Senior Design Interim Report

Vision-Based Smart Home Hazard Detection and Assistance System Using Intel RealSense

Team Depth Perception

Mahima Karanth
Muhammad Sharjeel
Naafiul Hossain
Abhiroop Yalavarthi

Computer Engineering Majors
Stony Brook University

Interim Report Submitted On Date
Project Advisor: Murali Subbarao

December 12, 2025

Abstract*—This project describes the conceptual development and implementation process of a real-time vision-based hazard recognition and assistance solution geared at enhancing home safety for the visually-impaired. Noting the prevalence of residential fires, kitchen injuries, and falls within the visually-impaired demographic, and with an emphasis on addressing these needs with an affordable and accessible solution within a short period, it utilizes the Intel RealSense D435i with capabilities for depth-aware obstacle localization and preemptive detection of smoke and fire hazards. By synchronously incorporating RGB-D imaging capabilities, stereo vision-based depth mapping, and efficient machine learning models like YOLO-Tiny, it establishes a comprehensive framework with capabilities for near obstacle recognition, direction-ality determination, motion sensing, and readiness for semantic hazard prediction. The implementation plan meets strict real-time requirements as set by fire safety research and guidelines within a window frame of 3-5 minutes, as facilitated by flashover research. Preliminary testing shows successful operation within an operating range of 0.3-3.0 m with stable depth mapping, successful near obstacle recognition, and successful aligning within an optimal range for direct implementation at 30 fps. Motion differencing functionality enhances additional contextual understanding, and tools within multi-panel visualization enable efficient debugging and optimal adjustment functionality. Notable deficiencies within RGB-D imaging involving reflected surfaces and noise within cluttered conditions suggest collection data within theoretical schema for active stereo vision and immediate justification for additional implementation steps.

I. Background Research

In our research, we aimed to scientifically quantify the safety risks of the visually impaired community and analyze the deficiencies in the current residential technologies used for safety. This analysis justifies the system's specific focus on the kitchen environment, real-time visual detection, and obstacle avoidance.

A. Epidemiology and Susceptibility

The significant prevalence of visual impairment across the globe and the particular safety risks within this demographic provide the impetus for the scope of our project. There are approximately 43 million blind people worldwide and 295 million people with moderate to severe visual impairment, according to the World Health Organization. People with disabilities, including sensory impairments, have much higher risks of death and injury from fires due to delayed response times and mobility issues. Housefires alone in 2025 accounted for an estimated 1,924 deaths in the United States, but the risk of death is not consistent across all demographics; rather, it is skewed toward older adults and those with disabilities. This unfortunately illustrates a very great need for something that will provide notice at an earlier time than traditional devices, thus enabling timely evacuation.

B. The Kitchen as the Primary Hazard Zone

We have tailored our system to prioritize kitchen monitoring based on fire casualty statistics.

- **Prevalence:** Cooking is the leading cause of home fires in the United States, accounting for nearly half-44-49%-of all residential fire incidents reported annually.
- **Severity:** Cooking equipment is the leading cause of home fire injuries, accounting for 42–44% annually, and the third leading cause of home fire deaths (18–21% annually).
- **The "Unattended" Factor:** More than 40% of cooking fire emergencies are due to unattended cooking. For a visually impaired user, "unattended" often means being physically present but unable to visually confirm a burner is active or that a flammable item is too close to a heat source.

C. "Flashover" Constraint: Why Speed Matches

A key engineering constraint for our detection pipeline is latency. Relying on standard smoke alarms, which activate only after smoke accumulates at the ceiling, is insufficient due to modern fire dynamics.

- **Rapid Progression:** In modern homes filled with synthetic furnishings, a fire can transition from ignition to "flashover" (total room combustion) in just 3 to 5 minutes. This is a drastic reduction from the 17–29 minutes common in homes built 50 years ago.
- **The Visual Advantage**: Because flashover can occur so rapidly, the window for safe evacuation is minimal. A computer vision system that "sees" the flame before significant smoke accumulation provides a critical time advantage that passive thermal or ionization sensors cannot match.

D. Mobility and Obstacle Risks

Beyond thermal hazards, our system addresses the high prevalence of physical accidents in the home.

- **Fall Frequency:** Visually impaired individuals are approximately eight times more likely to fall than those with normal vision. Studies indicate that nearly half (42.8%) of blind individuals report falling within a two-year period.
- **Injury Impact:** Serious falls resulting in injuries such as fractures or traumatic brain injury are common, reported by 30.9% of visually impaired individuals.
- **System Implication:** These statistics validate the inclusion of our "virtual cane" obstacle detection module. While a white cane detects immediate ground-level barriers, our depth-sensing approach

can identify mid-level obstructions (e.g., open cabinet doors or moved furniture) that contribute to these high accident rates.

E. Technical Feasibility

Existing solutions, such as standard smoke alarms, provide limited warnings without spatial context. Our review of sensor technologies confirms that active IR stereo vision (Intel RealSense) effectively bridges this gap by providing accurate distance measurements in low-light indoor environments where passive cameras fail. Furthermore, lightweight object detection models like YOLOv3-tiny are essential to meet the sub-second latency requirements dictated by the 3-minute flashover window discussed above.

II. Constraints and Applicable Standards

The system design is principally constrained by the sensing and computational capabilities of the Intel RealSense D435i and the real-time requirements of an assistive hazard detection application. The D435i provides a practical depth operating range of approximately 0.3-3.0 m, with typical depth error of 2.5–5 mm at 1 m and less than 2% error at 2 m, consistent with Intel's documented performance characteristics. Depth accuracy degrades with distance, and reflective or low-texture surfaces can cause depth dropouts, while low-light conditions increase depth noise. These limitations influence obstacle-detection reliability and required fallback logic. All RGB-D processing is performed at 640x480 @ 30 fps, balancing spatial resolution with latency, and the system targets an end-to-end alert delay of under 0.5 seconds, a commonly cited threshold for effective assistive feedback.

