End-to-end Reinforcement Learning of Robotic Manipulation with Robust Keypoints Representation

Tianying Wang,[†‡] En Yen Puang,[§¶] Marcus Lee,[‖] Wei Jing,[**††] and Yan Wu[§¶]
[†] Key Laboratory of Education Ministry for Modern Design Rotor-Bearing System, Xi'an Jiaotong University, Xi'an, China
[‡] School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an,China
[§] A*STAR Institute of Information Research (I^2R), Singapore.
[¶] A*STAR Institute of High Performance Computing (IHPC), Singapore.
[‖] National University of Singapore (NUS), Singapore.
[**] Alibaba DAMO Academy, Hangzhou, China.
[††] Corresponding author, email: 21wjing@gmail.com

***Abstract*—We present an end-to-end Reinforcement Learning (RL) framework for robotic manipulation tasks, using a robust and efficient keypoints representation. The proposed method learns keypoints from camera images as the state representation, through a self-supervised autoencoder architecture. The keypoints encode the geometric information, as well as the relationship of the tool and target in a compact representation to ensure efficient and robust learning. After keypoints learning, the RL step then learns the robot motion from the extracted keypoints state representation. The keypoints and RL learning processes are entirely done in the simulated environment. We demonstrate the effectiveness of the proposed method on robotic manipulation tasks including grasping and pushing, in different scenarios. We also investigate the generalization capability of the trained model. In addition to the robust keypoints representation, we further apply domain randomization and adversarial training examples to achieve zero-shot sim-to-real transfer in real-world robotic manipulation tasks.**

I. Introduction

With the recent advancement in deep neural network and computational capacities, Reinforcement Learning (RL) has demonstrated the capabilities of achieving superior performance in many applications such as atari games [1], go [2], and complex games [3]. RL has been used for various robotics tasks [4]. Among the existing applications, using RL for the robot to learn primitive manipulation tasks is particularly attractive, since it could be a promising paradigm to help the robot learn to minimize the effort of robot programming, especially in ad-hoc and high-mix-low-volume settings. Additionally, RL is capable to handle uncertainties and novel scenarios because of its sequential decision and generalization capabilities [4], [5], [6].

End-to-end RL learns the robot motion command with direct sensor input (typically vision) for robot manipulation task [7], [8], [9]. The end-to-end approaches minimize the effort of manually handcrafting features, such that the same framework could be reused for different manipulation tasks with little modification. With the rapid advancement of deep learning in recent years, the Convolutional Neural Network (CNN) is able to extract features from the image input directly. Several end-to-end robotic RL methods have been proposed with the deep CNNs to process the visual input and generate motor policy [9], [10].

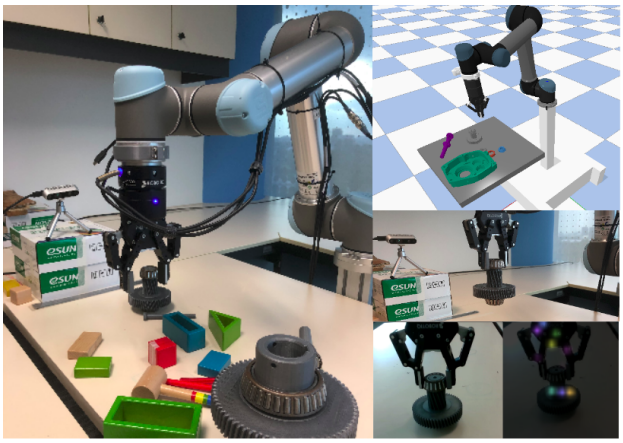


Fig. 1: End-to-end RL of robotic manipulation with keypoints representation in simulation and real-world

Though end-to-end robotic RL has been studied for robotic manipulation, there are still several issues that limit its performance in real-world applications. Firstly, efficient sensory data mapping into simulation is required. Since real-world robot data collection is expensive and ad-hoc for different task settings, collecting data of direct visual input in the simulation is often desired. Furthermore, training a robust RL control policy with direct camera input is challenging and slow to converge, especially when only simulation data is available for training. To tackle this problem, robust state representations and reliable sim-to-real transfer methods are needed.

