SafePicking: Learning Safe Object Extraction via Object-Level Mapping

Kentaro Wada, Stephen James, Andrew J. Davison

Dyson Robotics Laboratory, Imperial College London

{k.wada18, slj12, a.davison}@imperial.ac.uk

Abstract— **Robots need object-level scene understanding to manipulate objects while reasoning about contact, support, and occlusion among objects. Given a pile of objects, object recognition and reconstruction can identify the boundary of object instances, giving important cues as to how the objects form and support the pile. In this work, we present a system, *SafePicking*, that integrates object-level mapping and learning-based motion planning to generate a motion that safely extracts occluded target objects from a pile. Planning is done by learning a deep Q-network that receives observations of predicted poses and a depth-based heightmap to output a motion trajectory, trained to maximize a safety metric reward. Our results show that the observation fusion of poses and depth-sensing gives both better performance and robustness to the model. We evaluate our methods using the YCB objects in both simulation and the real world, achieving safe object extraction from piles.**

I. INTRODUCTION

6D manipulation of objects enables robots to alter object poses to a target state, to achieve tasks such as part assembly [28], [34], object extraction from clutter [32], [38], and arrangement in a specific configuration [10], [22]. Traditionally, manipulation pipelines have been composed of perception that builds explicit scene representation with objects (e.g., 6D poses) followed by planning that searches a collision-free arm trajectory using the representation. As an alternative, a learning-based approach has emerged recently, which directly infers actions from observations (usually raw sensor information) with implicit scene understanding.

Although the traditional pipeline has been successful in structured environments, robotic manipulation in cluttered environments is still challenging due to close contacts and occlusions among objects. Prior work tackled this by combining state-of-the-art object pose estimation and collision-based motion planning [32], [38]; however, this pipeline has struggled to safely and efficiently extract occluded objects in a single step, causing potentially destructive effects on the objects, which is particularly important for fragile objects.

For this challenge in the traditional pipeline, we replace the collision-based motion planning with learning-based planning [7], [15], [21], which receives as input some observation (e.g., images, object poses) and predicts the next best action to take. To avoid undesirable consequences on objects in a pile, we train the model with reinforcement learning while penalizing the translations of non-target objects during the extraction of a target object (e.g., Figure 1).

Despite the use of raw sensor observation being common in prior work, semantic information such as object poses can give important cues as to how robots should manipulate
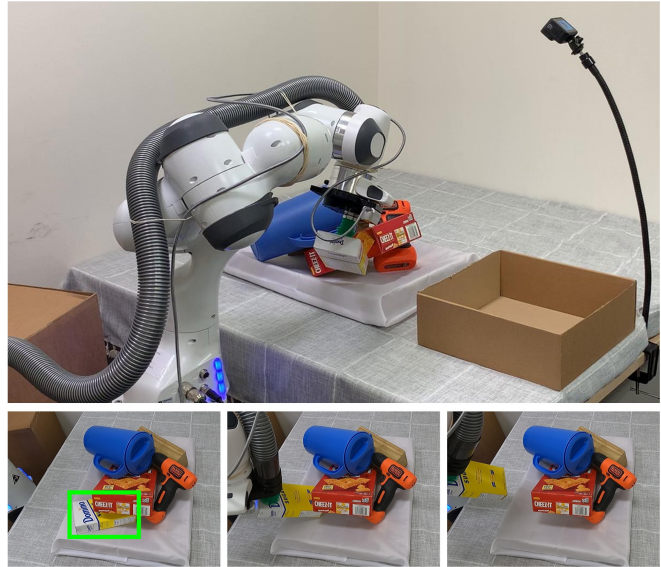

Fig. 1: **SafePicking** extracts target objects with minimum destruction by generating a safe end-effector trajectory given a raw depth observation and object poses from an object-level map.

objects. Object extraction requires a proper understanding of the occluded parts of objects, and we empirically observe that a learned model given object poses as input generates better motions even with errors in the pose estimate.

On the other hand, however, when the pose estimate has errors such as misdetection and pose-difference, a model that only uses object poses (as in the learning-based game agents [1], [2]) would only deteriorate its performance. To handle these errors, we introduce raw observation along with object poses, which enables a model to gain high performance from the pose information as well as robustness from the raw observation.

