Fine-Grained Object Detection and Manipulation with Segmentation-Conditioned Perceiver-Actor*

Shogo Akiyama[1], Dan Ogawa Lillrank[1] and Kai Arulkumaran[1]

***Abstract*— Prior work has shown the benefits of using a 3D representation space—in particular, voxels—for 3D manipulation tasks. However, computation with voxels requires N^3 memory, which limits the possible observation size. While the structure of voxels convey spatial information, limited resolution can obscure semantically-relevant information. In this work, we show this can be overcome by conditioning a 3D-based agent, Perceiver-Actor, on additional segmentation information, which allows it to successfully distinguish between similar objects for manipulation tasks. This is achieved by using pretrained segmentation and text-image models to extract segmentation masks for relevant objects in a zero-shot manner. We demonstrate our model on a real robot, where we show it can correctly interact with objects with fine-grained differences, such as a "Cola" can versus a "Dr. Pepper" can.**

I. INTRODUCTION

The last few years in machine learning have been dominated by "the bitter lesson"—that scaling (relatively) simple models and data can surpass many ingenious, hand-engineered solutions [1]. This has held in domains where data is plentiful, such as text and images, but notwithstanding large industrial efforts [2], this seems infeasible for robotics. The field of robot learning has therefore sought to combine the best of both worlds—pretrained deep learning models, with established solutions for robotics.

One such example is CLIPort [3], an agent that can perform a wide range of tabletop manipulation tasks by combining the general semantic information learned by the CLIP text-vision multimodal model [4], with the spatial inductive biases of Transporter networks [5]. CLIP has been widely used in many domains, and robotics is no exception [3], [6], [7], [8], [9], [10], [11].

However, we are still a way from a *general* solution for manipulation, as dictated by the large-scale vision-guided manipulation RLBench benchmark [12]. Challenges include using image sensors to locate objects, controlling an arm with several joints, and performing manipulation with 6 DoF (translation + rotation). Significant progress on this benchmark was achieved by the C2F-ARM agent [13], which uses 3D (voxel) observation and action spaces, far outperforming the best 2D agent [14]. Notably, the C2F-ARM agent is data-efficient, outperforming baselines with 100 demos, when it itself is only given 10 demos. C2F-ARM was then superseded by the Perceiver-Actor (PerAct) agent [10], which replaced the "coarse-to-fine" hard attention mechanism in C2F-ARM with a Perceiver IO Transformer [15] to efficiently process the voxel observation space.

Fig. 1: 720×1280 image view from the RGBD camera (top) vs. 100^3 voxel view (bottom). Fine-grained semantic information somewhat diminished in the voxel view.

However, PerAct is still fundamentally limited by the size of the raw observation space. The authors use an input and output size of 100^3 voxels, representing 1m^3, and train their model on a batch size of 16 over 8 NVIDIA V100 GPUs. With the chosen hyperparameters, a backwards pass (with the LAMB optimiser [16]) with a single sample requires ∼11GB. As seen in Fig. 1, whilst it is possible to distinguish between similar objects, such as different soft drink cans, in the 720×1280 RGB image, this information can be more difficult to extract when voxelised to 100^3, and even more so at even lower resolutions. Furthermore, PerAct's feature space (Sec. II) biases its policy towards spatial location/object shapes. As such, the PerAct agent often chooses the wrong object to interact with if it is difficult to distinguish from other objects in the scene.

We propose a relatively simple solution—conditioning the agent on an additional object mask—a solution often employed by grasp controllers [17], [18], [19], [20], [21]. If

*Work supported by JST, Moonshot R&D Grant Number JPMJMS2012.