In this work, we propose a robust keypoints representation-based RL framework for self-supervised, end-to-end robotic manipulation tasks. We use the keypoints to encode the geometric information of the input image, as well as the

spatial relationship of the tool and the target in a robust and compact manner. We demonstrate that the proposed keypoints representation is able to outperform traditional methods for visual feature extraction in robotic manipulation applications. In addition, the overall RL framework is able to generalize the tasks well and achieve good success rate and fast convergence in training as well as zero-shot sim-to-real transfer in several manipulation tasks. The main contributions of this work are as follows:

- We propose a novel end-to-end RL methods with self-supervised keypoints as the state representation in robotic applications.
- We achieve reliable zero-shot sim-to-real transfer of end-to-end RL by directly applying the learnt model to real-world robot manipulation tasks.
- We conduct extensive experimental studies with different task scenarios in both simulation and real-world experiment, as well as studies on task generalization and transfer learning, to validate the effectiveness of the proposed algorithm.

II. Related Work

A. Reinforcement Learning for Robotic Manipulation

A number of RL algorithms have been developed and applied to robotic manipulation in recent years [4], [13]. Many of these techniques are demonstrated on robot manipulation tasks using camera inputs as the sensory feedback. However, state representation and explainability still remain as open questions [14] as end-to-end RL algorithms using images as input is difficult to train. Compressing the images to a lower dimensional meaningful state space could be helpful for faster and reliable training. [15] learns feature extractors prior to RL training to achieve good results. End-to-end task learning also requires outputting low level motion commands or trajectory from sensory input [16]. Various formulations have been studied, such as end-to-end visual servoing [17], [18] and end-to-end motor policies [16], [10], [19]. RL methods with direct visual input have also been explored [20], [21], [22] for different robotic manipulation tasks. In this work, we focus on improving the end-to-end RL with a robust keypoints representation, while the proposed method could work with most actor-critic style RL approaches.

B. Representation Learning for Visual Inputs

Processing the direct image input is a crucial step towards end-to-end RL in robotic manipulation. A natural approach is to use CNN to extract the visual features for direct feeding to an end-to-end RL framework [21], [8]. Using estimated poses of the objects as state for vision based RL is also a common approach [23], [24]. However, it requires a good object detector and a pose estimator, which also separate the pipeline into two stages. One other approach learns a low dimensional latent space representation of the input image with an autoencoder [25], [26], [27]. However, these latent space features usually encode the entire physical world in the camera view which limits its generalizability. Recently, keypoint-based approaches have been used in many computer vision applications such as image generation [28], face recognition [29], tracking [30] and pose estimation [31]. Keypoints have also been used as a representation for robotic manipulation tasks. For example, [32], [33] use keypoints for manipulation planning of category tasks with supervised learning; and [34] proposes to learn a task-specific keypoints representation for manipulation tasks via self-supervised learning. To improve the representation robustness, we propose a self-supervised keypoints representation to encode geometric information and relationship of manipulator and manipulation target. The proposed keypoints representation is an improved version of [18], with improved architecture and transformation loss to address the RL framework and static camera output.

III. Methodology

The overall architecture of the proposed method is shown in Fig. 2. The proposed method includes a keypoints extraction module using a CNN-based autoencoder with additional loss functions to capture the keypoints. The RL module is a model-free, actor-critic style method [35] that maps the keypoints and robot states to the motion command and Q-value. The two-stage training is conducted entirely in simulation by using the simulation ground truth to self-supervise the training. The first stage trains an autoencoder for keypoints detection, while the second stage trains the RL module for control policy. The trained policy could be directly used in real-world experiment.

A. Learning Keypoints Representation

We use keypoints as a compact latent representation to describe the visual information of the tasks as depicted in Fig. 2, and hence to improve the robustness and efficiency of vision-based end-to-end RL for the manipulation tasks. In summary, [18] is adopted to simplify the state representation from input image to keypoints for objects in the scene of robotic manipulation task. An autoencoder is used to train the extraction of keypoints with reconstruction loss and keypoint bottle-neck. We use the ground truth of semantic segmentation and depth map as the targets for reconstruction loss. Additional losses as soft constraints are also introduced to push the keypoints away from each other, and pull them within the segmentation areas of the targets.

First, the encoder $f : \boldsymbol{x} \mapsto \boldsymbol{z}$ compresses the input image $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ to extract features $\boldsymbol{z} \in \mathbb{R}^{H' \times W' \times K}$. The keypoints function $\Phi : \boldsymbol{z} \mapsto \boldsymbol{k}$ then transforms the feature map into keypoints $\boldsymbol{k} \in \mathbb{R}^{K \times 3}$ by computing channel-wise centroid on the softmax of $\boldsymbol{z}$ for each of the K channels:

$$\boldsymbol{j}_i^* = \sum_{\boldsymbol{j} \in \Omega} \boldsymbol{j} \frac{\exp\left(\boldsymbol{z}_i\right)}{\sum_\Omega \exp\left(\boldsymbol{z}_i\right)} \tag{1}$$

$$\alpha_i = \sigma\left(\max_\Omega\left(\boldsymbol{z}_i\right)\right) \tag{2}$$

