CLIPort with Safety Constraints

Sai Deepak Mutta
Robotics and Autonomous System (Artificial Intelligence)
Arizona State Univeristy
vmutta@asu.edu

Dheeraj Kallakuri
Robotics and Autonomous System (Artificial Intelligence)
Arizona State Univeristy
dkallak1@asu.edu

Sai Manikanta Badiga
Robotics and Autonomous System (Artificial Intelligence)
Arizona State Univeristy
sbadiga@asu.edu

Raghavendra Dinesh
Robotics and Autonomous System (Artificial Intelligence)
Arizona State Univeristy
rdinesh2@asu.edu

1 Introduction

The work focuses on building an end-to-end system that is capable of taking a linguistic goal and performing it in a real-world scenario. For building such a system we require a model that is aware of the spatial distribution of objects in its targeted environment and be able to link it to the linguistic target. The problem becomes exceedingly complex considering the variability of shapes, sizes, and colors of targeted objects and the reachabilities and positions of their end poses. To execute this plan, we wanted to employ the CLEVR [1] dataset and Transporter Networks to process images and translate them into end-effector poses. We became aware of the complexity involved in handling objects of varying shapes, sizes, and colors along with their end poses, but we had planned to overcome it. Our objective was to generalize the system to handle different daily objects so that we could avoid the need for predefined object classes.

2 Related Work

Referring to the latest advancements in this field, we chose CLIPort [4] [3] as a reference. CLIPort is a language-conditioned imitation-learning agent that integrates the semantic understanding of CLIP by OpenAI with the spatial precision of Transporter by Google. It is a two-stream architecture that uses data from CLIP to identify "what" and transPORTer[6] [5] networks to identify the "WHERE" parts of the goal. CLIP is a pre-trained model by OpenAI that is capable of combining visual and textual features. It has been trained on a large dataset of image-question pairs and is capable of predicting the probabilities for the most reasonable label for a given image. In our case, we use it to extract the word embeddings to identify the target region. Instead of using visual image features, Transporter networks use spatial symmetry features to identify the poses. It creates an attention map and divides the field into key and query, which are then convolved over the field to identify the required end

Preprint. Under review.

pose. Using convolution gives us the capability to identify the end pose in all possible orientations irrespective of the initial object pose. The images are passed down to transporter networks which use the spatial and color data from RGB-D images and give us the required region of interest.

CLIPort uses a two-stream architecture, comprising the Semantic stream and the Spatial stream. The Semantic stream uses a frozen CLIPResNet50 to encode RGB input, and the decoder layers are conditioned with language features from the CLIP sentence encoder. The Spatial stream encodes RGB-D input, and its decoder layers are laterally fused with the Semantic stream. The final output is a map of dense pixel-wise features used for pick or place affordance predictions. This same two-stream architecture is employed in all three Fully Convolutional Networks (fpick, ϕquery, ϕkey). The fpick network is used to predict pick actions, while ϕquery and ϕkey networks predict place actions. CLIPort employs ReLU activations after each convolution and identity block without any Batch Normalization. Note that we repeat the depth input to match the dimensions of the RGB image RHxWx1$\rightarrow$RHxWx3 following Transporter. CLIPort is implemented in PyTorch, using the pre-trained checkpoint released by the authors for CLIP. Here is the image of the architecture of CLIPort.

