Project Goals and Development Process

Group 8 - C.A.V.E: Cave Assessment and Visualization Equipment

Abdul Rahim Khan
Andrew Brink
Gokberk Yilmaz
Nicholas Trimble
Tabish Faisal

September 27, 2025

1 Project Description

There are no publicly available models of the crevice caves of the Niagara Escarpment. These caves pose a significant challenge to existing mapping methods, which are typically designed for environments such as roads or well-lit indoor spaces. In addition, the 3D Li-DAR sensors which are ideal for mapping tasks are very expensive and therefore out of reach for most hobbyists. This project aims to develop a system which will obtain 3D models of challenging environments such as crevice caves at a relatively low price.

2 Goals

2.1 Main Goals (Level 1)

2.1.1 Create a 3D model of a tight space with little to no natural light

The primary goal of the project is to make accurate models of crevice caves. These often have no light from the outside and require the caver to bring their own lighting. The space is also constrained, often narrowing to passages not much larger than what a person can pass through. The exact format of the output is not yet determined but could be a point cloud or a 3D mesh.

2.1.2 Total cost of project is under $750

Typically, professional systems for mapping spaces use 3D Li-DAR scanners and cost multiple thousands of dollars. The total cost of the sensors and hardware used needs to be limited for the requirements of the capstone, but keeping a low cost would be an important goal regardless of this. We will be able to reuse equipment purchased by a previous capstone group, which will lower our expenses, but even if the sensors we already have are included in the budget, the cost should remain well under $750. The goal is to lower the barrier to entry for these types of mapping tasks while maintaining a reasonable level of accuracy.

2.1.3 The operator's ability to explore crevice caves is not hindered

Spelunking is a physically demanding activity. The system should be wearable by the user in a way that does not impede their ability to navigate through challenging caves. The space that the system uses should be small and the components be located to minimize disruption of the caver's experience. The system should also be light, to avoid fatiguing the user, and the battery charge should last long enough to avoid having to halt a caving session for recharging.

2.1.4 Use of the system does not pose any danger to the operator or others

There should be no risks to the user caused by the mapping system. Specifically, the system should not decrease the effectiveness of existing safety equipment such as harnesses or helmets. Furthermore, the batteries should be stored in a secure manner such that a fall does not cause further issues related to battery safety . This is related to the previous goal, but the focus of this goal is on the safety of the user and any others who could be affected.

2.2 Extra/Stretch Goals (Level 2)

Goal	Description
Local Data Processing	Implement the capability of processing the collected data from the sensors on the system in real time.
Camera Improvements	Design a camera mounting system with 30° range of motion, with 1° micro-movement via an interface. Add a RGB camera to capture color information to generate colored 3D models.
Phone Application	Develop an application to access previously made images and 3D mappings.
Machine Learning	Implement a learning mechanism that allows the option to reject images, and will train on accepted images.

Table 1: Stretch goals and detailed description

3 Development Process

3.1 Project Task Workflow

No.	Step	Acceptance Criteria
1	Set up Git workflow	Initialization of a GitHub repository for version control and tracking changes to the development of software.
2	Code review	Review existing code and functionality and validate compatibility with project's requirements. Should have a high-level understanding of what was done previously, what aspects were successful and what wasn't.
3	Sensor stress testing in controlled environments	Collected data is usable and consistent in best-case (simple cave, dim light), average-case (some terrain twists and dark spots) and worst-case (complex cave in pitch black darkness) scenarios. Obtain the extreme cases that keep the visualization accuracy at approximately 90%.
4	Sensor performance evaluation	Have expert knowledge on the full capabilities, strengths and limitations of the sensors we are using. *(i.e: time-of-flight sensors and Li-DAR scanner)*
5	Choose algorithms for sensor data fusion	Try different methods for efficient and functional fusing with the help of external libraries like Eigen, OpenCV, Open3D, and PCL. Should have functioning code that properly converts the sensor data into the appropriate visualization, that is at least 90% accurate.

continued...

No.	Step	Acceptance Criteria
6	Determine suitable mounting position for sensors	The chosen placement should provide stability, a large field-of-view and not obstruct the user's movement while still taking into consideration image quality.
7	Develop the software and the physical electrical system	The code for the wearable system and visualization is written in the suitable languages like Python and/or C/C++ with the help of libraries like Matplotlib and Seaborn. It should compile, run without any errors and function as intended. The electrical system should function and communicate properly between the code.
8	Design a mechanical system fusing all the hardware seamlessly	The sensors, Raspberry Pi, along with any other hardware are integrated securely onto the appropriate wearable gadget/accessory. The user should be able to comfortably wear the gadget and traverse naturally.
9	Test functionality in controlled environments	Obtain a good understanding of how our product functions in a controlled environment such as offices, rooms with re-arranged furniture and obstacles.
10	Test functionality in real-world environments	Be able to get a strong grasp of the capabilities and performance of the product in real-life environment, more specifically, in low-light crevice caves.

Table 2: Required workflow steps and its acceptance criteria

3.2 Primary Member Roles

All members are expected to broadly understand all parts of the project, including mechanical, electrical, and software components. To make best use of each team member's skills, certain people will focus on certain domains, with others helping as needed.