Our system, *SafePicking*, shown in Figure 2 is composed of 1) **object-level mapping** to build a map with object poses while exploring a target object, and 2) **learning-based motion planning** to generate an end-effector 6D trajectory from raw (depth images) and pose observations (predicted object poses). The grasp point is determined to be the centroid of the visible surface of a target object, and the observations are transformed into the grasp point coordinate to be agnostic to the grasp point (canonicalization). This combination of object-level mapping and canonicalization makes the subsequent learned model general to the object position in a workspace, enabling the model to learn faster.
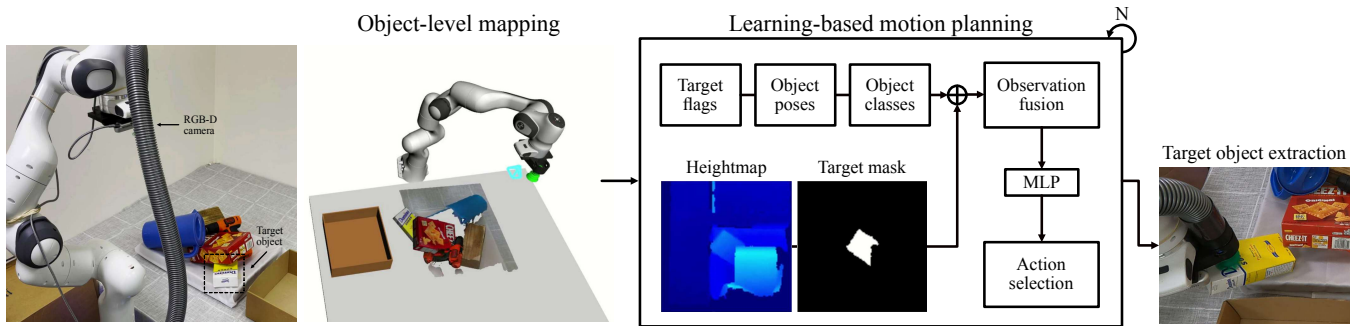
Fig. 2: **System overview**, which consists of 1) object-level mapping with volumetric reconstruction and pose estimation of detected objects with an on-board RGB-D camera, and 2) learning-based motion planning with a model that recursively generates end-effector relative poses with N steps for object extraction using estimated object poses and a raw depth observation in the form of a heightmap.

To our best knowledge, this is the first work that tackles safe object extraction, in which a robot picks occluded target objects with a single grasp minimizing the destructive effect to the surrounding objects. In the experiment, we demonstrate our integrated system in the real world.

In summary, the main contributions of this paper are:

1) **Introducing safe object extraction** as a novel manipulation task, where a robot extracts a target object with a single grasp while minimizing destructive effects.
2) **Fusion of a raw observation and object poses** in learning-based motion planning, achieving high performance and robustness to estimation errors.
3) **Integrated robotic manipulation system** that demonstrates safe object extraction in the real world.

II. RELATED WORK

A. Manipulation with visual scene understanding

Manipulation of a grasped object has been a long-standing problem that dates back to the beginning of robotic research. Traditionally, 6D object manipulation has been tackled via collision-based motion planning in the configuration space of a scene constructed by another perception system or manual annotations [5], [8]. Although this approach has been widely used in industry, the planning component requires complete (i.e., no missing objects) and fairly accurate scene understanding, which is challenging in a cluttered scene. There have been several attempts to increase the robustness of visual perception [30], [4], [29]; however, object detection and pose estimation can still struggle with complex object piles due to close contacts and occlusions. To handle imperfect perception, recent work [27], [32], [38] integrated perception with heuristics-based motion generation, which works with an uncertain estimate of object configurations. Although promising, this heuristics-based approach struggles to handle arbitrary object configurations and requires restricting the scene and target object that can deal with (e.g., requiring exhaustive distractor removal before picking the target). In this work, we use learning-based motion planning to maintain robustness even with imperfect perception.

B. Learning robotic manipulation

Recently, the use of deep learning for robotic manipulation has become prevalent with its progress in visual recognition with convolutional networks and observation-to-action policy learning [20], [35], [14]. With convolutional networks, prior work [21], [25], [37] demonstrated robotic grasping from a single view without explicitly modeling object geometry. For optimizing the policy for a sequential motion, other work has applied deep reinforcement learning for discriminate (i.e., targeted) grasping [7], [9], indiscriminate grasping [15], [36], and retrieval [19]. Our work is along the line with the work on discriminative object manipulation [7], [9], [19], but instead focusing on how to manipulate objects after grasping, which has not been well explored in previous work. Moreover, we exploit object-level scene understanding in the learning-based manipulation model by feeding estimated poses along with a raw depth observation.

