Action-Centric Vision-and-Language Manipulation Using Perceiver-Actor on VLMbench

Xihang Yu, Ziyun Chi

***Abstract*— Recent progress in embodied AI makes robotics close to reality but it is still an open question how the agent can do manipulation tasks following human guidance. It is essential for robot agents to complete manipulation tasks with language guidance from humans. By following the language instructions, robots can 1) pay more attention to the target place or objects that humans are concerned about and 2) have more information in scene understanding like inter-object relation. Among all existing algorithms, the state-of-the-art robot-centric Perceiver-Actor works well in multiple housing tasks (e.g. pick, place and stack) in RLBench[1]. This project re-implements Perceiver-Actor [2], tests the result on the tasks on VLMbench dataset [3] that has not been tested on Perceiver-Actor yet.**

I. Problem Statement

A. Language-Conditional Manipulation

Language-conditioned manipulation is a widely studied area. Existing vision language manipulation (VLM) algorithms [4] [5] [3] [6] [7] have demonstrated impressive performance on object manipulation tasks. While CLIPort proposed by [4] only deals with 3 DoF desktop object rearrangement tasks, the follow-up work [3] extends CLIPort to 3D environment with 6-DoF manipulation. Unlike the other three papers, StructFormer proposed in [6] uses object-centric model in object rearrangement (i.e. directly predicting the relative pose of objects) and has good results on the specific shape formation tasks. StructDiffusion [7] improves the performance of StructFormer by incorporating diffusion model in the pipeline. However, StructFormer and StructDiffusion focus on object rearrangements. In addition, [6] and [7] use pre-segmented pointcloud which is not accessible in general. Perceiver-Actor proposed by [5], which achieves record-breaking performance in general language-conditional tasks. All of the above works solve language-conditional tasks in static environment. Our work will generalize to dynamic grasping tasks with language inference.

B. Transformer for Manipulation

Transformer was first outperformed in Natural Language Processing impressively [8], before which it shows a revolutionary level of performance in computer vision and biology [9][10][11]. In manipulation, applying transformer on imitation learning for manipulation [12] can increase the success rate of not only grasp but also flip on dual-arm robots dramatically(more than 50%). Also, by using Perceiver Transformer, others can encode language goals and voxel observations for better data enhancement[13].

C. End-To-End Manipulation

It witnesses a huge success on end-to-end manipulations, like applying on vision-based manipulation[14] that can improve the task of grasp success rate[15]. End-to-end learning is also applied in interactive estimation and demonstrative-guided goal strategies[16][17]. Also, end-to-end learning of binding vision and control has shown an incredible success rate increase, even achieved 100% with hanging a coat hanger[18].

II. Introduction

The task of our interest is to encode language goals and RGB-D visual observations and output end-to-end actions until completion of the goals. As shown in Fig. 1, the algorithm we choose is Perceiver-Actor, a behavior-cloning agent to complete multi-task for 6-DoF manipulation. The core of this framework is the encoding of visual images as voxel sequence and the use of transformer-like PerceiverIO which provides a strong structural prior for efficient learning.

There are two contributions in this project:

1) Testing Perceiver-Actor on VLMbench to test the robustness of the framework.
2) Open source code for re-implemention of Perceiver-Actor architecture and its APIs to VLMbench dataset. https://github.com/XihangYU630/perceiver-actor-vlmbench.git

III. Preliminary

A. Transformer

It is worth mentioning that Perceiver-Actor uses voxels from multi-view observations to represent objects which is an alternative compared to previous methods that use pointcloud[6], key points [19]. This leads to a question that how to design a network that fits in the memory of modern GPU to encode sequential language and voxel data while reasoning the relation among language tokens and voxels. Transformer[8] is the natural choice for sequential data processing. This network is originally from Natural Language Processing community but has shown huge potential in recent robot learning [6] [20] [21].