[1]Shogo Akiyama, Dan Ogawa Lillrank and Kai Arulkumaran are with Araya Inc., Tokyo 107-0052, Japan {akiyama_shogo, dan_ogawa, kai_arulkumaran}@araya.org

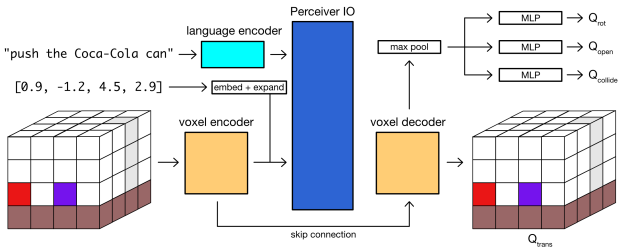


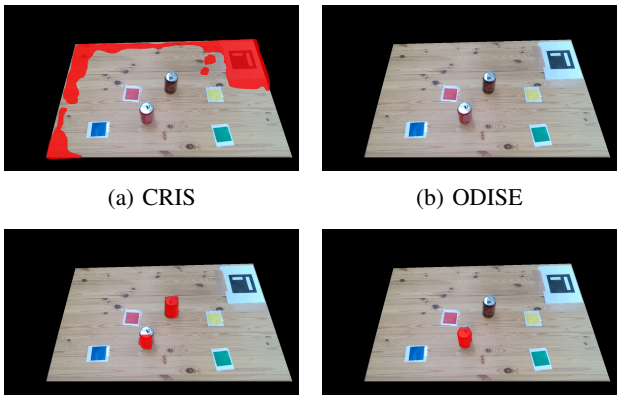
Fig. 2: PerAct architecture. The voxel and text inputs are separately encoded, with proprioceptive inputs embedded and then concatenated with the output of the voxel encoder. These are then processed by a Perceiver IO Transformer, and decoded back to voxels. The voxels are used to directly predict the end-effector translation, and further processed through pooling and fully-connected layers to predict the rotation, gripper state, and motion planner mode.

a mask can be extracted from the raw 2D RGB(D) image, then it can be included in the voxels along with the RGB values. To be scalable, we require a segmentation model that can extract a wide variety of objects, and performs robustly on real-world images. Ideally such a model should operate in a "zero-shot" fashion, where rather than predicting masks based on preset classes, we can query it for our objects of interest. Recently this has been enabled by text-conditioned segmentation models [22], [23], [24]. However, in initial experiments we found these were not robust. Instead, we found that the recently-released SAM model [25], that can propose label-agnostic segmentation masks for the whole image, was robust enough to provide good quality segmentation masks on our real robot setup without requiring further training. By combining this with text-image similarity search, we could thereby extract masks for objects of interest. By conditioning PerAct on these masks, we enabled it to push objects with similar appearance in the real world, where PerAct without this information failed.

II. METHOD

The core of our method is the PerAct agent [10]. The model takes voxels and a text-based task condition as input, and produces a structured action space for the end-effector as output: translation, rotation, gripper state, and motion planner mode (Fig. 2). Based on prior work on RLBench [14], the agent's prediction is fed into a motion planner (which must avoid collisions when moving in open space, or "not" if it needs contact with an object), resulting in an observe-plan-execute loop. The authors chose to use a pretrained CLIP model [4] for the language encoder, which provides vision-guided language embeddings, but PerAct learns vision from scratch, which means that it does not benefit from generalisation across the visual domain—only language.

The PerAct agent is trained using supervised learning, similarly to behavioural cloning, on a set of demonstrations. For full architectural and training details, we refer readers to the original work [10]. We added two further regularisation methods to improve PerAct's performance on the real robot:



(a) CRIS (b) ODISE

(c) CLIPSeg (d) OWL-ViT + GrabCut

Fig. 3: Segmentation masks for "Cola" from different models for the image in Fig. 1. Mistakes include selecting parts of the workspace, not making any selections at all, selecting several objects (which may/may not include the target object), or producing incomplete masks.

TABLE I: Segmentation GPU inference time and memory usage, evaluated with 1 720×1280 image and 1 text prompt.

Name	Time (s)	Memory (GB)
CLIPSeg	0.70	1.68
CRIS	0.72	3.30
ODISE	4.82	16.3
OWL-ViT + GrabCut	1.82	1.60
SAM + CLIP	4.07	6.14

dropout on the point clouds (at a rate of 30%), and additive noise on the robot joint positions ($\sim \mathcal{N}(0, 0.1)$).

