Training Robot Navigation behaviors in real indoor environments

Varsha Sankar
Stanford University
svarsha@stanford.edu

Abstract

In this paper, we attempt to enable autonomous navigation in real indoor environments by training the robot to learn specific navigation behaviors and imitation learning. For training in real world indoor scenarios, we generate the dataset of images that the robot would view if it moves along the path that an expert agent would take given the map (floor plan) and the start and goal positions.

In this paper, we first review the literature in section[2] for similar methods developed to solve similar problem. We then present the dataset in section[3] and discuss its nuances. Section[4] discusses the problem statement and the setup, while Section[5] covers the approach that we took to solve the defined problem. Section[6] discusses the results that we have obtained from our methods.

1. Introduction and Motivation

Autonomous navigation of robots in unknown indoor environments by training them to learn certain navigation behaviors (like, passing through a doorway, moving along a hallway, etc.) and imitation learning is an interesting task. Broadly we aim to navigate the robot based on imitatation learning, where it learns from expert demonstrations. Given only samples of trajectories from the expert agent, the robot should learn the underlying navigation policy directly. The main objective however is to learn inherent patterns and behaviors in navigation. Human beings while navigating in novel environments, reason about free-space, obstacles and topology of the environment using common sense and previous experiences drawn in similar conditions. For example, the task of navigating from one room to another can be broken down into several subtasks. Firstly, exit the current room by crossing the doorway, enter the hallway and navigate along the hallway, finally cross the doorway and enter the desired room. Our goal is to enable expertise in robot navigation by learning these behaviors from generated sequence of images that would be seen by the robot taking the trajectories given by an expert agent.

Classically robot navigation is mostly seen as a goal directed navigation, where the mapping and planning are carried out separately. Thus the underlying obvious patterns which could be exploited to allow the robots to make decisions like the ”common sense approach” in human beings is not utilized well. Here, we aim to train the robot to learn such behaviors and at the same time learn path planning using imitation learning based on trajectories taken by an expert agent. The main motivation behind this to exploit the inherent structure in the real world data and make informed decision about the action rather than doing uninformed explorations at unnecessary instances.

2. Related Work

Navigation is a fundamental problem in robotics. Mapping and Planning are the two fundamental components of navigation. Classically, the two were treated as independent problems and a survey of various such approaches can be seen in [7].

Instead of separating out discrete mapping and planning phases, Reinforcement Learning (RL) [6] methods can be used which is concerned with taking actions in order to maximize some sort of reward. However determining appropriate reward function is a difficult task. Thus, inverse reinforcement learning provides a more supervised approach. Here, we observe an expert demonstrating a task that we wish to perform. We can think of the expert as trying to maximize a reward function that can be expressed as a linear combination of known features and this can be recovered and thereby the policy can be obtained using Reinforcement Learning. But this approach is indirect and slow. On the other hand, Imitation learning [5, 4] and Apprenticeship learning [1] use the underlying principle of Inverse reinforcement learning to learn the policy directly by observing the expert, rather than recovering the reward function itself. These model-free approaches are faster and provide a good approximation of the expert behavior and thus result in significant performance gains.