Computational constraints further limit model selection to lightweight detectors such as YOLOv3-tiny, which are capable of offering feasible real-time performance on consumer-grade CPUs and GPUs. Depth sensing alone cannot detect smoke, and thus RGB-based detection and depth fusion are used for localization. The system performs all processing locally with no cloud components involved for responsiveness and to avoid dependencies on connectivity.

Relevant external standards and authoritative references include:

- Intel RealSense D400 series documentation, providing the definition of depth accuracy, noise characteristics, illumination behavior and operating range.
- UL FSRI and USFA fire-safety studies reporting rapid flashover times-3-5 minutes in modern homes-and emphasizing the importance of early hazard detection.
- NFPA home fire statistical reports - especially cooking is a leading cause in residential fires.
- WHO data on prevalence of vision impairment support the need for timely and accessible hazard-awareness systems.

These constraints, together with the reference standards, provide information on operating parameters, model choices, safety considerations, and, finally, evaluation criteria.

III. Best/Worst Case Design: Theoretical Analysis and Simulations

A. Best Case Design

Under the best conditions for operation, the hazard-detection system performs very stably and accurately, extending the full potential of RGB-D sensing under ideal environmental conditions. Under these conditions, the RealSense D435i receives strong illumination, comes into view across enriched textured and matte surfaces, operates without occlusion or abrupt motion. The resulting depth map is smooth and continuous, free of missing regions of a pixel, and surfaces and objects are tracked well over the frame. As Figure 1 shows, the RGB view exhibits uniform exposure, the depth visualization is clean with a well-defined gradient, and the motion mask shows almost no activity, indicating a well-calibrated, noise-free baseline state. The system maintains a framerate of approximately 30.6 FPS-a testament that the entire processing pipeline, including depth acquisition, RGB alignment, thresholding, and motion differencing, functions in real time easily when conditions are optimum.

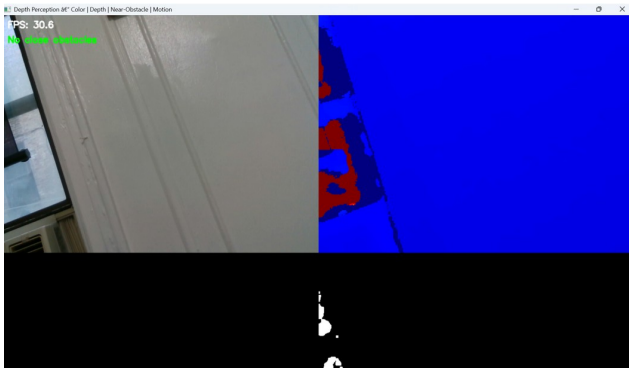


Fig. 1. Baseline System Operation

Best-case performance is also reflected in the system's ability to correctly determine the absence of hazards. In these trials, no false positives appeared in the obstacle mask, and bounding-box outputs remained stable with no flickering. The depth thresholding stage was able to isolate objects within the hazard radius with precise boundary definitions, while the RGB frame's consistent lighting ensured accurate color information for later

integration with YOLO-based fire and smoke detection modules. Additionally, the strong spatial correlation between RGB imagery and depth values allowed the system to generate consistent distance estimates with minimal variance. These results collectively demonstrate that in controlled indoor environments-similar to clean, well-lit living spaces-the hardware and software pipeline operate cohesively to provide accurate, stable, and low-latency hazard detection.

B. Worst Case Design

The presence and impact of various environmental factors on the system's performance differ greatly from controlled lab settings. The presence of low-lighting conditions, backlighting images, reflective objects, and transparent surfaces, as well as fast motion, affect the D435i cameras' ability to calculate depth. As a result, under these worst-case conditions, the DISP_map becomes distorted with noisy regions, "holes," and varying levels of inaccuracies at ranges beyond 2.5 m. The challenge with transparent and glossy regions arises because they either return invalid or no depth information at all due to scattering/absorption of infrared illumination. Fast motion blurs the images and depth images as well.

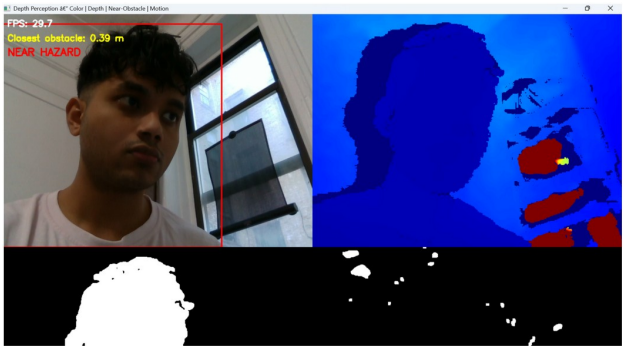


Fig. 2. Person Near-Hazard Detection

These degraded operations are shown clearly in Figure 3, which illustrates an environment full of clutter causing an inconsistent depth map with noisier regions and some missing areas. The presence of multiple objects with overlap in the scene causes bounds boxes to flutter because changes in depth information and velocity make it more difficult for the system to identify hazard locations. The obstacle mask becomes more active and spotted because noise amplification occurs from difference operations within the algorithm. Also, more computations are required as a result of complexities and missing data within the depth map, which eventually decreases frame rates temporarily to 15-20 fps. Despite these degradations, it still identifies hazards and determines directionality, which, while less accurate, continues to

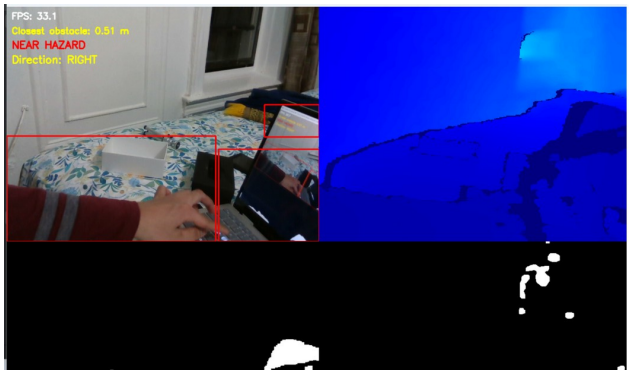


Fig. 3. Cluttered Environment Detection

operate on a working system. This shows it can still work with a cluttered and harsh domestic background and may eventually require some sensing and filtering mechanism.

