Project Report : Visual Embodied Navigation without Navigation Sensors (Habitat Challenge 2020 - PointNav)

Akshay Krishnan
akrishnan86@gatech.edu

Mason Lilly
jmason.lilly@gatech.edu

Prateek Vishal
prateekvishal26@gatech.edu

Haard Shah
haard@gatech.edu

Abstract

In this project, we propose a deep-learning based approach to solve the problem of visual Embodied Navigation in an indoor environment. We train a model in simulation to predict the pose of a navigational agent which can be fed to a policy that controls the motion of the agent. Unlike recent methods that train navigational agents in simulation, we do not assume the availability of noiseless sensors and actuators or navigational sensors like GPS/Compass. We evaluate our approach by comparing the performance of an agent that uses the predicted pose as opposed to other existing methods that use navigational sensors.

1. Introduction

The problem of embodied navigation aims to build agents that can autonomously navigate from one point to another in indoor environments. It is also referred to as the PointGoal or PointNav task. This problem has had a long history in research. Classically, it has been approached by geometric methods that mathematically model the dynamics of agents and the geometry of the scene. One such example is to track the position of the agent in a 3D map of the scene using a filtering algorithm like Kalman filtering, a planner, and a low level controller to guide the robot to the goal. While these methods are intuitive and computationally inexpensive, they are not robust to changes in the environment like illumination and noise in the sensors and actuators of the robot. Recent developments in the application of deep models trained using supervised learning for semantic perception and reinforcement learning for control has motivated researchers to solve the PointGoal task using deep models. These deep learning based approaches have been shown to generalize better to unseen environments and do not require a 3D map. The authors in [1] formalized this area of research and defined the PointGoal task as the task of navigating to a goal location with respect to the initial position of the agent.

The Habitat Embodied Navigaiton Challenge's PointNav task has accelerated research in this area. In this task, an agent is spawned in a random location in a scene and is required to navigate to a goal location using only RGB-D data. The agent can take 4 actions - move forward, turn left, turn right or stop. The task in 2019 asked participants to submit agents that can navigate to a PointGoal using noiseless RGBD sensors and noiseless actuators. One approach that performed well on the challenge used hierarchical deep models, factoring the problem into mapping, pose estimation and planning blocks [2]. Each of these blocks was trained using independent loss functions but they were trained jointly since their data distributions are inter-dependent. Another approach uses a visual encoder to learn a representation that is fed into a policy along with the PointGoal. The whole model could either be trained end-to-end [6] or using auxiliary loss functions [3]

A shortcoming of existing approaches is the failure of models trained in simulation to generalize to real world environments [4]. The Habitat Challenge 2020 aims to address this fact by introducing noise in sensing and actuation and removing the GPS sensor. This adds extra realism to the simulation, since it is not practical to have reliable odometry or GPS indoors. The challenge for participants, therefore, is to introduce models that are robust to noise and can navigate without odometry. [2] addresses the first problem using modular deep architectures. They use a pose estimator and mapper that can infer spacial geometry in a way that is robust to observation noise, a local policy that is robust to actuation noise and a global planner that can adapt to errors in perception and the local policy. However, their system still use input from odometry on the agent. We build upon their approach to train a pose estimator without using GPS, compass or odometry. This pose estimator can be used along with the local policy and global policy, or any other deep model that currently solves the PointNav problem using these sensors. We evaluate our pose estimator

using (a) RMS error with Grond truth pose (b) Distance to goal of a policy that uses the pose estimate. We made use of the Gibson Dataset [7]. This dataset contains full video visualizations from indoor 3D spaces. This data is adapted for the PointNav task of the habitat challenge, that provides splits of train, val and test for this dataset.