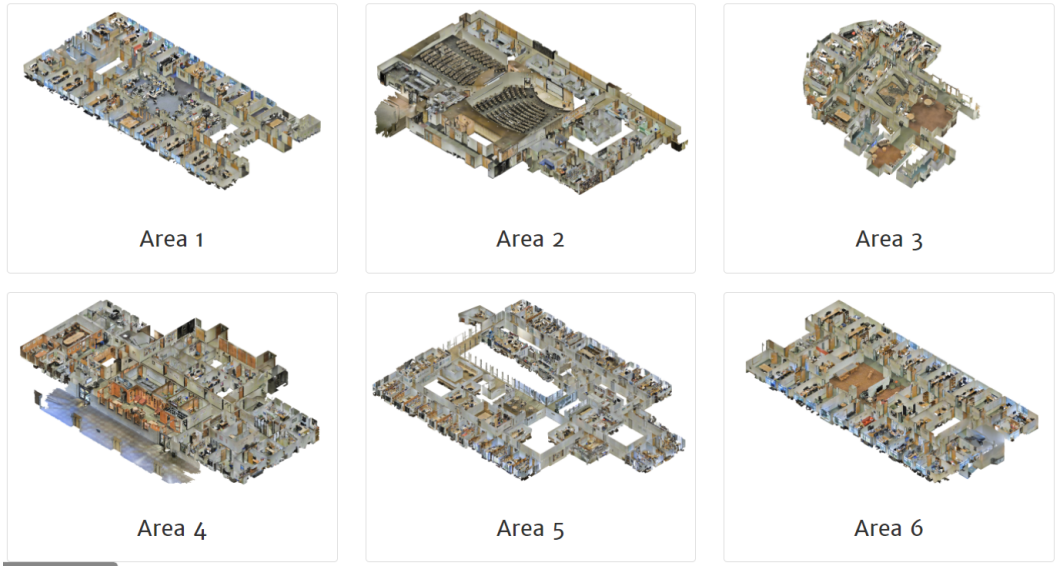

Figure 1. Visualizations of the point cloud data. [2]

In our problem, we use the real world 3D indoor environment data from the 2D - 3D - S Dataset [2] for training our robot. [3] have used the same for robot navigation, but using a different approach where the mapping and planning components are tied together and the robot learns jointly thus making it more robust to errors. Also their model can be used in an online manner in novel environments without requiring pre-constructed maps.

3. Dataset

The dataset used for this purpose was the 2D - 3D - S Dataset which was collected in 6 large-scale indoor areas from 3 different buildings of mainly educational and office use. The dataset along with 360°visualizations is available for download at `http://3Dsemantics.stanford.edu/`. It consists of data in various formats as discussed below.

The dataset consists of regular RGB, equirectangular RGB images and corresponding depth, surface normals with semantic annotations. The 3D reconstructions are also available in the form of textured meshes along with the corresponding 3D semantic meshes. This mesh data for each scanned area was obtained from the matterport camera. This is also presented in the form of colored 3D point clouds by densely sampling the mesh data. Data in the form of semantic meshes and point clouds were primarily used in this work. The point cloud data is presented at the level of each object/component (like walls, ceiling, floor, chair, table, etc.) present in each room constituting an indoor area. This data, without considering the ceiling, was used to get the range of coordinates of an indoor area and then construct the floor plan after thresholding the z-coordinate of each point. The mesh data on other hand was used to render camera views that the robot will observe when it takes a particular trajectory in an area.

Figure 1 shows the 3D visualizations of the point cloud data corresponding to all the 6 areas in the dataset. Each area has a number of office rooms, conference rooms, hallways and pantries. Thus this dataset becomes ideal for generating real world data for learning specific navigation behaviors.

4. Problem Statement

The focus of the problem is to enable robot navigation in indoor environments by learning particular behaviors. This can be broken down into the following subproblems; a) generating the floor plans, b) generating trajectories of an expert agent, c) generating the images that would be seen by the robot when taking those trajectories, d) training the robot with the above data.

The experiments are first conducted in simulated environments. We model the robot as an object having a fixed height and radius, rather than a point object. We assume that it is equipped with vision sensors (like RGB or depth cameras), mounted at a fixed height and orientation with a particular Field of View. We assume that the robot can look around only by rotating the vision sensors across the vertical axis, i.e., there would be yaw, while pitch and roll are assumed to be zeros. Thus the robot could take two actions;

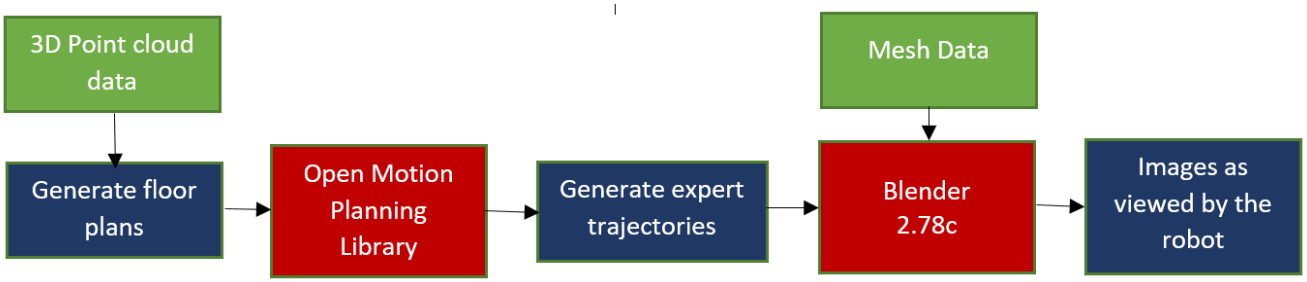

Figure 2. Flow graph of the approach.

a) Moving forward, b) Rotate across the vertical axis, in order to navigate between the start and the goal positions in any given environment.

