Object Recognition and Path Smoothing Robot

Jake McKenzie
jake314@uw.edu

Alex Marlow
alexmarlow117@gmail.com

Ammon Dodson
ammon0@uw.edu

Prepared: 11/1/2018 Rev: 0.2.2

Revision History		
v0.1	10/18/2018	Initial Specification
V0.2.0	11/01/2018	Additional information on ROS
V0.2.1	11/01/2018	Added ethical concerns
V0.2.2	11/03/2018	Modelled the overall system architecture

Table of Contents

1. Introduction

The Object Recognition and Path Smoothing Robot is systems project with multiple objectives, they are, in sequence:

- Implement for verification Dr. McCourt's ~~path smoothing~~ delay mitigation algorithm.
- Demonstrate Simultaneous Localization and Mapping (SLAM)
- Implement a "finder robot" by integrating machine vision with the SLAM functionality

Additional objectives may include:

- Using beacon triangulation and/or GPS to fuse additional location information into the SLAM or finder functionality
- Implement functionality to allow the system to report back on mapped spaces or found objects

Remote Control

Remote control systems have many useful applications; The typical use case involves using robots to interact with an environment that is too hazardous for a person. Any such remote control involves some delay of both outgoing control signals and incoming sensor data. In some situations, this delay may impair the intended function of the remote system.

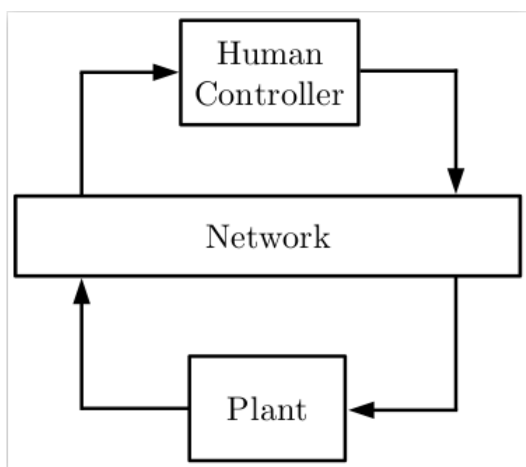

Figure 1: A typical remote-control system

Dr. McCourt has developed a set of filters intended to be placed in such a delayed, closed-loop control system. These filters apply a mathematical transformation on both incoming and outgoing loop signals such that ~~timing~~ communication delays are mitigated.

Autonomous Control

There may also be use cases for robots in hazardous environments where direct human control is impossible. Such a robot must be able to autonomously navigate and interact with an unknown space. SLAM is a fundamental technology for such autonomous activity, allowing the robot to navigate. Additionally, such an autonomous machine must be able to sense and recognize an objective before being able to interact with it. Computer vision is another fundamental technology for sensing a real-world environment. Sensing a condition is a necessary first step for being able to take action based on the current environment.

2. Requirements

This robot will use the TurtleBot3 Burger as a base. All the hardware and software modules of the remote component must fit within the constraints of that platform.

2.1. Remote Control

In order to demonstrate the McCourt filter the robot must be remotely controllable. Control signals must be routed through a delayed communication medium. Ideally, or as a second stage, the human controller's feedback data should also be routed through the delayed medium.

The Human controller we have available currently is a USB game controller. This controller will have to be connected to some base station that can connect to the TurtleBot3 wirelessly.

The McCourt filter requires processing hardware available at both ends of the wireless communication. Presumably, because this filter is some sort of matrix transformation, it may be preferable that the filter hardware have SIMD vector processing instructions.

2.2. Autonomous Control

In autonomous mode the robot should be able to completely explore and map a real-world environment. Potentially it should also be able to identify, locate, and report on the location or condition of some object in the environment.

2.3. Scope

This specification covers the following:

- Any and all hardware modifications to the TurtleBot3 Burger
- Software installed on the burger bot insofar as it deviates from the stock installation
- Base station control software setup insofar as it deviates from stock installation
- Any necessary techniques for integrating a Human Interface to the base station installation
- Any methods used to implement a communication delay between the robot and its base station
- Implementation of the McCourt filter
- An overview of implementing SLAM on the TurtleBot3 Burger
- Integration of OpenCV into the TurtleBot3 Burger

3. System Architecture

3.1. Robot Operating System (ROS)

ROS is a middleware library that creates interfaces for many common robot components and allows them to communicate in standardized ways. Each ROS process is called a node. Nodes communicate through topics or services. A service provides a classic server-client model where the client makes a request and the server respondes. A topic implements a logical many-to-many data bus. Nodes can publish to, or subscribe from any topic without knowing anything about other nodes on the topic.