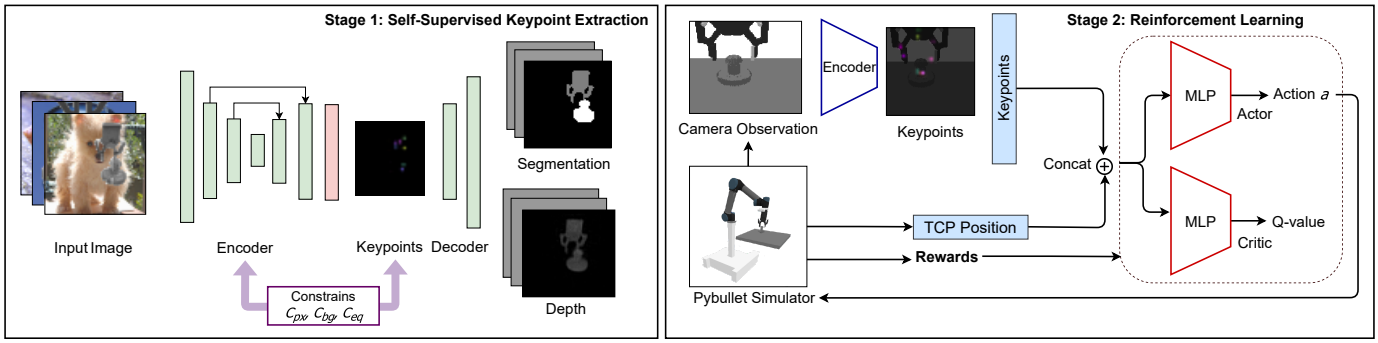


Fig. 2: (Right) Overall model architecture of the proposed reinforcement learning framework with keypoints representation. (Left) Model architecture for keypoints extraction: the encoder is a DenseNet [11] backbone with a U-Net [12] architecture to enhance multi-scale feature extraction; the reconstruction objectives are depth image and semantic segmentation, with the ground truth from the simulator. We also include additional loss to constrain the formation of keypoints.

where $\boldsymbol{j} = (j_1, j_2)$ represents the indices in the 2D axes in $\boldsymbol{z}$, and $\boldsymbol{j}^*$ is the centroid of the 2D feature map Ω. Moreover, keypoint confidence α is computed from the sigmoid $\sigma(\cdot)$ of the channel's maximum activation. The complete keypoint representation comprises $\boldsymbol{k}_i = (\boldsymbol{j}_i^*, \alpha_i)$ given $i \in [1, \ldots, K]$.

Two soft constraints are used in keypoint formation during training to improve object localization. The first constraint $\mathcal{C}_{px}$ encourages distributed keypoints formation by promoting L2-norm among centroids. The second constraint $\mathcal{C}_{bg}$ contains the keypoints within object of interest by penalizing centroid located on the background mask.

In order to further improve the robustness and performane of keypoints learning for RL tasks with static camera, we propose an additional transformation loss $\mathcal{C}_{eq}$ to improve consistency of keypoint extraction:

$$\mathcal{C}_{eq} = \left\| (\Phi * f * T)(\boldsymbol{x}) - (\Phi * T * f)(\boldsymbol{x}) \right\|_2 \tag{3}$$

where T is a geometrical transformation function that can be applied to both input image $\boldsymbol{x}$ and extracted feature map $\boldsymbol{z}$, and the * represents concatenation of the functions. This constraint requires the encoder to be consistent in object localization over geometrical transformations. It hence penalizes the differences in extracted keypoints between when transformation is applied before and after the encoder.

B. RL framework for Robotic Manipulation

The robotic manipulation tasks discussed in this paper are formulated as Markov Decision Processes (MDPs) $M =< S, A, T, R, \gamma >$, where S is a set of states, A_i is a set of actions ; T_s is the state transition; $R : S \times A \longrightarrow R$: is the rewards; $\gamma \in [0, 1]$ is the discount factor. The total episodic return is then the summation of discounted rewards: $r_{total} = \sum_{i=0}^{T_e} \gamma^i r_i$, where T_e is the total steps of an episode.

In our manipulation tasks, we use the keypoints pixel positions and robot end-effector state to represent S in MDP; the action $a \in A$ is the robotic end-effector velocity command in continuous space. We adopt an off-policy, actor-critic style RL algorithm [35] for the policy learning. As shown in Fig. 2, two parallel multilayer perceptron networks (MLPs) are used to estimate the control policy and Q-value independently.