5. Approach

As described in section 4, the first step was to generate the floor plans corresponding to the 6 large indoor areas from the 3D data, such that the white regions correspond to ”free” regions, while black corresponds to ”busy” in the floor plan. Next, using the floor plans, we generated expert trajectories between random valid start and goal positions. Using the paths, we obtained the images that would be viewed while navigating along them. This flow can be clearly seen in the Figure 2.

5.1. Generating Floor Plans

We used the data from 2D - 3D - S dataset in the point cloud format to generate the floor plans. Since each of the 6 areas had different dimensions, the first step was to determine the range of coordinates for each area. First approach was to use the mesh data and obtain the coordinates using Blender software. Though this was fast, since the data was huge and required high memory, we used the point cloud data to obtain the range of coordinates. The point cloud data was divided semantically across the components constituting each scene and was in the form of text file with each line containing the X, Y, Z coordinates and R, G, B values corresponding to a point.

After obtaining the range of coordinates, the z-coordinates of each point which is not a part of the ceiling, floor or beam, was checked to note if it falls within a threshold marked by the height of the robot to determine if the point would correspond to a potential obstacle, i.e., if the point has a z-coordinate which is lower than the height of the robot, then it is an obstacle (”busy”), otherwise it corresponds to ”free” space. Using this algorithm, each ”busy” point was marked as a black pixel and each ”free” point as a white pixel in the floor plan. All the floor plans can be seen in Figure 3.

5.2. Generating trajectories

After obtaining the floor plans, expert trajectories were generated using Open Motion Planning library (OMPL). OMPL is an open source library written in C++ consisting of several state-of-the art sampling based motion planning algorithms and has python wrappers. The inbuilt algorithm for RRT-Connect (Rapidly-exploring Random Trees) was used to generate the expert trajectory given the floor plan as an image along with the start and goal positions. Here, the black regions in the map were assumed to be ”free” and the white ones ”busy”. Thus the floor plan obtained as described in Section 5.1 had to be inverted, i.e., white regions where thresholded and changed to black and vice-versa. The visualizations of the map, plan and choosing of positions were all carried out using openCV 3.

We have support for two modes. In the first, the floor plan would be displayed and the start and goal positions can be manually selected and the solution path visualized. In the other mode (batch mode), several paths can be generated in one go by letting the start and goal positions to be randomly and automatically chosen, provided they are valid (corresponds to ”free” region or black pixels). The path obtained from OMPL in terms of the X and Y coordinates with repect to the floor plan image (in terms of the row and column number) along with the orientation (just yaw, pitch and roll are assumed to be 0) is saved as a .txt file. Yaw is allowed to achieve a more human-like behavior, so that the turning of the robot doesn’t appear to be very abrupt. For each point in the path, the label of the pixel which gives information about which room (office or conference room, etx.) it belongs to, is also saved in that file.

5.3. Generating image dataset

Using the paths generated above, the camera views that would be observed by a robot moving along that path is captured using Blender and stored as images. The mesh data for a given indoor area is loaded into Blender and using the Blender python library, a python script is used to read the path off the .txt file as sequence of coordinates in each line.