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \qquad (1)$$

At the core of Transformer is the dot-product attention function(1). Attention module takes a length-n input and output a sequence of the same length. Each input is linearly mapped to a query, key and value. The output is computed

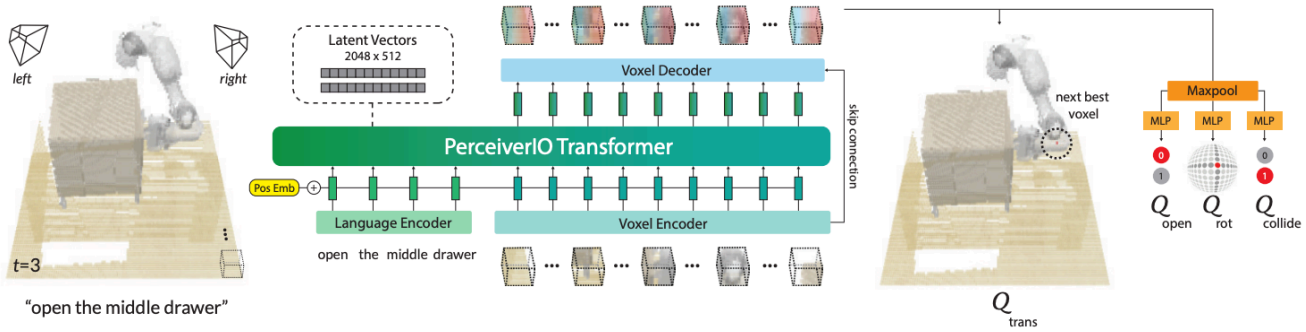

Fig. 1: PerceiverIO Transformer: A behavior-cloning agent to complete multi-task for 6-DoF manipulation. Given space discretized voxels and language instruction and outputs next waypoint gripper translation, orientation and gripper open state.

as a weighted sum of the values. The weight of each value is computed based upon query with its key. We refer readers to the original Transformer paper[8] for more details.

B. Q-Learning

Q-learning is an algorithm in model-free reinforcement learning, i.e. unknown transition function. The formal definition of Q-learning uses Bellman equation, which basically refers to maximize a reward function by random select best actions according to a Q function as follows:

$$Q(s,a) = Q(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

where (s, a) is the current state-action pair, (s', a') is the next state-action pair, α the learning rate and γ the discount factor. The Q function is to measure potential future rewards from being in this states. Q function design will be introduced in Sec. IV.

C. Large Language Model

Large Language Model is used to capture the semantic information in natural language. Since the language model is trained on large text corpora, it outputs word token embedding and serves as pre-processing of language. One large language model is CLIP[22], which is trained with large amounts of image-caption pairs. It serves as a semantic prior for grounding visual concepts like colors, shapes and object categories. For instance, in a sentence "pick up the red cube in the front of the green bottle." CLIP may relate color "red cube" and "green bottle" in the sentence with the corresponding "red cube" and "green bottle" in a image, thereby network pay more attention to the two parts of manipulation interests.

IV. Perceiver-Actor

Perceiver-Actor is a Transformer-based behavior-cloning network. Given space discretized voxels and language instruction and outputs next waypoint gripper translation, orientation and gripper open state. This action is then executed with motion planner to command joints of manipulator until convergence to goal position.

The language instruction l is encoded using pre-trained large language model. In [5], it uses CLIP[22]. Learned position embeddings are incorporated along with language embeddings. For images data, they are firstly used to construct 3D scene as 100^3 and then the voxels v are divided into patches of size 5^3 and finally vectorized into a flattened sequence. However, there is an input of $(100/5)^3$ = 8000 patches which is hard to fit in the memory of modern GPU if we use plain transformer. Instead, Perceiver Transformer [23] is used. This transformer computes the cross-attention between inputs and selects a much smaller size of latents. The original paper used 2048 latents but in our experiment, 512 is used due to smaller memory size of local GPU.

