**Definitions:**

* Platform:
* Protocol: a given set of comma
* Operating System: the base system running on the platform that allows users interface (Windows or Linux).
* Firmware: an embedded program on a platform that will not change, written in C or Python or both.
* Software: the program that is executed on the platform, written in C or Python or both.
* Hardware:
* RTOS: Real time operating system.
* Augmented Reality:
* Media: the material that is used to partially fill the volume of the enclosure.
* Gradient:
* BSP: Board Support Package
* Global variable:

**Problem Statement:**

To develop software that can capture topographical data by use of an RBG+D camera in real time.

**Project Goals:**

* To control various pieces of modern hardware using Python.
* To apply machine learning & computer vision concepts by exploiting the hardware/software interface of a host platform to analyze a system in real time.
* To aggregate data and display concurrently in a real time system.
* To develop a system whose latency is below the detectable threshold for human eyes, making it appear as if changes are happening simultaneously (I.E. augmented reality).

**Constraints/Restrictions for Simplicity:**

The following is a list of assumptions or constraints that we are using to implement our software in a controlled environment:

* Camera will be fixed and mounted above/over a region that can be manually changed to show efficacy
* Media will be easy to modify and shape to show the changing contours in real time, simulating panning a region
* There will be a secondary process to show the processed data, which will be displaying topographical contour lines on region from a projector mounted above/over the region

**General Operation:**

This device is an augmented reality sandbox used to demonstrate the capabilities of real time topographical mapping of a static environment. The structure will have a large internal volume that is partially filled with a fine sand-like material that is free from any/all debris. The structure will be supported by a mechanically sound “cart” with 4 wheels for easy moving of the entire setup (two caster, two rigid bi-directional). Above the sandbox will be a cantilevered structure that supports both a RBG+D camera and a digital projector both of which are fixed to the structure. Some minor built in adjustments may be implemented to allow for easier troubleshooting of the system during development but will be fixed and rigid for demonstrations. Cabling will be run from the sandbox to an adjacent laptop computer which will be running the software controlling the system. Once the system is powered on, the camera will take a depth measurement of the entire sandbox (z-axis height of the sand at all locations contained within the structure only, the program will ignore any data retrieved from outside of this area).

The data will then be parsed into levels of similar height which will be determined from the z-axis step size for grouping of similar data. The data will then be assigned a color based on its groupings and a line segment will be drawn linearly between each of the data points in the same group of that particular color. This will occur for all groupings of data for each color (red to blue). Once all the data has been parsed, the final contour image will be projected back down onto the media in the structure. This process will be in a loop routine and occur over and over again until user input is received to stop. The timing of this loop for processing the data is the most critical aspect since we require that this appear as if it is happening in real time. At any time during this processes if the user wishes to capture the current environment (I.E. the current topology of the sand), a keypress can be made on the computer which will save the entire data set to a file. This will then be immediately processed using either Matlab or Python and will 3D plot the data onto the screen which the user can then save. From this point, the saved data set can then be sent over to a connected 3D printer where it will initialize a print of the saved workspace from ABS material. Due to useable printer platform sizes, the 3D print may be scaled to a fraction of the original data taken.