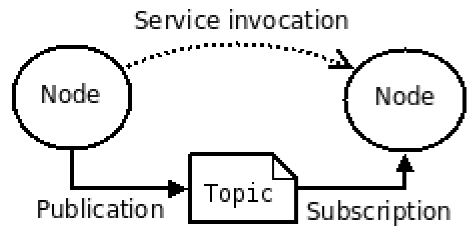

Figure 2: ROS basics. From: http://wiki.ros.org/ROS/Concepts?action=AttachFile&do=view&target=ROS_basic_concepts.dia Used under Creative Commons Attribution 3.0

3.2. The McCourt Filter

The control signals from the human controller will take the form of a 1x2 integer vector indicating speed and direction.

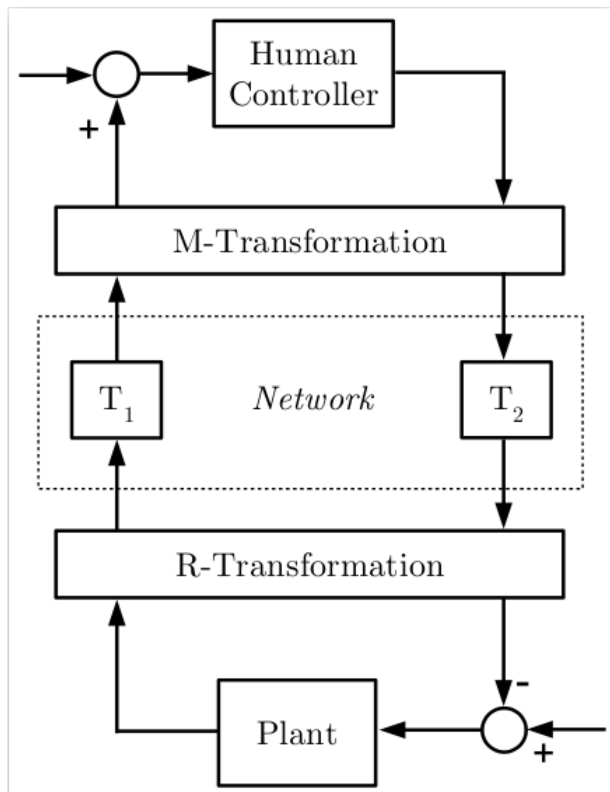

Figure 3: The McCourt Filter

3.3. Human Controller

The human interface device availible is a commercial game controller. ROS has a node that is capable of publishing the current cotroller state as a topic.

3.4. TurtleBot3 Burger

4. System Design

4.1. Hardware Design

Our hardware is largely preset

4.1.1. Objectives

4.1.2. Constraints

- Raspberry Pi 3 (32-bit ARM Cortex-M7)
- OpenCR (32-bit ARM Cortex-M7)