6 self-attention layers are used to encode the latents and output a sequence of patch encodings. Then the output is decoded to 64-dimension voxel features by passing through an upsampling 3D convolutional layers. The skip-connection step is like in U-Net [24]. The voxel features are used to predict actions: translation, orientation, gripper open state, collision state. Q functions are as follows. For translation (x, y, z), the voxel features are reshaped into 100^3 grid scale to form 3D Q-function of actions $Q_{trans}((x, y, z)|v, l)$. Orientation, gripper open state, collision state are predicted by first passing through max-pooling layer to reduce dimension and then MLP linear layer to predict actions. Euler angles (ξ, θ, ϕ) are used to represent orientation. For each state, we choose the action that maximize the Q functions: $Q_{rot}((\xi, \theta, \phi)|v, l)$, $Q_{open}(\omega|v, l)$, $Q_{collide}(\beta|v, l)$.

Perceiver-Actor chooses loss function like cross-entropy as follows:

$$L_{total} = -E_{trans}\left[log(V_{trans})\right] - E_{rot}\left[log(V_{rot})\right]$$
$$-E_{open}\left[log(V_{open})\right] - E_{collide}\left[log(V_{collide})\right]$$

where

$$V_{trans} = \text{softmax}(Q_{trans}((x, y, z)|v, l))$$
$$V_{rot} = \text{softmax}(Q_{rot}((\xi, \theta, \phi)|v, l))$$
$$V_{open} = \text{softmax}(Q_{open}(\omega|v, l))$$
$$V_{collide} = \text{softmax}(Q_{collide}(\beta|v, l)).$$

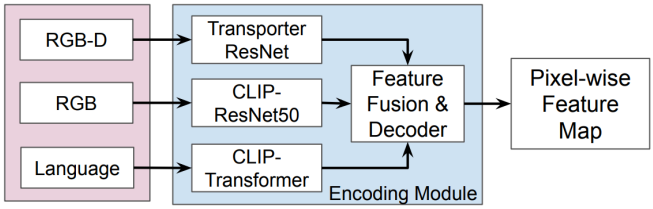

Fig. 2: Encoding module of CLIPort-6D. There are three streams which encode RGB-D images, RGB images and language data separately and then fuse them together. Output is a pixel-wise feature map.

V. CLIPort-6D

To serve as baseline, we introduce CLIPort-6D, a CLIPort-based behavior cloning agent introduced in [3]. This agent takes in multi-view RGB-D images and language goal as Perceiver-Actor and outputs 6 DoF pose waypoints until complishment of work.

The core of this network is the encoding module that fuse visual and language data to obtain a pixel-wise feature map as shown in 2. This feature map serves as a concise scene representation that are used for object detection and grasp pose selection, i.e. higher value areas are more likely to be the parts of our interests. For more architecture details, we refer readers to the original paper[3].

VI. Experiment

The experiment is set up in CoppeliaSim and PyRep. We use open door dataset of 36.8GB that is one task in VLMbench. We use multi-view camera images collected from VLMbench and long language description as input. To reconstruct high-quality voxelized scene representation shown in Fig. 3, we use five cameras in the front of table, at the left shoulder of table, at the right shoulder of table, at the wrist and overhead. The experiment metric is as follows: If the distance error between predicted keyframe translation voxel (q) and ground truth ($\tilde{q}$) is less than a fixed threshold (δ), we say that the experiment is successful. In the experiment, threshold is set to be 5.2 voxel length.

$$e = |q - \tilde{q}| \leq \delta$$

A. Model

Serving as the first try, we test the performance of CLIPort-6D. After observing the poor performance of the baseline algorithm, we test Perceiver-Actor with some minor implementation difference. To be more specific, we use the aforementioned CLIPs [22] language encoder as in Perceiver-Actor. We input language embedding from CLIP and RGB-D voxel observations with a PerceiverIO[23]. In action inference, action output is from Q-learning network composed of Maxpooling layer followed by MLP.

B. Training Protocal

All experiment is done on CoppeliaSim simulator. Different from original Perceiver-actor which uses LAMB[25] optimizer, we use Adam optimizer with 0.00001 learning rate for multiple epochs. This is because we notice the phenomenon of vanishing gradient to use Lamb. The batch size is set to be 1 to fit the model on a single local NVIDIA 3080ti GPU. We train the Perceiver-Actor for 1-4 epochs for 30-100 minutes. The voxel size is set to be 100^3 so that the total number of voxels in the space is 1 million. We use 512 latents in PerceiverIO.

