BRADLEY University

ECE497 Project Proposal

Project Title: Area Coverage Optimization using Heterogeneous Mobile Robots

Eric Jones and Dakota Adra
Advisor: Dr. Suruz Miah

September 27, 2018

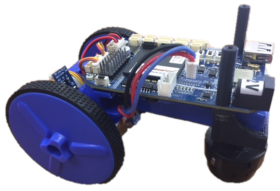
Contents

1 Introduction

Area coverage optimization has a variety of uses in robotic control applications. Most notability in roles where a robot is preferable to a human or when a task is simply impossible for a human to perform. Typical applications for area coverage optimization in the field of mobile robotics include search and rescue, surveillance, environmental monitoring, cooperative estimation, and indoor navigation, among others. These problems are usually solved using an array of networked mobile robots operating collectively. Recently the implementation of area coverage algorithms has been restricted to a fleet of homogeneous robots at high monetary cost. The purpose of this research is to lower the costs of entry due to the selected robot platform by providing an easily accessible framework for heterogeneous robots that can be implemented both expediently and efficiently. Note that we will be using the terms robots and agents interchangeably from now on.

This research is a continuation of a previous project conducted by Jacob Knoll and Dr. Miah wherein they implemented an area coverage algorithm with a time-invariant density in real-time. Density, in our project, refers to the importance of a particular point in a given two-dimensional plane. The current research seeks to extend this idea by implementing a density that is time-variant. In addition, we are looking to rigorously define a framework that multiple agents will use to perform this algorithm. We will be introducing a Multi-Agent Framework using Open-Source Software (MAFOSS) to standardize this and similar algorithm's implementations for open-source environments. Finally, as a stretch goal we will be using MAFOSS to allow a user operating a smart phone to provide input on the system regarding the location of the high-density points. Additional stretch goals include the implementation of external sensors and collision detection or mapping.

To conduct this research we will be testing the algorithm on a trio of mobile robots, such as the modified Beaglebone Blue eduMIP platform. This has already been developed by the project team members through our previous research. Figure 1a depicts the modified Beaglebone. Two other robots we will be using to conduct this research are the Pioneer 3-DX and the Khepra IV. These mobile robots have been designed by independent companies and are already in the Robotics Lab. These two robots can be seen in Figure 1b and Figure 1c, respectively. The Beaglebone, Khepra, and Pioneer are heterogeneous in the sense that they have different actuators, physical dimensions and processing capabilities. To test the heterogeneous case we will be using the Khepra, Pioneer and one modified Beaglebone. For the homogeneous case, we will be using three modified Beaglebones.



(a) Modified eduMIP

(b) Pioneer 3-DX

(c) Khepra IV

2 Milestones

We have decided to split up the implementation of the project into three main parts. A homogeneous case, a heterogeneous case, and various stretch goals that we will attempt to implement. In both the heterogeneous and homogeneous cases we will be testing the systems functionality with a static (time-invariant) and variable (time-variant) density.

- Homogeneous Case
  - Static Density
  - Variable Density
- Heterogeneous Case
  - Static Density
  - Variable Density
- Stretch Goals
  - Smart Phone Application
  - Interfacing External Sensors for Collision Detection or Mapping

3 Functional Requirements

3.1 High-Level System Block Diagram

As can be seen in figure 2 below the Heterogeneous area coverage system works by receiving a defined convex area, density sources, and agent positions from the user. The system then moves the robots around using the MAFOSS framework until they maximize their coverage metric or meet some coverage criteria. At this point, the agents will be in their final and optimal positions. Additional user input can be given via the smart phone application functionality such as, altering the density.

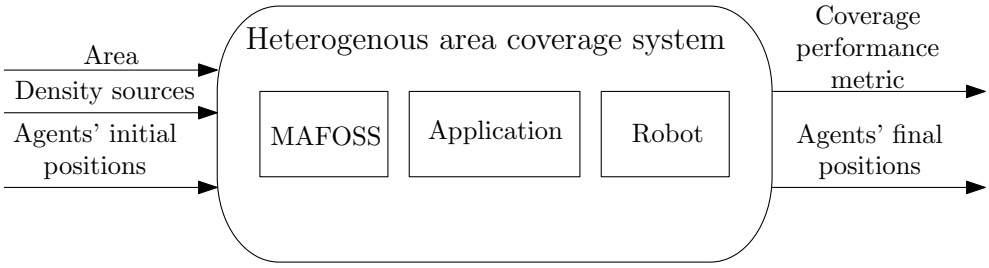


Figure 2: Functional Diagram

3.1.1 Inputs

The inputs for our system are the area that the robots will be operating in, as well as the locations of the density sources in this area. The robots initial positions are also given as this is crucial to the algorithm's operation.

