

Unlike robots such as autonomous cars or drones with significant interactions with the road or the air, robotic manipulation involves a wide range of objects with vastly differentiated physical properties and functional behaviors, making it a challenging task to build a learning system for robotic manipulation [5]. With the availability of a wide selection of robotic hardware with differentiated engineering specifications and functional performances, how to make an informed selection and integration of robot hardware becomes the first question to ask when transferring learned models to reality.

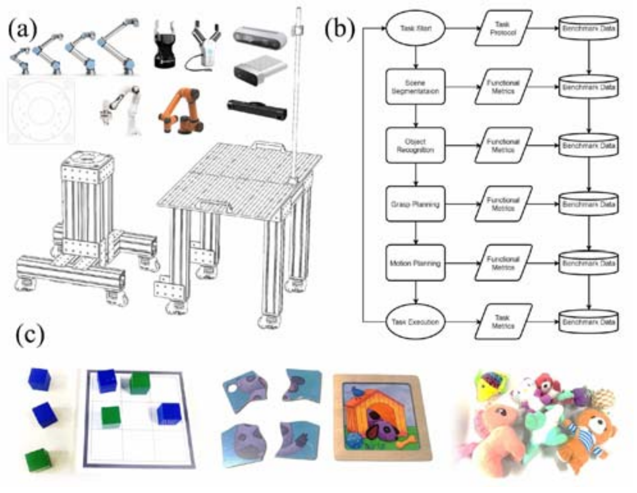

Fig. 1: The design of DeepClaw platform towards a reproducible and shareable benchmark for learning robotic manipulation, including (a) a standardized robot station design, (b) a modular pipeline for functional integration, and (c) exemplified task variations.

In this paper, we propose the DeepClaw in Fig. 1 as a minimum robot cell design to benchmark the robotic hardware for learning object manipulation, which involves three levels of embodiment.

- Modular frame design: aims at a standard mounting frame for all hardware components involved in a robot learning task, which can be easily sourced from global suppliers and local manufacturing shops with open access engineering drawings and assembly instructions for a baseline experiment environment.

*This work was supported by the National Natural Science Foundation of China (No. 51905252) and the Science and Technology Innovation Committee of Shenzhen Municipality, China (No. KQJSCX 20170728162555608). FW was employed by the company AncoraSpring, Inc. The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

[1]Fang Wan is with SUSTech Institute of Robotics, Southern University of Science and Technology and AncoraSpring, Inc., Shenzhen, Guangdong 518055, China.

[2]Haokun Wang, Xiaobo Liu, Linhan Yang are with the Department of Mechanical and Energy Engineering, Southern University of Science,Shenzhen Guangdong 518055, China.

[2]Chaoyang Song is the corresponding author and is with the Department of Mechanical and Energy Engineering, Southern University of Science,Shenzhen Guangdong 518055, China. songcy@sustech.edu.cn

- Functional pipeline hierarchy: enables a unified software architecture of system integration, algorithm pipeline implementation, data communication.
- Game tasks & objects: demonstrates three benchmarking tasks, namely tic-tac-toe, claw machine and jigsaw puzzle, with spatial-temporal task metrics and specific functional metrics for hardware benchmarking.

In the rest of the paper, we first review the related work on benchmarks of robot learning in Section II. In section III, we present a detailed description of the DeepClaw. Section IV presents the experiment design and results of three game-based benchmarks using DeepClaw with three sets of robotic hardware. Final remarks are enclosed in section V, which ends this paper.

II. Related Work

While it remains a question of using machine learning to aid the design of a manipulation system for robot learning [6], there is a wide range of robotic hardware to build up the initial understanding. Collaborative robots, as a lightweight version of the industrial robots, provide ease of access to robotic manipulation with force-limited safety, cost-effectiveness, and user-friendly interfaces [7]. Although recent research by Google shows the potential of using large-scale picking data to train models of robot learning [1], it is unrealistic to custom build and deploy such a large number of robotic hardware for continuous research. The recent proposal of the RoboNet demonstrates an improved effort using data collected from various robot cells with similar setups and experiment environment in an open-access format [8], which focus mainly on visual foresight tasks. Other efforts such as the ROBEL [9] and REPLAB [10] aim at a low-cost infrastructure for reinforcement learning using robotic hardware with limited motor function. The RLBench is another effort to build a simulated environment for reinforcement learning using a single robot of Franka Emika through 100 task variations [4].