III. OBJECT-LEVEL MAPPING

To build a map of objects and find a target object in a scene using an RGB-D camera sequence, we use object-level mapping similar to prior work, MoreFusion [32]. The mapping process consists of 1) single-view object detection with learning-based object detection; 2) occupancy-based volumetric reconstruction for object tracking and multi-view fusion, and 3) pose estimation using the volumetric reconstruction. To find a specific object, we query the object geometry from the map using the class ID of the object, which provides its mask and estimated pose.

A. Object detection from a single view

We use a state-of-the-art object detection model, Mask R-CNN [11], which receives an RGB image and predicts object classes and masks. Using the detected mask, we extract the visible surface of an object from a depth image and accumulate it in the following multi-view volumetric fusion.

Like prior work, we train an object detection model for the YCB objects [3] using existing datasets [32], [33], which contains both real and synthetic images. Although this model is fairly accurate, it can generate false positives with a low threshold of the detection confidence, so we use a relatively high confidence threshold of 75%, as per [32]. Although this high threshold favors false negatives in detecting objects, the multi-view mapping allows the model to run in different frames to find objects with a confident prediction.

B. Volumetric reconstruction in multiple views

We use occupancy volumetric reconstruction to track objects and accumulate their depth observation in multiple views. For efficient accumulation, we use an Octree-based occupancy volumetric fusion framework [12]. Octree structure efficiently queries the associated voxel v for a 3D query point in a new observation z_t, and the occupancy probability of the voxel $p(v|z_{1:t})$ at time t is updated with a new observation $p(v|z_t)$ with Baysian update:

$$\mathcal{L}(v|z_{1:t+1}) = \mathcal{L}(v|z_{1:t}) + \mathcal{L}(v|z_t) \quad (1)$$

$$\text{where} \quad \mathcal{L}(x) = \text{logit}(p(x)) = \log(p(x)/(1-p(x))).$$

As we collect observations z_t with a moving camera, we fuse them into an Octree-based volume, which filters sensor noises to build accurate reconstruction.

To associate the 3D point from an observation z_t to the corresponding voxel $v^o \in V^o$ of an object o, we have two tracking mechanisms for a camera and objects. Camera tracking gives the transformation of an observation z_t to the global coordinate system (i.e., map frame), and object tracking gives the corresponding octree map V^o to which a new observation is accumulated. For camera tracking, we use forward kinematics of a robotic arm and the rigid transformation from the robot to the attached camera. For object tracking, we render the mapped objects in the live camera frame M^o. This rendered mask is compared with the detected masks D_i by object detection with computing the intersection over the union (IoU) between them: $(M^o \wedge D_i)/(M^o \vee D_i)$. When the IoU is over a threshold of 0.4, D_i is recognized as another observation of the object o and accumulated into its octree V^o along with a observed depth, otherwise a new octree is initialized to map the new object.

C. Pose estimation

Using the volumetric reconstruction of objects, we estimate an object pose to replace it with a CAD model. This CAD model replacement gives full geometry of an object, whereas the volumetric reconstruction only gives partial geometry of the visible surface. We use a state-of-the-art pose estimation model in MoreFusion [32], trained for the YCB objects [3]. Following [32], we exploit different views from a moving camera to acquire confident pose estimates. Given a set of estimated poses, we compute the point-to-point distance among the CAD models transformed with the poses. We replace the volumetric reconstruction with a CAD model when this process finds several matches.

IV. LEARNING OBJECT EXTRACTION

To train the motion planning model, we use deep Q-learning [23], an off-policy, model-free reinforcement learning algorithm. This algorithm learns a policy that maximizes the cumulative reward in an episode, using the collected episodes through exhaustive exploration in action space. We use the translations of objects as the negative reward (i.e., penalty) in this algorithm so that the model learns manipulation that minimizes destructive effects.

