AN ANALYSIS OF THE FIELD D* ALGORITHM FOR PATH PLANNING IN THE RETURN AND DELIVERY JOURNEY OF GROUND BASED COURIER ROBOTS

By

Joshua Daly

Supervisor(s): Mr. Arnold Hensman

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
B.SC. (HONOURS) IN COMPUTING
AT
INSTITUTE OF TECHNOLOGY BLANCHARDSTOWN
DUBLIN, IRELAND
February 26, 2015

Declaration

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of **B.Sc. (Honours) in Computing** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfillment of the requirements of that stated above.

Dated: February 26, 2015

Author: ____________________________________

Joshua Daly

Abstract

Abstract should be clear, concise, and should cover the entire project in a fraction of the space, consider:

- Keep the word count low around 250 words.
- Avoid an jargon or ambiguous language.
- Do not use abbreviations.
- Do not reference anything.
- Briefly cover the motivation, problem statement, approach, results and conclusions.

Acknowledgements

Remember to thank the following people:

- My family and friends for putting up with me during the course of this project.
- Arnold Hensman for providing supervision and leading me into robotics.
- Tucker Balch for producing an excellent tutorial on grid based navigation.
- Sven Koenig for taking the time to respond to my queries.
- The Arduino, Raspberry Pi, and DFRobot communities for being awesome.

Table of Contents

List of Tables

List of Figures

Abbreviations

DARPA	Defence Advanced Research Project Agency
SLAM	Simultaneous Localisation and Mapping

Chapter 1

Core Implementation

1.1 Building the Project

1.1.1 Obtaining the Source

The latest version of the project's source code can be checked out via *git* using:

git clone https://github.com/swordmaster2k/botnav.git

Or downloaded as a ZIP file from `https://github.com/swordmaster2k/botnav`.

Alternatively the most update to date version at the time of printing is available on the CD at the front of this thesis.

1.1.2 Compiling the D* Lite Cython Module

The planning algorithm D* Lite must be compiled as a Cython module, the original source code was provided by Maxim Likhachev of CMU and Sven Koenig of USC in C. It has been modified to make it compatible with the core Python system using Cython, as Python is implemented in C [**?**] it is inherently compatible with the sample of D* Lite that is provided by its authors.

To build D* Lite you will need Python3.4, the Python3.4 headers, gcc, and make. It **must** be built for each platform on which it will execute as C compiles to machine code making it *target dependent*. From a terminal navigate to the source code directory *BotNav/algorithm/dstarlite_build/*.

```
cd BotNav/algorithm/dstarlite_build/
```

The make file contains two build rules:

1. *make* - which builds the module *dstarlite_c.so*
2. *make clean* - cleans all previous output files from the build process

Once the module file *dstarlite_c.so* has been successfully built for the target platform it can simply be dropped into the parent directory *BotNav/algorithm/*. The Python source code contains a reference to the module and will automatically link it in at execution time.

1.1.3 Running it in Python3

Explain how you go about running the project once it is compiled, by default it will run a simulation example. Give the command line arguments for running it from a terminal.

1.2 Running Simulations

Talk about how simulations provide an easy means to test the system against predefined use cases, speeds up the testing process, and provides a means for reproducing results. Discuss these points here.

1.2.1 Configuration

Outline the changes that need to be made to the configuration file, basically the simulation flag will need to be set.

1.3 Using a Real Bot

Mention how using a real robot differs by:

- The need for communications.
- Introducing drift wheel slippage etc.
- By proving the practical application of the planner.

1.3.1 Configuration

Outline the changes that need to be made to the configuration file, highlight the fact that communications medium is required USB, Bluetooth, or Wifi. Explain the different variations.

1.4 How the Planner Works

Explain how the planner was implemented in code, the control logic, looping structure, reacting to change, and the conditions for terminating the planner.

1.4.1 Five Simple Steps

Every planner in robotics splits the problem into five simple steps, include them under this section as numbered items. State any recursive operations that take place. Also include a flowchart or diagram in some form.

1.4.2 Abstracting Away from the Algorithm

This section will cover the abstract model that the Algorithm class enforces, every planning algorithm has a common interface which allows them to be interchanged. Explain how this is achieved using abstraction and talk about the advantages.

1.5 Open Field D*

Core of the project very important, state that every implementation of Field D* to date is closed source NASA's code is not available, nor is Carnegie Mellon's. Open Field D* is significant because it bucks this trend making it open to ITB students and others.

1.5.1 Modifying D* Lite

Point out the key differences between D* Lite and Field D* from a coding perspective, nodes to cell corners how this is represented, linear interpolation. Using Georgia Institute of Technologies D* Lite code state the modifications required to get Field D*.

1.5.2 Basic Implementation

Cover basic implementation of Field D*, most importantly state any problems encountered, or variations/optimisations made during the coding stage.