2. Background - NeuralSLAM

We build upon the approach taken by NeuralSLAM [2] to solve the PointNav task. NeuralSLAM was originally proposed for the task of embodied exploration of an environment, but it is also the current state of the art on the Habitat Embodied Navigation Challenge. We choose this approach as it has performed better than other approaches even in the presence of actuation and sensing noise. NeuralSLAM divides the embodied agent into 3 blocks:

- NeuralSLAM module: This deep model estimates a 2D map of the environment as the agent navigates to the goal and also estimates the pose of the agent in the environment using RGB images and odometry (pose sensor) alone.
- Global policy (along with a classical planner): The global policy takes as input the map and the current position of the agent from the NeuralSLAM module and outputs an intermediate point that the local policy can move to.
- Local policy: The local policy takes as input the current pose, the intermediate goal and the RGB images from the camera and navigates to the intermediate point.

The ablations in the NeuralSLAM paper reveal that this modular approach has a better sample efficiency (takes fewer iterations when compared to other end to end approaches) and generalizes better to noisy sensors and actuators.

NeuralSLAM, however cannot be directly used for the current version of the Habitat Challenge due to the following reasons:

- The architecture is designed with the aim of embodied scene exploration. The goal of the global policy is to propose a point in the space that is optimal for exploration - a direction that the local policy can pursue. This is however different for a pointgoal task. The direction in which the local policy must move towards is simply the point goal.
- The pose estimator requires as input the difference in pose of the agent between each time step, which is provided by the GPS+Compass sensor. This sensor is not available in the current version of the challenge in order to ease Sim2Real transfer.

3. Our approach

Similar to NeuralSLAM, we divide the problem of PointGoal navigation without using navigational sensors into two: estimating the pose of the agent in a 2D map, and learning a policy to guide the agent to the goal using the learnt pose and visual representations from RGB images.

3.1. Pose estimation

We build upon NeuralSLAM's architecture to estimate the pose of the agent in the absence of odometry. We do so by using an expected change in pose between the last timestep and the current timestep instead of the odometry.

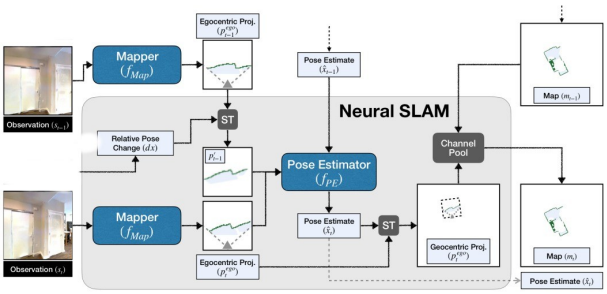

Figure 1. The neural SLAM module from [2]. We compute the δp that is fed in as the relative pose instead of using data from a sensor.

Denoting the position of the agent in 2D by

$$p(t) = [x(t), y(t), \theta(t)]^T$$

and the previous action by $a(t-1)$, the expected change in pose between the last timestep and the current timestep $\delta p(t-1)$ is:

$$\delta p(t-1) = [\delta x(t-1), \delta y(t-1), \delta\theta(t-1)]^T$$

If $a(t-1)$ = move_forward:

$$\delta p(t-1) = \delta T * [cos(\theta), sin(\theta), 0]^T$$

If $a(t-1)$ = turn_left:

$$\delta p(t-1) = [0, 0, -\delta\omega]^T$$

If $a(t-1)$ = turn_right:

$$\delta p(t-1) = [0, 0, \delta\omega]^T$$

Here δT is the translation of the agent in one forward action and $\delta\omega$ is the rotation angle of the agent in one turning action. $\delta p(t-1)$ is 0 for a STOP action.

The actual change in pose of the agent δp_{GT} of the agent is different from δp and can be written as

$$\delta p_{GT} = \delta p + n$$

where n is the effect of noise due to actuation. The role of the deep model is two fold: learn to predict n given the images of the previous and the current scene, and to learn to predict a binary map that is a 2D grid with two channels. The first channel of the binary map is 1 at a location if the location is being occupied and the second channel is 1 if the particular location has been explored. The pose of the agent is obtained by adding the predicted n to the input δp.

3.2. RL policy

The role of the RL policy is to output one of the 4 actions to control the agent using the observations from the cameras and the estimated pose. We explore 2 options for the RL policy:

- A hierarchical approach that uses a global planner and the local planner (in a manner similar to Neural-SLAM): The global planner takes as input the predicted map, pose and the goal location and outputs an intermediate goal location. The local planner takes as input the intermediate goal location and RGB images and outputs the action to be taken by the agent.