A. Grasp point selection

We select the centroid of the visible surface of a target object as the grasp point: $\mathbf{p} = [p_x, p_y, p_z]^\mathsf{T}$. The surface geometry and mask of a target object are extracted from the object-level map created in the preceding process. For the orientation, we compute a quaternion $\mathbf{q} = [q_x, q_y, q_z, q_w]^\mathsf{T}$ that gives the minimal transformation to align the axis of a suction cup $\mathbf{v}_g$ to the surface normal of the grasp point $\mathbf{v}_s$:

$$[q_x, q_y, q_z]^\mathsf{T} = \mathbf{v}_g \times \mathbf{v}_s \quad (2)$$

$$q_w = \sqrt{\sum_i \mathbf{v}_{g,i}^2 + \sum_i \mathbf{v}_{s,i}^2} + (\mathbf{v}_g^\mathsf{T} \cdot \mathbf{v}_s). \quad (3)$$

B. Fusing raw and pose observations

We take advantage of both raw and pose observations by feeding them as input to the model. The raw observation is formed as a heightmap generated from depth images, and the pose observation is extracted from the object-level map. These two observations provide different properties. The raw observation has few processes before being fed into the model, so it is less prone to estimation errors. In contrast, pose observation gives complete semantic and geometry (e.g., occluded parts of the objects), which is missing in the raw observation. The fusion of these two observations enables the model to achieve both high performance in object extraction (less destructive effects on a pile) and robustness to estimation errors (e.g., misdetection, pose-difference).

Raw observation

For raw observation, we build a heightmap from depth images. The heightmap represents the heights of object surfaces on a planar workspace and gives information about the visible surface of objects. We build the heightmap centering the XY coordinates of the grasp point p_x, p_y in the image coordinate of the heightmap for canonicalization regarding the grasp point. We use $0.004m$ as the size that each pixel represents and 128 as the image height and width dimensions, giving $\pm 0.256m \ (= 0.004 \cdot 128/2)$ XY bounds.

Pose observation

As for the pose observation at training time, we extract the ground truth object poses and classes from the simulator that we use to train the manipulation model. At test time, we extract object poses from the object-level map, which we build using an on-board camera on a robot. We represent the pose observation with target flags (O,) with a binary vector, object classes (O, C) with one-hot vectors, and poses (O, 7) as a pair of position (3,) and quaternion (4,), where O is the number of detected objects and C is the number of classes.

Before we feed the pose observation into the model, we canonicalize it similarly to the heightmap by centering the grasp point in the pose coordinate. We subtract the XY coordinates of the grasp point p_x, p_y from each pose of the object, which aligns the pose observation along to the raw observation (heightmap).

C. Model design

The model represents a Q-function that predicts the discounted return given an observation o_t at time t. Once trained, we evaluate this Q-function over a set of actions $a \in A$ to take the highest-valued one $\hat{a}$ (the next best action):

$$\hat{a} = \arg \max_a Q(a, o_t). \qquad (4)$$

Action

We formulate the action as a relative 6D end-effector transformation discretizing each axis of translation and rotation. We discretize the translation space in increments of 0.05 m and discretize the Euler rotation space in increments of 22.5 degrees. Each axis has either 0, positive or negative value, and their combinations give $3^6 = 729$ actions.

This relative end-effector action is taken N times to construct a 6D arm trajectory. To give the information of previously taken actions, we feed the previous end-effector poses as input to the model along with the other visual observations (a heightmap; object poses).

Network architecture

Figure 3 shows the network architecture. The input heightmap is downsampled by a convolutional (Conv) encoder to a feature vector. This feature vector is concatenated with object poses, and processed by a transformer encoder [31]. With this transformer encoder, the model can handle an arbitrary number and order of object poses given as input. The output features from this encoder are averaged (similar to [1]) to predict Q-value with a linear layer.

Along with the observations, the model receives two inputs: an evaluation action and a previous end-effector trajectory. The evaluation action is fed as a relative translation and rotation. The previous end-effector trajectory represents the actions the model has taken with a list of 6D poses, allowing it to generate a consistent next action.
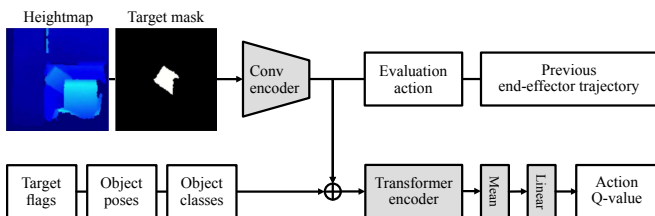

Fig. 3: **Network architecture**, which uses a heightmap and object poses to predict Q-value of 6D end-effector actions. We feed end-effector relative motions as evaluation actions, from which the best-scored action is selected as the next action.

