Spatial Language and Computer Vision based Object Search using DeepRL

Exiled Alliance: Anirudha Paul, Donglin Liu
Brown University

Abstract

When humans describe the position of an object in the real world, they tend to describe it in relation to the other object present near it. We rarely use a specific coordinate system to define the location of anything. However, this spatial language can be ambiguous, misleading, and subjective. But because we have a good perception of the surrounding world and inference capability, this kind of stochastic instruction does not become a problem. But it becomes a seriously challenging task when we expect the same behavior from an artificially intelligent system. In this paper, we are presenting a neural-network-based reinforcement learning architecture to achieve similar behavior in an AI agent. First, this architecture uses Spatial Language Object-Oriented POMDP to narrow the search space. Then it uses computer vision to make a good state representation of its surroundings to learn the optimum strategies for navigation based on spatial language. To test the effectiveness of this method, we deployed the proposed method in AirSim. In this realistic drone simulator, we gave language commands to a drone to find multiple connected objects in an unreal engine environment.

1. Introduction

Let's consider a real-world scenario. Suppose I am instructing a delivery agent that - "there is a truck in front of the Baja on Thayer street. I am sitting in the red car opposite of that truck. Deliver the package there."

For a human delivery agent, the instruction is pretty clear and straightforward. First, he will look at the google map to locate Baja and Drive there. Then find the truck in front of it using oculus observation. Then look the opposite side of it to locate my red car. If we can introduce the capability of inferring all the necessary action from just this linguistic instruction in an AI agent, it can interface with humans more intrinsically and search for objects more efficiently.

This problem is challenging because humans construct spatial language instruction based on their observation and prior experience of navigating the real world based on combining their observation with language structure and taking logical action. Yet, none of this knowledge of taking logical actions based on language instruction is available to the robot. In addition, the AI agent must generalize its strategy to take action based on its understanding of spatial language across the different layouts of the surroundings and must adjust to the stochasticity of observation.

That leads to breaking down the problem into two different sub-problems. The first step is to look at the annotated map and the instruction and check if the language can be linked to any landmark appearing on the map. Then make a distribution of beliefs over the map about the probable reference place to narrow down the actual search space significantly and move the agent there. For this part, we've decided to use an already established method called Spatial Language Object-Oriented POMDP (SLOOP). [6]

This SLOOP is a probabilistic observation model for spatial language which can interpret vague, context-dependent prepositions (e.g., front), calculate the frame of reference on the annotated map, and compute an online POMDP planner based on Monte Carlo Tree Search. This model gives a satisfactory solution for the first part of our object search problem - which is to narrow down search space over the whole map.

The second part needs to use vision to connect the information appearing in the language instruction with the real world and find an optimum policy to navigate the narrowed search space and pinpoint the final search object.

As this part is all about learning the optimum strategy for navigation based on the observation, it perfectly fits in the framework of Reinforcement Learning. In reinforcement learning, the agent observes the state of the real world and takes action based on the optimum learned policy given the state of the world. As the state, transition and reward everything is stochastic in our problem, we've decided to go with Deep Neural Network, specifically REINFORCE with baseline architecture to learn this policy.

We have developed an end-to-end solution for object search based on spatial language instruction by combining solutions for these two parts of the problem. Moreover, because our agent is learning optimum strategy in a stochastic environment, the model is adaptable to different situations, making it viable for practical use.

2. Related Work

Most prior works for object search based on spatial language grounding assume that the agent has access to the fully observable domain [4], [1], where the key objects have known coordinates or are within the field of view of the agent, and the task is generating navigation behavior according to the provided instruction.

Recent researches aim to connect these natural language instructions straight to low-level controls or generate motion planning using deep reinforcement learning or imitation learning. These methods require large datasets of instructions paired with demonstrations. [2]

Spatial language understanding in partially observable domains is a developing study area . Thomason et al. propose a method where the AI agent was tasked to reach a goal room, and the agent can discuss the goal's location during the task execution with an oracle. [5]

So our work targets to remove some of the limitations presented in these prior works. We are mainly focusing on developing a system where the agent can adapt to the stochasticity of the real world without relying on the information that can not be obtained in the run time.