- Area - The predefined two-dimensional operational area of the robots in our system. Note that this area must be convex for the algorithm to function.
- Density source - A simulated metric used to excite the system. In order to optimize their coverage the robots will try to cover regions that have higher densities. All other things being equal, the algorithm determines that a region of higher density requires coverage more than a region of lower density.
- Agents' initial positions - The agents' initial X and Y positions in a two dimensional X-Y plane.

3.1.2 Heterogeneous area coverage system

The heterogeneous area coverage system is the high level system that provides the networking infrastructure, agent movement, and application services.

- MAFOSS - A Multi-Agent Open-Source Framework that is used as the backbone for the multi-agent logistics.
- Application - The communication of a user's desired density to the system from a mobile device. Note that this sub-block of the Heterogeneous Area Coverage System is not required for it's operation.
- Agent - The mobile robot or vehicle that is being used to achieve area coverage.

3.1.3 Outputs

- Coverage performance metric - A description of how optimally the area is covered.
- Agents' final positions - The agents' final X and Y coordinates prior to the execution of the area coverage algorithm. The final positions of the robots should be at the centroids of their respective Voronoi regions.

3.2 System Architecture

- **User Interface** - The UI in the MAFOSS system framework takes user input from a smart device regarding the position of high density points and feeds that into the Application. This data is sent through the console, when the algorithm is running, and recorded in a ROS Log file.
- **Applications** - The Applications section relates to the control code running on the agents in the system. This code will be ran in a loop, sending and receiving the currentPose and desiredPose of the robots. The data transfer will be accomplished through the Robotic Operating System (ROS).
- **Robotic Operating System** - The Robotic Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. In our project we will be using ROS's networking abilities as well as it's ability to define individual agents as nodes to standardize the way we approach our algorithms implementation.

- **Control API** - The Control APIs in this architecture are each dependent on the target robot we are attempting to interface with. For the Beaglebone we use the well-documented *librobotcontrol* library, as it is specifically intended to be implemented on the Beaglebone with ROS. For the Khepra and the Pioneer, we will be using the proprietary interfacing firmware developed by their respective companies. The direct hardware interfacing will be done through the Linux Headers, as described in each API.

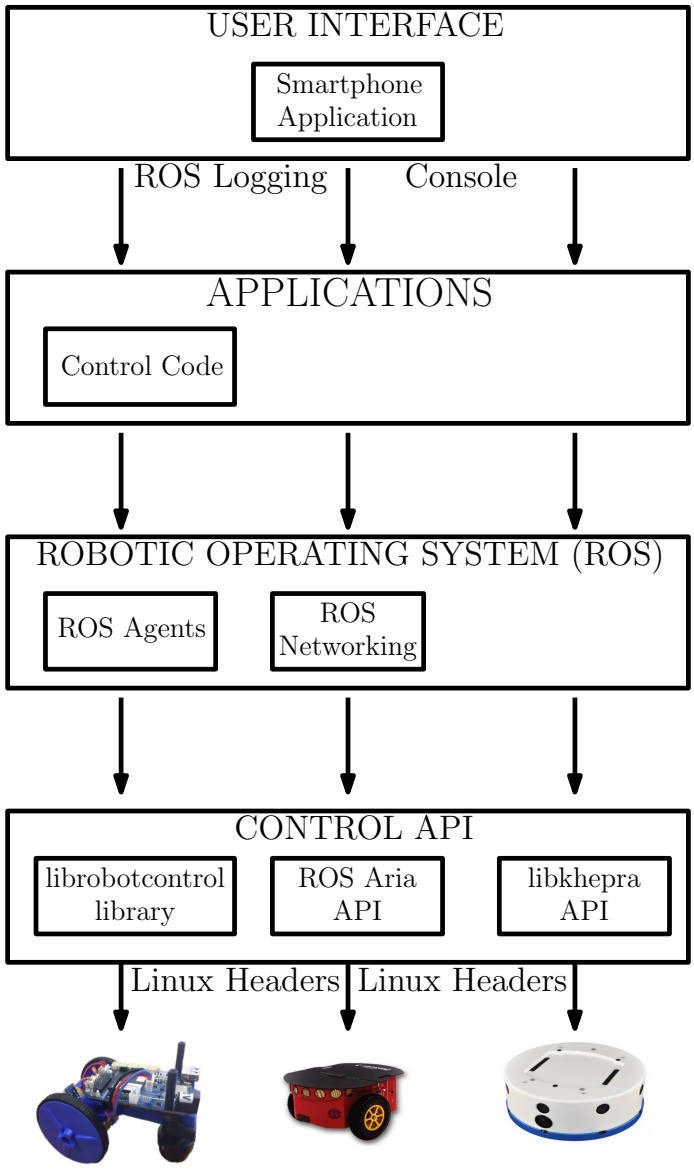


Figure 3: MAFOSS Software Architecture

3.3 ROS Networking

In the MAFOSS System architecture ROS handles the networking for us. In figure 4 we can see a detailed example of how ROS handles the connections between the agents.