- A more end-to-end approach where a visual representation is learned using a visual encoder and a recurrent RL policy is used to predict the action using the visual representation and the predicted pose.

3.3. Training

In an ideal scenario, the pose estimator and the RL policy must be trained together since their data-distributions are dependent (the input to one is the output to another). However, we found that it required a good policy to explore the space in a way that would provide meaningful data to the pose estimator, and starting from a random policy would end up in very short episodes, thereby delaying the training of the pose estimator. To side-step this problem, we used an optimal classical planning method (a shortest path follower that has access to the ground truth map), to collect data for training the pose estimator. We then used the trained pose estimator to train the policy.

The pose estimator is trained using supervised learning: The total loss for the pose estimator is:

$$L(PE) = \lambda_p ||p - p_{GT}||^2 + \lambda_m \text{cross-entropy}(m, m_{GT})$$

where m is the predicted map and m_{GT} is the ground truth map obtained from the simulator. λ_p and λ_m are hyper-parameters that weigh the individual losses.

3.4. Challenges

Although we explored both approaches to design an RL policy (hierarchical and end-to-end), we could not achieve stable training in the hierarchical case, and decided to pursue the end-to-end approach. The end-to-end RL policy was trained to optimize a reward that is a sum of a discrete reward for a successful completion and the negative of the distance to the goal upon termination. The pretrained weights for the model were used from Habitat's baseline models, and was fine-tuned by replacing the pose from odometry by the pose estimates.

While evaluating the existing baselines, we could foresee some problems moving forward. One of the greatest challenges was to adapt existing baseline models with the latest simulation environment which does not use the goal sensor. We planned to overcome this shortcoming by learning odometry from the visual embeddings and use that instead of having a goal sensor.
Another challenge was to make the simulation compatible with the version of the Habitat simulator used in Habitat Challenge 2020. This turned out to take most of our time. The starting point of our work was to try to use the code from the paper [2] which was the winner of Habitat Challenge 2019. We also worked on trying to use the code from [3] and some other baselines in the new simulation environment. Learning pose estimate turned out to be a very challenging task in the presence of noise. Initially, we planned on training the pose estimator in the presence of a pre-trained RL policy. However, this did not work since the RL policy did not perform well in the presence of noise. At this point, we decided to use the shortest path controller which makes use of the ground truth from the environment to give the best possible action. Also, we decided to train the RL policy in the presence of ground truth pose as a separate experiment. However training this policy took a lot of time and did not produce satisfactory results.

4. Experiments and Results

4.1. Learning Pose Estimation

We used the ground truth position from the simulator to train the pose estimator. The weights in the Neural SLAM block were trained using the dataset. The expert shortest path controller was used to run actions on the environment. Mean Squared Error was used as a metric to compare it to the actual ground truth pose. We also used this pose estimate along with the shortest path planner which makes use of the agent's pose estimate to return an action along the shortest path to the goal. However, the agent did not perform very well. Hence, we have just reported the training loss error for the pose estimator given in 2. We have multiplied the pose loss by 1000 to achieve centimeter level accuracy. We achieved an pose estimate accuracy of around $1 - 2cm$ while training.

Split	Distance to Goal	SPL	Success
Train	6.91	0.0	0.0
Val	6.60	0.0	0.0

Table 1. End to End PPO Baseline with actuation and sensor noise.

