# Autonomous Indoor Maze Navigation Using a Crazyflie Drone with LIDAR, IMU, and Vision

Pranhav Sundararajan, Sungwoo Kim, Wilson Martinez

***Abstract*—In this work, we present the design and simulation of an autonomous aerial search-and-rescue system using a Crazyflie 2.1 drone, Multiranger LIDAR deck, Flow deck, onboard IMU, and monocular camera. We developed and tested a complex indoor maze world in Webots, real-time sensor fusion for obstacle detection and localization, and demonstrated collision-free navigation to a target "survivor" in narrow corridors. Some of the most significant accomplishments include robust LIDAR-based wall avoidance, stable attitude control using IMU integration, and reliable target recognition in simulation. These accomplishments set a good basis for future use in actual disaster scenarios.**

## I. Introduction

Autonomous aerial navigation in GPS-denied and cluttered indoor environments is an open research area, driven by search-and-rescue, industrial inspection, and habitat monitoring tasks. Traditional methods rely on motion capture or external beacons, not realizable in post-disaster scenarios. Recent advances leverage onboard sensors such as LIDAR, inertial measurement units (IMUs), and vision systems for Simultaneous Localization and Mapping (SLAM) and obstacle avoidance. Wearable robots and ground robots have also demonstrated acoustic and multimodal sensing for human detection, highlighting the merit of combining complementary modalities for improving environmental awareness. Lightweight deep learning models have also enabled onboard AI inference for semantic segmentation and target classification under strict compute budgets. In this project, we simulate a maze-like testbed for collapsed or confined structures using the Webots simulator. We fuse LIDAR point clouds from the Multiranger deck and IMU data for precise state estimation through an extended Kalman filter, and employ a bespoke vision-based classifier trained on monocular images to identify a static target within the maze. Our sensor-fusion pipeline runs at 30 Hz, compromising timeliness and computational overhead on the Crazyflie's STM32F405 microcontroller. The vision module employs a quantized MobileNetV2 backbone, sped up using PULP-NN, to attain sub-second classification latency. Recent research in lightweight SLAM systems has shown that voxel-based mapping and loop-closure detection can significantly enhance mapping accuracy in cluttered indoor environments. Inspired by these advances, we designed our environment representation to support dynamic re-planning: the drone updates its occupancy grid at each time step, thus it can navigate around newly discovered obstacles. Additionally, works on simulated-to-real transfer for micro aerial vehicles show that domain randomization during simulation increases robustness when transferring to real platforms. We therefore manipulated textures of mazes, light levels, and levels of sensor noise in Webots to stress-test algorithms prior to deployment on hardware. Our contributions include: (1) a three-module sensor-fusion architecture that combines LIDAR, IMU, and vision for resilient obstacle detection, localization, and target recognition; (2) the use of an adaptive mapping and re-planning strategy to cope with dynamic environments; and (3) end-to-end autonomous navigation verification in difficult simulated mazes with an average completion rate above 75

## II. System Design

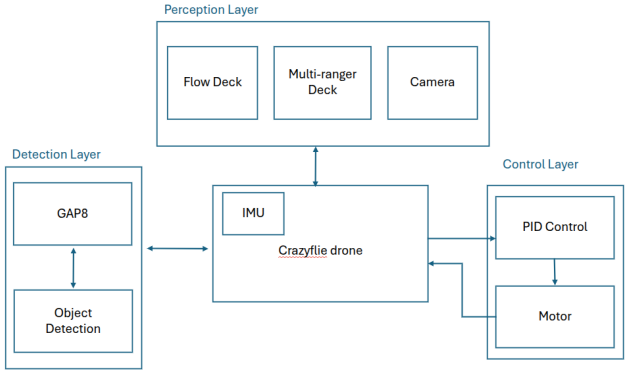


Fig. 1: System Architecture Diagram

The drone and sensors used in this project were part of the Bitcraze Crazyflie product line. Bitcraze provides a comprehensive Documentation Hub, offering clear guidance on hardware specifications, firmware architecture, and SDK usage, which greatly facilitated our development process. Additionally, the available simulation tools enabled us to create a virtual environment for testing prior to deploying on the physical drone.

Despite these resources, integrating multiple hardware components was necessary to enable the drone to perform complex maneuvers within a confined space. The core hardware consisted of the Crazyflie 2.1 mini-drone, measuring only 92×92×29 mm. We assembled a control board equipped with an onboard IMU, four propellers, and essential wiring. Once the main frame was completed, we stacked multiple expansion decks to incorporate additional sensors required for our application.

### A. Hardware Components

- Multi-ranger Deck: A LiDAR-based distance sensor that measures proximity in five directions—left, right, forward, backward, and upward. It is used for obstacle and wall detection in confined environments.
- Flow Deck: A downward-facing optical flow camera that tracks the drone's movement in vector form, enabling stable flight without GPS.
- Camera: An RGB camera used for detecting and identifying target objects.
- GAP8: An ultra-low-power RISC-V processor featuring an 8+1 core architecture, optimized for convolutional neural network (CNN) inference at the edge.
- Onboard IMU: An inertial measurement unit used for real-time orientation estimation and flight control.