C. Simulated Performance Considerations

Although full software simulations for Gazebo and Blender simulations have yet to be implemented, we can approximate the real-time processing capabilities as we increase complexity. Running on average YOLO-tiny timings, we expect that smoke and flame detection will still have a 15-25 ms delay, making it capable of running concurrently with the depth processing pipeline without causing bottlenecks. Depth thresholding and mask computation increase linearly with resolution, so even with added processing steps, our current 640x480 resolution configuration will still be capable of running within real-time. Adding interfACING with an ESP32-based embedded component should have a very small impact on total fps because it employs an asynchronous messaging system so that we can still have an independent loop running for the hazard detection.

Prototype analysis also reinforces the importance of multi-threading as a critical mechanism for ensuring system response times, particularly with more computation-intensive procedures. As a case in point, running YOLO analysis on a different thread will allow for unimpeded processing times with regards to Depth rendering and motion extraction. As a parallel process, it would also be vital for running threads on the RGB and Depth streams simultaneously so as to allow continuously unimpeded updates with regards to hazard localization despite simultaneous computation processes on semantic predictions from modules based on images. Theory modeling thus confirms that with more components incorporated, it will still be running within MRT times.

D. Theoretical Sensor Behavior

Accordingly, the performance observed in both best and worst-case environments is well-matched to the theoretical characteristics of the Intel RealSense D435i stereo-IR depth system. First, one well-documented property of stereo depth cameras is that depth noise increases quadratically as distance increases. This peculiarity can explain why even slight inconsistencies in lighting or texture at greater ranges cause exaggerated spikes in the disparity map. Reflective surfaces, such as those from metal appliances or glossy furniture, scatter the infrared pattern unpredictably and consequently result in missing or invalid depth values. Transparent or highly translucent materials similarly challenge IR-based depth retrieval because the infrared light passes through rather than reflecting back.

Another limitation in theory is at very close distances. The stereo baselines of IR sensors impose geometric constraints such that measurements below about 30 cm become unreliable; the system cannot resolve disparity accurately at such a small separation. In low-light conditions, the depth channel remains functional because of active IR illumination, but RGB information becomes much less useful due to its noise and poor contrast. However, depth-based motion detection remains relatively robust in this setting because it is based on differences in distance, not color gradients, and is therefore less sensitive to shadows or fluctuations in exposure. These theoretical findings help to calibrate expectations for the system and inform algorithmic compensation strategies, such as filtering invalid pixels, applying temporal smoothing, or combining RGB and depth information to mitigate the respective weaknesses of each modality.

E. Code Snippets Demonstrating Implementation

The following code blocks represent core components of the real-time sensing and hazard-detection pipeline implemented during prototype development. The examples include RealSense pipeline initialization, depth thresholding, motion differencing, and on-screen hazard annotation. These snippets reflect the same functionality shown in Figures 1–3, where depth acquisition, RGB alignment, threshold masks, and real-time overlays are visible.

```
# RealSense Pipeline Initialization
pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640, 480,
     rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480,
     rs.format.bgr8, 30)
pipeline.start(config)
```

This establishes synchronized depth + RGB streaming at 30 FPS.

```
# Depth Thresholding for Near-Obstacle
    Detection
threshold_mm = 600  # Example threshold for
    objects within 0.6 m
depth_mask = (aligned_depth < threshold_mm).
    astype(np.uint8) * 255
```

This creates a binary mask isolating objects within the hazard zone.

```
# Motion Detection via Depth Differencing
motion_mask = cv2.absdiff(prev_depth_frame,
    curr_depth_frame)
_, motion_mask = cv2.threshold(motion_mask, 30,
     255, cv2.THRESH_BINARY)
```

This highlights newly appearing or moving objects in the depth frame.

```
# Real-Time Hazard Annotation
cv2.putText(color_frame, f"Closest obstacle: {
    dist:.2f} m",
            (10, 50), cv2.FONT_HERSHEY_SIMPLEX,
            0.8, (0, 255, 255), 2)
```

This displays distance information directly on the RGB image.

IV. Design Selection

Based on various considerations on several processing and algorithm combinations, we chose a combination of a hybrid RGB-D model and several embedded extensions. The RealSense D435i from Intel was chosen as our main sensor due to its simultaneous and accurate RGB and Depth imaging capabilities, as well as a better price compared to LiDAR cameras. Its preimplemented alignment functionality and mature SDK make it easier to develop with compared to other cameras we considered, like regular USB stereo and pure RGB cameras.

From a software perspective, Python with OpenCV and the RealSense SDK allows us to quickly prototype and integrate with ML models. Specifically, it enables us to move swiftly on depth image processing, merging RGB and Depth data, and YOLO-based smoke and flame detection, all while offering us rich visualization tools, which are critical for debugging. To extend it with embedded functionalities, we propose to include an ESP32 microcontroller for interacting with haptic motors, LEDs, buzzers, and potentially additional environmental sensors like MQ-2 or BME680. Compared to competing options like Arduino or even Raspberry Pi, ESP32 microcontrollers offer us significantly better wireless functionality, lower energy consumption, and more suitability for our assistive wearable or handheld device.