The viability of adding segmentation masks into PerAct depends on the quality of current pretrained (zero-shot) segmentation models. Although these have improved significantly in recent years, we found that most were not robust in cluttered scenes (the area outside of the table workspace), and even when restricted to the tabletop[1] failed to associate the correct object with its corresponding text label, if at all [22], [23], [24]. We also tried an alternative approach, using the OWL-ViT zero-shot object detector [26] + the bounding-box-based segmentation algorithm GrabCut [27], but either algorithm could fail and introduce errors into the process. Fig. 3 shows some failure cases with these methods.

In comparison, SAM, which is simply trained to predict object-agnostic masks given a variety of spatial-conditioning prompts, provides a comprehensive segmentation of the entire image input (Fig. 4). Although the authors of SAM trained a text-conditioned variant, this model was not released. Therefore, we first use SAM's automatic mask generator mode to propose masks for the whole scene, tuning its hyperparameters to reduce over-segmentation (Fig. 4). We then embed the (image) parts using CLIP's vision encoder, and compare these to query text embeddings from CLIP's text encoder using cosine similarity. As taking the the

[1]Information which is already required for PerAct's voxelisation process.

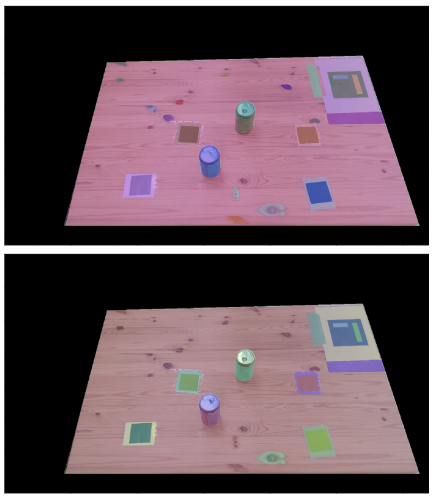


Fig. 4: Mask predictions from pretrained SAM model with default (top) and tuned (bottom) hyperparameters. By default SAM tends to over-segment areas in an image.

segmented object with the highest similarity can sometimes result in selecting a similar object to the desired one, we also query for all relevant objects on the table and use the Hungarian method to solve for assignments. Although this is one of the more resource-intensive methods (see Tab. I), it is little overhead compared to running PerAct.

PerAct's voxelisation process involves taking an RGBD image from a camera with known extrinsic parameters, turning this into a point cloud, and then constructing the voxel grid with respect to a fixed frame (the robot base). PerAct uses a 10D voxel grid: 3 for RGB values, 3 for Cartesian coordinates, 1 for occupancy, and 3 for voxel grid indices. We augment this with an 11^{th} dimension for a binary segmentation mask: 1 for the object, and 0 otherwise.

III. EXPERIMENTS

For our experiments we use a Franka Panda robot with its standard gripper, with an Intel D435i camera with RGBD images captured at a resolution of 720×1280. We use ROS [28] to interface with the robot and the MoveIt package [29] with RRT-connect [30], restricted to Cartesian path planning, as PerAct's motion planner. Demos were collected by specifying gripper positions[2] for the motion planner with an XBox One controller, with sensory inputs synchronised and sampled at 30Hz. To compute the camera extrinsic parameters we used ArUco markers [31] and the default hand-eye calibration package from MoveIt.

The environment is as shown in Fig. 1: the robot is placed in front of a tabletop with 4 coloured markers, and 2 objects (one target, one distractor). The set of objects we use are a "Cola" can, a "Dr. Pepper" can and a "black bottle" (Fig. 5). The task is for the agent to push the target object to a target marker, specified via text, e.g., "push the Cola to the green marker". Both when collecting training data and evaluating the agent, the objects are placed between

[2]Which are used as the keypoints for PerAct's training.

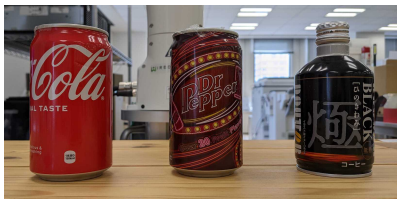


Fig. 5: Objects: "Cola", "Dr. Pepper" and "black bottle".