Our framework is suitable for any off-policy, actor-critic style RL methods, since it requires estimation of Q-value and policy, as well as efficient sampling with replay buffer. For Deep Deterministic Policy Gradient (DDPG) algorithm, the expected total return is:

$$J_\pi(\theta) = \sum_{i=0}^{T_e} \mathbb{E}_{a,s}(\gamma^i r(a, s))) \tag{4}$$

The Q-value update is as follows:

$$y_t = r(s_t, a_t) + \gamma Q_\phi(s_{t+1}, \pi_\theta(s_{t+1})) \tag{5}$$

And the policy gradient for DDPG is approximated as:

$$\nabla_\theta J \approx \frac{1}{N} \sum_i \nabla_a Q_\phi(s_i, a)|_{a=\pi(s_i)} \nabla_\theta \pi_\theta(s_i) \tag{6}$$

where s_t, a_t are the state and action at time t; Q is the network that estimates Q-value; $\pi(s_{t+1})$ is the policy network to estimate action at state s_{t+1}; y_t is the target to be used to train Q-network; θ, ϕ are the parameters of policy network and Q network.

The Soft Actor Critic (SAC) [36] includes an additional entropy item to encourage exploration:

$$J_\pi(\theta) = \sum_{i=0}^{T_e} \mathbb{E}_{a,s}(\gamma^i r(a, s) + \alpha H(\pi(.|s))) \tag{7}$$

The target Q value is calculated as:

$$y_t = r + \gamma(Q(s_{t+1}, a_{t+1}) - \alpha log \pi(s_{t+1})) \tag{8}$$

Policy update could be done by taking gradient from

$$\nabla_\theta J \approx \nabla_\theta \mathbb{E}(Q_\phi(s, a(s, \xi)) - \alpha log \pi_\theta(a(s, \xi|s)) \tag{9}$$

where ξ is a fixed distribution such as multi-variant Gaussian for the stochastic policy. For simplicity, detailed formulations of DDPG and SAC are omitted.

In the training process, we use a guided reward r_g to address the sparse rewards problem in the RL training. The total reward is then denoted as $r = \alpha_1 r_g + \alpha_2 r_s$. The guided reward r_g guides the target object towards the goal position, so it gives less penalty if the target object is closer to the goal position. r_s is the regulation rewards which constrains the robotic end-effector to maneuver within the effective workspace. The total reward for robotic grasping task is:

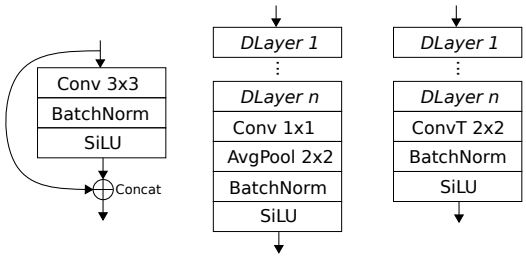
$$\begin{aligned} r(x, y, z, d_{og}) &= -\alpha_1 r_g + \alpha_2 r_s \\ &= -\alpha_1 max(0, min(d_{og}, 1)) + \\ &\quad \alpha_2(\psi(x) + \psi(y) + \psi(z)) \end{aligned} \tag{10}$$

where $\psi(x)$ is given as:

$$\psi(x) = \begin{cases} 1 & \text{for} \quad x_{min} < x < x_{max} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where d_{og} is the distance between the target object position and the goal position, we clamp the guided rewards r_g to better model the loss. x, y and z are the coordinates of the robotic end-effector. We use x_{max}, y_{max}, z_{max} and x_{min}, y_{min}, z_{min} as upper and lower bounds to softly constrain the end-effector in the effective working space.

C. Network and Training Details



(a) DLayer (b) Contracting Block (c) Expanding Block

Fig. 3: Configuration of each block in the DenseNet-based autoencoder for keypoints extraction. Each block contains 3 or 4 layers, the convolutional dilation increases accordingly in each layer.

1) Training and Implementation of Keypoints: For the keypoints extraction module, we adopt the autoencoder network architecture from [18]. The keypoints extraction module consists of a fully-convolutional U-Net [12] with DenseNet[11] block and skip-connections in the encoder. Fig 3 depicts the configuration of blocks used in the autoencoder architecture of the keypoints extraction module. The total learnable parameters for keypoint extraction network is about 1M.