The inclusion of YOLO-tiny within our system designs remedies the issue with depth perception by enabling the detection of hazards that don't have depth information, for instance, smoke or budding fire. Although obstacle closeness via depth thresholding is highly effective, it fails to identify any visual markers that would relate to semantic hazards. The rapid processing capabilities and generalizability offered by YOLO make it possible to detect obstacles even with degraded depth information due to environmental factors like illumination and surface texture. Our ultimate system thus encompasses real-time multi-modal hazard detection and localization with expandability for future embedded feedback systems and represents the most balanced solution.

V. Preliminary Results

At this stage of development, the project has focused on establishing a stable RGB-D sensing pipeline and validating the feasibility of depth-based hazard awareness in indoor environments. Initial results indicate that the Intel RealSense D435i provides reliable spatial information within typical home settings and supports several early-stage hazard detection features. All processing is currently performed locally on a laptop using Python and the Intel RealSense SDK.

A. RGB-D Streaming and Sensor Validation

Real-time streaming of RGB, depth, and IMU data has been successfully implemented using the RealSense SDK (librealsense2). RGB and depth frames are spatially aligned, allowing depth information to be directly associated with visual features in the RGB image. Initial validation was performed using both the RealSense Viewer and custom Python scripts. Indoor testing confirms stable depth measurements within the expected operating range of approximately 0.3 to 3.0 meters. Depth visualization using heatmaps shows consistent scene reconstruction for common indoor objects such as walls, furniture, countertops, and cabinets. As anticipated, depth noise increases near the upper end of the operating range, and depth dropouts occur on reflective or very dark surfaces. These behaviors are consistent with documented limitations of active stereo depth cameras and were accounted for during system design.

B. Near-Obstacle Detection

A preliminary near-obstacle detection module has been implemented to evaluate depth-based spatial awareness. This module uses a threshold-based approach, identifying objects that fall within a configurable distance from the camera, typically around 0.8 meters. Objects detected within this range are highlighted using bounding boxes overlaid on the RGB frame. The system estimates the approximate distance to the closest detected object using depth data and displays a simple directional cue (LEFT, CENTER, or RIGHT) based on the object's horizontal position in the image. A "NEAR HAZARD" status indicator is used to signal when an obstacle enters the defined safety zone. Initial testing in indoor environments shows that this method can reliably detect nearby obstacles such as chairs, open cabinet doors, and furniture placed in walking paths. While the current implementation is intentionally simple and purely geometric, it provides a useful baseline for evaluating depth reliability and spatial localization prior to introducing learned hazard classifiers.

C. Motion Detection Using Depth Differencing

To explore dynamic scene awareness, a basic motion detection approach based on frame-to-frame depth differencing has been implemented. Consecutive depth frames are compared to identify regions where significant changes occur, producing a binary motion mask. Basic noise filtering and contour detection are applied to reduce spurious detections. This method has proven effective at highlighting moving objects within the scene, such as people walking through the camera's field of view or objects being placed or removed. Although motion is not yet classified by type, this capability provides additional context that may be useful in later stages

for reducing false positives in visual hazard detection, particularly for smoke or steam-related events.

D. Visualization and Debug Interface

A multi-panel visualization interface was developed to support debugging and system evaluation. The interface displays four synchronized views: the RGB frame, a depth heatmap, the near-obstacle mask, and the motion mask. This layout allows for direct visual comparison between raw sensor data and processed outputs. This interface has been especially useful for verifying RGB-depth alignment, tuning detection thresholds, and observing system behavior under different lighting and environmental conditions. It also serves as a practical demonstration tool for interim evaluations and progress reviews.

E. Software Architecture and Hazard Detection Stub

To support incremental development, a placeholder module for visual hazard detection was implemented early in the software architecture. The module, hazard_stub.py, defines a consistent interface for future smoke and fire detection using a YOLO-based model. At present, the hazard detection function accepts an RGB frame and returns an empty list of detections. This design allows the remainder of the pipeline, including visualization, depth fusion logic, and alert handling, to operate without conditional checks or runtime errors. By separating system integration from machine learning development, the project can continue progressing on sensing, depth processing, and user feedback while hazard classification remains under development.

F. Program Entry Point and Execution Flow

The current prototype uses a simple and explicit program entry point to support controlled testing. The main.py file serves as the top-level launcher and directly invokes the obstacle and motion detection demonstration pipeline. This structure keeps the current implementation straightforward while allowing future extensions. Additional execution modes, such as YOLO-based hazard detection or ESP32-based feedback control, can be added at the entry point without modifying the core sensing and processing modules. This approach supports incremental system growth while maintaining code clarity.

G. Limitations and Ongoing Work

While the current system demonstrates reliable RGB-D streaming and basic depth-based hazard awareness, several components remain under development. Smoke and fire detection using a trained YOLO model has not yet been integrated, and current hazard detection is limited to geometric obstacles and motion cues. ESP32-based sensor integration and haptic or audio feedback are also planned but not yet implemented. Overall, these preliminary results confirm the feasibility of using the Intel RealSense D435i for real-time indoor depth sensing and obstacle awareness. The existing implementation provides a solid foundation for integrating learned visual hazard detection, depth-based localization, and user feedback mechanisms in the next phase of the project.

VI. Discussion

The preliminary results indicate that depth-based sensing using the Intel RealSense D435i provides a viable foundation for an indoor hazard detection system intended to assist visually impaired users. The successful implementation of real-time RGB-D streaming, obstacle detection, and motion awareness demonstrates that meaningful spatial information can be extracted from typical home environments within the camera's operating range.