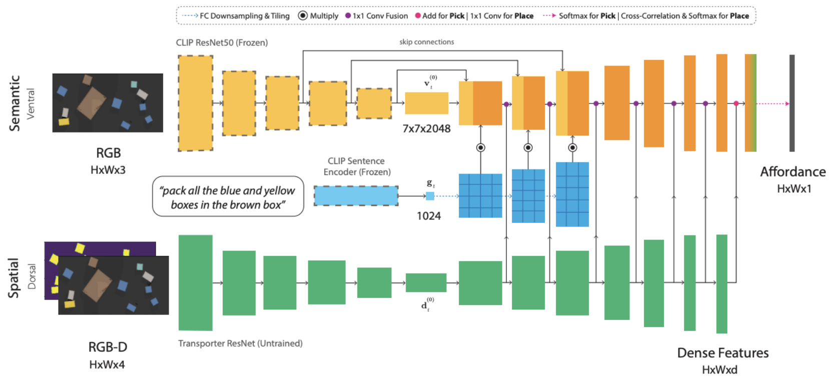


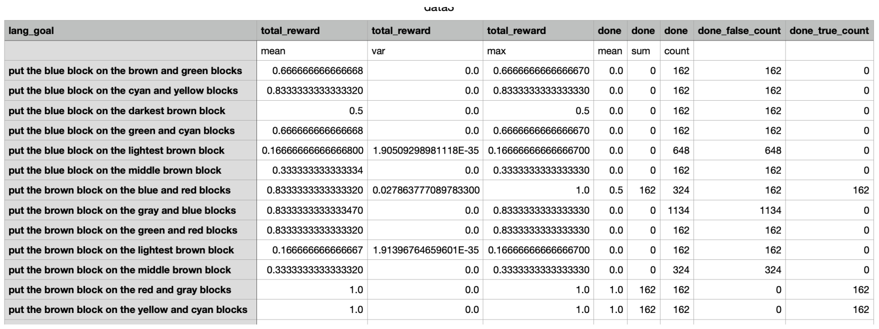
Figure 1: CLIPort Architecture

3 Baseline Results

Considering the requirement of heavy computational resources we decided to run it on the agave cluster that provided us with high-end GPUs and a greater amount of RAM that was required to get the model running. The simulation of CLIPort has been run over 150 times and the results have been recorded. Each time the model ran we were able to record the linguistic goal and the reward awarded for the step. Overall we were able to record about 11,000 observations and then perform analysis on the samples. Each execution of the demo consists of 10 episodes and each episode runs for up to a max of 10 steps or till the task is fulfilled. The below screenshot shows us a few entries in the results. Here we are considering all the different language goals received in each demo and extracting the success factor for each test case and the average reward for all the successful samples. We also created a performance metric where we calculated the ratio of successful episodes to the total number of episodes and we got about 80%.

4 Proposed Solution

The idea of the work is to improve the scores of the given baseline and be able to implement it in real-time. Specifically, the CLIPort model could provide pathways for the robot to manipulate objects based on both visual and linguistic input. The task "avoid-region" involves picking up a green block and placing it in a pink square region while avoiding a blue forbidden region. The image below (Figure 3) gives an idea about the environment in which the arm with perform the required task. However, the robot failed to follow this language goal because the Transporter Network, which provides the two points Tpick and Tplace, could not take into account the forbidden region.



lang_goal	total_reward	total_reward	total_reward	done	done	done	done_false_count	done_true_count
	mean	var	max	mean	sum	count		
put the blue block on the brown and green blocks	0.666666666666668	0.0	0.666666666666670	0.0	0	162	162	0
put the blue block on the cyan and yellow blocks	0.833333333333320	0.0	0.833333333333330	0.0	0	162	162	0
put the blue block on the darkest brown block	0.5	0.0	0.5	0.0	0	162	162	0
put the blue block on the green and cyan blocks	0.666666666666668	0.0	0.666666666666670	0.0	0	162	162	0
put the blue block on the lightest brown block	0.166666666666666800	1.90509298981118E-35	0.166666666666666700	0.0	0	648	648	0
put the blue block on the middle brown block	0.333333333333334	0.0	0.333333333333330	0.0	0	162	162	0
put the brown block on the blue and red blocks	0.833333333333320	0.027863777089783300	1.0	0.5	162	324	162	162
put the brown block on the gray and blue blocks	0.833333333333470	0.0	0.833333333333330	0.0	0	1134	1134	0
put the brown block on the green and red blocks	0.833333333333320	0.0	0.833333333333330	0.0	0	162	162	0
put the brown block on the lightest brown block	0.166666666666667	1.91396764659601E-35	0.166666666666666700	0.0	0	162	162	0
put the brown block on the middle brown block	0.333333333333320	0.0	0.333333333333330	0.0	0	324	324	0
put the brown block on the red and gray blocks	1.0	0.0	1.0	1.0	162	162	0	162
put the brown block on the yellow and cyan blocks	1.0	0.0	1.0	1.0	162	162	0	162

Figure 2: Results

Specifically, the network could only identify the best pick-up point (Tpick) and the best place point (Tplace) based on the current state of the environment.

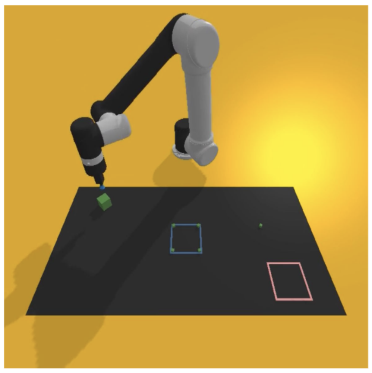


Figure 3: Task environment

To address this limitation of the Transporter Network and why the Transporter network failed, it can be understood by observing the following equation: Tpick = argmax (u,v) Qpick((u, v)|γt), Tplace = argmax$\Delta\tau$ Qplace($\Delta\tau$ |γt, Tpick). Where Qplace($\Delta\tau$ |γt, Tpick) is Cross-correlation between ϕkey and ϕQuery.

Qplace($\Delta\tau$ |γt, Tpick) = ϕquery(γt[Tpick])*ϕkey(γt)[$\Delta\tau$].

To evaluate the success of the robot in this task, an evaluation matrix was created to calculate the number of trajectory points that passed through the forbidden square region. This metric helps to quantify the extent to which the robot was able to avoid the forbidden region while completing the task.

The solution to avoid looks like a mathematical problem. The shortest distance between the initial and final points is a straight line. We were assuming the area that has to be avoided to be a polygon. In our case, we working with a square-type restricted area. We took the intersection point between the square and the straight line by using the Liang- Barsky[2] clipping algorithm and for the following case, there are about 25 possible trajectories (Figure 4). By using the initial intersection point a fixed offset is given in either x or y-axis to get the intermediate point so that the arm avoids the restricted area while performing the task.