3. Method

Given spatial language instruction, the object search process can be divided into two major stages.

3.1. Navigate using the map

For this task, we have used the vanilla architecture of SLOOP. In this architecture, first, we need to annotate the map with name and color-coding just like figure 1.

Then the model converts this color-coded annotated map to bitmap representation and feeds it through a convolutional neural network so that given a landmark name associated spatial key, it can estimate the frame of reference on the map.[Architecture: 2]

After guessing the frame of reference, it uses a deterministic function to make a probability of belief about the referred location over the map as seen here in figure 3.

And finally, it uses this belief to make planning using Partially Observable Markov Decision Processes. With this planning, the agent moves to the location with high belief. This belief also helps the agent to reduce the search space.

3.2. Navigate without the map

In this part, the search becomes easier as the search space gets narrowed down after the first step. But the problem is the model can no longer rely on the map as the object it is searching for is not annotated on the map.

So the agent now starts to rely on computer vision for this stage of the searching process.

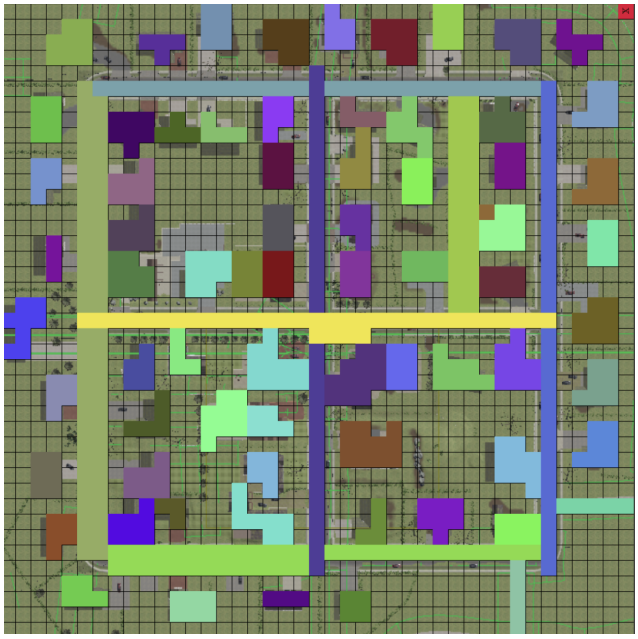

Figure 1. Color coded landmarks on annotated map

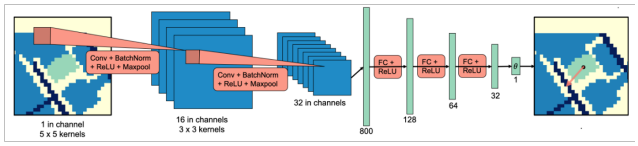

Figure 2. Frame of reference prediction using CNN

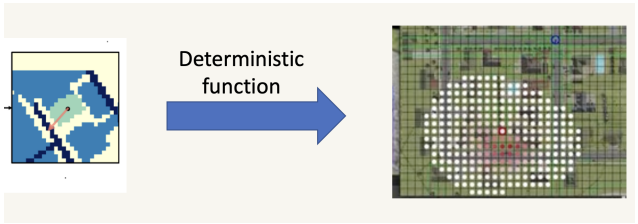

Figure 3. Distribution of belief

To achieve this at first, we've used YoloV3 [3] to detect all the objects appearing from the agent's viewpoint. This object detection system gives us the coordinate, height, and width of the detected object with respect to the image frame as seen in figure 4. Then we marked which things appeared in the language and what is the positional information (for example - front, back, etc.) associated with that object.

We combined this positional information and linguistic representation in the spatial information to make the state representation for reinforcement learning.

A sample state representation can be seen in the figure 5. We have also added the current co ordinate of the agent along with this vision based state representation.