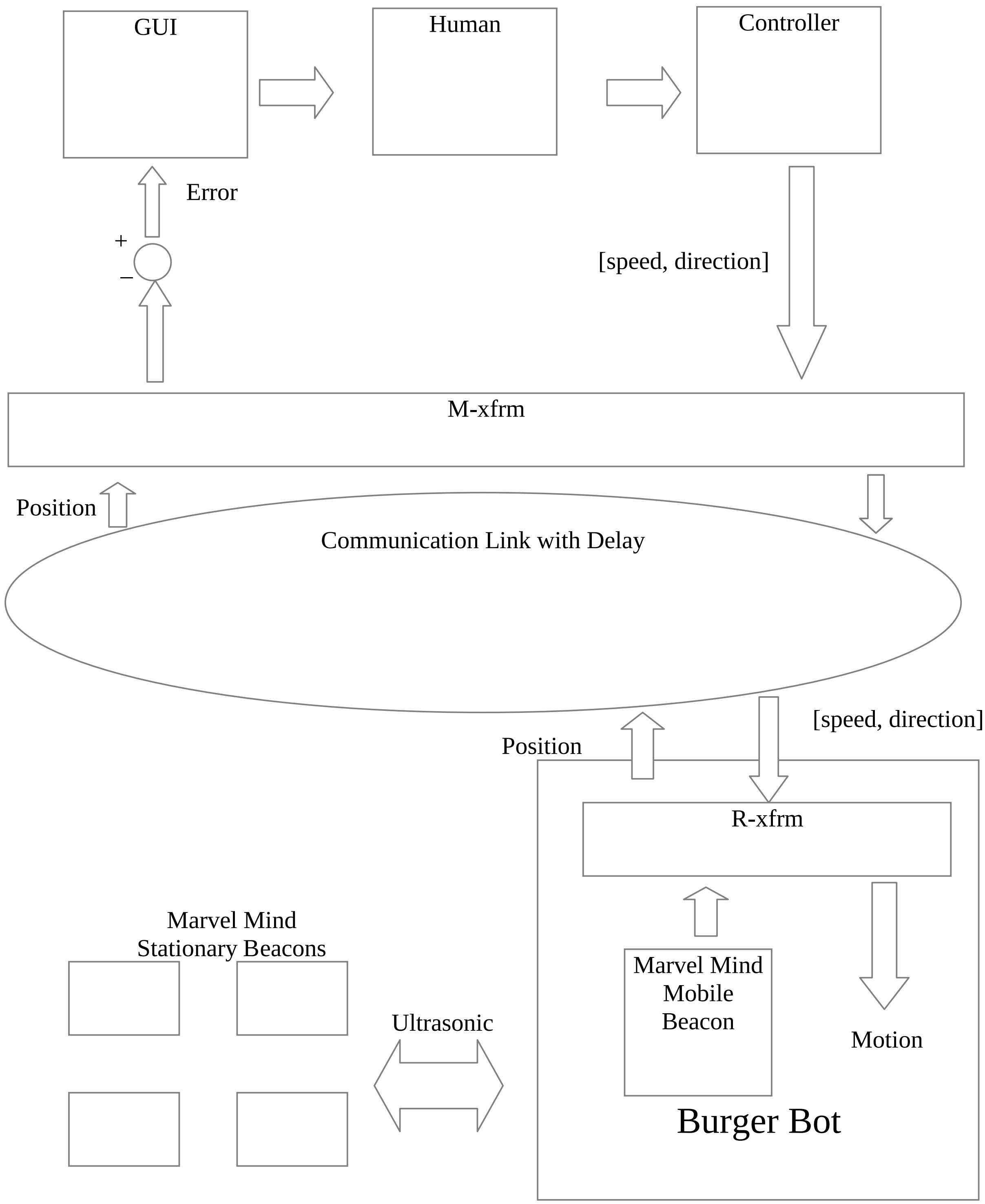

Figure 4: Overall system architecture for the ORPS Robot

4.1.3. Composition

4.1.4. Uses and Interactions

4.1.5. Interface

4.1.6. Resources

4.1.7. Details

4.2. Software Design

4.2.1. Objectives

4.2.2. Constraints

4.2.3. Composition

4.2.4. Uses and Interactions

4.2.5. Interface

4.2.6. Resources

4.2.7. Details

4.3. Human Interface Design

4.3.1. Functionality from the User's Perspective

When in remote control mode the robot can be controlled by a consumer game controller.

4.3.2. Interface Objectives

4.3.3. Interface Constraints

4.3.4. Use Cases

5. Ethical Considerations

In this project, our specific goals do not raise many ethical concerns, however, there are still things that could happen from this project that should be considered.

Future use of McCourts filter: The primary purpose of McCourt's algorithm is to help with the delay of human interacting in closed loop control systems. This algorithm could be used for military systems or other wepon systems. This is not directly our concern as McCourt's algorithms are not ours, but our work in enabling it could be considered helping. This concern is mainly dependent on the motives and intensions of the IP owner. With our countries current IP laws and our confidence in the owner of the IP we are not in any way currently worried about this ethical concern.

Complete testing: McCourt's filter could be use in the future in systems that could cause harm if they were to malfunction. Because our testing is a proof of concept, it is partly our responsibility to make sure this algorithm works in practice, so there is little chance of malfunctions in the future of this algorithms life.

Use of funds: This project is funded by Dr. McCourt who has funding from the University of Washington. We did not technicaly earn this money, so it is our responsibility to use this money efficiently, and to not cause money to be spent due to mistakes.

Everyone loves this project. It will be really good for America.

6. References

1. Turtlebot3 Manual: http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/
2. Robot Operating System: http://www.ros.org/
3. Open Source Computer Vision Library (OpenCV): https://opencv.org/

7. Errata

There is no previous version of this document.