The size of the input image $H \times W \times C$ and feature map $H' \times W' \times K$ is $64 \times 64 \times 3$ and 32×32 respectively. K is the number of keypoints, which is a heuristic parameter depending on the complexity of the task. The keypoints will be used with an image pixel coordinate representation ($K \times 2$ dimension) in the later RL stage. The geometrical transformations T in (3) is in-plane rotation at $90°$, $180°$ or $270°$. We randomize the camera positions and angle within a small range, to make the algorithm more robust and calibration-free. For each task, we pre-train the keypoints extraction module, using $50,000$ images generated from the simulation. We also use domain randomization [37] and adversarial examples [38] in the keypoints training to improve the performance and robustness for sim-to-real transfer. For the hyperparameters used during the training, the batch size is 64; the number of epochs is 16; Adam optimizer is used with learning rate of 0.001.

2) Training and Implementation of RL: For the reinforcement learning module, we adopt Soft Actor Critic (SAC) [36] with Hindsight Experience Replay (HER) [39] to learn the control policy. The policy and critic networks are both 64×64 MLP networks with layer normalization. For the hyperparameters used during the training, the batch size is 512; the learning rate is 0.0003; the random exploration rate is 0.2. As for HER, the number of artificial transitions to generate for each actual transition is 4, and the buffer size is 50,000. All training experiments are conducted using several dual Nvidia GTX 2080Ti GPU workstations with Intel i9-9900X CPU.

IV. Experiments and Results

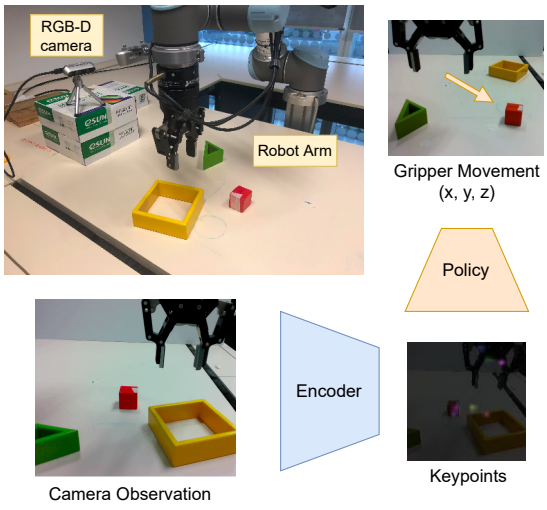


Fig. 4: Experiment setup and inference pipeline

A. Setup

We use PyBullet simulator [40] for the training and simulation, with OpenAI gym [41] interface. Our RL algorithm is implemented based on Stable Baselines RL framework [42]. The environments we train and test the policy on are *UR5Grasp* and *UR5Push* implemented with PyBullet simulator. All simulations are run 10 times to plot the mean and variance in the figures in Section IV-B2. Actual experiments are carried out on a Universal Robot UR5 with a Robotiq 2F-85 gripper and an Intel Realsense D435 camera. The camera is

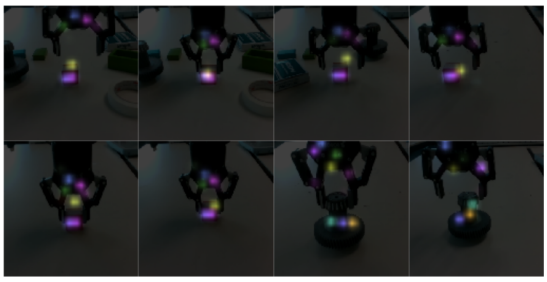


Fig. 5: Keypoints extracted from the camera observations in real-world experiment

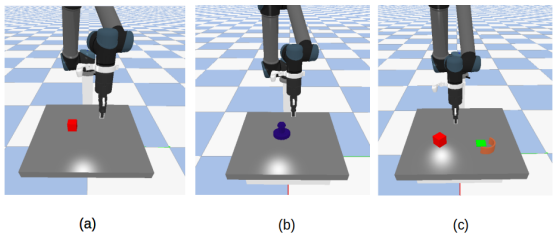


Fig. 6: Setup of simulation environment: (a) *UR5Grasp* env-1 (b) *UR5Grasp* env-2 (c) *UR5Push* env-1

placed at a fixed location at about 45° view of the workspace; camera calibration is not required for our system. The real-world experiment setup and inference pipeline is shown in Fig. 4.

B. Results

1) Keypoints Representation: As visualized in Fig. 5, the keypoints are constrained on either the target or the gripper, with the designed geometric loss. In addition, the keypoints extraction module also demonstrates robust tracking performance with real-world images, with the model trained only in simulated environment. Thus, with these characteristics, keypoints representation is able to reduce the input state dimensions effectively, and help to learn the control policy robustly.

