Session 3 – Robotics and Autonomous Agents

Dr Claire Rocks  
C.L.Rocks@warwick.ac.uk
Agenda

• What is a mobile robot?
• How do they get information about their environment?
• How can they use that to create maps?
• Once they have a map, how do they find a path through?
• Introduce the lab
What is a robot?
What is a robot?

A robot is:

1. An artificial device that can sense its environment and act purposefully in it
2. An embodied artificial intelligence
3. A machine that can autonomously carry out useful work

*Alan Winfield, Robotics: a very short introduction*
Microcontroller and software

Camera
Microphones
Proximity sensors
Accelerometer

Lights
Speaker
Motor/wheels
Robot arms
Pneumatic/Hydraulic pump

Battery

energy

A body
Task

• Chat to the person next to you to answer the following question:

  Why do robots need sensors?
Why do robots need sensors?

To measure parameters of

• The robot, and
• The environment

For various purposes such as control, monitoring, inspection and modelling

Increasing the sensor capability could

• Increase flexibility
• Save time
• Increase productivity
• Increase safety
The sensing process

1. Transducer
   - Converts a physical or chemical quantity into an electric signal

2. Signal conditioning
   - Want a signal to be linear and proportional to the quantity being measured. Signal may need to be amplified

3. Computer Interface

4. Data reduction and analysis
   - Data compressed (if required) and fused to generate a model of the world

5. Perception
   - Models analysed to infer about the state of the robot surroundings (or world)
Commonly used sensors

- Wheel/motor encoders
- Heading sensors
- Accelerometers
- Inertial Measurement Unit
- Light and temperature sensors
- Touch sensors
- Proximity
- **Range sensors**
- Beacons/GPS
- Vision Sensors
Task

• An autonomous unmanned security car is to patrol the University Campus.

• In groups of 3-4 discuss the following:
  • What internal and external parameters do you think the autonomous unmanned security car would need to monitor and why?
    • Can you take a guess at the sensors it would need to use?

• Feedback to the group
Fundamental Questions

Where am I?  Where am I going?

How should I get there?
All real robots interact with humans to an extent. However, the amount of interaction varies, and this reflects the extent to which a robot works for itself or relies on us.

Increasing human input

Autonomous | Commanded | Remote-controlled

- Difference between autonomy and automatic
- Dynamic autonomy
- Autonomy and Intelligence are not the same thing...
Fundamental Questions

Where am I?  
Where am I going?  
How should I get there?
This is hard!!

- World is uncertain, complex and dynamic
- Sensor measurements are noisy
- Control inputs are noisy

Moravec wrote, “It is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility.”
The Motion Planning Problem

- Find a motion from a starting position to a goal position while safely avoiding any obstacles
- A key component in the navigation stack of a robot

Given an initial pose (position and orientation) and a goal pose of a robot, find a path (in the form of a continuous sequence of robot poses):
- that do not collide or contact with any obstacle
- that will allow the robot to move from its starting pose to its goal pose
- and report failure if such a path does not exist
Dense Metric Maps

• A metric representation (or metric map) is a symbolic structure that encodes the geometry of the environment
• Dense representations attempt to provide high-resolution models of the geometry
• These models are more suitable for obstacle avoidance, or for visualisation and rendering
• Visually pleasant, also need a large amount of data

Feature-based Metric Maps

- Feature-based metric maps represent the scene as a set of sparse landmarks corresponding to identifiable features in the environment (e.g., lines, corners, interest points)
Topological Maps

- Topological maps drops the metric information
- Only leverages place recognition to build a graph in which the nodes represent distinguishable “places”, while edges denote reachability among places
- Not suitable for tasks such as obstacle avoidance
- Can add local metric information to the edges

Topological maps in COLD-TopoMaps Dataset [3]

Simultaneous Localisation and Mapping

- The SLAM Problem
  - A fundamental problem in mobile robotics, and providing some solutions is one of the main successes of probabilistic robotics.
  - A robot with sensors moves through a previously unknown, static environment, mapping it and calculating its own motion at the same time.
We need SLAM:

• When a robot must be truly autonomous (no human input)
• When little or nothing is known in advance about the environment (don’t have a map)
• When we can’t place or use infrastructure such as artificial beacons, or GPS
• And when the robot actually needs to know where it is

In SLAM:

• We build a map incrementally
• And localise the robot with respect to that map as it grows
How do you represent the robot’s map in the code?
Occupancy Grid Representation

- Use Grid to Represent an Area on the Ground
  - Choose a suitable square grid cell size

- For Each Cell
  - Store and update a probability of occupancy $P(O_i)$ that it is occupied by an obstacle

- Greyscale Visualisation
  - From black for *occupied*, to white for *empty*, while intermediate values are shades of grey
The Robot in the World (Workspace)

- To solve the motion planning problem, we need to position the robot in an appropriate space.
- Every point of the robot needs to be specified the space to ensure no possible collision.
- We can transform the robot and the obstacles in the workspace into the configuration space (C-Space).

The Configuration Space

- The space of all legal configurations of the robot.
- Regardless the geometry of the robot.
Configuration Space

- **C-Space Transformation**
  - Two robot locations $A_1$ and $A_2$ in the workspace
  - Two robot configurations $q_1$ and $q_2$ in the C-Space
  - WO: Obstacle region in workspace; CO: Obstacle region transformed into C-Space
  - $C_{\text{free}}$: free configuration space; $C_{\text{contact}}$: the contact space
Configuration Space

• Planning a Path in C free
  • Within the C-Space, the robot can be viewed as a point
  • Path planning can be done for this point in the C-Space
Finding paths through C-Space

Roadmap

Cell Decomposition

Sampling-based
Roadmaps

- Can be Constructed by Various Methods
  - Visibility Graph
  - Voronoi Diagrams
Supporting and Separating Lines
Roadmaps

- Can be Constructed by Various Methods
  - Visibility Graph
  - Voronoi Diagrams
Exact Cell Decomposition

- Decompose Configuration Space $C$ into
  - Discrete and non-overlapping cells
- Trapezoidal Decomposition
- 2D $C$-Space, and polygonal obstacles
- Vertical/Horizontal sweeps the $C$-Space, draw a line when meets a node of obstacle
- A path can be found from the associated connectivity graph
Approximate Cell Decomposition

- Decompose Configuration Space $C$ into
  - A grid of cells with predefined shape and size
  - Samples the workspace, and identify:
    - Free cells
    - Mixed cells
    - Occupied cells
  - Connectivity Graphs Built upon Free and Mixed Cells

Approximate Decomposition

Adaptive Decomposition
Sampling-based approaches

• Sampling C, and connecting those samples to solve the motion planning problem
  • Two main approaches:
    • Probabilistic Roadmap
    • Single-query planners

Probabilistic Roadmap
Rapid-exploring Random Tree
Connectivity Graphs

- A graph models a set of connections
- They are made up of nodes and edges
- A node can be connected to many other nodes, those nodes are called its neighbours
- Can be used to model paths through mazes – junctions where there is a choice of route are modelled as nodes
- Is there a path between nodes?
- What is the shortest path?
Task

• In pairs, draw the graph representing the following maze.
  Draw in the dead ends
Search Algorithms

• Given the C-Space Representation (a connectivity graph), search for a solution

• Different Types of Search Algorithms

  • **Uninformed search** - no additional information about the distance from the current state to the goal.
    • **Examples: Depth First Search and Breadth-First Search.**
  
  • Informed search – has additional information about the distance from the current state to the goal
Depth First Search

• Explores as far into the graph as possible before backtracking to visit unvisited nodes
• Often implemented using a recursive algorithm
• Uses a stack - LIFO
A B C G
VISITED

A B C G
VISITED

A B C G D
VISITED

STACK
A
STACK
B A
Breadth first search

- Visits the nodes closest to the starting point first
- A queue – (FIFO) - is used to keep track of the nodes to visit
Summary

• What is a mobile robot?
• How do they get information about their environment?
• How can they use that to create maps?
• Once they have a map, how do they find a path through?
• We be doing some of this in the lab tomorrow
The robot lab