Reward

The destructive effects on surrounding objects during manipulation can be classified as follows:

- Falling, which happens when overlapping objects are mainly supported by a target object and they fall after the extraction of the target;
- Sliding, which happens when surrounding objects have to be displaced to create a space to extract a target.

Both of these two effects are undesirable. The falling effect can damage objects, and the sliding effect can expand the pile, which can make subsequent task continuation harder.

To cover both effects in the reward, we use the sum of the translations of non-target objects in a pile as the metric. When an object falls a large distance, not only does the translation of the object itself become significant, but it can also hit other objects causing chain effects. By using the "sum" of translations, we can encourage the model to minimize the number of objects affected as well as the translations of individual objects.

D. Training the model

We compute the reward r_t at each time step t, and train a deep Q-network to maintain the Bellman equation:

$$q_{t,a} = Q(o_t, a) \qquad (5)$$

$$\hat{q}_{t,a} = r_t + \gamma \max_a \tilde{Q}(o_{t+1}) \qquad (6)$$

$$\mathcal{L} = |\hat{q}_{t,a} - q_{t,a}|, \qquad (7)$$

where γ is the time discount of the reward, Q is the live network updated every training step, and $\tilde{Q}$ is the target network, a copy of the live network updated less frequently.

We train the model in a physics simulation environment since, in the real world, it is difficult to track the motions of objects to compute the reward. It would be also time-consuming and challenging to build various configurations of objects for each trial of a robot learning in the real world while maintaining the safety of the robot and objects.

To let a robot experience various configurations of objects, we procedurally generate object piles by simulating feasible pile configurations. We define a 3D bounding box where object models can be spawned. Each step randomly selects a model and its pose, and we apply physics simulation until the spawned object stops moving. This process produces diverse object configurations.

V. EXPERIMENTS

We evaluate our method by assigning a robot to grasp and extract a target object from object piles in both simulation and the real world. For training the learning model in simulation, we use the YCB models provided by [3], [33].

A. Training detail

We implement the learning model with PyTorch [24] and the simulation environment with PyBullet [6]. We run a single process to update the model parameters using action-state paired data. This data is collected by multiple processes that asynchronously run the learned model to act in different environments for exploration. For this asynchronous data collection and training, we use an open-source framework [13].

As for the training hyperparameters, we use a batch size of 128, and Adam [17] as the optimizer with a learning rate of 0.001. From the start of training, we use epsilon greedy exploration to collect data until 5000 iterations. We use replay ratio 16 (number of updates per data collection), and synchronize the model parameters every 100 iterations

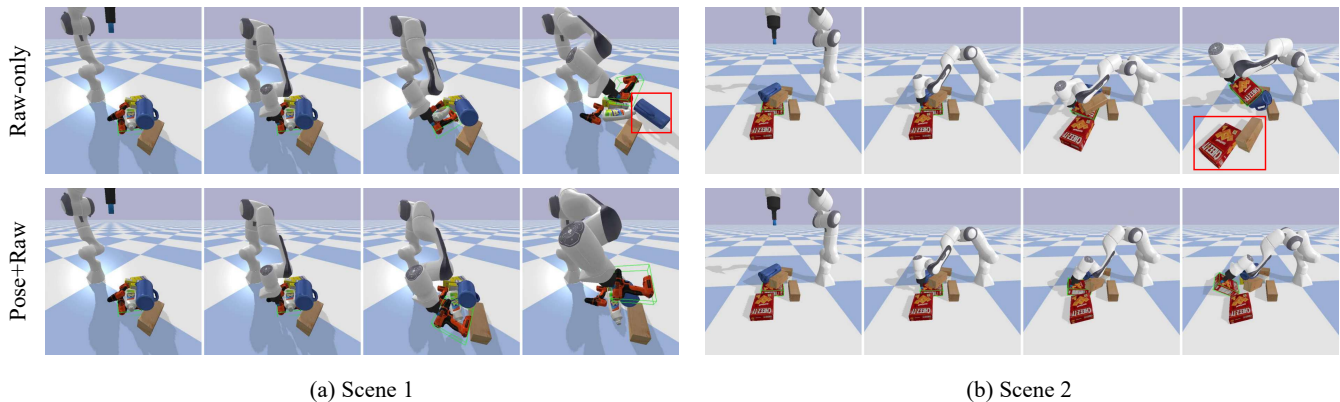|	Raw-only
	Pose+Raw

(a) Scene 1   (b) Scene 2

Fig. 4: **Qualitative results of the model ablation**, in which we compare *Raw-only*, which uses only a heightmap observation, and our full model *Pose+Raw*, which uses object poses along with a heightmap.