2) Results Comparison: We validate the proposed method on three different robotic manipulation scenarios including grasping and pushing tasks. The simulation setups of the three tasks are shown in Fig 6. We compare our methods with the methods that use Variational Autoencoder (VAE) and Autoencoder (AE) as feature extraction modules. We use the same encoding network architecture (without expanding layer for keypoints) for VAE and AE, a pre-trained and fine-tuned ResNet18 [43] for AE-FT. All feature extraction modules are trained with the same simulation image data and fixed during the RL training stage.

As shown in Figure 7, the proposed RL framework with keypoints representation significantly outperforms other methods with higher success rate and better training efficiency in all three tasks. Taking grasping task in Figure 7 for example, the proposed keypoints based RL methods started to converge after 20 epochs and achieves almost 100% success rate. Moreover, keypoints-based RL method demonstrates better stability over other methods in the grasping tasks. Note that the push task is more challenging compared to grasp task, because the vision-based pushing scenario only uses the camera image to evaluate the pushing process.

3) Number of Keypoints: We evaluate the performance with different numbers $(8, 10, 12, 14)$ of keypoints, as shown in Fig. 8. We find that the algorithm with 12 keypoints achieve best results in the grasping task. Nevertheless, the performance differences are minor with different numbers of keypoints, within a reasonable range. The results indicate that the performance of the proposed method is robust to keypoint numbers.

4) Generalization and Transfer Learning: As discussed in [44], [10], [6], when deploying a trained policy to new manipulation target object, the model performance usually drops. Therefore, we also investigate the task generalization capabilities of the proposed keypoints-based RL methods. For this study, we train the policy using a cube as a primitive manipulation target, then conduct a transfer learning on a rectangle and a computer mouse, as shown in Fig. 9. Fig. 10 compares the results of success rates and average returns of the TFS (training from scratch) and TF (transfer learning from the pre-trained policy). The pre-trained policy shows good initial success rate and converges faster. The generalization capabilities indicate that we could train the RL model on primitive geometries to achieve good efficiency for transferring the policy to new target object.

C. Real-world Experiments

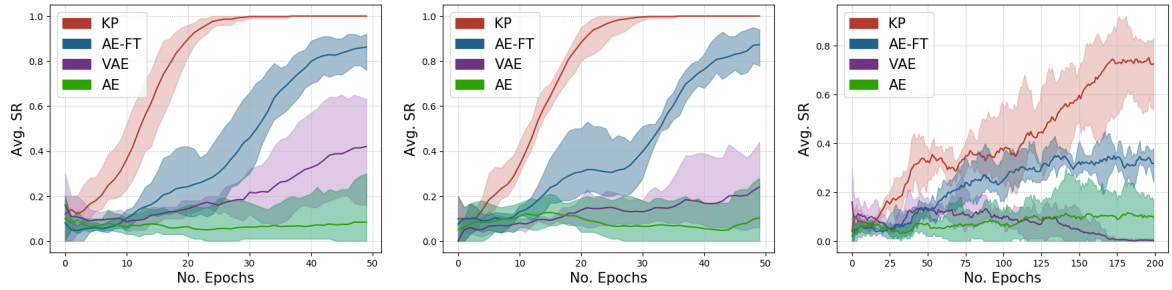
We also conduct real-world experiment directly using our policy model trained in the simulator, as shown in Fig. 4. We evaluate our methods in three scenarios, GB (grasping the block), GB-N (grasping the block in a noisy workplace) and GT (grasping the gearbox transfer component). The experiment processes are shown in Fig. 11 and Fig. 12. The success rate of three scenarios are shown in Table. I. In general, the proposed keypoints-based RL method demonstrates good success rate, as well as stability and generalization capabilities in real-world experiments. However, some failure cases are still observed, which are mainly due to real-world environment factors such as noisy workplace, camera distortion and noise, as well as slippery surface.

TABLE I: Success rate of real-world experiments.

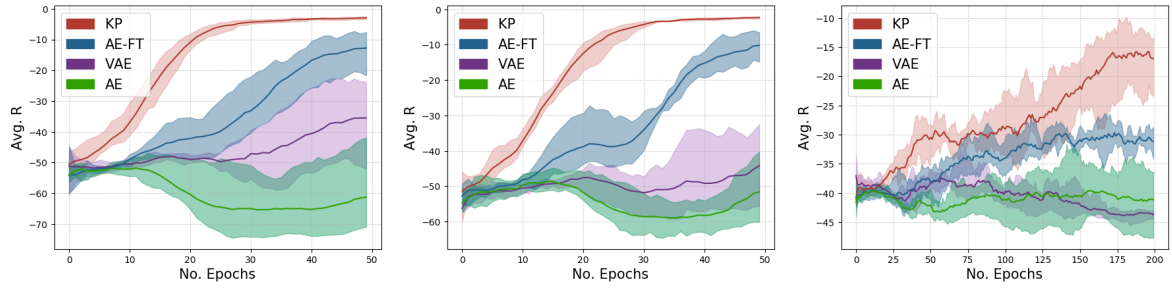
Scenario	GB	GB-N	GT
Success rate	80% (8/10)	80% (8/10)	70% (7/10)