• Working in pairs

• Your robot is part of a search and rescue mission. It needs to get to the victims located at the Goal. Once it has found the victims, it should lead them to safety via the fastest route.
2 steps

1. Explore the world and build the map in simulation
2. Use the map to provide waypoints to a real robot
Initially, we don’t know much...

```javascript
maze = {
    {1,1,1,1,1,1,1,1,1,1,1,1},
    {1,2,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,1},
    {1,1,1,1,1,1,1,1,1,1,1,1}
};
```

0 = unoccupied
1 = occupied
2 = starting cell
3 = goal
4 = unknown
This will return and array called neighbours with 4 elements, [?, ?, ?, ?].

- [0] will store a 0, 1, 2 or 3 depending on the cell to the North
- [1] will store a 0, 1, 2 or 3 depending on the cell to the South
- [2] will store a 0, 1, 2 or 3 depending on the cell to the East
- [3] will store a 0, 1, 2 or 3 depending on the cell to the West

The 0, 1, 2, or 3 stored in the array represent the following:
- 0 - Unoccupied
- 1 - Occupied
- 2 - Starting cell
- 3 - Goal

This checks the neighbours of the current cell and then checks that the first element in that array, [0], is not equal to 1 (its the ! that makes the check NOT equal to). If the element is not equal to 1 then we can move north by subtracting y from the current element, current_y -= 1.
Sets the size of the drawing canvass
Sets the frame rate
Used when drawing the grid

```java
int h = 0;
int w = 0;
int cellWidth;
int[][] maze;
Robot robot;

Library lib;
void setup() {
    frameRate(10);
    size(600, 600);
    robot = new Robot();
    lib = new Library();
    h = lib.getHeight();
    w = lib.getWidth();
    cellWidth = width/w;

    //initialise maze as unknown
    maze = new int[h][w];
    for (int i=0; i<h; i++) {
        for (int j=0; j<w; j++) {
            maze[i][j] = 1;
        }
    }

    //set special cases
    //walls around the edge
    for (int i=1; i<h-1; i++) {
        for (int j=1; j<w-1; j++) {
            maze[i][j] = 4;
        }
    }

    //set start node
    maze[1][1] = 2;
}

maze = {
    {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
    {1,2,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
    {1,4,4,4,4,4,4,4,4,4,4,4,4,4,4,1},
};
```
void draw() {
    //draw the maze
    drawMaze();
    //draw the robot in the maze
    robot.draw();
}

// Write your code here

void drawMaze() {
    for (int i=0; i<h; i++) {
        for (int j=0; j<w; j++) {
            switch(maze[j][i]) {
                case 0:
                    fill(255, 255, 255);
                    break;
                case 1:
                    fill(0);
                    break;
                case 2:
                    fill(0, 255, 0);
                    break;
                case 3:
                    fill(255, 0, 0);
                    break;
                case 4:
                    fill(127);
                    break;
            }
        }
    }
    rect(j*cellWidth, i*cellWidth, cellWidth, cellWidth);
}
Moving it to the robot

- Using LeJOS – lets us program the robot in Java
- Extract the waypoints from the grid map → upload them to the robot
import lejos.nxt.*;
import lejos.robotics.navigation.*;

class Robot {
    private DifferentialPilot pilot;
    private Navigator navbot;

    public Robot() {
        // set up the robot
        pilot = new DifferentialPilot(5.0, 17.5, Motor.B, Motor.C);
        // set up the navigator
        navbot = new Navigator(pilot);
        pilot.setTravelSpeed(30); // cm per second
        pilot.setRotateSpeed(45); // degrees per second
    }

    public void moveTo(int x, int y) {
        navbot.goTo(x, y);
        while(!navbot.isMoving()){}
    }
}

The Main Class

```java
import lejos.nxt.*;
import lejos.robotics.navigation.*;

public class RobotLab {

    public static void main(String[] args) {

        // set up the robot
        Robot robot;
        robot = new Robot();

        // write code to move the robot
        // e.g. robot.moveTo(20, 20);

        // press F9 to compile and upload the code to the robot

    } // end class main

} // end RobotLabDFS
```