Software Overview

Year: 2022 Semester: Fall Team: 05 Project: Metaporter

Creation Date: September 7, 2022 Last Modified: December 15, 2022

Author: Kris Kunovski Email: kkunovsk@purdue.edu

Assignment Evaluation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Score (0-5)** | **Weight** | **Points** | **Notes** |
| **Assignment-Specific Items** | | | | |
| **Software Overview** |  | x2 |  |  |
| **Description of Algorithms** |  | x2 |  |  |
| **Description of Data Structures** |  | x2 |  |  |
| **Program Flowcharts** |  | x3 |  |  |
| **State Machine Diagrams** |  | x3 |  |  |
| **Writing-Specific Items** | | | | |
| **Spelling and Grammar** |  | x2 |  |  |
| **Formatting and Citations** |  | x1 |  |  |
| **Figures and Graphs** |  | x2 |  |  |
| **Technical Writing Style** |  | x3 |  |  |
| **Total Score** |  | | |  |

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:

1.0 Software Overview

Metaporter will consist of a single handheld plug-in device with one firmware component. The firmware for this project will record a video of the subject, collect LiDAR sensor data, display the time elapsed, and accept user input on a 4x4 keypad.

The architecture of our microcontroller software most closely resembles a state machine. The microcontroller and its peripherals are initially configured during startup. Afterwards, the MCU waits for keypad input from the user. The possible options to choose from are calibration, data collection, and process termination. When a command is selected, an interrupt will get triggered to enact the subsequent functionality based on the button that is pressed.

Provided that the input is a calibration command, the microcontroller sets the corresponding register of the LiDAR sensor through I2C, and the sensor initiates the calibration processes. The LCD interface reflects this procedure by displaying “Calibration In Progress.” When the calibration process is complete, the software writes “Calibration Complete” to the LCD through SPI, and the device changes to a ready state and waits for user input.

Presuming the data collection command is chosen, the sensor is triggered to begin gathering data, and the Jetson nano is sent the instruction to initiate the camera’s recording. We are using an interrupt-driven architecture since we are working with multiple peripherals at once. A time-based interrupt will be used for the LiDAR, and once triggered, data will be read from the LiDAR using I2C and transferred to a buffer in memory. The LiDAR transfers two bytes of distance data to the MCU at a frequency of approximately 1000 Hz. If we choose to use an IMU, the same process will be used for this peripheral as well. In addition, a separate timer will raise interrupts to update the LCD interface. A final timer-based interrupt will trigger DMA transfer from memory to UART, and the Jetson nano receives this data on the other end. Meanwhile, the Jetson nano acquires the camera data using ROS publisher and subscriber nodes. As the collection process continues, the sensor data undergoes fusion with the sliced video frames. After the fusion, the sensor data is used by a SLAM algorithm to compute the camera pose (see the Description of Algorithms for more details). The camera pose and corresponding image is sent to an NVIDIA GPU-enabled computer using ROS nodes over Wi-Fi. This process is continued as long as data is being collected (until abort), and following the completion of collection, the Jetson nano sends a signal to the microcontroller to stop the timer indicating the time passed. On the host computer and the Jetson nano, the images will be stored in a directory, deleted, and then replaced with other ones for each new run. The data is run through NVIDIA’s Instant NeRF library [3] to train a model in a matter of minutes, and the 3D reconstruction will be presented in the NeRF GUI.

If the abort instruction is selected by the user, the microcontroller software ensures the peripherals stop gathering data. The LiDAR sensor immediately halts writing to its distance data register, and a signal is sent to the nano using UART in order to terminate the camera’s recording. To reflect these changes, the timer controlling the elapsed time is also halted which stops updates to the LCD interface.

2.0 Description of Algorithms

Metaporter requires the implementation of two primary algorithms. The first algorithm necessary is SLAM or simultaneous localization and mapping. The Jetson nano will use a ROS package called ORB-SLAM [1]. This package will use basic SLAM features along with a survival-of-the-fittest approach for accurate point estimates and the reconstruction of a trackable map.

Sensor fusion will be an essential step in recreating a 3D model of our subject. For this process, we will use an Extended Kalman Filter (EKF) in MATLAB [2] to get the most accurate fusion data for our localization and mapping algorithm.

3.0 Description of Data Structures

There is one particular data structure necessary to ensure cohesive data transmission from the microcontroller to the Jetson nano through UART. A header pertaining to each block of data transmitted on the UART is shown below in Figure 3.1. The header must contain a flag for an input command, a data type, and the number of data (depending on type). If the command flag is not set, the nano will understand that data transmission will occur for the amount of data specified in the second byte. The type of data needs to be distinguished from bytes, ints, longs, and floats in order for the nano to keep track of each dataset for processing, so this is set in the four most significant bits of the first byte. By including a second byte for size, the nano knows how much data to expect from the microcontroller since UART can only send byte-by-byte. If a command is set in the flags, the nano will accept the instruction and fulfill the corresponding functionality. For example, the command to calibrate would transmit 8’b00000001 in the first byte and a zero in the second byte. This formatting strategy would allow for future commands to be created, if necessary (i.e. up to 24 possible commands).

A picture containing header diagram

Description automatically generated

Figure 3.1 – Header

4.0 Sources Cited:

[1] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.

[2] “Extended Kalman Filters,” *Extended Kalman Filters - MATLAB & Simulink*. [Online]. Available: <https://www.mathworks.com/help/driving/ug/extended-kalman-filters.html>. [Accessed: 09-Sep-2022].

[3] J. Stephens, “Getting started with Nvidia instant nerfs,” *NVIDIA Technical Blog*, 01-Aug-2022. [Online]. Available: <https://developer.nvidia.com/blog/getting-started-with-nvidia-instant-nerfs/>. [Accessed: 10-Sep-2022].

Appendix 1: Program Flowcharts

Main Program

*Diagram

Description automatically generated*

Data Collection

*Diagram

Description automatically generated*

Calibration Abort

*Diagram

Description automatically generated*

Appendix 2: State Machine Diagrams

*Diagram

Description automatically generated*