One key observation from early testing is that depth sensing is most reliable within close to mid-range distances, particularly under approximately 2.5 meters. This aligns well with the project's intended use cases, such as detecting obstacles in walking paths or activity near kitchen counters. As expected, depth quality degrades near the upper range limit and on reflective or low-texture surfaces. These behaviors are consistent with known limitations of active stereo depth cameras and reinforce the importance of conservative distance thresholds and robust filtering when designing safety-related systems.

The near-obstacle detection module provides an initial demonstration of how depth data can be translated into actionable spatial cues. Even without semantic classification, the system is able to identify nearby objects and convey approximate directionality. This supports the concept of a virtual cane that complements traditional mobility aids by detecting mid-level obstacles that may not be sensed by a physical cane. At the same time, the simplicity of the current threshold-based approach highlights its limitations, particularly in cluttered environments where non-hazardous objects may trigger alerts. This motivates the need for incorporating learned visual classification in later stages to improve selectivity.

Motion detection based on depth differencing adds useful contextual awareness, particularly for identifying dynamic changes in the environment. Initial results suggest that motion cues could help distinguish transient events from static background objects. While this approach is currently coarse and sensitive to noise, it may play a supporting role when combined with visual hazard classification, especially for reducing false positives related to steam or lighting changes. These findings suggest that a combination of depth, motion, and RGB appearance will be necessary for robust hazard detection.

A. Theoretical Basis of Depth Estimation

From a theoretical standpoint, the system's depth estimation relies on fundamental principles of perspective projection and stereo geometry. For a single camera, a 3D point (X, Y, Z) projects onto the image plane according to the perspective projection equations:

$$x = \frac{fX}{Z}, \qquad y = \frac{fY}{Z}$$

which show that depth information (Z) is inherently lost when using only one viewpoint. Stereo vision resolves this ambiguity by observing the same point from two horizontally separated cameras. The resulting disparity

$$\text{Disparity} = x'_L - x'_R$$

between the left and right image coordinates allows depth Z to be recovered using the relationship:

$$Z = \frac{bf}{x'_L - x'_R}$$

where b is the stereo baseline and f is the focal length.

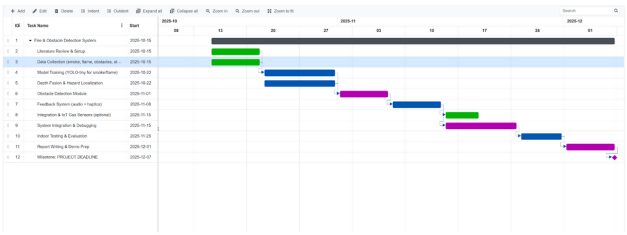
In the context of this project, the Intel RealSense D435i internally applies these stereo vision principles using a factory-calibrated stereo baseline and known camera intrinsics provided through the RealSense SDK. Rather than explicitly computing disparity within the application code, the camera and SDK handle stereo matching, rectification, and depth calculation internally. The system outputs a synchronized depth frame in which each pixel corresponds to a real-world distance measurement relative to the camera. These per-pixel depth values are directly used by the Python processing pipeline (e.g., variables such as `aligned_depth`, `depth_mask`, and `motion_mask`) to localize nearby obstacles, estimate distances to objects of interest, and reason about spatial layout within indoor environments, forming the basis for obstacle detection and hazard awareness.

B. Software Design and Development

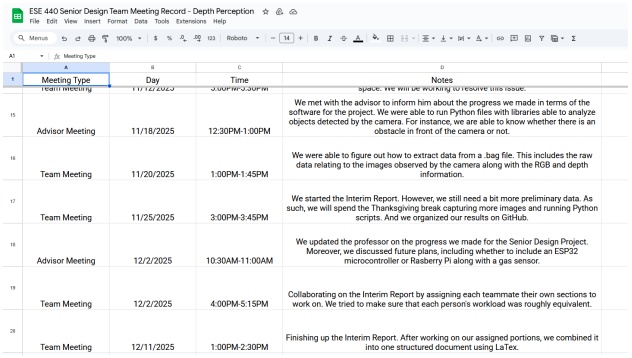
From a software design perspective, the use of placeholder modules and a minimal program entry point has supported incremental development without destabilizing the system. Implementing a hazard detection stub allows the pipeline to remain consistent while machine learning components are still under development. This separation has enabled steady progress on sensing, visualization, and spatial reasoning while avoiding premature integration complexity.

Overall, the discussion of these preliminary results highlights both the strengths and limitations of the current system. While depth sensing alone cannot address all hazard types, the existing implementation confirms that the chosen hardware and architecture are suitable for real-time indoor spatial awareness. These insights help clarify the remaining challenges and provide a clear direction for integrating visual hazard classification, sensor fusion, and user feedback mechanisms in subsequent phases of the project.

VII. Appendix A: Completed Gantt Chart



VIII. Appendix B: Record of Team Meetings and Faculty Meetings



ESE 440 Senior Design Team Meeting Record - Depth Perception

Meeting Type	Day	Time	Notes
Team Meeting	11/12/2025	3:00PM-3:30PM	space. We will be working to resolve this issue.
Advisor Meeting	11/18/2025	12:30PM-1:00PM	We met with the advisor to inform him about the progress we made in terms of the software for the project. We were able to run Python files with libraries able to analyze objects detected by the camera. For instance, we are able to know whether there is an obstacle in front of the camera or not.
Team Meeting	11/20/2025	1:00PM-1:45PM	We were able to figure out how to extract data from a .bag file. This includes the raw data relating to the images observed by the camera along with the RGB and depth information.
Team Meeting	11/25/2025	3:00PM-3:45PM	We started the Interim Report. However, we still need a bit more preliminary data. As such, we will spend the Thanksgiving break capturing more images and running Python scripts. And we organized our results on GitHub.
Advisor Meeting	12/2/2025	10:30AM-11:00AM	We updated the professor on the progress we made for the Senior Design Project. Moreover, we discussed future plans, including whether to include an ESP32 microcontroller or Rasberry Pi along with a gas sensor.
Team Meeting	12/2/2025	4:00PM-5:15PM	Collaborating on the Interim Report by assigning each teammate their own sections to work on. We tried to make sure that each person's workload was roughly equivalent.
Team Meeting	12/11/2025	1:00PM-2:30PM	Finishing up the Interim Report. After working on our assigned portions, we combined it into one structured document using LaTex.