Member	Role
Andrew Brink	Liaison/Chair: Act as leader of the group for presenting to or interacting with TAs and professors, set meeting agendas and goals. Algorithms: Take point on the selection or development of the algorithms necessary to produce maps of low-light areas and caves.
Abdul Rahim Khan	3D Modeling/Design (Solidworks): Take point on mechanical design of the gadget. Algorithm: Also pay special attention to the development of the data processing algorithm

continued...

Member	Role
Berk Yilmaz	CI/CD: Set up and manage the various GitHub integrations necessary for continuous integration and deployment of the project's software.
Tabish Faisal	Python: Ensure best practices for Python structure and packages are being used and attempt to optimize the code towards the project's requirements. Reviewer: Pay special attention to merges and other reviews that take place to make sure all code and documentation is up to the group's standard.
Nicholas Trimble	Scribe/notetaker: Take minutes during the meeting to have a written record of what happened and to convert action items into GitHub issues. Meeting Chair: Keep meetings productive by ensuring the conversation continues moving and that the group does not become sidetracked or mired down by a single topic.

Table 3: Team members and their respective roles

3.3 Team Meeting Plan

In all meetings, Nicholas will write meeting minutes to keep track of the points discussed, the members present, and any actions items to be documented as tickets in GitHub. Currently, these will be uploaded to the Teams channel; in the future an effort will be made to put them under version control for tighter integration.

3.3.1 Weekly Meetings

A meeting will be held at minimum once per week during the school term in the first hour of the Capstone tutorial time (Thursday, 4:30-5:30pm). Unless agreed upon beforehand or in extreme circumstances, the meeting will be held in-person. A group study room will be booked by Berk, or it will be held in the room booked by the University for the tutorial time. Once weekly TA meetings begin, this meeting may be moved to one of the other times allocated for Capstone if it is no longer convenient to meet at that point of the week. During weekly meetings, members will update one another on the issues they are working on or have completed and voice any concerns or blockers affecting their tickets or the project.

3.3.2 Supervisor Meetings

Meetings with the project's supervisor, Dr. Giamou, should be held at least once a month, if his schedule allows. Due to personal matters, he will not be easily available during the middle of the fall term, so fewer meetings are anticipated during this time.

3.4 Technologies Used

Throughout this Capstone project, there will be various software and hardware components utilized in the development process. Here is a list of all the necessary tools that will be used[1]:

Software	**Use case**
Git/GitHub	Version control and track changes in the development of the source code as well as documents/deliverables.
C/C++ and/or Python (v3.13)	Programming languages used in the software development process. C/C++ will be the main language used for the implementation of the algorithms. Python will be used in interfacing with the sensors.
Robot Operating System (ROS)	Interact with and integrate sensors, inertial measurement units (IMUs) and cameras to the system.
SOLIDWORKS 2025	Develop 3D designs for the mounting of hardware onto a selected wearable accessory.
KiCAD 8.0	Design and fabricate custom PCBs.
Visual Studio Code	Code editor used for software development.
Microsoft Teams	Group communication and file management.

Table 4: Software and their uses

Hardware	**Use case**
Time-of-Flight (ToF) sensors	Capture the depth data of the cave environment.
2D LiDAR Scanner	Scans the environment and generates a 2D point cloud.
Inertial Measurement Unit	Track orientation of the scanning for simultaneous localization and mapping (SLAM).
Raspberry Pi 5 (RPi5)	Store and process the data received from the sensors.
3D Printer	Fabrication of the hardware mount design to a wearable accessory.
Power Delivery Module	Maintain and regulate power for the RPi5 and other components.
Power Bank	Portable power supply for the system to keep running while in the cave.

Table 5: Hardware components and their uses

[1]Subject to change.

3.5 Documentation and Coding Standards

The software running on the helmet and the data visualization code will be checked into GitHub for version control. The Git server will have continuous integration set up in the form of GitHub Actions, where unit tests, linters, and static analyzers will be run. Issues raised by static analyzers will be tracked in version control as needed.

Software environments (e.g. Python and system packages) will be checked in with an appropriate format (e.g. requirements.txt for Python) to ensure continued development of the project.

Outstanding tasks should be tracked in issues on the Git repository, with commits linked to issues if relevant. Outstanding tasks should also be highlighted with TODO comments in code if relevant.

Git commit messages should be descriptive and concise. Empty commit messages are not allowed. The commit message should be professional and relevant to the work checked in. As a guideline, conventional commits may be used to standardize commit messages.

Checking in magic files is not allowed. A magic file is defined as a file whose origin cannot be tracked down, such as a pre-compiled binary with no apparent means to obtain it outside of the repository. Any file checked into version control must be comprehensible by another team member. Furthermore, the CI/CD process must be able to provide releases without requiring such magic files.

Hardware setup such as OS version, electrical connections, and packages installed on the wearable device will be documented in a LaTeX document checked into the Git repository. PCB designs and mechanical designs must have the CAD files checked into version control. Other aspects of the hardware must be documented as appropriate with the aim of providing a clear path from the repository to the working product.

Builds with no errors that pass all unit tests shall be considered passing builds. The Git server will attempt to produce a passing build by running the CI/CD pipeline after each push. Passing builds shall be provided as releases via GitHub Actions.

Critical parts of code should have comments to explain functionality. For example, long blocks of math should have a brief description or the name of the equation/algorithm. As another example, hard-coded constants should have a comment explaining why that value was chosen.

The development process should be platform-agnostic. In other words, the development process shall not depend on factors such as OS or IDE choices that may vary between developers.