|  |  |
| --- | --- |
| **Requirement Number/Requirement Specification** | **Requirement Description** |
| **1. Hardware** | |
| 1.1 Platform | Platform must support required peripherals (see section X). |
| 1.2 RBG+D Camera | Camera must be capable of taking depth data with a minimum resolution of 1cm. |
| 1.3 Digital Projector | Digital projector must have HDMI connection and have a minimum display resolution of 1080p. |
| 1.4 Cabling | Cabling will be purchased through an approved supplier and will not inhibit the overall data speed transfer rates set by the platform or other hardware. |
| 1.5 3D Printer | 3D printer must be capable of taking in X, Y, Z coordinates or point cloud information and printing in ABS material with a minimum resolution of 0.5 mm. |
| 1.6 RTC | Real time clock will be on the platform to be used as a time stamp for any offloaded or saved data sets taken from the device. |
| **2. Software** | |
| 2.1 Operating System of platform | Operating system will be up to date to manufacturer’s current distribution. |
| 2.2 Operating System of 3D Printer | 3D printer will be up to date to manufacturers approved firmware version |
| 2.3 BSP’s | BSP’s will be used to hasten development time if they are currently supported by the manufacturer. |
| 2.4 3D Plotting Software | Matlab or Python will be used to plot the changes of the sandbox at set intervals determined in the software. |
| **3. Language** | |
| 3.1 Embedded Software Language | All software will be written in either C or Python or a combination of the two. |
| **4. Protocols** | |
| 4.1 HDMI | HDMI will be used to display the information from the platform to the projector and onto the surface. |
| 4.2 USB 3.0 | USB 3.0 will be used to transfer data from the RBG+D camera to the platform |
| 4.3 SPI | SPI will be used if required to communicate to peripherals through full duplex. |
| 4.4 I2C | I2C can be used as an alternative solution if there are a large number of hardware that are more easily addressable in this fashion. |
| **5. Platform** | |
| 5.1 Power | Platform will have a power supply sufficient to power itself and must be able to plug into AC mains 110-125V. |
| 5.2 Peripherals | The platform must have the following peripherals: HDMI, USB 3.0, SPI & I2C. |
| 5.3 Performance | The platform must have at least one core. |
| 5.4 Processor Speed | The platform must have a minimum processor speed of 1 GHz. |
| 5.5 |  |
| **6. Overall Dimensions** | |
| 6.1 Exterior Dimensions (Prototype) | The overall outside dimensions of the prototype will be 12” x 12” x 6” ± 0.5” (L x W x H). |
| 6.2 Interior Volume (Prototype) | The overall interior volume of the prototype will be no greater than 864 (12” x 12” x 6”). |
| 6.3 Exterior Dimensions of the final design | TBD. |
| 6.4 Interior Volume of final design | TBD. |
| **7. Media** | |
| 7.1 Particle size | Material selected must be finer than 1mm. |
| 7.2 Gradient | Particle size must be fine enough to produce a smooth gradient on the surface. |
| **8. Speeds** | |
| 8.1 Refresh rate |  |
| 8.2 Program execution time |  |
| 8.3 Latency |  |
| **9. Power** | |
| 9.1 Main power | Power will be 110-125 VAC provided from mains via a wall outlet. |
| 9.2 Battery power | This revision of the device will not be designed to operate off of any battery power. |
| **10. Z-axis Step Interval** | |
| 10.1 Z-axis step interval | Will be a #define or similar global variable that can be adjusted in one location and will change everywhere in the program. |
| **11. Topographical line color palate** | |
| 11.1 Topographical line color palate | This will red to blue ranging from the minimum to the maximum and the color transition from one line to the next will be determined by the Z-axis step internal (Requirement #10). |
| **12. Optimization** | |
| 12.1 Program optimization | Techniques such as loop unrolling and the use of intrinsics may be used to speed up the execution time of the program if required or desired for the appearance of real time. |
| **13. 3D Printer** | |
| 13.1 Usable printer area | Due to the constraints of the 3D printer’s usable workspace, the rendered print may need to be a scaled down version of the original data set. In this case, a secondary data set will be generated from the original which will be the reduced data necessary to fit on the printer’s platform. |
| **14. Cost** | |
| 14.1 Total Cost of the Project | TBD. |
| 14.2 Total investment from the department of electrical engineering | $600. |
| 14.3 Total prototype cost |  |
| 14.4 Total estimated production costs |  |
| **15. Data** | |
| 15.1 Topographical Data set | Each captured data set will contain all data points prior to being sorted. Each subsequent data captured will over-write the previous data. This will keep the database from “blowing up” in size and filling up digital space. |
| 15.2 Topographical Data set to output file | When the user triggers the dataset to be saved to an output file, this will be time stamped and saved to the working directory. This data is never destroyed. The user could trigger many “save data to output file” and each will be saved uniquely to be reviewed or analyzed at a later time. |
| 15.3 Output file naming convention | When the user triggers the program to save the current dataset to an output file, the filename will be the date and time the user triggered the save event. |
| 15.4 |  |
| **16. User Interaction** | |
| 16.1 Start Program | When the program is initialized, to start the program press the “ENTER” key on the keyboard. |
| 16.2 End Program | To end the program, press the “ESC” key. |
| 16.3 Save data to output file | To save the last data to an output file, press the “S” key. |
| 16.4 Send data to 3D printer | To send the data to a 3D printer, press the “P” key. |
| 16.5 Send data to be 3D Plotted | To print the topographical map of the last data set, press the “M” key. |
|  |  |
|  |  |
|  |  |