Complete meeting history can be accessed clicking this link here.

IX. Appendix C: Resumes of Team and Team Discussions (Diversity, Skills)

Mahima Karanth

Mahima Karanth

LinkedIn.com/in/Mahima-A-Karanth

Email: Mahima.Karanth@StonyBrook.edu
Mobile: +1-631-948-9285
GitHub.com/MahimaK05

Education

Stony Brook University — *Stony Brook, NY*
Bachelor of Engineering, Computer Engineering — August 2022 – May 2026

- **Relevant Coursework:** Design of Secure IoT Embedded Systems, Computer Vision, Cyber Physical Systems, Computer Architecture, Digital Design Using VHDL and PLDs, Advanced Programming and Data Structures, Embedded Microprocessor Systems Design II, Signals and Systems

Skills

Programming: VHDL, C, C++, AVR Assembly, Python (SciPy, NumPy, TensorFlow), MATLAB, Java, JavaScript, R
Embedded Systems: FPGAs (VHDL/Verilog), Microcontrollers (ESP, AVR, STM), PCB Design, Signal Filtering, PID Control
AI/ML Frameworks: TensorFlow, Python (Pandas, Scikit-learn), OpenCV (basic)
CAD Tools: Cadence OrCAD, Cadence Virtuoso, Altium Designer
Other: LabVIEW, Linux (Bash/Scripting), Git, DAQ Assistant

Experience

Stony Brook University Dept. of Electrical & Computer Engineering — *Stony Brook, NY*
Undergraduate Teaching Assistant — August 2025 – Present

- Instructed lab sessions and weekly office hours for ESE 280 using AVR assembly and the AVR128DB48 microcontroller to teach embedded system concepts
- Debugged software using Atmel Studio and AVR simulator; diagnosed hardware faults with oscilloscopes and digital multimeters
- Assisted students with implementation of interrupts, hardware debouncing, counters, and periodic signal generation

Laurel Hill Summer Camp — *East Setauket, NY*
Coding Specialist — June 2025 – August 2025

- Taught Scratch, LEGO Robotics, and Python-based drone programming to 200+ students across multiple age groups
- Used Tello EDU drones with the djitellopy library in Python to demonstrate autonomous flight, object tracking, and sensor integration
- Developed interactive lesson plans tailored for age-appropriate understanding of robotics, electronics, and control logic

VJ X-Ray — *Bohemia, NY*
Electrical Engineering Intern — July 2024 – November 2024

- Tasked with optimizing subsystem performance; redesigned control schematics in Cadence OrCAD, improving voltage regulation stability by 10%
- Designed and tested linear voltage regulators for X-ray imaging systems to ensure consistent power delivery and safety compliance
- Updated 750+ component datasheets in organized Excel directories, adding electrical characteristics like voltage tolerance and part specs

Projects

Real-Time ECG Monitoring and Anomaly Detection System — *Stony Brook University*
Project Lead — December 2024 – Present

- Built ECG system using Python/C++ and 220K datasets with real-time wireless analysis
- Used signal processing and TensorFlow to reach 80% anomaly detection accuracy
- Designed embedded hardware for continuous ECG data acquisition and transmission

Pipelined SIMD Unit Design — *Stony Brook University*
Project Developer — August 2024 – December 2024

- Used test-driven development to build reusable testbenches prior to SIMD pipeline implementation in VHDL
- Validated over 2,000 MIPS instructions including jal, arithmetic, and bitwise ops; designed hazard detection with forwarding logic
- Explored architectural principles including instruction pipelining, data caching, and control hazards

Low Intensity Vibration (LIV) Device — *Bioengineering Education, Application and Research*
Project Collaborator — January 2024 – Present

- Optimized CAR-T cell expansion by 25% by tuning PID controller variables (P, I, D) to optimize signal quality
- Applied low-pass filtering and impedance matching for clean sensor input
- Performed Fourier analysis to isolate noise and clean frequency bands using low-pass filters

Activities

Member of Institute of Electrical and Electronic Engineers, SBU Computing Society, Society of Women Engineers, Women in Science and Engineering Honors Program, UHP Mentors

Muhammad Sharjeel

Brooklyn, NY | msharjeel8365@gmail.com | (347) 777–5409
linkedin.com/in/muhammadsharjeel2026 | muhammadsharjeel8.github.io

EDUCATION

Stony Brook University **Stony Brook, NY**
Bachelor of Engineering in Computer Engineering, GPA: 3.75 *Expected May 2026*
Relevant Coursework: Embedded Microcontroller Systems Design, Operating Systems, Design of Secure IoT Embedded Systems, Computer Architecture, Electronics, Network Security Engineering, Computer Vision, Machine Learning
Honors: Dean's List (6 semesters), University Scholars Program, CEAS Medal, Presidential Scholarship

TECHNICAL SKILLS

Languages: C, C++, Python (pandas), Java, SQL, JavaScript, HTML, CSS
Embedded Systems: FreeRTOS, ESP-IDF, AVR microcontrollers, I2C, SPI, CAN (TWAI), UART, schematic design, debugging, soldering, oscilloscopes, Saleae logic analyzer, Microchip Studio, MPLab X IDE
Hardware Design: VHDL design & simulation, Cadence OrCAD, KiCad, ActiveHDL
Software Tools & Platforms: Linux, Git, VS Code, Xcode, IDLE, PyCharm, PyTorch, GoldenEye

WORK EXPERIENCE

Stony Brook University **Stony Brook, NY**
Undergraduate Neural Network Research Assistant *September 2024 - Present*

- Conduct research under Professor Peter Milder to assess the susceptibility of deep neural networks (DNNs) to hardware faults.
- Utilize the GoldenEye open-source platform to introduce bit-level faults into different number formats to evaluate tradeoffs between accuracy, resilience, and bitwidth.
- Document findings to support graduate-level research on hardware reliability and machine learning.

