Software Requirements Specification (SRS)

https://cshadd.github.io/fetch-bot/

Project Fetch Bot

Authors: Christian Shadd, Joseph Damian, Thanh Vu, Maria Verna Aquino, Giovanni Orozco

Customer: Prof Taehyung Wang

Instructor: Prof Taehyung Wang

1 Introduction

This Software Requirements Specification document provides an overview of the functionality of the Fetch Bot project. This document will cover the scope, organization, description, constraints, requirements, and the prototype of the Fetch Bot project.

1.1 Purpose

The purpose of this document is to describe the functionality and specifications of the design of the Fetch Bot project. The expected audiences of this document are the developers and end users of the application.

1.2 Scope

The project is designed to generate an AI robot called Fetch Bot with both hardware and software. The Raspberry Pi will control most of the software and the Arduino will control some of the hardware. The end users will be able to control the robot from a remote computer connecting to a web interface from the Raspberry Pi. The web interface and program are stored on an Apache HTTP Server that connects to a Java program locally on the Raspberry Pi.

1.3 Definitions, Acronyms, and Abbreviations

AI (Artificial Intelligence) -capability of machine to imitate intelligent human behaviour.

Hardware - tangible components of a computer system.

Software - intangible components of a computer system.

Raspberry Pi- a low-cost single board computer.

Arduino- open-source electronics platform or board, including the Arduino software

Web Interface - consists of web server and web page. Interaction between user and software running in the server.

Computer - electronic device that stores and processes data.

Java -a general-purpose computer-programming language.

JSON (JavaScript Object Notation) - a lightweight data-interchange format

OpenCV - an open-source library used mainly for real-time computer vision.

Apache HTTP Server - a free and open-source cross-platform web server

Robot - a machine capable of carrying out a complex series of actions automatically

Live Stream - to transmit or receive live video and audio coverage over the Internet

Verbose - how much detail is logged or sent to the console when needed for debugging

1.4 Organization

The remaining portions of this document are decomposed into four major sections, followed by references and a point of contact. The first section will provide an overall description of the project and the next part will give more detailed requirements. Following the requirements, there are models describing the project and the description/use of the robot.

2 Overall Description

This section provides a high level description of the entire application. It describes the product perspective, functionality, and characteristics of an expected user, constraints, assumptions and dependencies.

2.1 Product Perspective

This project is constructed respectively and independently for the specific processors. There needs to be an Apache HTTP Server for JSON communications between the web interface and the main program. The web interface will have a smooth design to it that incorporates full control of the program.

2.2 Product Functions

Provides a simple AI robot that runs off of OpenCV for image processing which can process off of a live stream. Provides a web interface for controlling the robot. The end user can easily access the web interface and control the robot whenever they choose to do so.

2.3 User Characteristics

The user should be familiar with the basic functionality of the web interface. He/She should be familiar with the different buttons: Auto, Manual and Idle that set the robot's mode. As well as the kill button that will end the robot's execution. He/She should be able to provide the robot with a target from a list of available pre-trained ones in the web interface. The user should finally be able to start the robot by typing the command "bash/install.sh" in terminal.

2.4 Constraints

The Raspberry Pi has limited memory and space. The OpenCV image processor needs to be pre-trained and may consume memory and space during processing as the neural network needs memory to learn. Another area of concern is timing. The commands may have a few seconds of delay for the processor to transmit data back and forth. Another constraint is the concurrency. Currently there should only be one end user controlling the robot. Finally this project runs off of specific hardware so unless the end user is willing to make changes within the repository to accommodate other hardware, there will be errors when compiling onto different hardware due to native libraries not being included. To accommodate this, the end user may change the libraries needed.

2.5 Assumptions and Dependencies

The project has multiple features that we will utilize. The Raspberry Pi incorporates most of the features needed through a package distribution manager. The package distribution manager can install most of the dependencies. Our project will use all of these and should work on the specified hardware given.

Really all the end user will need to do is input commands through the web interface after setting all hardware up. As long as the end user follows the proper installations per each dependency and hardware setup, they should get a functioning robot.

2.6 Apportioning of Requirements

Depending on deployment, we may delay the usage of a full pathfinding AI. However the neural network AI will still come into play. This is due to the time constraints given on the project that the Alpha “Cinnamon” version may not incorporate.

3 Specific Requirements

This section provides further details on the specific requirements of our robot. Each requirement is given along with a description of the requirements.

1. The system shall be able to start up. During start up, the bash script, “bash/install.sh” is ran in the terminal. The bash script sets the monitor to never turn off. The script also kills any existing Java processes and starts the jar file which is the main program itself. It then opens up the chromium web browser on the monitor. It starts the program that keeps the mouse cursor turned off on the monitor to get the screen to only display the face of the robot.

2. The system shall be able to turn off. Once kill the button is pressed on the web interface, a log file is created. Serial ports are disconnected and the jar file is terminated.
3. The system must be able to recognize a target. In order to recognize a target, the robot is set to autonomous mode. Once the robot is set on autonomous mode by clicking the Auto button on the web interface, the robot uses the neural network on the raspberry pi to match images to preloaded data. It processes the image and if a 90% match is obtained, the robot reports the image as a target. If any error occurs for the neural network or the target being found, the error is saved to a log file and the user is notified.
4. The system must be able to calculate a path towards the target. In order to calculate the path to the target, the robot is set to Auto mode. During auto mode, if the target is not found, it chooses where to go next and generates a graph as it moves around. Once the target is found, a straight path is generated towards it and is followed. If an obstacle is detected, another path is calculated and the graph is updated. This repeats until the target is reached. If any error occurs during the calculation and setup, the error is saved to a log file and the user is notified.
5. The system must be able to move. The robot can move in both Manual and Auto mode. In order to move during Manual mode, the user can choose to press the move buttons: Forward, Left and Right in the web interface in order to move the robot. In order to move in Auto mode, the user only needs to specify a target from a list of pretrained data in the interface. If the motors fail or the processor was not able to receive data, the error is saved to a log file and the user is notified.
6. The system must be able to detect obstacles. Data is received from the sensors and the arduino sends this to the raspberry pi via Serial which then gets sent to the jar file in the raspberry pi. The raspberry pi processes the data and interprets it into distances that will be sent to the web interface.

Index