# Please Wait to Be Seated: Empty Chair Detection and Navigation with Movo

Jasmine Liu, Andrew Hou

May 2018

## 1 Abstract

Object oriented navigation is a key problem in robotics that leads to many real-life applications. However, current work falls short in detecting objects consistently due to occlusion and lighting. It is also hard for robots to make decisions in a dynamic environment where human activity is in the loop. We present an interactive system that allows the Movo robot to detect, save, and intelligently navigate to empty chairs in a dynamic environment. Our approach uses the state-of-the-art convolutional neural network Fast R-CNN to detect objects (chairs) in a consistent manner. Our work also includes SLAM to accurately locate the robot and a finite state machine to help the robot adapt to a dynamic environment. Our system allows robots to perform object oriented navigation in the setting of finding and navigating to an empty chair in a dynamic environment.

## 2 Introduction

Finding an empty chair is one of the most subtle and common problems in everyday life. Restaurants need servers to guide customers to an empty table; students have to squeeze through the crowd to find an empty seat in a big lecture hall. This tiring process can be replaced by a robot that can autonomously detect chairs, navigate and adapt to the environment.

Our approach for solving this problem can essentially be divided into two major tasks:

1. Chair detection (distinguishing and detecting chairs that are empty as opposed to those that are occupied) and finding the real world location of each detected chair

2. Navigation planning/Decision Making (how should Movo direct someone to an empty chair)

Our final checkpoint provides solutions for both tasks, which we discuss in subsequent sections.

# 3 Related Work

There are a lot of current applications in robot indoor searching. One paper "Object Search and Localization for an Indoor Mobile Robot" by K Sjo[1] uses classic computer vision techniques to detect, recognize and estimate the distance of the object. However, the method requires the object to be within certain range and without any occlusion. Occlusion is generally too hard to deal with in indoor object searching and we are using R-CNN to tackle this problem. A lot of the applications do not consider a dynamic environment so the robot is unable to correct itself once the environment changes, such as when someone gets out of their chair or suddenly takes a chair in the case of this project. (Unfortunately we are not able to finish this part in this semester. This becomes one of our future works).

# 4 Technical Approach

Our approach can be divided into three parts: perception, navigation and decision making. Perception involves detecting and localizing the chair in both the image and the real world. Navigation involves the robot exploring the environment and navigating to a specific chair. Decision making controls the whole system flow, including the initial exploring stage and navigation stage.

## 4.1 Perception

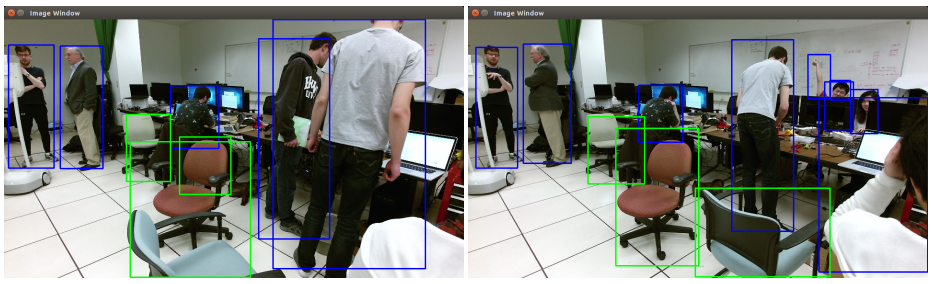


Figure 1: People and chair detection result: blue bounding box indicates people and green bounding box indicates chair

#### 4.1.1 Object Detection

We use the state-of-the-art Faster Regional Convolutional Neural Network(Faster RCNN) [2] with the ResNet 101 network pretrained on the VOC-Pascal dataset[3]. The VOC-Pascal dataset includes images with 20 classes. We use two classes from the detection result: person and chair. The network detects people and chairs in Movo's view in Fig.1 and returns bounding box coordinates for each object.

#### 4.1.2 Object Localization

In order to localize the object in the real world, we use the Kinect depth information. We converted depth information into the xyz real world coordinates. We use the coordinates of the centroid of bounding boxes detected by RCNN as the location of the object. This may not be the best solution. Sometimes, the centroid is located at the background and causes errors in location estimation due to incorrect depth (the depth of the background is generally greater than the depth of the chair). We were nonetheless able to navigate through the entire lab space and generate a map with all of the chairs located on the map as shown in Fig.2 and Fig 3.