After designing a reasonable state representation that contains all the necessary information for the agent, the next

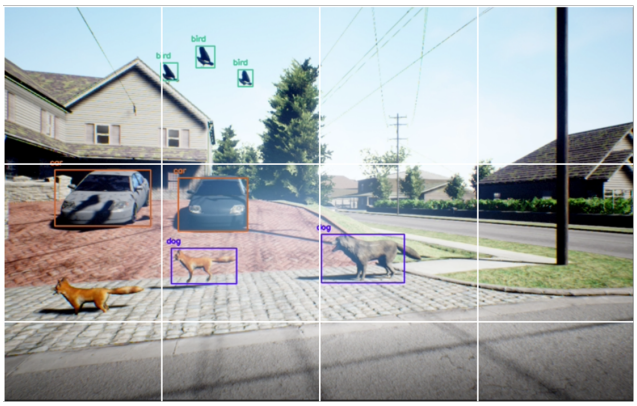

Figure 4. Detected objects using YoloV3

Object	Language Positional Encode	x	y	w	h	c
Stop Sign	0 (Not present)	-50	-50	0	0	0
Bird	5 (Goal)	12	5	7	3	75
Car	1 (Top)	6	12	161	51	67
Dog	3(Behind)	56	123	32	10	63

Figure 5. Sample state representation for DeepRL

goal was to make a rational reward model for the AI agent in order to give it an incentive to learn optimum policy. As the project of this scope is limited - we hardcoded the reward system. If the agent detected a nongoal object appeared in the language in its viewpoint, it was given a reward of 1. For the final goal object, the reward was 10. To incentify moving towards the goal, this reward was multiplied by the height and width of the object in the image frame. The reason is the closer the agent gets to the object, the height and width of the object appearing in the image frame would be bigger, resulting in higher reward gain.

Now we have a reasonable state representation and reward model. So with these two components in place, we designed a deep neural network with REINFORCE with baseline (Actor-Critic model) architecture and fed state representation. The output of this network is a probability distribution over the action representing how good each action might be given the state. In our case, there were three actions - move front, rotate left, and rotate right. After sampling from this distribution and taking the step, we took gradient descent based on the following state and the reward gained from it. We frequently updated the deep neural network weights until the agent could get the most reward with minimum steps. We limited the agent to take no more than 20 actions for training purposes. If it could not reach the final goal within 20 moves, we updated the model with the current reward and restarted the episode.

Setup	Episodes needed for first convergence
One object	3
Two objects	12

Table 1. Number of episode needed before successfully reaching the goal

Setup	Successrate after first convergence
One object	91%
Two objects	87%

Table 2. Successrate in the next 100 episodes after reaching the goal for the first time

The whole architecture of learning optimum policy can be seen in figure 6.

4. Results

For testing our model, we have set up a neighborhood map in Unreal Engine 4, which simulates a suburb area. Then, we deployed Microsoft Airsim on top of it to simulate the drone as an AI agent. As the whole setup was complex and real-time simulation took a much longer time to complete, we had to restrict the environment to make the training faster. We only let the agent take 20 steps in each episode before ending it, and we only run for 100 episodes for each snapshot of the model. And we only tested on one or two reward objects present in the scene.

With this setup, we ran our model and observed its behavior. First, we checked the number of episodes needed for the first time to the end goal for both one and two objects. The result can be found in table 1.

You can watch the demonstrations for one object and two objects search on YouTube.

Then after the first convergence, for the next 100 episodes, we measured the success rate of completing the goal to see whether the model actually learned an optimal policy. The result can be found in table 2.

With this result, it was pretty apparent that the agent learned a policy to navigate toward relevant objects. So in that aspect, the project is successful. But because of limited test attempts, we can not conclude whether it can actually understand the spatial cue like the - front, behind, etc. For this, we need more rounds of testing with more versatile layouts.

4.1. Technical Discussion

Our model raises several questions, especially in terms of interpretability. For example, how can we make sure that our model actually understands the language, especially its spatial structure? And if it takes the right step without un-

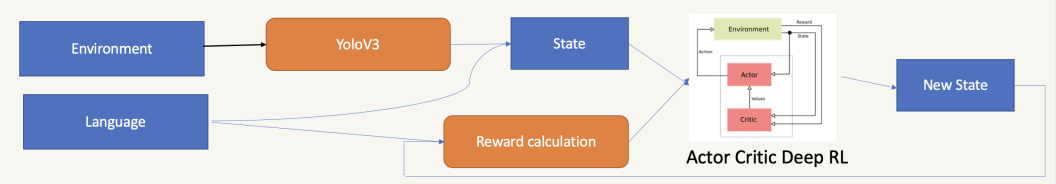

Figure 6. Our Deep Reinforcement Learning Architecture

derstanding the intrinsic meaning of the language, does it really matter? These are the questions we would like to look forward.