To evaluate the task, we created a basic metric that tracks the end effector and identifies the number of steps it has taken through the obstacle region. Our goal is to minimize the number of steps taken through the region. The following graph(Figure 5) explains the plot between the number of episodes and the number of trajectory points through the obstacle region for 100 episodes.

5 Future work

Add a pathway to focus on the region that is needed to be avoided. Change the reward formulation in such a way that, it is awarded at each instance, instead of being awarded at the end. In this way, a negative reward can be awarded for every instance that passes through the obstacle region. This helps us to train it like a reinforcement learning model that can avoid a certain region based on the goal. We would also like to create a GUI for using the system and convert it into an application instead of a terminal implementation. After having all the above things in place we would like to implement it on a real-time robot like a turtle bot or a similar mobile robot that could perform the goal.

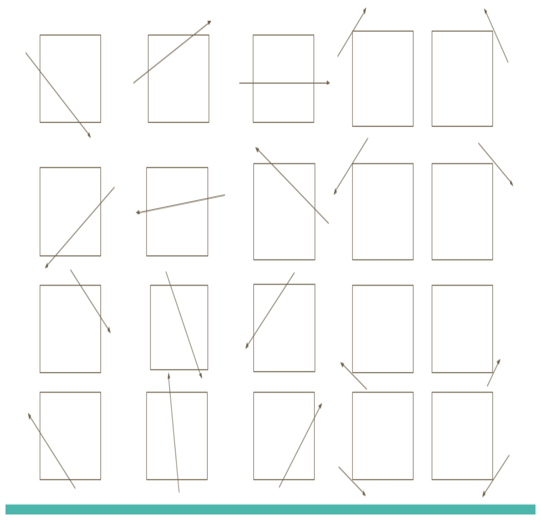


Figure 4: Possible trajectories

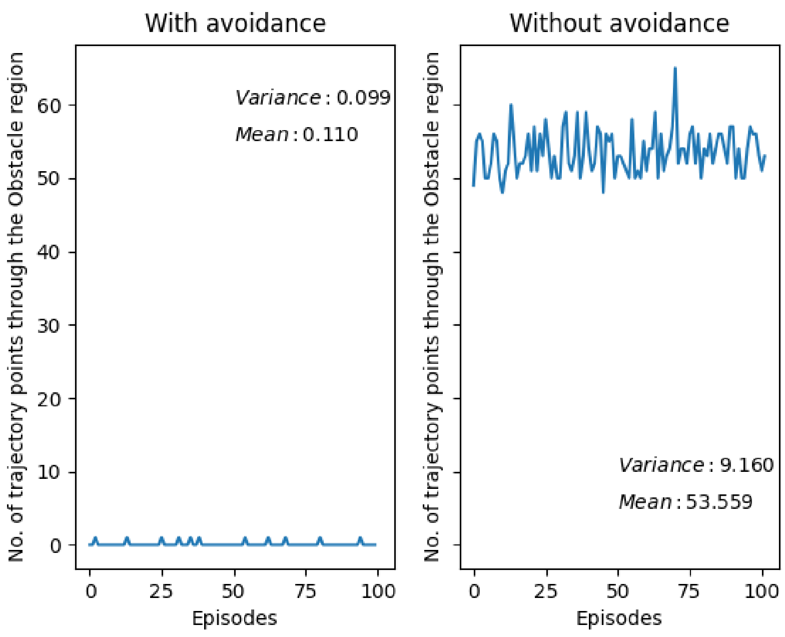


Figure 5: Metrics comparison

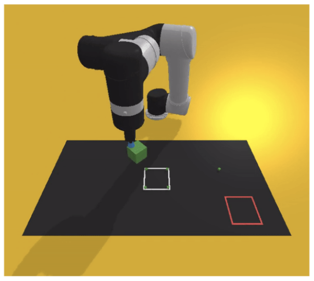


Figure 6: Arm passing through the restricted area

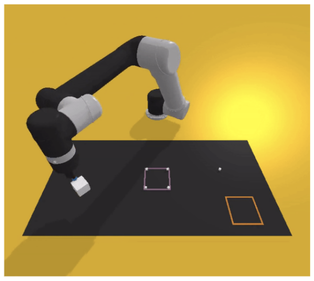


Figure 7: Arm avoiding the restricted area

References

[1] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016.

[2] You-Dong Liang and B. A. Barsky. A new concept and method for line clipping. *ACM Trans. Graph.*, 3(1):1–22, jan 1984.

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[4] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[5] Elias Stengel-Eskin, Andrew Hundt, Zhuohong He, Aditya Murali, Nakul Gopalan, Matthew Gombolay, and Gregory Hager. Guiding multi-step rearrangement tasks with natural language instructions. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1486–1501. PMLR, 08–11 Nov 2022.

[6] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, 2020.