TABLE I: Performance of CLIPort-6D in multiple tasks

task	door	drawer	place	pour water	wipe table
success rate	0.01	0.09	0.036	0.004	0.026

VII. Results

A. Baseline

The first experiment is to test five tasks on VLMbench using baseline algorithm CLIPort-6D. As shown in Table. I, among all five tasks, only opening drawer scores success rate more than 5% while pouring water and opening door have less than or equal to 1% success rate. The poor result may due to 3D cloud points projection onto 2D plane which lose part of geometric information. This experiment motivates us to explore other algorithms like Perceiver-Actor, which preserves 3D information by using 3D voxels.

B. Comparison with Zero-Shot Model

In Fig. 3, we demonstrate the performance comparison between zero-shot model and trained model. As we can see in the leftmost figure, red voxels are the predicted translation and the blue voxel is the ground truth value. The turquoise gripper is the visualization of the predicted orientation of the end effector. We can see that the predicted translation and orientation is far from ground truth. In contrast, the right 4 figures from the same sequence demonstrate the successful prediction. In this demo, the end effector gets close to the door, grasp the door gripper and slightly open the door. The predicted translations and orientations are close to ground truth, which shows the effect of the model.

C. Comparison Through Epochs

We compare the performance of convergence using Perceiver-Actor as the number of epochs goes up in Fig. 4. We can observe an increasing trend with a convergence to 52.23% after 2 epochs. Notice that the number of epochs range from 0 to 3. Notice the number of epochs is small when in witness of convergence. This may due to the fact that open door task in VLMbench lacks diversity. It may also due to lack of generalization of Perceiver-Actor.

VIII. Discussion

In this project, we test the performance of CLIPort-6D. After fail case analysis, we choose Perceiver-Actor to finish vision and language tasks. After re-implementing the code, we also test it on VLMbench which tests the robustness of the framework. The voxel representation of visual scene understanding is quite inefficient and expensive in terms of resource consumption. As shown in Fig. 5, we count the time for training alone. With batch size 1 and only 3 epochs, it already takes around 2 hours for training solely on opening door dataset for one NVIDIA 3080ti GPU. Actually, in the

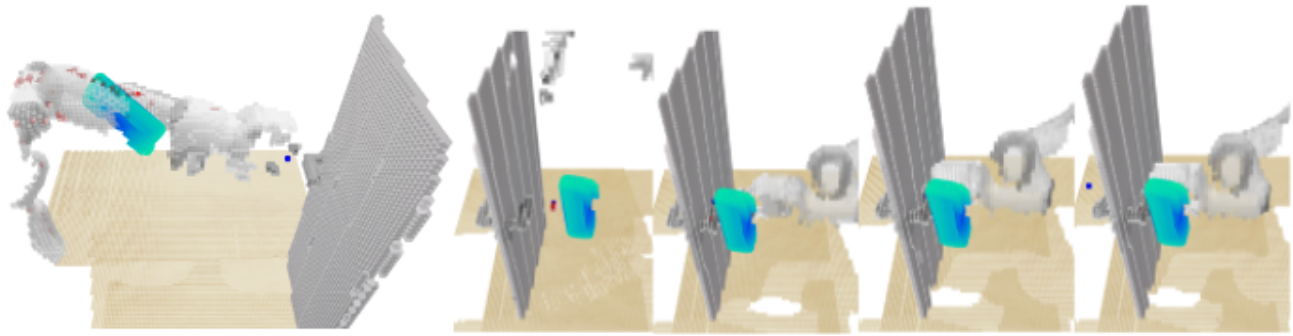

Fig. 3: Performance comparison between zero-shot model and trained model. Leftmost figure is the predicted keyframe for zero-shot model. 4 figures on the right are the predicted waypoints in a single play. Red voxels denote the predicted next step translation. The turquoise gripper (without fingers) demonstrates the predicted orientation.

