**Monte Carlo Localization Simulation**

**Release 1.0**

**User Manual**

Miralda Rodney, Savanh Lu, Stephen Kristin, Julian Pryde

24 April 2016

Embry-Riddle Aeronautical University

**Table of Contents**

**1.0 General Information**  (Savanh)

**2.0 System Summary** (Miralda)

2.1 System Configuration

2.2 User Requirements

2.3 Contingencies

**3.0 Getting Started** (Stephen)

**4.0 Using the System** (Julian)

4.1 User Input

4.2 Chart

**1.0 General Information**

The Monte Carlo Location Simulator is a two-dimensional simulation that allows a user to define an environment for the operation of a robot and its sensor. The user defines where the robot will start and end the simulation. The program randomly generates the point locations in the program. The user also specifies the range and accuracy of the sensors, the amount of error in the sensors, and the amount of error in the robot’s movement itself. The accuracy of the robot’s operation will be documented, displayed, and saved.

**2.0 System Summary**

The program is a java application with a graphical user interface. The application will run as an executable .jar file. To properly execute the program, the user needs to have the basic system requirements which are enumerated below.

**2.1 System Configurations**

The MCL program will operate on Windows operating systems in which there is a Java Runtime Environment (JRE) that allows for the execution of java application programs. If the system being used does not contain the JRE, the execution of the program will fail. To assure that the program will execute, the user needs to verify that they have the JRE required installed on their system by going onto the Java ORACLE website and downloading the most recent version of the java runtime environment.

To run the MCL program, the user must download the project zip file which contains the program JAR file. The user has to click on the JAR file without changing the configuration of the submitted file.

**2.2 User Requirements**

Users with the previously described system configurations will be able to execute the MCL simulation application.

**2.3 Contingencies**

The user needs to read the instructions provided under the “help” tab on the user input stage if they are in need of further assistance to perform the simulation.

**3.0 Getting Started**

When downloading the application, make sure to download the entire zip file and keep the file system intact. This is required for the application to run properly and for the file input and output to work. To start the application select the MCL.jar file. If the file will not run try updating your java runtime environment on your computer. If the problem persists please contact customer service at 555-123-4567.

**3.1 Program Start**

The program interface will pop up on screen upon successful load of the MCL.jar file.

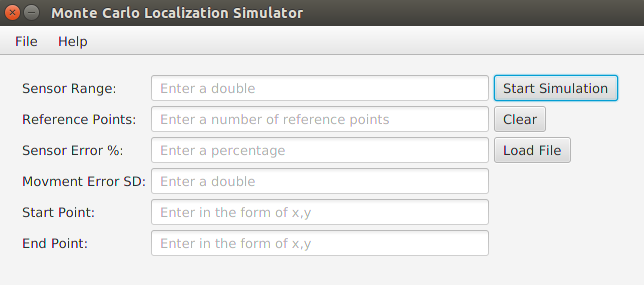
**3.2 Exit Program**

The program can be exited using either the X button in the top right for Windows users, or the X buttons in the top left for Linux and Macintosh users.

**4.0 Using The System**

**4.1 The Main Menu Interface**

The main menu contains most of the controls for the program. From here, the user can control the parameters of the robot’s movement and reference point detection as well as load a file from system memory and display a graph of the robot’s positions for that run.



***Figure 1: A screenshot of the main menu.***

**4.1.1 Sensor Range Text Field**

This field allows the user to control the maximum distance from itself that the robot can sense a reference point.

**4.1.2 Reference Points Text Field**

This field allows the user to choose how many reference points will be exit in the 100-unit-by-100-unit field.

**4.1.3 Sensor Error Percent Text Field**

This allows the user to control how precisely the robot will be able to detect reference points. Each of the robot’s detected points will be off from its actual point by the user-defined percentage of the range from the robot to the point.

**4.1.4 Movement Error Standard Deviation**

Each time the robot moves, it generates a random error on a Gaussian curve with a user-defined standard deviation and add that error to its intended position.

**4.1.5 Start and End Points**

This allows the user to define the starting and ending positions of the robot’s movement. The robot moves one unit along this path at a time and detects a new set of points at each stop along the way.

**4.1.6** **Start Simulation**

This button runs the program and opens a graph of the robot’s positions at the end.

**4.1.7** **Clear**

This button clears all information from the text-fields.

**4.1.8 Load File**

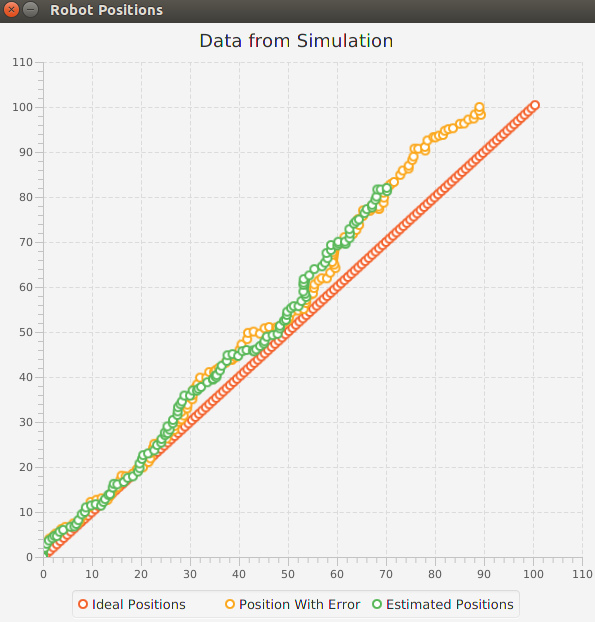
This button opens the interface to load and display information from a previous run.

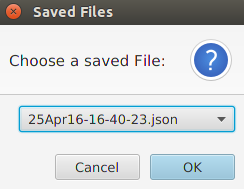
**4.2 Loading a File**

The load file function allows the user to load a log file from system memory and open a graph of the robot’s movements from that run.

**4.3 Graphing**

After each run and when the user selects a file saved in memory, the program opens a graph of the robot’s positions from that run.





***Figure 2: A screenshot of the file***

***choice interface***

***Figure 3: A screenshot a***

***graph of the data from a single run***