The broad selection of robotic hardware and a lack of consideration of their mechanical differences motivate the need for a benchmark of robotic devices for learning object manipulation. A recent survey shows that there are more than 30 collaborative robots with different functional performance and engineering specifications [11]. To further reduce the dimensionality of the learning problem, parallel grippers [12], [2], as a mature, robust, and cost-effective design of end-effector, are widely adopted instead of using robotic hand with multi-fingers [13], [5]. The object variations and task specifications add further complexity in setting up the robot cell for learning [4]. While many machine learning research much relies on simulated data to test algorithm as a first attempt [14], researchers also become increasingly aware of the gap between simulation and reality when transferring the learned model for hardware validations [15], [16].

After reviewing the efforts of making shareable benchmarks for manipulation from literature in Table I, we summarize five aspects of consideration to develop an in-depth understanding of the physical interaction in object manipulation.

A. Manipulation Task

Manipulation task focuses on a goal-oriented benchmarking of the integration between advanced robotics and machine learning. For arm-type manipulators, the general task design spans from basic human skills such as picking, placing, pushing [8], etc., to a wide range of advanced tasks in daily life activities [17]. A hierarchical decomposition of the manipulation task is usually a preferred choice to establish reusable skills for learning system design [18], which requires a systematic task protocol design.

B. Target Object

The target object is a minimum collection of the physical features in manipulation, including shape, weight, size, color, material, etc., which must be perceived by the robot system with accuracy. It represents an object-centric generalization in robot learning, such as the YCB object dataset [19]. Simulated object generation also provides an alternative source of data for offline training [2], [3].

C. Operating Environment

The operating environment requires reliable and robust support to maintain a quality data collection protocol and a safe operation of the robot with human, if necessary. It usually involves a flat desk [3] or an object tray [1] as the manipulation arena. The robot can be mounted on the desk or a dedicated pedestal with a fixed spatial relationship with the desk [4]. In the meanwhile, one should also consider the mounting frame and angle of the cameras for calibration and view range, which can be on the hand, to the base, or on the robot [20].

D. Robotic Hardware

Robotic hardware usually involves a manipulator as the arm, a vision sensor as the eye, and an end-effector as the hand for an image-based robot control system [8]. As shown in Table I, while there is a wide range of hardware available for integration, researchers gradually converge to a selection guideline based on collaborative robots, RGB-D cameras, and 2-finger parallel grippers or a single suction cup to build the robot cell [21].

E. Algorithm Pipeline

The algorithm pipeline represents the control logic of the learning system, which relies on a quality collection of the interaction data of the robotic hardware against the target object in the operating environment for a specific manipulation task [18]. The main input data is collected through passive perception such as 2D, 3D, or depth cameras towards an image-based robot control for large-scale data collection. A potential drawback is the limited use of robotic hardware in manipulation tasks and learning system design.

III. DeepClaw Benchmark Design

A. Design Overview

DeepClaw is a benchmarking system for robot manipulation designed to be reconfigurable, shareable, and reproducible on real robots and the environment. As shown in Fig. 1, DeepClaw consists of four components:

1) A standardized robot cell design.
2) A set of unified driver interfaces that serves as a connection layer between the benchmarking algorithms and the hardware, including robot arms, robot hands, visual sensors, tactile sensors.
3) A collection of baseline algorithms for segmentation, recognition, grasp planning, and motion planning.
4) A pipeline for task definition and functional integration.

The standard hardware design and functional pipeline of DeepClaw aim to accelerate the benchmarking process. Tasks defined in DeepClaw will be easily reproduced on different robot hardware setups by changing the configuration files while keeping the same task pipeline. The software is mainly written in Python for ease of use. However, as some hardware suppliers only provide APIs in C++, some of the DeepClaw drivers use a mix of Python and C++ as well. The key features of the design of the DeepClaw are described in the following four aspects. More comprehensive documentation and codes are provided at https://bionicdl-sustech.github.io/DeepClaw/.

B. System Integration

The mechanical design of the DeepClaw station involves aluminum extrusions commonly available from global suppliers such as Misumi, with part numbers shown on the right of Fig. 2. For the convenience of assembly, we also introduced a few aluminum plates with through holes, which can be easily machined from local shops. A robot mounting plate is also included with threaded holes matching the base mount of the robots used in this paper, including Universal Robots and Franka Emika, which is also suitable for other manipulators from suppliers such as AUBO. We included detailed design files of the DeepClaw on the project Github page so that others can make informed modifications to suit their robot system.