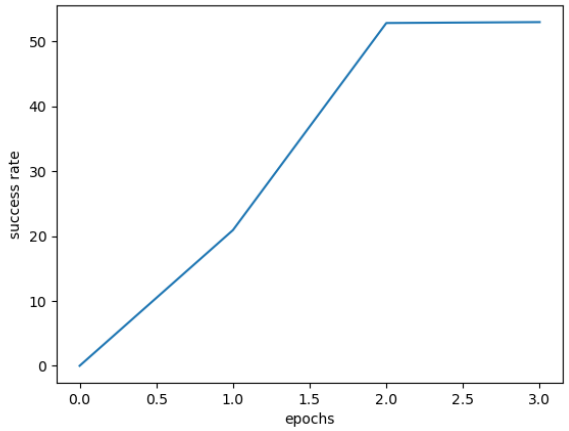

Fig. 4: The success rate against number of epochs. 50% success rate is achieved in opening door dataset.

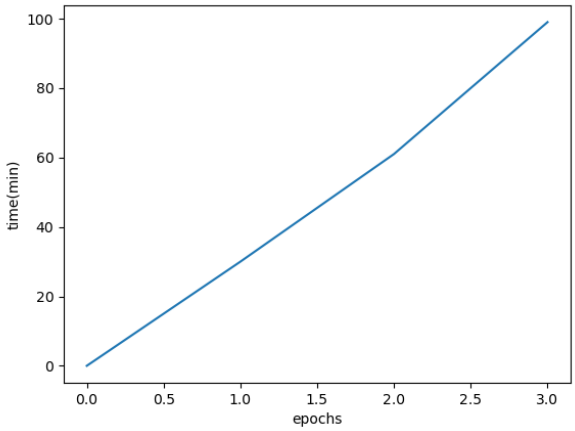

Fig. 5: The success rate against number of epochs. It takes 100 minutes to train the network with batch size 1 for 3 epochs.

original paper, it takes 16 days for training with batch size 16 on 8 NVIDIA V100 GPUs. Moreover, in testing step, it inference at the frequency of 1Hz. The long training time and low testing frequency is both due to large number of voxel inputs and thereby large computation.

IX. Future Work

One issue with the current framework is that it only predicts several discretized wayframes along the path. As a result, it is not applicable to dynamic tasks that require real-time closed-loop reaction. In the field of dynamic manipulation, the manipulator should detect moving objects and interact with the dynamic scene in a short time. Due to the nature of transformer architecture, action sequence can be generated instead of only one action. Also, the network may be used to predict velocity of objects in real time as mentioned in Perceiver-Actor. Considering these two facts, we are planning to extend the current work to finish grasp task in dynamic environment. There are a lot successful works in this field we can refer to. [26] learns residual velocity to compensate the uncertainty of a physics-based controller (known dynamic model like gravitational field). [27] predicts a predefined vector flow for articulation objects from vision and selects the grasp point such that its vector flow with highest magnitude. To grasp moving objects, a closed-loop visual feedback controller for dynamic environment features [28]. After the best grasp pose has been predicted, a velocity command is computed to make gripper converge to the target pose. [29] also formulates dynamic grasp as a closed-loop policy generation problem. In contrast to [28], [29] formulates action policy generation as a Markov Decision Process. An object-centric approach by [30] uses RNN to predict future object pose sequence from a sequence of instantaneous poses. [31] formulates the dynamic problem as a "move-and-grasp" game and use adversarial reinforcement learning to train the grasping policy and object moving strategies jointly. However, semantic inferences using language instruction are not included in the learning process [31] [28] [29] [30]. The Perceiver-Actor model may infer the object dynamic model using language instructions and output a sequence of actions to grasp the object. For instance, for a sentence "Pick the red cube circulating the green plate". Language implies the circular trajectory of the red cube and hence policy generator and output actions to capture the red cube.

REFERENCES

[1] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[2] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," *arXiv preprint arXiv:2209.05451*, 2022.

[3] K. Zheng, X. Chen, O. C. Jenkins, and X. E. Wang, "Vlmbench: A compositional benchmark for vision-and-language manipulation," *arXiv preprint arXiv:2206.08522*, 2022.