D. Discussion

The proposed RL framework with keypoints representation demonstrates reliable manipulation behaviors in differ-

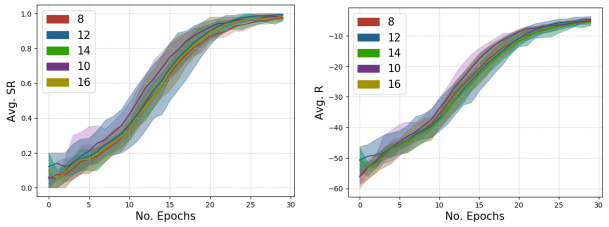


(a) Success rates of *UR5Grasp* env-1 (b) Success rates of *UR5Grasp* env-2 (c) Success rates of *UR5Push* env



(d) Average returns of *UR5Grasp* env-1 (e) Average returns of *UR5Grasp* env-2 (f) Average returns of *UR5Push* env

Fig. 7: Performance Benchmark of keypoint-base, VAE-based and AE-based methods for extracting features in the raw image input for the RL module. The first row is the success rate of the manipulation task, while the second row is the corresponding episodic return. The comparisons are based on *UR5Grasp* env-1, *UR5Grasp* env-2, *UR5Push* env. Each task has been run 10 times to plot the mean and variance.



(a) Success rates (b) Average returns

Fig. 8: Performance comparison with different numbers $(8, 10, 12, 14)$ of keypoints on grasping task

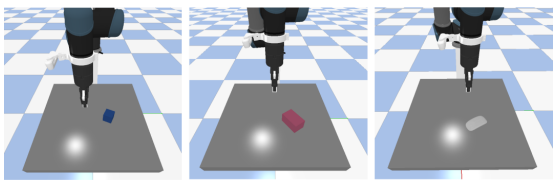
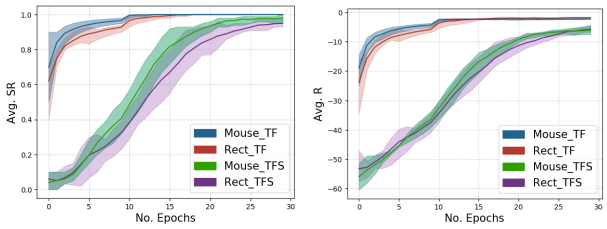


Fig. 9: Generalization and transfer learning experiment setup: (a) Cube (b) Rectangle (c) Computer mouse

ent robotic manipulation tasks, by encoding the geometric information for manipulation task efficiently. As designed in our loss function, the keypoints are forced to fall into the gripper and target object area, through the available ground



(a) Success rates (b) Average returns

Fig. 10: Generalization and transfer learning: comparison of performance on grasping rectangle on transfer learning (Rect_TF) and training from scratch (Rect_TFS), as well as grasping mouse on transfer learning (Mouse_TF) and training from scratch (Mouse_TFS)

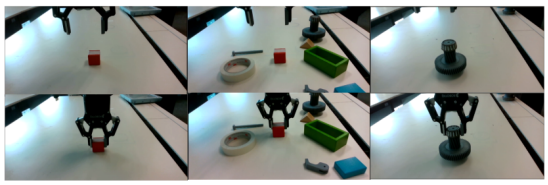


Fig. 11: Experiment process from the camera viewpoint in three scenarios: (a) GB (b) GB-N (c) GT

truth to describe their geometric features. In addition, unlike the keypoints representations in [33], [34] that mainly encode robotic task information, our keypoints representation encode geometric information and relationship of the tool and the target in robotic manipulation, and thus it is lightweight and easy to train. In addition to the robust keypoints representation, we also include domain randomization and adversarial examples to improve the sim-to-real transfer. Moreover, we only use grey-scale images in the training and real-world scenarios to minimize the information gap and possible texture information in the real-world.