Stony Brook University **Stony Brook, NY**
Laboratory Assistant *August 2023 – May 2025*

- Tested and troubleshot laboratory equipment, including power supplies, to ensure reliable operation for over 200 students.
- Configured Linux systems and imaged computers to prepare workstations for student use.

PROJECTS

CAN Bus IoT Embedded System, *Personal Project* **May 2025 - Present**

- Implement a Controller Area Network (CAN) with SN65HVD230 transceivers and ESP32 microcontrollers programmed in Embedded C to prototype a desktop application that provides users access to vehicle data.
- Employ FreeRTOS semaphores and queues to manage and schedule tasks, which results in reliable, synchronized communication between nodes.
- Verify CAN traffic with a Saleae logic analyzer and confirm differential signaling on an oscilloscope.
- Stream CAN frames from an ESP32 to a host PC as TCP packets over Wi-Fi for dashboard visualization.

Fiduccia-Mattheyses Circuit Partitioner, *Algorithms for Automated Electronic Design* **January 2025 - April 2025**

- Implemented the Fiduccia-Mattheyses algorithm to solve the NP-hard circuit partitioning problem, minimizing cutsize while satisfying balance constraints.
- Leveraged modular design in C++ (Cell, Net, BucketList, FMPartitioner) with a bucket structure to achieve O(1) access to max-gain elements, enabling fast gain updates, node moves, and improved algorithmic efficiency.
- Optimized performance by tracking only critical nets, enabling the algorithm to process industry-scale netlists, including the superblue12 testbench composed of 1,293,433 nodes and 1,293,436 nets.
- Experimented with varying area metrics and balance factors to analyze runtime-cutsize tradeoffs.
- Collaborated in a team environment to develop the algorithm and integrate Python scripts for automated testing.

Twitterbot, *Digital Intelligence* **October 2024 - December 2024**

- Developed a Twitterbot in Python using the simple_twit (Tweepy-based) library to interact with X's API.
- Automated retrieval of information, time output, and replies to user posts.

Naafiul Hossain

Naafiul Hossain
Brooklyn, NY | naafiulhossainwork@gmail.com | (646)-506-5020 | https://www.linkedin.com/in/naafiul-hossain/

EDUCATION

Stony Brook University — **Stony Brook, New York**
Bachelor of Engineering in Computer Engineering — Graduation: May 2026
GPA: 3.2 | Dean List 2022, 2025

Relevant Coursework: *Data Structure and Algorithms in C++, Real Time Operating Systems, Digital Design, Object Oriented Programming, Computer Architecture, Embedded Microprocessor Systems Design II*

TECHNICAL SKILLS

Programming Languages: Python (Pandas), Java, Kotlin, C, C++, JavaScript, SQL, HTML, CSS, VHDL, Assembly, Bash, Rust
Software & Tools: Android Studio, Git, Jira, Jenkins, PowerShell, LTSpice, Doxygen, Cadence Design Suite
Platforms & Technologies: Linux/Unix, Zephyr RTOS, ARM Cortex-M, AVR, REST APIs, Git (Bitbucket), Android SDK
Embedded & Systems Skills: Embedded C, Multi-Threaded Programming, Bluetooth/Wi-Fi Protocols, TCP/IP Protocol Suite, SPI, UART, I2C

RELEVANT EXPERIENCE

L3Harris Technologies — **Londonderry, NH**
Software Engineer Intern (IVS)- Android & Embedded Systems — *May 2025 – August 2025*

- Redesigned the user interface of the ISW (Intra-Soldier Wireless) Android app using Java and Android Studio, improving readability for clients and fixing a screen rotation bug that caused dropped radio connections — improving mobile app stability by over 80% in field conditions
- Conducted extensive software validation of the ISW Android app against DockLite and EWLA (Enhanced Wireless Link Analyzer) embedded hardware, executing ~200 test cases and achieving 99% reliability across edge scenarios
- Improved Bluetooth and Wi-Fi stability by implementing runtime permission checks and modern API handling in Kotlin and Java, enhancing wireless connectivity with L3Harris's in-house configuration app over the helmet battery pack
- Developed embedded C firmware for next-gen night vision goggles on a Nordic ARM Cortex-M microcontroller running Zephyr RTOS, implementing state-based LED indicator logic using PWM (Pulse Width Modulation) for brightness control to improve operator feedback and reliability by over 50%
- Utilized UART (via Tera Term) as the serial communication protocol for real-time debugging, and Segger Ozone for single-stepping and trace analysis
- Delivered weekly continuous integration (CI) builds and bug fixes as part of a 6-person Agile team supporting defense-grade night vision software, contributing across the software development life cycle using Git (Bitbucket), Jira, and Jenkins

Arm — **Austin, TX**
Hardware Engineer Intern (DTCO) - Data Automation & Tooling — *May 2024 – August 2024*