to allow the model in the exploration processes to collect episodes with the new parameters.

B. Evaluation metrics in simulation

To evaluate the performance of the model, we define metrics that represent how safely a robot extracted objects. As we discussed in §IV-C, this metric should represent the destructive effects on non-target objects caused by falling and sliding, and for that, we use the followings:

- Sum of translations, which evaluates both falling and sliding effects;
- Sum of max velocities, which primarily evaluates falling effects, as a larger distance fall gives higher velocity.

C. Baseline comparison

1) Naive motion: As the simplest motion, we use joint linear interpolation from a grasp pose to a reset pose where the end-effector is located in free space and the suction cup faces down. When a target object is overlapped by distractor objects, this extraction motion introduces many collisions and causes falling and sliding effects on the distractors, and shows the lowest score in Table I.

2) Heuristic motion: As a simple heuristic for extracting objects from a pile, we use a motion that extracts a grasped object with a straight motion towards the +Z direction of the world coordinate (the opposite direction of the gravity). This motion gives better results than the naive motion (Table I).

3) Collision-based motion planning: As another baseline, we use a collision-based motion planning, RRT-Connect [18], which is a widely used motion-planning algorithm with estimated object poses. Although this motion planning can give non-destructive extraction motion when it finds a collision-free path, it struggles to find *relatively* safe trajectories when a complete collision-free path does not exist. In this case, the motion planner can end up giving the naive motion, which causes significant movement of surrounding objects. Table I shows that RRT-Connect gives comparable results as the heuristic motions (better in translation, worse in velocity), but our learned model (SafePicking) gives a much better result.

TABLE I: **Baseline comparison**, in which we compare the motion of our learned model with baseline methods using the safety metric, in 600 unseen pile configurations in simulation.

Method	Input	Noise	Safety metric translation↓	velocity↓
Naive	-		0.701	1.919
Heuristic	-	no	0.578	1.624
RRT-Connect	pose		0.520	1.643
SafePicking	pose, heightmap		**0.465**	**1.419**
RRT-Connect	pose	yes	0.532	1.645
SafePicking	pose, heightmap		**0.465**	**1.433**

D. Model ablation

We evaluate 3 model variants with different inputs to compare the effect of different observations given to the learned model (Table II). To evaluate the robustness, we add noises to the pose observation (misdetection, pose-difference) based on the visibility of objects. *Raw-only* receives as input only the heightmap generated from a depth image. *Pose-only* uses object poses as input, which gives more complete information about objects, and gives better results than Raw-only even when noises are introduced in the input object poses. The full model, *Pose+Raw*, receives both heightmap and object pose as input, and compensates the noises in object poses using the information about the object's visible surface from the heightmap. The results show that this fusion of object pose and heightmap observations works best in the presence of pose noises. As one would expect, when given perfect pose information (no pose noises), there is no benefit to having the additional heightmap information; however, in reality, estimated object poses will always have errors. Figure 4 shows qualitative results.

E. Real-world evaluation

We evaluate our system in the real world on a Franka Emika Panda robot, integrating using the Robotic Operation System (ROS) framework [26]. To capture the short-range depth (∼0.1m), we used Realsense D435 [16] as the RGB-D camera mounted on the wrist of the robotic arm.

TABLE II: **Model ablation**, in which we compare the variants of the learned model with/without adding noises to the pose observation, and test in 600 unseen pile configurations in simulation.