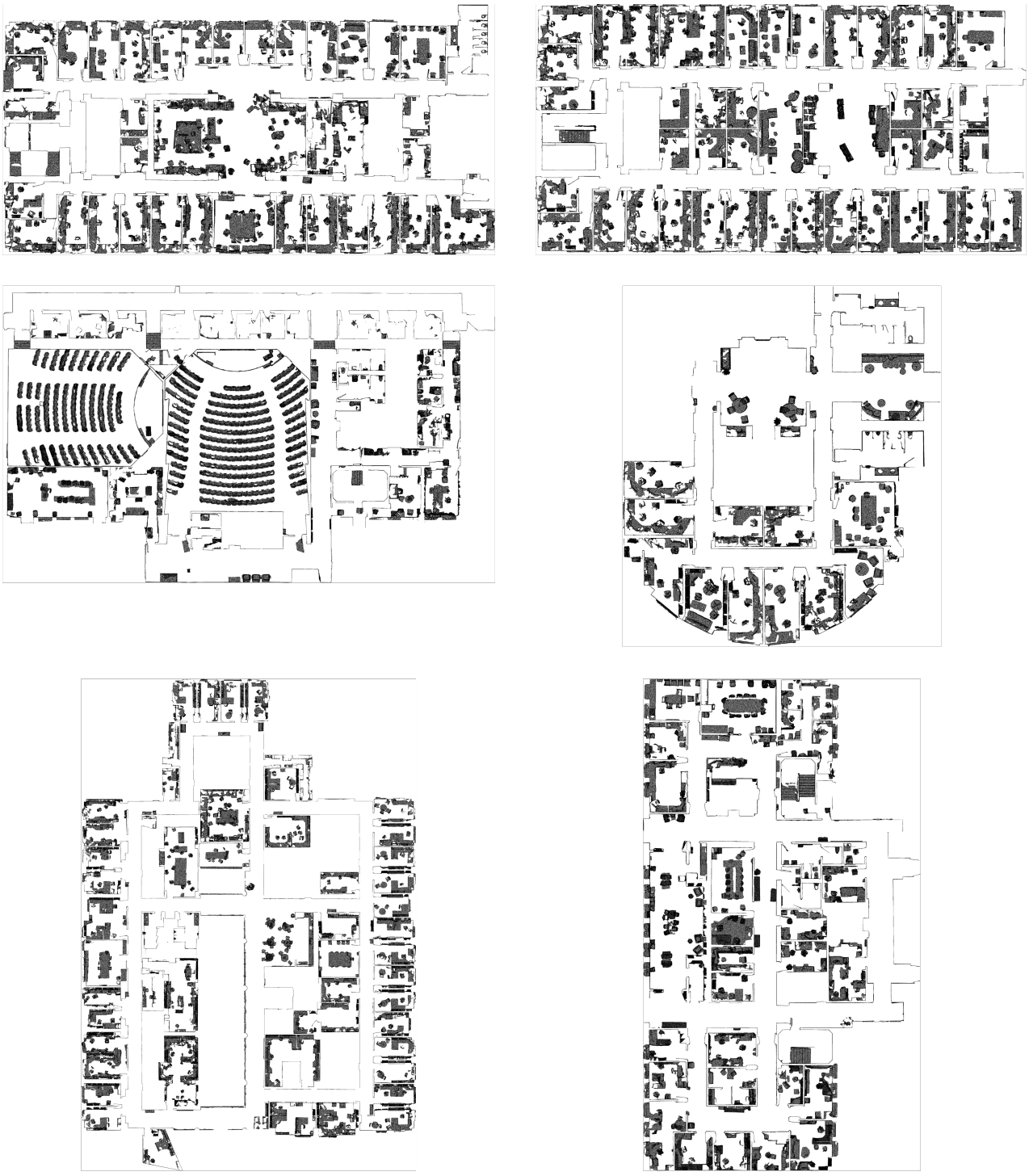

Figure 3. Floor plans of all 6 areas.

Figure 4. Sample path on Area 1.

Figure 5. A simple path on Area 1.

These coordinates are scaled and translated to the world coordinates corresponding to the mesh data and the camera location is changed accordingly. At each position, the camera views are captured and saved. Using the python script helps us to run Blender in the background, thus speeding up the process considerably. These images along with the label of the position indicating which room it belongs to can then be used to train the robot navigation behaviors.

Figure 6. Sequence of images observed when turning towards the goal from initial orientation.