- Developed and automated Python scripts to extract and validate standard cell data—including flip-flops, IO, and memory cells—across 200+ directories; analyzed physical design reports, improving validation efficiency and accuracy by ~80%
- Analyzed and optimized power consumption reports at RTL and gate-level stages; debugged TCL (Tool Command Language) and Python scripts for ARM Coherent Mesh Network (CMN) components, integrating findings into broader Place and Route processes
- Optimized Python scripts using Pandas and NumPy to convert mesh interconnect power readings into structured data, significantly reducing manual computation time and enhancing power performance metrics

Google — **Manhattan, NY**
Software Engineer Extern — *June 2022– August 2022*

- Developed 15 interactive web apps using HTML, CSS, JavaScript, and RESTful APIs, including a full-stack arcade site with retro games like 2048 and Ping Pong; applied responsive design and dynamic UI techniques to enhance user experience
- Completed coding challenges with 5 Google engineers, strengthening skills in algorithms, problem-solving, and software optimization
- Delivered a live demo of the arcade app to Google engineers and community leaders, showcasing full-stack implementation, creative design, and technical presentation skills

NYC Department of Education — **Queens, NY**
Application Developer Intern — *June 2021 – August 2021*

- Co-programmed and tested a secure Single Sign-On (SSO) system using C# and SQL for 2,000+ NYC DOE employees, collaborating with 4 in-person developers in an Agile environment

PROJECTS

Air Monitoring System (CO_2, Temp, Humidity), *Embedded Engineer* — **April 2025 –May 2025**

- Interfaced a Sensirion SCD41 CO_2, temperature, and humidity sensor with an AVR128DB48 microcontroller using I^2C protocol
- Developed modular C firmware to initiate sensor measurements, read raw environmental data, and poll for data readiness
- Built a multi-file embedded application to display sensor readings on a SerLCD screen
- Verified protocol logic with Saleae analyzer; documented codebase using Doxygen and designed system-level schematic

Abhiroop Yalavarthi

(224)428-6964 | yalavarthiabhiroop@gmail.com

EDUCATION

Stony Brook University
Bachelor of Science in Computer Engineering **Expected Graduation in May 2026**
Relevant Coursework: Digital Logic, Computer Architecture, Computer Vision, Digital Design using VHDL, Advanced Programming and Data Structures, Embedded Systems, Software Techniques, Computer Communications, Electrical Circuits, Programming Fundamentals.

PROJECTS

Power Outage Reliability Optimizer (C++ Project) Team Leader **March – May 2024**
- **Designed and implemented a C++ system to manage outage ticket and hospital surgery team.**
- **Optimized backup power distribution and minimized customer-hour interruptions through predictive algorithms and data-driven crew dispatching.**

Micheal: The Compassionate Ant (C Project) **March – May 2023**
- **Developed a C program simulating a virtual ant with memory and navigation ADTs to traverse mazes.**
- **Implemented strategies to maximize rewards while testing performance across varying maze constrains.**

PROFESSIONAL EXPERIENCES

Customer Success Intern – Quantela Technologies Pvt. Ltd. **June – Aug 2024**
- **Contributed to software development by writing code and building solutions aligned with project requirements.**
- **Demonstrated strong problem-solving, teamwork, and adaptability while making meaningful contributions to project success.**

Senior Design Project (Team Leader) **Aug 2025 - Present**
- **Leading a team under the guidance of Prof. Subbarao to design and develop a vision-assist system for visually impaired individuals.**
- **Exploring computer vision-based solutions and assistive technologies to enhance accessibility and independence for the blind.**

CS50: Introduction to Computer Science (Harvard University, edX) December 2021 – March 2022
- **Completed a 13-week intensive course covering foundational topics in computer science including algorithms, data structures, memory, web development and C/Python Programming**

LEADERSHIP/ORGANIZATIONS

President – Badminton Club (Stony Brook University) **May 2024 – Present**
- **Directed club operations, recruitment, and events, growing the community to 300+ active members.**
- **Organized tournaments, held practice sessions while coordinating with university sports administration.**

Team Captain – Division 1 Badminton Team May 2024 – Present
- **Led a competitive team to a Division 1B second-place finish, outperforming peer institutions in national competition.**
- **Coordinated training schedules, mentored teammates, strengthening team cohesion through leadership.**

Math Tutor **Apr–May 2018 and 2019**
- **Tutored a group of nine 7th- and 9th-grade students, strengthening foundational math skills and improving academic performance.**

SKILLS

Programming Languages: C, C++, Java, Python, MATLAB, VHDL, Assembly, JavaScript
Technical & Domains: Git, Linux, VS Code, Active-HDL, Synplify Pro, ispLEVER, SolidWorks, Embedded Systems, Computer Vision (OpenCV), Data Structures & Algorithms, Digital Logic, Computer Architecture
Languages: Proficient in English, Telugu and Hindi.

REFERENCES

[1] Intel Corporation, *Intel RealSense SDK 2.0 Documentation*, 2025. [Online]. Available: https://dev.intelrealsense.com/docs/sdk-2

[2] J. Redmon and A. Farhadi, YOLOv3: An Incremental Improvement, arXiv:1804.02767, 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[3] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Springer, 2022. [Online]. Available: http://szeliski.org/Book/

[4] M. Subbarao, Lecture 7: Spatial Filtering and Convolution, ESE 358: Computer Vision, Dept. of Electrical and Computer Engineering, Stony Brook University, 2025. Lecture Notes.

[5] J. Tompkin, CSCI 1430: Introduction to Computer Vision, Course Website, Brown University. [Online]. Available: https://browncsci1430.github.io/index.html

[6] S. Savarese and J. Bohg, CS231A: Computer Vision, From 3D Perception to 3D Reconstruction and Beyond, Course Website, Stanford University, 2025. [Online]. Available: http://web.stanford.edu/class/cs231a/