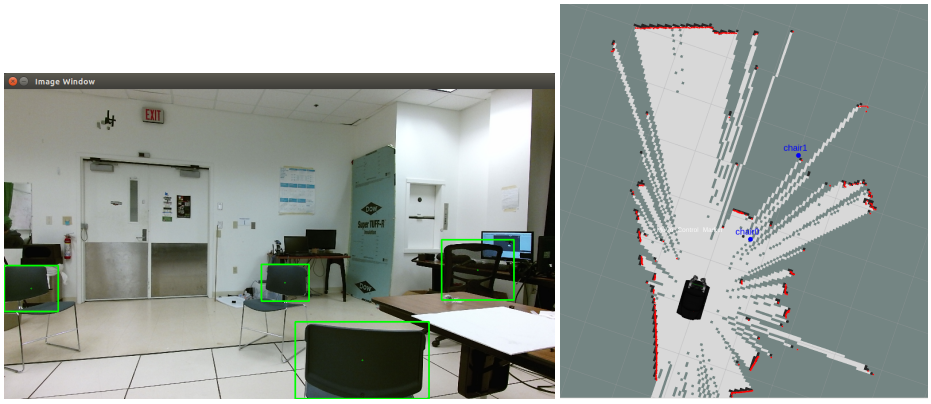


Figure 2: the left figure shows the bounding box detection and the right figure shows the location on the map

### 4.2 Navigation

In the navigation stage, we load the map and the chair locations from the exploration stage. After estimating the robot's location in the map, we are able to pick the closest empty chair. We set up an action client that can communicate with the ROS navigation stack and send the chair location to the action server. The ROS navigation stack is able to plan a path for the robot to avoid any obstacles on the way.

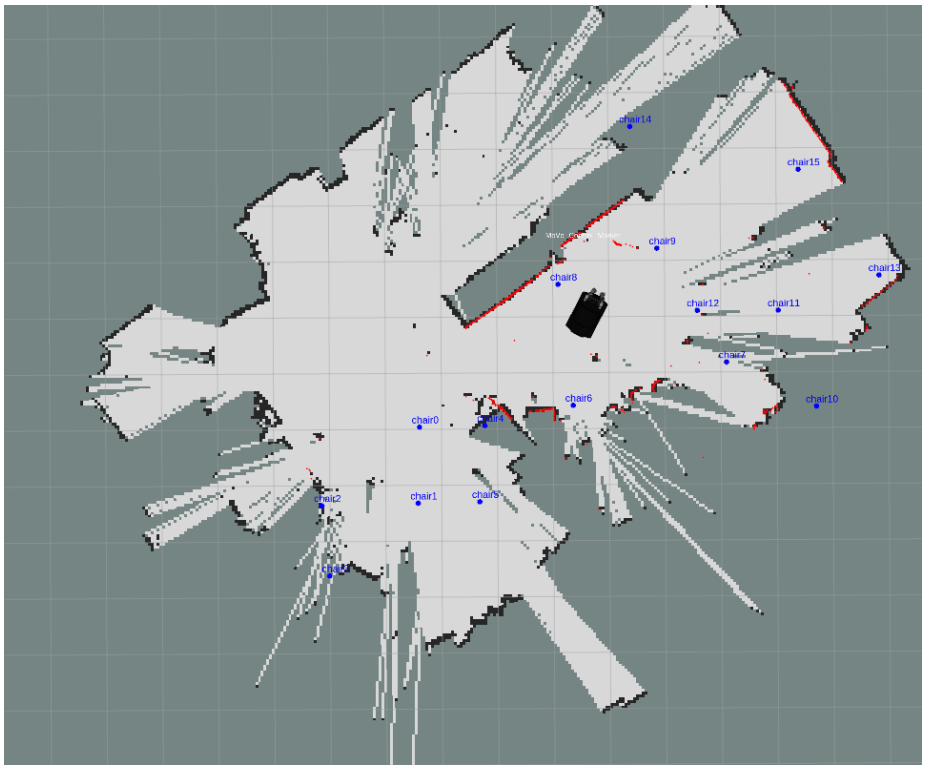


Figure 3: Map of the entire lab space with empty chairs located

#### 4.2.1 Selecting an Appropriate Empty Chair

The ROS costmap has 3 layers: *static map* that represents the unchanging part of the costmap, *obstacle map* that tracks the obstacles and *inflation map* that inflates obstacles and provides configuration space for the robot. In order for motion planner to successfully plan a path, we choose the closest empty chair that's not within the inflation radius of any obstacles (set to 0.55m for Movo).

### 4.3 Decision Making

Decision making is the high level description for the whole system flow. It can be represented as a finite state machine in Fig 4. In the exploration state, we manually drive the robot to go around the entire environment to build a map. While the robot is moving, it detects chairs and associates their locations on the map. After we explore the whole area, we enter the second stage: the navigation stage. In this stage, the robot will start execution upon detection of a person. The robot will use the map generated in the first stage and navigate to the closest empty chair that is not within the inflation radius of any surrounding obstacles.