We have implemented quite a few drivers for hardware shown in Fig 2 including robot arm drivers for UR5, UR10e, and Franka; robot hand drivers for HandE from Robotiq, RG6 from Onrobot, gripper from Franka, and standard suction cup; visual sensor drivers for Realsense 435/435i, Azure Kinect, PhoXi M from Photoneo; force-torque sensor driver for OptoForce from Onrobot. Based on these hardware devices, we have built three robot stations, as shown in Fig. 2. Robot cell one consists of an assembled station with a tabletop, a Franka robot arm with a suction cup, and an Realsense 435i. Robot cell two consists of an assembled station with a tabletop, a UR5, an RG6 gripper, and a Realsense 435. Robot cell three consists of a manufactured optical table, a Realsense 435, an Azure Kinect, and a Robotiq HandE. All the cameras are mounted about one meter above the table top and facing downward. The hardware setup is defined in the software part of DeepClaw through configuration files. The same task should be easily reproduced on different hardware setups if the configuration file is adequately defined.

We also design a 6cm$\times$6cm 3D printed calibration board mounting on the tool flange without affecting the end-effectors, as shown in Fig. 2. A 3x3 checkerboard pattern is printed and stick on the calibration board. The hand-eye calibration is implemented by registering the coordinates of the checkerboard for the robot arm and camera. Since the transformation from the tool center point (TCP) to the checkerboard is known, the hand-eye matrix can be easily obtained from the singular value decomposition (SVD) method [22]. Scripts that automatically run the hand-eye calibration and to verify the accuracy of the calibration are provided in the calibration module of the DeepClaw.

C. Tasks vs. Objects

In DeepClaw, a manipulation task involves three hierarchical concepts: task, sub-task, functionality module. A task protocol should clearly define the main purpose of the task, the target objects, the robot, and hardware setup, procedures to fulfill the task and execution constraints [19]. Each task may consist of several repetitive sub-tasks. A pipeline of functional modules can accomplish each sub-task.

The most similarity between game and dexterous manipulations enable reproducible experiments in various environment. All of the game manipulation tasks can be classified from two different perspectives: spatial reasoning and temporal reasoning. Compared with human daily dexterous manipulations, game manipulations have a noticeable distinction in spatial and temporal dimensions. "Jigsaw Puzzle," for example, requires a meaningful pattern at finally by placing certain pieces in settled spatial position and orientation using robot cell. We can summarize that "Jigsaw Puzzle" focuses on spatial reasoning rather than temporal reasoning sine chronological operations to finish the puzzle are needless during the whole placing process. "Tic-tac-toe" is the contrary that emphasizes moving chess chronologically rather than its spatial position and orientation (distinguish the type of pieces rather than each piece individual). Claw machine is another popular game that involves picking and placing to clear the toy tray. We hypothesize that both robot cells and intelligent algorithms lead to performance differences when executing game manipulation tasks.

D. Functional Pipeline

In DeepClaw, a sub-task is defined by a pipeline of modules, including segmentation, recognition, grasp planning, and motion planning, as shown in Fig 3. The pipeline takes color/depth images, force feedback, hardware limitation, and environment information as input and gives actions to the manipulation system and pushes data and results to data monitor.

Segmentation and recognition involve analyzing information gained from the perception system. *Segmentation* is the

TABLE I: Review of shareable manipulation benchmarks with physical and simulated robots (shaded) sorted by year.

	Work name	Year	Robot	Gripper	Camera	Task
1	RoboNet [8]	2019	Sawyer, Franka,Kuka, WidowX, Baxter, Fetch, GoogleRobot	Weiss Robotics WSG-50, Robotiq, WidowX, Baxter, Franka, Kuka	Monocular RGB	Grasping, Pushing, Placing
2	RoboTurk [20]	2019	Sawyer	Sawyer	Kinect2 and Webcam	Object search, Tower creation, Laundry
3	REPLAB [10]	2019	WidowX	WidowX	Creative Blasterx Senz3D SR 300 RGB-Depth	Grasping
4	QT-Opt [21]	2018	Kuka LBR IIWA	Self-build two-finger gripper	Monocular RGB	Grasping
5	ACRV [22]	2017	Baxter	Suction cup	Kinect2, Realsense SR300	Shelf Picking
6	YCB [19]	2015	HERBïŒ PR2	HERBïŒ Model T, Model T42 PR2	Kinect2, Webcam, PR2	Pitcher-Mug, Gripper Assessment, Table Setting, Block Pick and Place, Peg Insertion Learning, Box and Blocks
7	MOC [23]	2015	Baxter	Baxter	Baxter	Grasping
8	RLBench [4]	2019	Franka	Franka	Monocular RGB, Stereo camera	100 manipulation tasks
9	Meta-World [24]	2019	Sawyer	Sawyer	Not mentioned	50 manipulation tasks
10	Simitate [14]	2019	TIAGo Sawyer	TIAGo Sawyer	Kinect2, 12 "OptiTrack PRIME 13"	Human daily activities
11	ROBOTURK [3]	2018	Sawyer	Sawyer	Monocular RGB	Lifting, Picking, Assembly
12	SURREAL [25]	2018	Sawyer	Sawyer	Monocular RGB	Block Lifting, Block Stacking, Bimanual Peg-in-hole, Bimanual Lifting, Bin Picking, Nut-and-peg assembly
13	OpenGRASP [26]	2011	ARMAR-III, PA-10	ARMAR-III, Barrett hand Shadow, Schunk SAH and SDH	Not mentioned	Grasp planing, Motion planning, Clean the table