TABLE II: Training and evaluation object pair settings.

	Target Obj.	Distractor
Training + Evaluation	Cola	Dr. Pepper
	Dr. Pepper	Cola
Evaluation	black bottle	Cola

the markers such that either the target or distractor object might be closer to the target marker, but a solution trajectory would not cause a collision between the objects. We collected a total of 40 demos (2 object pairs $\times$ 4 markers $\times$ 5 positions) for training, and performed evaluation 48 times per model (3 object pairs $\times$ 4 markers $\times$ 4 positions), using both in-domain pairs and an unseen object pair to test zero-shot generalisation ability (Tbl. II). During training we randomised the object positions, but used fixed positions during evaluation to keep these consistent across models.

Due to hardware limitations for training PerAct, we changed several hyperparameters: a voxel grid of 60^3 (which was shown in the original work to have relatively close performance to 100^3 [10]), a hidden size of 512, a latent size of 512, and a batch size of 4 (which still requires gradient accumulation on our hardware). The original authors used a multitask model for their main results, so it is possible to train single task models using reduced compute and memory. Indeed, despite the reduction in capacity, we were able to successfully replicate performance on several RLBench tasks with our single task models, so proceeded with these hyperparameters for experiments with the real robot.

Conditioning PerAct on segmentation masks successfully improved its ability to distinguish between objects, and even interact with an object that was not seen in the training data (Tbl. III). Although on average the standard PerAct agent achieves a 50% success rate on the in-domain object pairs, almost every failure occurs due to pushing the wrong object (to the correct marker). On the other hand, when conditioned on segmentation, most failures were due to the segmentation failing—so making this more robust should improve results further. As further evidence of this hypothesis, preliminary experiments on the "lift numbered block" task in RLBench

TABLE III: Success rate (%) of PerAct models.

Target Obj.	Distractor	PerAct	PerAct + Seg.
Cola	Dr. Pepper	37.5	81.2
Dr. Pepper	Cola	62.5	75.0
black bottle	Cola	25.0	50.0

using ground truth segmentation masks resulted in high success rates, versus the standard PerAct agent.

In the zero-shot setting, the standard PerAct agent usually pushed the "Cola" can, overfitting to this object from its training data. When conditioned on segmentation masks, PerAct typically pushed the "Cola" can if the segmentation failed, but even when segmentation was successful it sometimes failed to complete the task successfully. Some of the failures came from the gripper colliding with the "black bottle" when coming down, hitting the area to the side of the cap. We hypothesise that since the training data consisted exclusively of objects of perfectly cylindrical shapes, the model made such mistakes.

IV. DISCUSSION

In this work, we introduced segmentation-conditioned PerAct. Using the power of pretrained segmentation and text-vision models, we can cheaply generate useful semantic information for robotic control, without requiring further training of the pretrained models. In our experiments on a real Franka Panda we demonstrated improved success rates on a task involving distinguishing between similar objects.

The success rate could be improved by collecting more data, with more varied data to improve generalisation. Although adding segmentation information improved zero-shot performance, a wide enough dataset could also provide this improvement. However, collecting demonstrations is a time-consuming process, so methods that are more sample-efficient are still relevant for robotics applications.

Although the generalisation ability of SAM enabled us to extract good masks, its "segment anything" mode did require further processing. Text-image similarity search with CLIP embeddings was not always successful, and having a single model perform zero-shot text-conditioned object segmentation could alleviate this—in particular, the knowledge that there are several known objects in the scene and matching these can reduce false positives. Further work on improving this post-processing phase should lead to downstream improvements for segmentation-conditioned policies.

REFERENCES

[1] R. Sutton, "The bitter lesson," *Incomplete Ideas (blog)*, 2019.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "RT-1: Robotics transformer for real-world control at scale," *arXiv:2212.06817*, 2022.

[3] M. Shridhar, L. Manuelli, and D. Fox, "CLIPort: What and where pathways for robotic manipulation," in *CoRL*, 2022, pp. 894–906.

[4] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021, pp. 8748–8763.

[5] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *CoRL*, 2021, pp. 726–747.