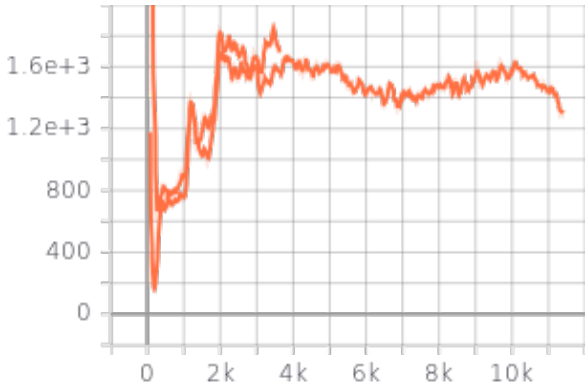

Figure 2. Training Loss for the Pose Estimate

Figure 3. Running pose estimation with classical shortest path follower

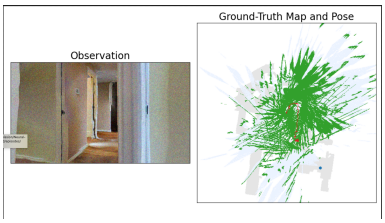

Figure 4. Running pose estimation with classical shortest path follower

4.2. RL Policy

As a baseline for comparison, we attempted to train one of the example policies provided by the Habitat system in the 2020 version of the environment, using the provided trainer scripts. The policy selected was an fully End-to-End actor-critic trained using Proximal Policy Optimization [5]. The agent received RGB-D images and the global position of the goal. The visual layers were encoded with a CNN, and then this encoded and the goal location were passed through a simple RNN, which output a distribution over the action space.

We trained this policy for around 33 hours on a Google Cloud instance. Despite the lengthy training time, we were unable to achieve satisfactory performance. We attribute this training difficulty to the noise aspects of the environment inducing greater variance than expected.

Experimental results can be found in 1.

5. Conclusions and Future Work

Navigating through the environment without a goal sensor was one of the challenges in Habitat Challenge 2020. We tried to tackle this problem by learning SLAM from visual representation. We faced some challenges while training the pose estimate in the presence of noisy actuation. A noisy pose estimate would result in the sub par performance of the navigation policy as well. Hence, future work could focus on developing more robust policy that would generalize well in the presence of noisy inputs. One could also explore end to end learning methods to achieve this which was something we did not explore too much in our analysis. For better better understanding of our SLAM model, we could have given more emphasis to trying to overfit the dataset. During training, we were limited in training speed because of the size of our dataset. We came across other researchers that have reduced the size of the dataset by overfitting the model to a single scene and then utilizing it for the entire dataset. An approach similar to this could be explored to better understand the features that our model learns.
While exploring the policy that could be integrated with our SLAM model, we explored End to End RL policy. However, this took a very long time to train. Without access to multiple GPUs, we were not able to get satisfactory results. In future, we could look at behaviour cloning using the actions from the shortest path follower. Also, we could use something similar to DAgger for training for better generalizability.

6. Work Division

Our work division can be found here 2.

Student Name	Contributed Aspects	Details
Akshay Krishnan	Neural SLAM	Adapting NeuralSLAM to work on the new version of the simulator Retrieving ground truth from simulator (time consuming) Developed the modified pose estimator architecture and training pipeline Report: Introduction and Approach
Prateek Vishal	Implemented Shortest Path Policy	Trained the Pose Estimator using Shortest Path Policy
	Setup GCP	Fixed errors to install simulator
	Report	Training, Challenges, Experimental Results and Conclusion
	Setup GCP	Setup GCP Deep Learning VM's with the appropriate software and available GPU resources
Haard Shah	Setup environments	Setup up Habitat to work headless on Google Colab and GCP (with Prateek)
	Experiment with SOTA	Ran pretrained SplitNet and NeuralSLAM without any changes on older pointgoal navigation and environment exploration task
	Pose estimation training	Debug issues with pose estimator (with Prateek/Akshay) and run training (implemented tensorboard logging and real-time visualization of images)
	Trained E2E PPO	Trained PPO baseline end to from scratch on GCP
Mason Lilly	Evaluation E2E PPO	Performed evaluation for end-to-end PPO task

Table 2. Contributions of team members.

6.1. References

References

[1] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *ArXiv*, abs/1807.06757, 2018. 1

[2] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 3

[3] Daniel Gordon, Abhishek Kadian, Devi Parikh, Judy Hoffman, and Dhruv Batra. Splitnet: Sim2sim and task2task transfer for embodied visual navigation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1, 3

[4] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation, 2019. 1

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 4

[6] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames, 2019. 1

[7] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018. 2