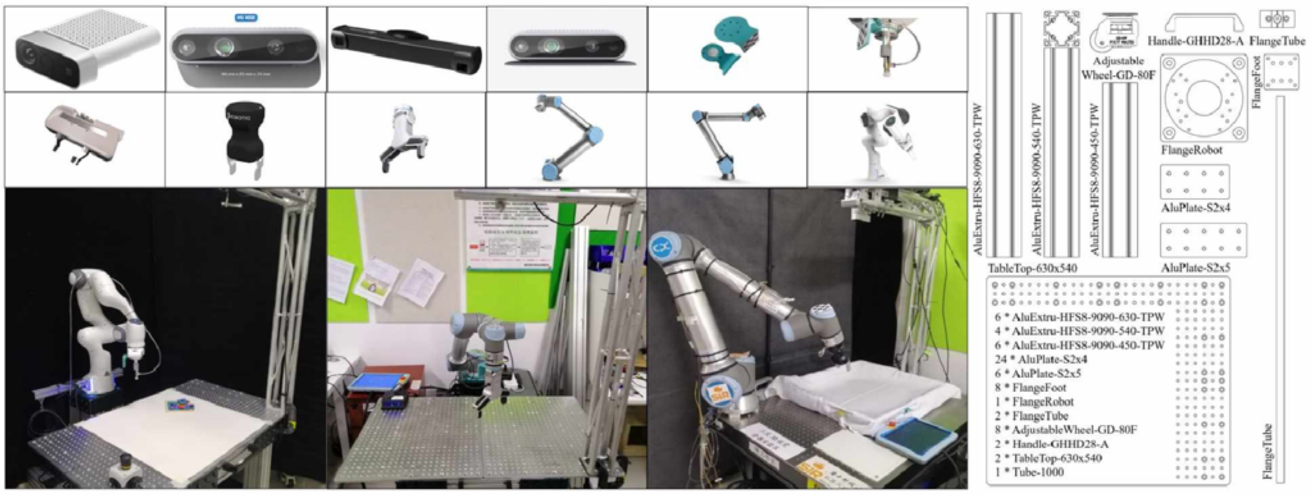

Fig. 2: Design overview of the robotic hardware in DeepClaw, including the supported cameras, end-effectors, and robotic manipulators (top), the three robot cells used in this papers (bottom), and the all the mechanical parts for a full assembly (right).

process that robot cell collecting environment information and representing spatial information of the target objects by using perception algorithms. The output of the segmentation module can be pixel-wise masks or bounding boxes. DeepClaw includes basic segmentation based on contour and edge detection in Opencv [23].

Recognition is the process of extracting features of the target object beyond location information. In this step, the robot cell infers the category of the target object by applying specific methods, such as support vector machine(SVM) and convolutional neural network [24]. Some of the end-to-end neural networks infer the location and category of the target object at the same time [25], [26].

Grasp planning aims to find the optimal pose for the robot arm and end-effect to approach the target objects, which is highly dependent on both the end-effector and the objects. Recent years, research interests have shifted from analytic method [27], [28] to data-driven method [2], [12], [29]. DeepClaw has implemented an end-to-end grasp planning model based on fully convolutional AlexNet, which was trained on 5,000 random grasps with labels.

Motion planning utilizes information above, such as grasping pose, force sensor data, constrain of the robot system, and limitation of working space, to obtain collision-free trajectories. Currently, waypoint-based motion planning is used through our tasks. For UR5 and UR10e, we utilize the *movej* command implemented in UR's controller to plan and execute a path between waypoints. For Franka, we utilize a fourth-order motion generator in the joint space provided by the libfranka software.

Researchers can assess different robot cells in the same manipulation task by standardizing sub-tasks in the above DeepClaw pipeline. Moreover, a new algorithm for each functionality of the pipeline can be evaluated by implementing the algorithm as a module and integrate it into the task pipeline. Besides, DeepClaw pipeline provides the possibility to analyze the variance between end-to-end neural network models and traditional step-by-step algorithms.