There are some trade-offs of this model. Though it presents an end-to-end solution for object search problems, a huge portion of it depends on external factors. For example - object detection relies on the YoloV3 model, and map-based planning relies on SLOOP. So any imperfection of those models carries forward to our solution. Another thing is we can not train the model for millions of episodes, unlike other deep learning projects, which can reduce the confidence of this model for practical use.

But in the end, if we consider language as the reward and computer vision as the state, it can lead to pretty exciting results. And all of our changes are done to prove this point.

4.2. Societal Discussion

Any intelligent system comes with its own social concern. Our model is no different.

For example, one concern was raised - if big companies leverage their resources to scale this automated system to streamline their delivery system and wipe out small competitors. Our counterargument is - if this system is embedded in every consumer drone, anyone, including small businesses, can buy drones according to their capability and scale their delivery system. Because currently, the cost of keeping a dedicated delivery man is too high, and Uber Eats and DoorDash kind of services take a big chunk as commission. With automated AI, the small businesses will have full control over the quality and revenue of their delivery system.

The second concern was the privacy of the collected huge amount of video feed. First of all, it is not feasible for any company to store those data and the second thing is drones will be operated in public spaces. They will only be looking for objects by recognizing the surrounding landmark and objects. We don't see any reason to be concerned about collecting facial or other sensitive information.

The third concern was whether the knowledge learned from airsim is transferrable to the real world. It is hard to tell at this point, but given the AirSim environment is stochastic, so any policy learned in that environment should be generalizable. As a result, it should be transferrable to the real world in theory.

The fourth concern was about no-fly zones. Again, with the current state of GPS technology, those situations can be easily avoided.

The fifth concern was about the responsibility taker for the unintended event. As this is a legal issue, it should be judged case by case in the legal framework we currently have in place.

The sixth concern was usability and user-friendliness for end users. Those mainly depend on industrial design, which is out of this project's scope. We are only focusing on developing a generalized strategy for object search.

Another concern was delivering to the wrong address. Our model uses a map for the first step. So given the map is accurate and GPS is good, this kind of issue should not happen.

The final concern was about night time situation. As this is a hardware issue, we are not addressing this concern here. In the nighttime, our model will continue to work like daytime if any object recognition system is developed for nighttime.

5. Conclusion

In this paper, we have presented a way to combine spatial language and vision data to make a good state representation of the real world which we've successfully used to train an AI agent to learn a reasonable policy for finding objects. It paves the way to naturally interact with robots and make them work more efficiently.

Moving forward, we will focus more on its language component, especially more research is needed to automate the process of reward shaping using natural language processing.

References

[1] Juan Fasola and Maja J Matarić. Using spatial semantic and pragmatic fields to interpret natural language pick-and-place instructions for a mobile service robot. In *Social Robotics*, Lecture notes in computer science, pages 501–510. Springer International Publishing, Cham, 2013. 2

[2] Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay on the path: Instruction fidelity in Vision-and-Language navigation. May 2019. 2

[3] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. 2

[4] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy.

Approaching the symbol grounding problem with probabilistic graphical models. *AI Mag.*, 32(4):64–76, December 2011. 2

[5] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 394–406. PMLR, 30 Oct–01 Nov 2020. 2

[6] Kaiyu Zheng, Deniz Bayazit, Rebecca Mathew, Ellie Pavlick, and Stefanie Tellex. Spatial language understanding for object search in partially observed city-scale environments. In *2021 30th IEEE International Conference on Robot Human Interactive Communication (RO-MAN)*, pages 315–322, 2021. 1

Appendix

Team contributions

Anirudha Paul has done the work to set up the project environment, re-implement the SLOOP model, design the deep reinforcement learning architecture, and made the literatures to present the whole project.

Donglin Liu has worked to make helper functions for the simulated environment so that in training and testing time, the AI model can call those APIs to take actions and observe its impact in that virtual world.