1 Design

This section details and analyzes the design choices we took as a team. All of our choices were mediated through experimentation, reference to the background literature, as well as availability and feasibility.

1.1 Hardware and Software

The Pioneer 3DX (P3-DX) is one of the most popular research robots [21] and the model we used. It is a compact differential-drive mobile robot and is fully assembled with an embedded controller, motors with 500-tick encoders, 19cm wheels, a tough aluminum body, 8 forward-facing ultrasonic (sonar) sensors, 8 optional rear-facing sonar, and 1, 2 or 3 hot-swappable batteries [22]. Because of its modest and balanced size combined with reasonable hardware, it is best suited for indoor navigation [21]. The robot was additionally equipped with a Hokuyo URG-04LX laser sensor, which can report obstacles in ranges from 20mm to 5600mm (with a 1 mm resolution) in a 240°arc (with a 0.36°angular resolution). It has a scanning time of 100ms/scan [23].

The software we used was the Robot Operating System (ROS), a Linux based software framework for operating robots [24]. It is open source and multi-lingual [24], and has language support for C++, Python, Octave, and LISP, and experimental ports for Java and Lua [25]. We used python (2.7) for our programming needs.

The core concepts of ROS are packages, nodes, topics, messages and services. A node is an executable program that takes data from the robot's sensors and passes it to other nodes. Nodes communicate with each other through strictly typed data structures called messages. Messages always travel via special ports called topics, which are strings such as odometry or map. A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics [24]. All related nodes are combined in one package that can easily be compiled and ported to other systems. The packages are necessary to build a complete ROS-based autonomous robot control system [21].

1.2 Control Architecture

For the overall control design we followed a behaviourally inspired hybrid approach. We felt this best fit the problem scope; we knew exactly what tasks we needed to execute (and in what order), and we were dealing with a semi-structured environment. Our challenge was thus to integrate the a-priori world knowledge with reactive behaviours such as obstacle avoidance.

Our hierarchical planner was structured into a series of sequential steps as per the task requirements. These are, in order:

```
1  Localise in respect to the map
2  Navigate to the hard-coded coordinates of the first room and position to
       scan the room.
3  Scan the room checking for the presence of people
4  IF person detected and confirmed, goto 6
5  ELSE set first room as the meeting room and goto 9.
6  Navigate to the coordinates of the second room and position to scan the
       room.
7  IF person detected and confirmed, goto 9.
8  ELSE set second room as the meeting room and goto 2.
9  Roam the area searching for people.
10 Upon detecting a person ask them if they would like to attend the meeting
       .
11 IF person confirms then drive back to the meeting room then goto 9
12 ELSE goto 9 (continue roaming)
```

During execution of the plan steps, appropriate reactive behaviours and responses are selected according to the robots peripheral environment as reported by the sensors. The navigation algorithm, for example, selects a point-to-point path consistent with the plan specification and a perceptually running obstacle avoidance algorithm results in the robot avoiding any local and unforeseen obstacles.

1.3 Localization

As a team we decided to adopt the ACML localisation implementation included as part of the ROS Navigation Stack package over a bespoke MCL implementation we had previously developed. As discussed in the background literature, the rationale of using a Monte Carlo approach to localisation is clear: they dont possess the independence assumption of Markov localisation techniques (in their basic implementations), and unlike Kalman filters (again, in their basic implementations), particle filters can approximate a wide variety of probability distributions, not just gaussian distributions. They are also computationally efficient, as they focus resources on regions in state space with high likelihood [13].

The immediate advantage of the ACML implementation over our MCL approach is that it uses adaptive sampling (adjusting the sample size during the localisation process). Experimental results (See 5.2) also show the ROS ACML implementation being much quicker and more robust than our bespoke MCL algorithm. Experimental data (5.2) also informed design decisions on the respective parameter configurations for the odometry drift, rotation, and translation noise.

1.4 Navigation and Exploration

For Navigation, we used the ROS Navigation Stack, described as a 2D navigation stack that takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base [26]. It uses ACML for localisation and position tracking (when supplied with an a-priori map), and a grid-based costmap for motion planning [27]. It was developed for an autonomous mobile robot, PR2, to robustly navigate through a real-world dynamic office environment. PR2 achieved 26.2 miles of autonomous navigation [27]. The Navigation Stack also boasts several recovery strategies: when a robot finds itself entrapped by obstacles, for example, a number of increasingly aggressive recovery behaviors are executed to attempt to clear out space [27].

Considering its demonstrated robustness and the problem domain the ROS navigation stack was developed for (which is very similar to ours), there was no need to research other navigation methods at the expense of losing time to solve the other sub-tasks.

We tested two possible exploration strategies for our robot. One strategy was allowing the robot to free roam and the other was setting a fixed path through hard-coded points. Through qualitative experimentation (presented in 4.3) we found the fixed point approach to be more suited to the task. Whilst it covered less area in absolute terms than the random free roaming navigation strategy, the areas covered were more sensible to the problem scope (such as corridors as opposed to corners and dead-ends) and also covered significantly faster.

1.5 Obstacle Avoidance

Obstacle avoidance is included as default in the ROS Navigation Stack and has been proven to be robust (which is corroborated by our experiment data in Section 5). Obstacles are inferred strictly from geometric information from range sensors that can produce point clouds [27]. In our case, this was the front-mounted Hokuyo laser.

1.6 Person Detection

As per the sub-task of identifying persons, we adopted a laser based leg-detection system as developed by Macro Becerra, a demonstrator for this module. The algorithm is detailed thoroughly in the Literature Review section 3.4.

Whilst we implemented the basic algorithm as described in in section 3.4 in our robot, there are several improvements we could have considered. One such improvement would be integrating face detection with leg detection to arrive at a more accurate conclusion on the presence of a person. Another (more difficult) improvement would have been to track the person/s in real time. This would have the immediate advantage of eliminating some false positives from the leg and face detection (i.e. a person can't just appear/disappear in the middle of the scene). Tracking a walking person, however, is not a trivial task. There are many factors which may influence the human trajectories, such as the environment, the number of people, and the interactions among them. In fact, the unpredictability of such behaviors have been modeled by some researchers as Brownian motion [29]. Typically, Kalman Filters are used to track persons, as they are fundamentally designed to predict a processs state and

then use measurements to correct or update these predictions. In this regard, their purpose and function in tracking persons is not at all dissimilar to that in motion tracking.