[4] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[5] ——, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[6] W. Liu, C. Paxton, T. Hermans, and D. Fox, "Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6322–6329.

[7] W. Liu, T. Hermans, S. Chernova, and C. Paxton, "Structdiffusion: Object-centric diffusion for semantic rearrangement of novel objects," *arXiv preprint arXiv:2211.04604*, 2022.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[9] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[11] K. Tunyasuvunakool, J. Adler, Z. Wu, T. Green, M. Zielinski, A. ídek, A. Bridgland, A. Cowie, C. Meyer, A. Laydon, S. Velankar, G. J. Kleywegt, A. Bateman, R. Evans, A. Pritzel, M. Figurnov, O. Ronneberger, R. Bates, S. A. A. Kohl, A. Potapenko, A. J. Ballard, B. Romera-Paredes, S. Nikolov, R. Jain, E. Clancy, D. Reiman, S. Petersen, A. W. Senior, K. Kavukcuoglu, E. Birney, P. Kohli, J. Jumper, and D. Hassabis, "Highly accurate protein structure prediction for the human proteome," *Nature*, vol. 596, no. 7873, p. 590596, August 2021. [Online]. Available: https://europepmc.org/articles/PMC8387240

[12] H. Kim, Y. Ohmura, and Y. Kuniyoshi, "Transformer-based deep imitation learning for dual-arm robot manipulation," *CoRR*, vol. abs/2108.00385, 2021. [Online]. Available: https://arxiv.org/abs/2108.00385

[13] A. Jaegle, S. Borgeaud, J. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, O. J. Hénaff, M. M. Botvinick, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver IO: A general architecture for structured inputs & outputs," *CoRR*, vol. abs/2107.14795, 2021. [Online]. Available: https://arxiv.org/abs/2107.14795

[14] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," *CoRR*, vol. abs/1707.02920, 2017. [Online]. Available: http://arxiv.org/abs/1707.02920

[15] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *CoRR*, vol. abs/1806.10293, 2018. [Online]. Available: http://arxiv.org/abs/1806.10293

[16] C.-H. Min and J.-B. Song, "End-to-end robot manipulation using demonstration-guided goal strategie," in *2019 16th International Conference on Ubiquitous Robots (UR)*, 2019, pp. 159–164.

[17] N. Hudson, T. Howard, J. Ma, A. Jain, M. Bajracharya, S. Myint, C. Kuo, L. Matthies, P. Backes, P. Hebert, T. Fuchs, and J. Burdick, "End-to-end dexterous manipulation with deliberate interactive estimation," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2371–2378.

[18] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *CoRR*, vol. abs/1504.00702, 2015. [Online]. Available: http://arxiv.org/abs/1504.00702

[19] L. Manuelli, Y. Li, P. Florence, and R. Tedrake, "Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning," *arXiv preprint arXiv:2009.05085*, 2020.

[20] C. He, R. Li, S. Li, and L. Zhang, "Voxel set transformer: A set-to-set approach to 3d object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8417–8427.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[22] 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 *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[23] 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 preprint arXiv:2107.14795*, 2021.

[24] 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.

[25] 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 preprint arXiv:1904.00962*, 2019.

[26] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[27] B. Eisner, H. Zhang, and D. Held, "Flowbot3d: Learning 3d articulation flow to manipulate articulated objects," *arXiv preprint arXiv:2205.04382*, 2022.

[28] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The International journal of robotics research*, vol. 39, no. 2-3, pp. 183–201, 2020.

[29] S. Song, A. Zeng, J. Lee, and T. Funkhouser, "Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4978–4985, 2020.

[30] I. Akinola, J. Xu, S. Song, and P. K. Allen, "Dynamic grasping with reachability and motion awareness," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9422–9429.

[31] T. Wu, F. Zhong, Y. Geng, H. Wang, Y. Zhu, Y. Wang, and H. Dong, "Grasparl: Dynamic grasping via adversarial reinforcement learning," *arXiv preprint arXiv:2203.02119*, 2022.