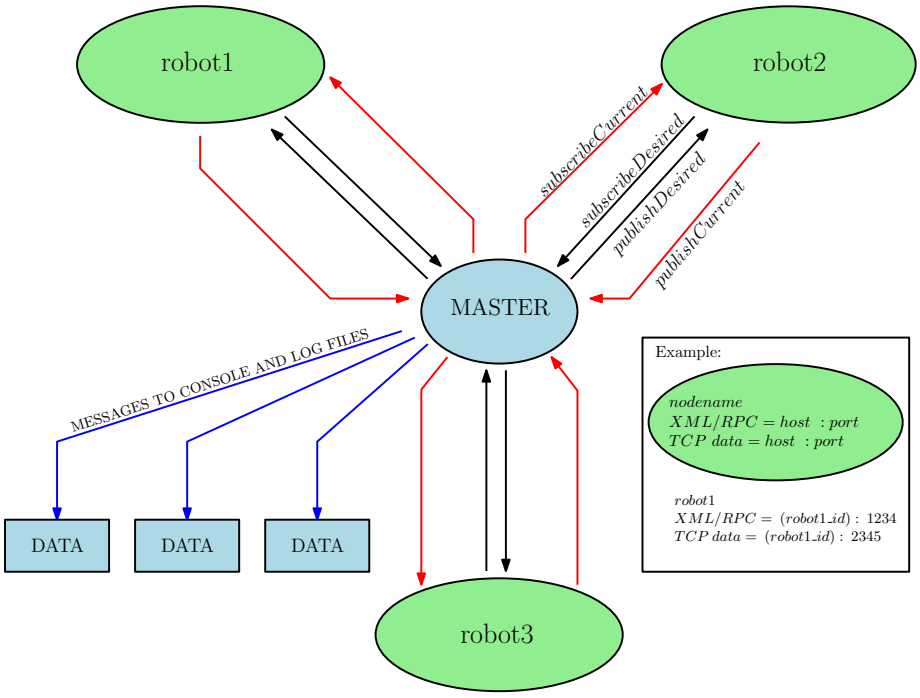


Figure 4: ROS Networking

- **Nodes** - Nodes are executables that uses ROS to communicate to other nodes through TCPROS across a corresponding Topic. In the above graph, the nodes in question are *robot1*, *robot2*, and *robot3*.
- **Topics** - Topics are named buses over which nodes exchange messages. In general, nodes are not aware of who they are communicating with. Instead, nodes that are interested in data *subscribe* to the relevant topic; nodes that generate data *publish* to the relevant topic. The graph lists the 4 topics associated with this system as *subscribeCurrent*, *subscribeDesired*, *publishCurrent*, and *publishDesired*. Each node communicating with the master in the system will get its own set of 4 topics. For robot one a full topic name could be *robot1PublishDesired*: this convention will be carried throughout the system independent of the number of agents.
- **Master** - The ROS Master provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes have located each other they communicate with each other peer-to-peer.
- **Output Data** - Output Data in our system will be sent to the laptop computer running the master node as the raw x and y positions. This data will also be sent to the corresponding ROS system log file for each node. For simulation purposes, we will be using the log files

created after running tests to create graphs relating our robot's desired position and it's actual position.

- **Example** - Note the example next to the graph. This gives a more in-depth view at what occurs inside the nodes on the graph. Given a publisher URI, a subscribing node negotiates a connection, using the appropriate transport, with that publisher, via XMLRPC (Extensible Markup Language Remote Procedure Call). The result of the negotiation is that the two nodes are connected, with messages streaming from publisher to subscriber. Each transport has its own protocol for how the message data is exchanged. For example, using TCP, the negotiation would involve the publisher giving the subscriber the IP address and port on which to call connect. The subscriber then creates a TCP/IP socket to the specified address and port. The nodes exchange a Connection Header that includes information like the message type and the name of the topic, and then the publisher begins sending serialized message data directly over the socket.

For our system, the above graphic gives a holistic view on its operation. Note that the ports specified to the TCP/IP sockets for the nodes are arbitrary. Further, the full URI for the master node will include the standard 11311 port.