E. Result Reporting

One of the key problems in benchmarking is consistent evaluation metric and automatic data collection process. The DeepClaw provides a monitor module to record the experiment data automatically and keep a snapshot of the necessary codes and configurations to be able to reproduce and reference the experiments in the future. Evaluation metrics can be divide into function-wise and task-wise. The former metric only evaluates a single functionality of the pipeline. Examples are Intersection-over-union (IoU) for segmentation [26], recall and precision for recognition [30], force-closure for grasp planning [31], length and clearance for motion planning [32]. These metrics require a good knowledge of the truth hence require extensive manual annotations. The latter is defined to evaluate the completeness and efficiency of the task as a whole. A universal and easy-to-get performance metric is the time cost of each step in the task pipeline, which is a key metric when comparing between different algorithms and robot platforms. DeepClaw has implemented the time cost metric, task-wise metrics such as success rate of grasping, and some function-wise metrics, which will be described in Section IV.

The raw data produced by the robot arm and sensors, e.g., robot poses, RGB images and depth images, is also recorded throughout the pipeline depending on the tasks requirement. The raw data and metric data are all in well defined format, hence the same analysis script can be used.

IV. Experiment Benchmarks

We conduct three benchmarking tasks as examples of applications based on DeepClaw. Each task represents a specific task of board games, assembly, and grasping, respectively, and is repeated on all the three robot cells we have built.

A. Tic-Tac-Toe: Board Games as Adversarial Interaction

In this section, we elaborate on how we implemented the Tic-Tac-Toe game as a benchmark test for adversarial interaction. Tic-Tac-Toe game is a temporal reasoning related task, which required two players moving pieces alternately. To simplify this game as a baseline, the two players use the same placing strategy, namely the Minimax algorithm with depth 3, and are both executed by the robot arm. We use green and blue cubes from Yale-CMU-Berkeley objects set [19] representing two types of pieces. At the start of the game, 3×3 checkerboards printed on an A4 paper is placed in front of the robot base, and the two types of pieces are lined on the left and right side of the chessboard as shown in Fig 4(a). The task is to pick a type of piece and place it on one of nine boxes on the checkerboard in turns until one player wins or ends with a tie.

The full task of the Tic-Tac-Toe game can be divided into two repetitive sub-tasks, as shown in Fig 4. In the picking sub-task, we locate the cubes from contour segmentation and select one the cubes with the same color randomly.

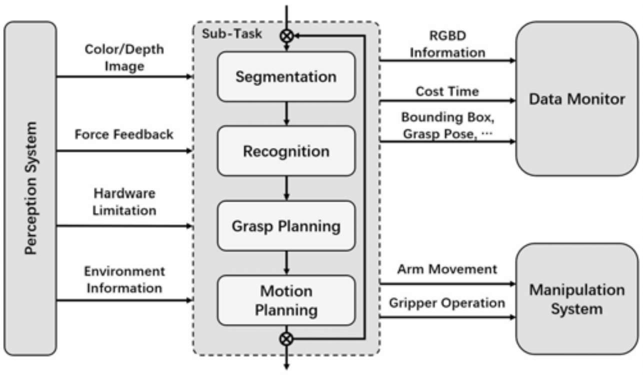

Fig. 3: Functional pipeline of the DeepClaw for structured data-driven benchmarking, consists of four steps: segmentation, recognition, grasp planning and motion planning. Sensors, which are parts of perception system, provide plentiful perceptual information to algorithms among steps. Data monitor collects valuable data during the process. Actions are given to the manipulation system.

In the placing sub-task, we use the Minimax algorithm to determine the placing location. The time to complete a pick and place sub-task t_{sub} is recorded as a performance metric. To compare the performance of the robot arm, we purposely exclude the gripper closing time from t_{sub}. We performed the bin-clearing task ten times on each robot cell, and the performance metric is averaged over ten repeated tasks.

The results show that all the repeated tasks were completed, given the simplicity of the game and the robustness of the contour segmentation. The time costs of a pick and place sub-task are 9.6 s, 17.9 s, and 19.5 s for Franka, UR5, and UR10e, respectively. Note in this benchmark, the joint speed and acceleration limits are set to 0.7 m/s and 1.6 m/s^2 for UR5 and UR10e and are set to 1 m/s and 2.5 m/s^2 for Franka. With the same speed setup, UR5 is slightly more efficient than UR10e. Franka cut down almost half of the time with 1.5 times joint speed.