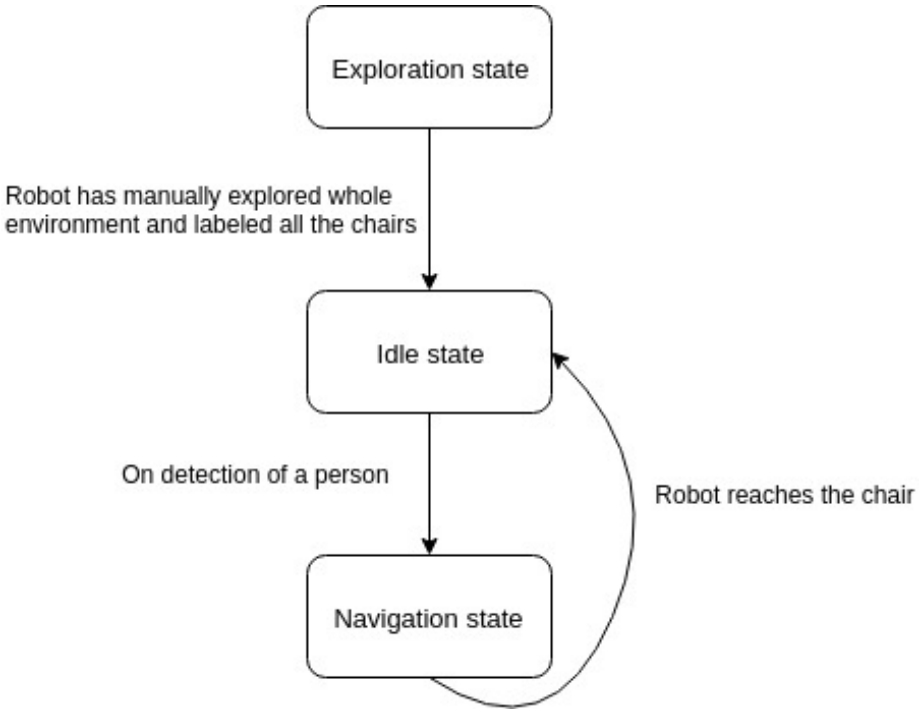


Figure 4: Finite state machine for the system

### 4.4 Human-robot interaction

We enabled the speaker on Movo to allow for some user interaction. When the navigation stage starts, Movo will say "Welcome". When Movo detects a person, it will say "Please wait to be seated" before beginning navigation to the closest chair. When Movo reaches the destination, it will say "Here is your seat. Enjoy your dinner." This simple vocal interaction can be expanded into conversations and voice commands in the future.

## 5 Evaluation

Given our pipeline, Movo is able to save the locations of all chairs in a room, and when prompted will direct a user to the closest empty chair that isn't surrounded by obstacles. Movo also interacts with the user by saying "Please wait to be seated", notifying the user that it is about to perform navigation. Movo can also notify the user once navigation is complete and the closest empty chair is reached. A demonstration of our system is linked below:

https://www.youtube.com/watch?v=UldPAEIwEEO&feature=youtu.be

# 6 Conclusion

Given that our current pipeline allows Movo to detect and save the locations of chairs as it moves across the room and enables it to direct a user to the closest obstacle-free empty chair, we can say with confidence that we've made substantial progress this semester towards creating an ideal system that allows robots to direct people to empty chairs. There are also several directions for future work that exist given the current state of this project, which we believe would be worthwhile to pursue to create a more adaptive and error free system. The first improvement involves chair detection. Since we only use the depth at the centroid of the chair's bounding box to predict the location of the chair, this can often lead to inaccurate estimates of chair locations on the map, particularly if the centroid captures a wall or door at a much higher depth. One possible solution would be to use segmentation for example, Mask-RCNN, rather than bounding boxes to ensure that any point we use to estimate the depth of the chair truly belongs to the chair. Another solution would be to scan through all points in the bounding box and pick the point that yields the minimum depth (ignoring points with no depth estimates).
Another line of work is allowing Movo to better adapt to a dynamic environment. For example, if Movo is navigating to a chair and an obstacle suddenly is pushed in its way, it should be able to update the map of the room and redo navigation planning. Also, if the chair that Movo was directing a user to is taken, Movo should be able to choose a new chair as a goal and redo navigation. We also have a third potential problem if the chair that Movo was navigating towards is suddenly pushed away. In this case, Movo should be able to check if it has actually arrived at a chair, and if not, should direct the user to a new chair.
Ultimately, the final goal will be to produce an interactive system that allows Movo to accurately detect all available empty chairs in the room and dynamically direct a user to the closest empty chair with no surrounding obstacles, all while adapting to changes in the environment. We believe that our current work is a large step towards this end goal, and that the project has high potential for future development.

# References


[1] Kristoffer Sjö, Dorian Gálvez López, Chandana Paul, Patric Jensfelt, Danica Kragic *Object Search and Localization for an Indoor Mobile Robot*

[2] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.*

[3] The PASCAL Visual Object Classes Challenge 2007 *http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html*