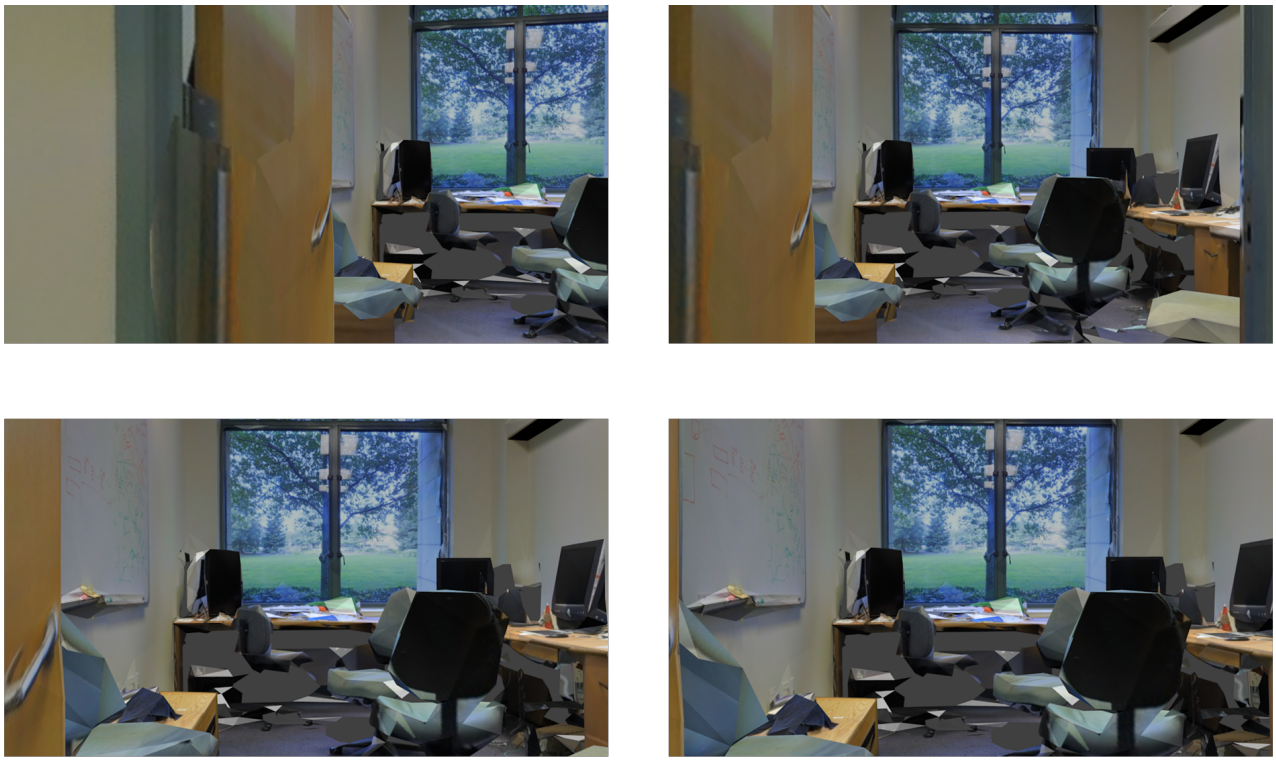

Figure 7. sample images observed when moving along the path.

6. Results

The floor plans that were generated from the dataset can be seen in Figure 3. The trajectories were generated by randomly choosing valid start and end goals after inverting the binary image of the floor plans. As we can see, in this image the ”free” areas are in black, while the ”busy” regions are white. To ensure that the path obtained using the RRT-Connect algorithm is approximately near the center of the free space which is more natural and human-like and also since we assume that our robot has real dimensions and not a point object, we internally increase the thickness of the walls and other obstacles. The robot also is assumed to have a smooth yaw to ensure that the views observed transition smoothly during turns instead of abrupt changes. An example showing a path between two positions on Area 1 can be seen in Figure 4. Here, the blue dot indicates the source position and the green dot indicates the goal position.

After generating a few thousands of such paths from each area, we generate camera views by moving along those paths using Blender. Each path results in hundreds of

camera views. Let us consider some of the camera views for a small and short path as shown in Figure 5.Here the robot is moving from the hallway into an office room. Figure 6 shows the camera views as the robot turns towards the desired direction from its initial orientation at the start position. Figure 7 shows some sample images as observed while moving along the path.

Though we can see from the camera views that they are of low-poly resolution, it gives a good approximation of the real world, thus making it suitable for training the robot in simulation and later adapting it for the real world.

7. Conclusion/Future Work

The images that the robot would view by moving along the expert trajectories were generated using blender. These images could be used to train the robot to learn the specific navigation behaviors. One issue with generating these images using blender was that it was very slow, thus to speed up the process, blender was made to run in the background. However, it still takes a considerable amount of time to generate images corresponding to one single path. This can however be cut down by sampling only few points along each path and obtain the camera views only at those points, thereby expediting the process and also saving lesser images occupying lesser memory.

The immediate next step is to use the image dataset obtained to train the robot to learn navigation behaviors. After training, the robot could be made to navigate based on semantic based instructions to move from a particular physical room to another based on the label associated with each position.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, Feb. 2017.

[3] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 2017.

[4] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[5] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 1, page 6, 2011.

[6] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[7] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.