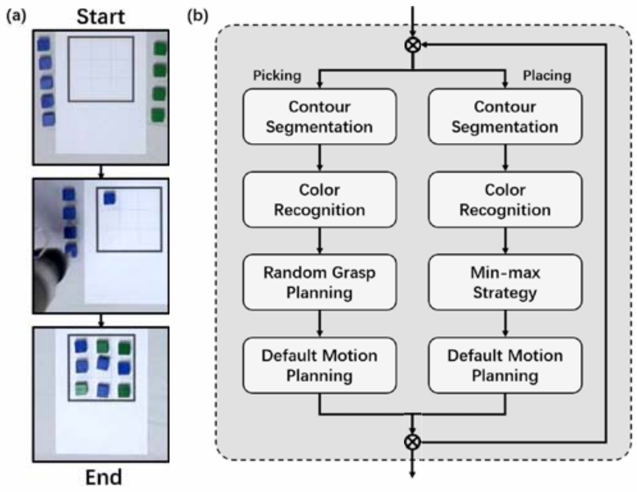

Fig. 4: (a) The first row shows initial configuration and the third row shows the state of Tic-Tac-Toe task completion. (b) The pipeline of the baseline Tic-Tac-Toe benchmark.

B. Claw Machine: End-to-End Manipulation Benchmarking

In this section, we used a claw machine scenario to illustrate a bin-clearing task. This benchmark measures the performance of a learned policy for predicting robust grasps over different robot cells. At the start of the task, a 60cm$\times$70cm white bin stuffed by eight soft toys and an empty 30cm$\times$40cm blue bin are placed side by side on the table top as shown in Fig 5. The task is to transport the toys to the blue bin one by one until clearing the white bin. We restrict the gripper to grasp vertically, allowing only rotations along the z-axis of the robot base.

Each sub-task contains a pick and place cycle. The pipeline predicts the optimal grasping pose using the fully convolutional AlexNet trained from a dataset of 5000 random grasps of soft toys. We replace the last three fully connected layers by 1×1 convolutional layers so that the model can process images with various resolutions. The CNN model takes an RGB image as input and outputs relatively dense map of success probabilities of grasps at pixel level. At each pixel with prediction, the model give success probabilities for 18 rotation angular bins, the angle with the highest success probability is selected, as shown in Fig. 5(b), where the radius of circles represents the probability, and the white lines indicate grasp orientations. The optimal grasp center (u, v) and the associated orientation θ is then obtained by finding the pixel with the highest probability in the image space. Then grasp center (u, v) is transformed to (x, y) with reference to the robot base using depth at (u, v) from the depth map, the intrinsic parameters of the camera, and hand-eye matrix. The grasp depth z is fixed slightly above the table top. For performance assessment, we report robot arm execution time t_{pick} for a single pick task, and the total picking success rate $r_{success}$, which equals to eight over the number of total grasp attempts before clearing the bin. We performed the bin-clearing task on three robot cells and ten times on each robot cell. The performance metric is average over ten repeated tasks. We also excluded the gripper closing time from t_{pick} to compare the efficiency of robot arms. The closing time is 0.80 s, 2.0 s, and 0.33 s for Franka, RG6, and HandE, respectively.

The picking success rates for Franka, UR5, and UR10e are 0.80, 0.91, and 0.83, respectively. The results demonstrate the grasp abilities variation of Franka, RG6, and HandE, among which RG6 is most capable due to its sizeable graspable area. The robot picking times t_{pick} are 8.3 s, 8.6 s and 11.2 s respectively for Franka, UR5 and UR10e. In this benchmark, the joint speed and acceleration limits are set to 0.7 m/s and 1.6 m/s^2 for all three robot arms. Franka and UR5 perform similarly while UR10e takes the longest time as in the Tic-Tac-Toe benchmark.

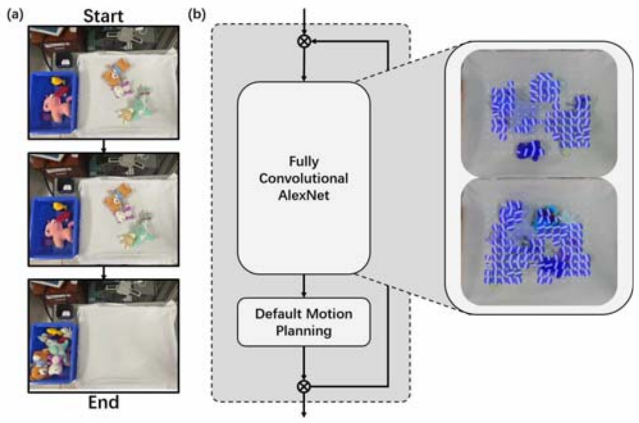

Fig. 5: (a) The first row shows initial configuration and the third row shows the state of claw machine task completion. (b) The pipeline of claw machine benchmark.