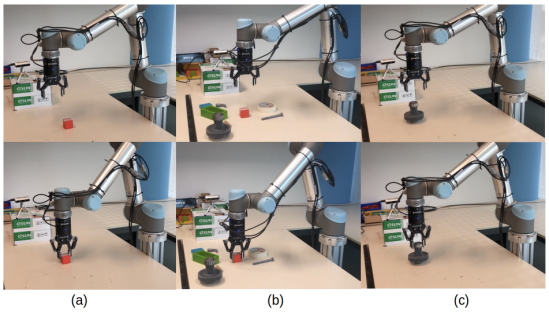


Fig. 12: Experiment process from the third-person viewpoint in three scenarios: (a) GB (b) GB-N (c) GT

V. Conclusion

We proposed an end-to-end reinforcement learning method with robust keypoints representation in robotic manipulation tasks. The keypoints represent compact geometric information from the camera input image, which is more explainable and robust compared to other traditional latent space representations. Moreover, using the keypoints representation trained with data augmentation, domain randomization and adversarial examples, we are able to achieve zero-shot sim-to-real transfer in real-world robotic manipulation tasks with good success rate. Based on the results in this work, we believe that using simple and compact representations such as keypoints to encode the geometric information in manipulation tasks could be promising to improve the performance of the end-to-end learning and sim-to-real transfer for robotic manipulation tasks.

Acknowledgement

This research is supported by A*STAR, Singapore, under its AME Programmatic Funding Scheme (Project #A18A2b0046), and Alibaba Group through Alibaba Innovative Research (AIR) Program.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. P. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[3] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, Çaglar Gülçehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. P. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in starcraft ii using multi-agent reinforcement learning." *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[4] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[5] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," *arXiv preprint arXiv:2003.06085*, 2020.

[6] T. Wang, H. Zhang, W. Q. Toh, H. Zhu, C. Tan, Y. Wu, Y. Liu, and W. Jing, "Efficient robotic task generalization using deep model fusion reinforcement learning," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 148–153.

[7] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.

[8] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6284–6291.

[9] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.

[10] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Learning multimodal representations for contact-rich tasks," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 582–596, 2020.

[11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[13] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[14] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowledge-Based Systems*, vol. 214, p. 106685, 2021.

[15] A. Raffin, A. Hill, R. Traoré, T. Lesort, N. Díaz-Rodríguez, and D. Filliat, "Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics," *arXiv preprint arXiv:1901.08651*, 2019.

[16] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[17] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke, "Training deep neural networks for visual servoing," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[18] E. Y. Puang, K. P. Tee, and W. Jing, "Kovis: Keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation," *arXiv preprint arXiv:2007.13960*, 2020.

[19] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[20] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4238–4245.

[21] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1461–1466.

[22] A. Hundt, B. Killeen, N. Greene, H. Wu, H. Kwon, C. Paxton, and G. D. Hager, ""good robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6724–6731, 2020.

[23] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3343–3352.

[24] Y. Cheng, H. Zhu, C. Acar, W. Jing, Y. Wu, L. Li, C. Tan, and J.-H. Lim, "6d pose estimation with correlation fusion," *arXiv preprint arXiv:1909.12936*, 2019.

[25] A. Byravan, F. Lceb, F. Meier, and D. Fox, "Se3-pose-nets: Structured deep dynamics models for visuomotor control," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[26] E. Y. Puang, P. Lehner, Z.-C. Marton, M. Durner, R. Triebel, and A. Albu-Schäffer, "Visual repetition sampling for robot manipulation planning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9236–9242.

[27] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2901–2910.

[28] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi, "Unsupervised learning of object landmarks through conditional image generation," in *Advances in Neural Information Processing Systems*, 2018, pp. 4016–4027.

[29] S. Berretti, B. B. Amor, M. Daoudi, and A. Del Bimbo, "3d facial expression recognition using sift descriptors of automatically detected keypoints," *The Visual Computer*, vol. 27, no. 11, p. 1021, 2011.

[30] S. Hare, A. Saffari, and P. H. Torr, "Efficient online structured output learning for keypoint-based object tracking," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 1894–1901.

[31] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1510–1519.

[32] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kpam: Keypoint affordances for category-level robotic manipulation," *arXiv preprint arXiv:1903.06684*, 2019.

[33] W. Gao and R. Tedrake, "kpam-sc: Generalizable manipulation planning using keypoint affordance and shape completion," *arXiv preprint arXiv:1909.06980*, 2019.

[34] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese, "Keto: Learning keypoint representations for tool manipulation," *arXiv preprint arXiv:1910.11977*, 2019.

[35] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[36] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[37] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[38] C. Xie, M. Tan, B. Gong, J. Wang, A. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," *arXiv preprint arXiv:1911.09665*, 2019.

[39] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," *arXiv preprint arXiv:1707.01495*, 2017.

[40] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[41] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[42] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[44] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning," *arXiv e-prints*, pp. arXiv–2004, 2020.