[6] P. Sharma, B. Sundaralingam, V. Blukis, C. Paxton, T. Hermans, A. Torralba, J. Andreas, and D. Fox, "Correcting robot plans with natural language feedback," *arXiv:2204.05186*, 2022.

[7] T. Xiao, H. Chan, P. Sermanet, A. Wahid, A. Brohan, K. Hausman, S. Levine, and J. Tompson, "Robotic Skill Acquisition via Instruction Augmentation with Vision-Language Models," in *CoRL Workshop on Language and Robotics*, 2022.

[8] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," *arXiv:2210.06407*, 2022.

[9] P.-L. Guhur, S. Chen, R. G. Pinel, M. Tapaswi, I. Laptev, and C. Schmid, "Instruction-driven history-aware policies for robotic manipulations," in *CoRL*, 2023, pp. 175–187.

[10] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-Actor: A multi-task transformer for robotic manipulation," in *CoRL*, 2023, pp. 785–799.

[11] D. Shah, B. Osiński, S. Levine, *et al.*, "LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action," in *CoRL*, 2023, pp. 492–504.

[12] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE RA-L*, vol. 5, no. 2, pp. 3019–3026, 2020.

[13] S. James, K. Wada, T. Laidlow, and A. J. Davison, "Coarse-to-fine Q-attention: Efficient learning for visual robotic manipulation via discretisation," in *CVPR*, 2022, pp. 13 739–13 748.

[14] S. James and A. J. Davison, "Q-attention: Enabling efficient learning for vision-based robotic manipulation," *IEEE RA-L*, vol. 7, no. 2, pp. 1612–1619, 2022.

[15] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, *et al.*, "Perceiver IO: A general architecture for structured inputs & outputs," *arXiv:2107.14795*, 2021.

[16] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large batch optimization for deep learning: Training BERT in 76 minutes," *arXiv:1904.00962*, 2019.

[17] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robot. Res.*, vol. 34, no. 4-5, pp. 705–724, 2015.

[18] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-DOF grasping for target-driven object manipulation in clutter," in *ICRA*, 2020, pp. 6232–6238.

[19] A. M. Christopher Xie, Yu Xiang and D. Fox, "The Best of Both Modes: Separately Leveraging RGB and Depth for Unseen Object Instance Segmentation," in *CoRL*, 2020.

[20] Y. Li, T. Kong, R. Chu, Y. Li, P. Wang, and L. Li, "Simultaneous semantic and collision learning for 6-DoF grasp pose estimation," in *IROS*, 2021, pp. 3571–3578.

[21] S. Back, J. Lee, T. Kim, S. Noh, R. Kang, S. Bak, and K. Lee, "Unseen object amodal instance segmentation via hierarchical occlusion modeling," in *ICRA*, 2022, pp. 5085–5092.

[22] T. Lüddecke and A. Ecker, "Image segmentation using text and image prompts," in *CVPR*, 2022, pp. 7086–7096.

[23] Z. Wang, Y. Lu, Q. Li, X. Tao, Y. Guo, M. Gong, and T. Liu, "CRIS: CLIP-driven referring image segmentation," in *CVPR*, 2022, pp. 11 686–11 695.

[24] J. Xu, S. Liu, A. Vahdat, W. Byeon, X. Wang, and S. De Mello, "Open-Vocabulary Panoptic Segmentation with Text-to-Image Diffusion Models," *arXiv:2303.04803*, 2023.

[25] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment Anything," *arXiv:2304.02643*, 2023.

[26] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, *et al.*, "Simple open-vocabulary object detection with vision transformers," *arXiv:2205.06230*, 2022.

[27] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut" - Interactive foreground extraction using iterated graph cuts," *ACM TOG*, vol. 23, no. 3, pp. 309–314, 2004.

[28] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, 2009, p. 5.

[29] S. Chitta, I. Sucan, and S. Cousins, "MoveIt!" *IEEE RAM*, vol. 19, no. 1, pp. 18–19, 2012.

[30] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, 2000, pp. 995–1001.

[31] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognit.*, vol. 47, no. 6, pp. 2280–2292, 2014.