C. Jigsaw Puzzle: Tiling Game for Modular Benchmarking

The jigsaw puzzle benchmark is designed to evaluate learning models for object detection and recognition for tiling tasks. A jigsaw puzzle is a tiling game that requires the assembly of often oddly shaped interlocking and tessellating pieces. The jigsaw set used in this paper contains four thin wooden pieces with an image printed on one side and can form a 10.2cm×10.2cm picture when they are correctly assembled. We use a suction cup to complete the task on all three robot cells as the jigsaw piece is only 5 mm thick and is too challenging for grippers. At the start of the task, the four pieces are randomly placed on the table top, as shown in Fig. 6(a). The task is to detect and pick one jigsaw piece at a time and place it at the required location according to its shape and texture information, and finally assemble all the four pieces into one whole piece. We restrict the gripper to pick vertically, allowing only rotations along the z-axis of the robot base.

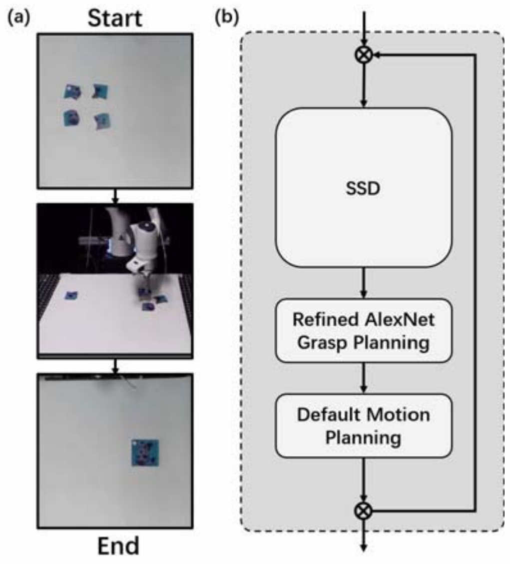

Fig. 6: (a) The first row shows initial configuration and the third row shows the state of jigsaw task completion. (b) The pipeline of the jigsaw benchmark.

The pipeline uses a fine-tuned Single-Shot Detection (SSD) model [25] for segmentation and recognition. SSD gives a bounding box of the piece without orientation and determines which one of four pieces it belongs. Also, a fine-tuned AlexNet is adopted to predict the grasping rotation angles. We collected about 80 images for the training of SSD and used data augmentation to avoid overfitting. The training data for AlexNet was automatically generated by rotating four template images. For performance assessment of the overall task, we report a 2D measurement of the area rate defined by

$$score = \frac{A_{standard}}{A_{actual}} \quad (1)$$

to evaluate the task completion, where $A_{standard}$ refers to the area of the finished jigsaw when the four fragmented pieces are perfectly aligned together, A_{actual} refers to the actual area of the minimum bounding box of the finished four jigsaw pieces. For performance assessment of functionalities in the pipeline, we report the IoU for segmentation, average precision (AP) for recognition, the success rate of picking and time cost of a pick and place sub-task as shown in Table II.

The results compare the performance of the task on the three robot cells. The IoU and AP are comparable on UR5 and UR10e and drops on Franka, possibly because the training data of the SSD model are collected on D435. The result is expected to improve on Franka if we include images from D435i as well. In this benchmark, the joint speed and acceleration limits are set to the same for the three cells. However, the execution time of Franka is significantly less than UR5 and UR10e. A plausible explanation is that 7 DoF robot arm is more agile and faster than 6 DoF robot arms for more complex manipulation tasks.

TABLE II: The jigsaw benchmark results for modularized tiling tasks.

Jigsaw Tiling			
metrics	**UR5 +D435**	**UR10e +D435**	**Franka +D435i**
IoU	0.876 ±0.0916	0.903 ±0.0157	0.794 ±0.1477
AP	0.950 ±0.1054	0.975 ±0.0791	0.875 ±0.2125
Success rate	1±0	1±0	1±0
Grasp time (s)	21.58 ±0.1854	20.81 ±1.4807	12.09 ±2.3475
score	0.828 ±0.0505	0.856 ±0.0223	0.940 ±0.2461

V. Conclusion and Future Work

This paper proposes DeepClaw, a minimum robot cell design, and software to accelerate benchmarking the robotic hardware and learning manipulation. To illustrate the use of such hardware and software design, we have integrated three robot cells involving three different robot arms, four cameras, and four end-effectors. We demonstrate three benchmarking tasks and our experiments show that tasks can be reproduced easily with different hardware by standardizing the pipeline of functionalities and the result reporting. However, the performance of the task depends on the hardware.