Variant	Input	Noise	Safety metric translation↓	velocity↓
Raw-only	heightmap		0.507	1.491
Pose-only	pose	no	0.477	1.430
Pose+Raw	pose, heightmap		**0.465**	**1.419**
Pose-only	pose	yes	0.487	1.449
Pose+Raw	pose, heightmap		**0.465**	**1.433**

1) Metric: In the real world, the safety metrics used in the simulation (the translation and velocity of all objects) are challenging to acquire. Therefore, we use the heightmap difference between before and after a task as the metric (Figure 5). The robot scans a pile and builds two heightmaps (Figure 5a, b), in which the region of a target object is excluded (since the target object is intentionally moved). These heightmaps are compared (Figure 5c) to compute the volume of the difference and the size of the mask at a threshold of $0.01m$.

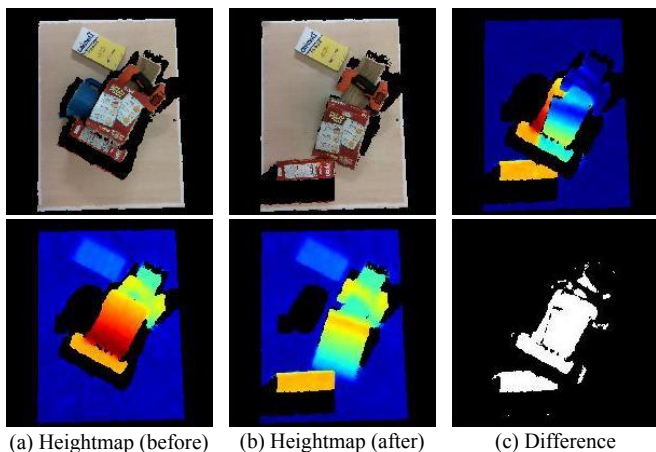(a) Heightmap (before) (b) Heightmap (after) (c) Difference

Fig. 5: **Heightmap-based evaluation metric** that compares the heightmaps of a pile built before/after manipulation. The region of a target object (blue pitcher in this case) is excluded from the comparison and is filled with black color in the heightmap visualization.

2) Results: Using the heightmap-based metric, we evaluate the variants of the learned models in the real world. As for the task setting, we build similar object configurations as the simulation, where objects are closely located in a pile and a target object is partially occluded by other objects, and the robot is tasked to extract the target object with a single grasp. Table III shows the comparison of the two learned models (Raw-only and Pose+Raw) in 20 configurations. Some of their qualitative results are shown in Figure 6. These comparisons show consistent results as the ablation study in the simulation, showing that object pose information enables the model (Pose+Raw) to generate efficient and safe motions compared to the one with only raw observations (Raw-only).

TABLE III: **Real-world model comparison**, in which we evaluate learned models comparing the heightmaps before/after each task. The models are tested in the same 20 configurations.

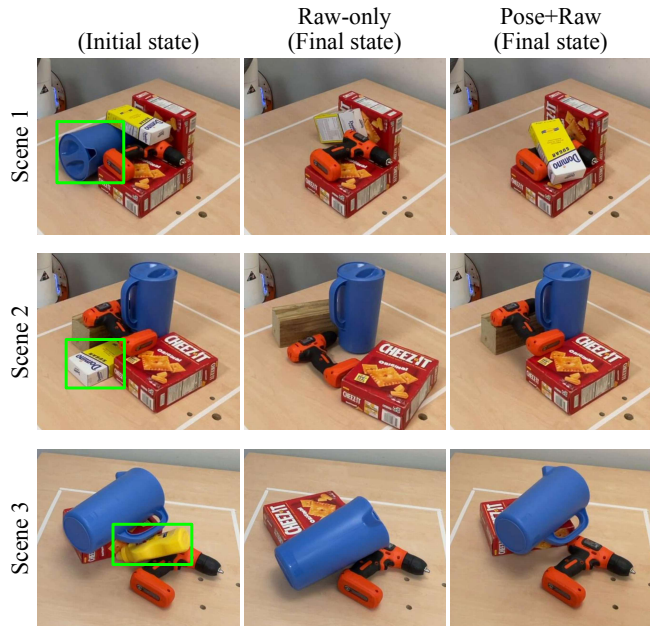| Variant	Safety metric Diff mask [%]↓	Diff volume [litter]↓
Raw-only	7.1	3.2
Pose+Raw	**4.4**	**2.1**

Fig. 6: **Pile configurations** out of 20 used in Table III, where we test the two learned models. We measure the performance with the heightmap-based safety metric. Consistently to the results in the simulation, the model generates better motions when given a fused observation of object poses and a heightmap (Pose+Raw).

