Requirements Specification

for

Visual SLAM Rover

Version 1.0

Prepared by Tyson Mergel, Jared Perlic, and Erik Sanders

Brigham Young University - Idaho

18 February 2023

Table of Contents

1. Introduction

1.1 Purpose

The goal of this project is to create a rover that can be used for exploration and/or surveillance. To that end, the rover, outfitted with two cameras and a Raspberry Pi, will generate a depth map using the Visual SLAM (Visual Simultaneous Localization and Mapping, aka vSLAM) algorithm.

1.2 Scope

For this project, a small rover about the size of an RC car will be constructed. It will move on the ground using two motors. However, it will not be capable of traversing rough terrain. A Raspberry Pi and two cameras will be installed. Near-ground level feed will be captured by the cameras. The Raspberry Pi will serve as the transmitter, receiving commands from the controller and sending data from the cameras. The controller will serve as the "brain," running the vSLAM algorithm and generating a depth map that provides information about the nearby area to the operator.

1.3 Overview

This project, named Visual SLAM Rover, is for CSE 499 - Senior Project. The project managers are Tyson Mergel, Jared Perlic, and Erik Sanders. This project does not have any sponsors.

This project is expected to take 13 weeks to complete, with each person contributing an average of 9–10 hours per week.

This project has been divided into three stages for planning purposes. The first stage of the project will be to build the rover and connect it to the controller. The second stage will be to get camera feed. The third and final stage will be to perfect the vSLAM algorithm in order to generate the depth map.

1.4 User Profiles

Users of the project will be able to control the rover directly by using an Xbox controller. Users will also receive a map of the rover's surroundings as output. The project can only be used by one user, or group of users, at a time because only one rover is being made. Project users will need basic motor skills to operate the controller, some degree of spatial awareness to interpret the output, and a rudimentary knowledge of how to use a computer.

1.5 Standards

All Python code used in the project will conform to PEP 8 standards. For tracking and managing changes to the software, Git will be used, and all code will be centralized in a GitHub repository. The design will follow the engineering design process, a common series of steps that engineers use to create functional products and processes.

2. Requirements

2.1 Motor Control

The rover’s left and right motors shall be controlled using an Xbox controller. This requirement is satisfied when the rover moves in response to a user-operated Xbox controller.

2.2 Video Capture

The rover shall capture video with a pair of cameras and transmit a live feed to the operator. This requirement is met when the operator receives live feed from the cameras.

2.3 Feature Matching

The software shall use an ORB detector and brute-force matching to perform feature matching. This requirement is satisfied when the software identifies at least ten feature matches between the two cameras for each input frame.

2.4 Feature Projection

The software shall calculate the position of matched features in 3-dimensional space. When the software precisely positions each matched feature in 3-dimensional space, this requirement is satisfied.

2.5 Object Detection

The software shall detect objects using Haar Cascades. This requirement is met when the software correctly identifies nearby objects 80% of the time.

2.6 Depth Map Generation

Using the projected features as a reference, the software shall generate a depth map for a single frame of the run. This requirement is met when the software creates an accurate depth map for the selected frame.

3. Stretch Requirements

3.1 Map Generation

Using the projected features as a reference, the software shall generate a map or point cloud of the nearby area. This requirement is met when the software creates an accurate map of the rover’s surroundings.

3.2 Autonomous Mode

When chosen, the rover will move autonomously rather than under manual control. This requirement is satisfied once "Autonomous Mode" has been successfully executed.

3.3 Rough Terrain Traversal

The rover shall traverse rough or uneven terrain while still performing its other functions. This requirement is satisfied when the rover successfully navigates obstacles.

3.4 Object Counting

During each run, the software shall count the number of identified objects. This requirement is met when the software correctly counts the number of identified common objects.

4. Design Overview

4.1 Workflow

4.1.1 Clone Repository

The user will navigate to the repository page on GitHub and clone the repository onto their machine.

4.1.2 Download Dependencies

The user will download all dependencies necessary for running the software. A current list of dependencies is maintained in the requirements.txt file.

4.1.3 Output

The user will obtain a depth map containing a map of the rover's surroundings.

4.2 Data at Rest

User information will not be stored, nor will the user require a profile to access the software or operate the rover. The output will be a depth map containing a map of the rover's surroundings.

4.3 Data on the Wire

After the completion of the run, the output file will be saved in the Pointclouds folder of the repository. It will be up to the user to decide what to do with the output file—to save or delete it.

4.4 Graphical User Interface

The software will not contain a Graphical User Interface (GUI). Instead, the user will see a live feature-matched feed from the two cameras. At the completion of the run, the user will navigate a 3-dimensional map of the rover's surroundings.

5. Verification

5.1 Demo

A 3- to 5-minute video of the rover using the software will be produced to show the rover's functionality and demonstrate the vSLAM algorithm. The video will be shot in a medium-sized space with adequate room for the rover to maneuver. The requirements outlined in this document will be turned into a checklist and checked off as the demo progresses.

5.2 Testing

The list of requirements will be transformed into a checklist. As requirements are tested, they will be compared to the checklist and either passed or failed. For the specifics of each requirement, see the Requirements and Stretch Requirements sections.