One of the challenges with benchmarking research is to provide long-term support for such an infrastructure so that it can be regularly updated for improvement and feedback. We are still in the process of adding drivers and support for new robotic hardware and intend to provide a benchmark for the various robotic hardware on mechanical perspective to facilitate the selection process, as well as a build-and-select interface for ease of use. In the meanwhile, the current DeepClaw still lacks a simulation environment, which will help tremendously for reinforcement learning research with robotic manipulation. As of now, we are in the process of adding support through an education-free version of PyRep, which is the V-rep repackaged with improved python support for learning tasks.

While our effort is to provide a standard design of hardware and algorithm pipeline, DeepClaw currently only features three benchmarking tasks on three different robot cells. This is a significant limitation whenever deals with hardware and can only be resolved by encouraging collaborations within the community. Future work will be focused on three directions: 1) adding modules of real-time robotic control with close-loop such as visual servoing [33], force control [34], and motion planning and 2) incorporating existing simulation effort from the community like RLBench [4] and 3) expanding the benchmarking tasks and dataset. There are several ongoing projects based on DeepClaw in our lab, including benchmarking new gripper designs for manipulation and visual–tactile sensor fusion, and we shall continuously update DeepClaw with our new results. We hope DeepClaw will contribute to and invite collaborations from the robotics and learning communities.

REFERENCES

[1] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, D. Quillen, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018. [Online]. Available: http://arxiv.org/abs/1603.02199v1

[2] 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.

[3] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay *et al.*, "Roboturk: A crowdsourcing platform for robotic skill learning through imitation," *arXiv preprint arXiv:1811.02790*, 2018.

[4] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *arXiv preprint arXiv:1909.12271*, 2019.

[5] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving Rubik's Cube with a Robot Hand," pp. 1–51, 2019. [Online]. Available: http://arxiv.org/abs/1910.07113

[6] D. V. Gealy, S. McKinley, B. Yi, P. Wu, P. R. Downey, G. Balke, A. Zhao, M. Guo, R. Thomasson, A. Sinclair, P. Cuellar, Z. McCarthy, and P. Abbeel, "Quasi-direct drive for low-cost compliant robotic manipulation," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 437–443, 2019.

[7] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3356–3363.

[8] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning," *arXiv preprint arXiv:1910.11215*, 2019.

[9] M. Ahn, H. Zhu, K. Hartikainen, H. Ponte, A. Gupta, S. Levine, and V. Kumar, "ROBEL: Robotics Benchmarks for Learning with Low-Cost Robots," 2019. [Online]. Available: http://arxiv.org/abs/1909.11639

[10] B. Yang, J. Zhang, V. Pong, S. Levine, and D. Jayaraman, "Replab: A reproducible low-cost arm benchmark platform for robotic learning," *arXiv preprint arXiv:1905.07447*, 2019.

[11] Robotiq, *Cobots Ebook: Collaborative Robots Buyer's Guide*. Robotiq, 2019.

[12] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 3406–3413.

[13] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, p. 027836491988744, 2019.

[14] R. Memmesheimer, I. Mykhalchyshyna, V. Seib, and D. Paulus, "Simitate: A hybrid imitation learning benchmark," *arXiv preprint arXiv:1905.06002*, 2019.

[15] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," *arXiv preprint arXiv:1610.04286*, 2016.

[16] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *arXiv preprint arXiv:1707.02267*, 2017.

[17] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 20–29, 2007.

[18] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis-A survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[19] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[20] F. Wan and C. Song, "Flange-based hand-eye calibration using a 3d camera with high resolution, accuracy, and frame rate," *Frontiers in Robotics and AI: Robot and Machine Vision*, 2020.

[21] L. Yang, F. Wan, H. Wang, X. Liu, Y. Liu, J. Pan, and C. Song, "Rigid-soft interactive learning for robust grasping," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1720–1727, 2020.

[22] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, April 1991.

[23] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision.* Springer, 2016, pp. 21–37.

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[27] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," 2003.

[28] F. T. Pokorny and D. Kragic, "Classical grasp quality evaluation: New algorithms and theory," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 3493–3500.

[29] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4304–4311.

[30] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

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

[32] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[33] F. Janabi-Sharifi, L. Deng, and W. J. Wilson, "Comparison of basic visual servoing methods," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 5, pp. 967–983, Oct 2011.

[34] Guanfeng Liu and Zexiang Li, "Real-time grasping-force optimization for multifingered manipulation: theory and experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 9, no. 1, pp. 65–77, March 2004.