VI. CONCLUSIONS

We have shown a 6D manipulation system to efficiently and safely extract target objects from a pile with a trajectory that minimizes destructive effects on the surrounding objects. Our system integrates object-level mapping and learning-based motion planning that uses raw observation from an RGB-D camera and estimated object poses from the object-level map. By fusing the raw and pose observations, the motion model maintains both capabilities of safely extracting objects and robustness to the errors in pose estimation.

In this work, we focused on post-grasp 6D manipulation of household objects; however, it would be interesting to extend this to long-term motion optimization with grasping and placement, and more general objects including soft and high-friction objects. Moreover, we believe there are still various possibilities in integrating object-level scene understanding with learning-based manipulation. This work has mainly exploited the completeness of pose estimates; however, object pose information can also be used in different ways such as for object placement in a specific pose.

ACKNOWLEDGEMENTS

REFERENCES

[1] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint:1606.01540*, 2016.

[3] B. Calli, A. Singh, A. Walsman, P Srinivasa S. and, Abbeel, and A. M. Dollar. The YCB object and Model set: Towards common benchmarks for manipulation research. In *International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015.

[4] Xiaotong Chen, Rui Chen, Zhiqiang Sui, Zhefan Ye, Yanqi Liu, R Iris Bahar, and Odest Chadwicke Jenkins. Grip: Generative robust inference and perception for semantic robot manipulation in adversarial environments. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2019.

[5] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit! *IEEE Robotics and Automation Magazine*, 2012.

[6] Erwin Coumans and Yunfei Bai. PyBullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[7] Coline Manon Devin, Eric Jang, Sergey Levine, and Vincent Van-houcke. Grasp2Vec: Learning object representations from self-supervised grasping. In *Conference on Robot Learning (CoRL)*, 2018.

[8] Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 2008.

[9] Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, Silvio Savarese, and Mrinal Kalakrishnan. Multi-task domain adaptation for deep learning of instance grasping from simulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[10] Wei Gao and Russ Tedrake. kPAM-SC: Generalizable manipulation planning using keypoint affordance and shape completion. *arXiv preprint arXiv:1909.06980*, 2019.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[12] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013.

[13] Stephen James. YARR: Yet another robotics and reinforcement learning framework for pytorch. `https://github.com/stepjam/YARR`, 2021–2022.

[14] Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. *arXiv preprint arXiv:2106.12534*, 2021.

[15] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2018.

[16] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[18] James J Kuffner and Steven M LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[19] Andrey Kurenkov, Joseph Taglic, Rohun Kulkarni, Marcus Dominguez-Kuhne, Animesh Garg, Roberto Martín-Martín, and Silvio Savarese. Visuomotor mechanical search: Learning to retrieve target objects in clutter. *arXiv preprint arXiv:2008.06073*, 2020.

[20] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1), Jan 2016.

[21] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *International Journal of Robotics Research (IJRR)*, 37(4-5):421–436, 2018.

[22] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kPAM: Keypoint affordances for category-level robotic manipulation. *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2019.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, 2019.

[25] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[26] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. *ICRA workshop on open source software*, 3(3.2):5, 2009.

[27] Max Schwarz, Anton Milan, Christian Lenz, Aura Munoz, Arul Selvam Periyasamy, Michael Schreiber, Sebastian Schüller, and Sven Behnke. Nimbro picking: Versatile part handling for warehouse automation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[28] Stefan Stevšić, Sammy Christen, and Otmar Hilliges. Learning to assemble: Estimating 6D poses for robotic object-object manipulation. *IEEE Robotics and Automation Letters*, 5(2):1159–1166, 2020.

[29] Zhiqiang Sui, Haonan Chang, Ning Xu, and Odest Chadwicke Jenkins. Geofusion: Geometric consistency informed scene estimation in dense clutter. *IEEE Robotics and Automation Letters*, 5(4):5913–5920, 2020.

[30] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems (NIPS)*, 2017.

[32] Kentaro Wada, Edgar Sucar, Stephen James, Daniel Lenton, and Andrew J. Davison. MoreFusion: Multi-object reasoning for 6D pose estimation from volumetric fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[33] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[34] Kevin Zakka, Andy Zeng, Johnny Lee, and Shuran Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[35] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics (T-RO)*, 2020.

[36] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245, 2018.

[37] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[38] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.