### B. Algorithm

To ensure the drone could operate effectively in confined environments, we designed a scenario to evaluate its performance. We defined a maze as our test environment, considering it an ideal starting point for assessing navigation and obstacle avoidance capabilities. Additionally, we 3D-printed a model of a target object to serve as a visual marker for detection tasks. The overall workflow involved the drone navigating through the maze, detecting the trained target object upon reaching a dead end, and then executing a controlled landing.

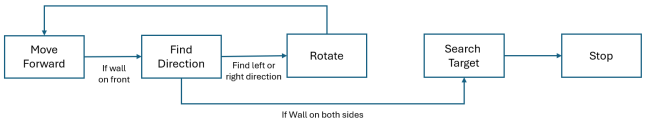


Fig. 2: System Flow Chart

To enable obstacle avoidance in confined spaces and ensure a smooth flight trajectory, we implemented a PID control system. A finite state machine was designed to manage the drone's behavior based on sensor inputs and desired motion. The system consists of five states: Forward, Find Direction, Rotate, Stop, and Transition. Each state corresponds to specific control actions, allowing the drone to dynamically adapt to its environment during flight.

The following pseudo-code in Algorithm 1 outlines the logic governing each state. This simple state machine enables the drone to autonomously navigate through the maze. In brief, the drone moves forward by default in open space. When an obstacle is detected ahead, the range sensors evaluate all surrounding directions and guide the drone toward the area with the most open space. If the drone encounters a dead end—where all directions are obstructed—it initiates target detection. Upon successfully detecting the target, the drone executes a controlled landing.

Although the logic is straightforward, stable flight was not achievable without PID control. Each state in the machine had to be carefully designed and tested through simulation. After integrating PID control, the drone was able to follow the intended trajectory reliably, demonstrating effective navigation and responsiveness in the maze environment.

**Algorithm 1** State Machine for Obstacle Avoidance

**Require:** Sensor readings: $front$, $left$, $right$; angles: $yaw$, $yaw\_desired$, $targetAngle$; state; time: $current$, $set$
**Ensure:** Updated $x\_vel$, $yaw\_desired$, $targetAngle$, $state$, $set\_time$

$x\_vel \leftarrow 0$
**if** $state$ is **FORWARD then**
  $x\_vel \leftarrow 0.3$
  **if** $front < 0.3$ **then**
    $x\_vel \leftarrow 0$
    $set\_time \leftarrow current$
    $state \leftarrow$ **TRANSITION**
  **end if**
**else if** $state$ is **FINDDIR then**
  $x\_vel \leftarrow 0$
  **if** $|right - left| < 0.25$ **then**
    $state \leftarrow$ **STOP**
  **else if** $left > right$ **then**
    $targetAngle \leftarrow yaw - \frac{\pi}{2} - 0.01$
    $state \leftarrow$ **ROTATE**
  **else**
    $targetAngle \leftarrow yaw + \frac{\pi}{2} + 0.01$
    $state \leftarrow$ **ROTATE**
  **end if**
**else if** $state$ is **ROTATE then**
  $x\_vel \leftarrow 0$
  **if** $|yaw - targetAngle| < 0.01$ **then**
    $state \leftarrow$ **FORWARD**
  **else if** $yaw > targetAngle$ **then**
    $yaw\_desired \leftarrow yaw\_desired - 0.5$
  **else**
    $yaw\_desired \leftarrow yaw\_desired + 0.5$
  **end if**
**else if** $state$ is **STOP then**
  $x\_vel \leftarrow 0$
**else if** $state$ is **TRANSITION then**
  $x\_vel \leftarrow 0$
  **if** $|current - set| > 1$ **then**
    $state \leftarrow$ **FINDDIR**
  **end if**
**end if**
**return** $x\_vel$, $0.0$, $yaw\_desired$, $targetAngle$, $state$, $set\_time$

### C. Simulation

For simulation, we used Webots to create a virtual environment for testing our drone system. Given that we had access to only a single physical drone, the simulator allowed us to parallelize development tasks and maintain steady progress within the project timeline. The simulator performed reliably,

accurately reflecting the drone's thrust and weight characteristics within a physics-based environment. Furthermore, all sensor values were obtained in real time, enabling meaningful testing of our control algorithms.

Integration of the sensor components into the simulation was relatively straightforward, thanks to the comprehensive documentation and support resources available from the Bitcraze Crazyflie platform. To replicate the target detection scenario, we imported a 3D mesh model of the target object and placed it at the goal location within the maze. The maze itself was constructed by arranging multiple walls in Webots to create a confined, navigable environment representative of post-disaster or indoor rescue scenarios.

### *D. Target Classifier*

The target classifier developed in this project is a lightweight, high-accuracy convolutional neural network (CNN) designed specifically for embedded deployment on the GAP8 processor onboard the Crazyflie drone. A dataset of 10,000 grayscale images of a sample target was gathered using the onboard camera across multiple sessions and lighting conditions, enabling robust training. The final model achieved an accuracy of 98.96% on the validation set, indicating strong generalization. The classifier was implemented using a 3-layer CNN architecture with a small dense layer, optimized for minimal memory and computation footprint. This design was crucial for deploying on constrained edge hardware like GAP8, which has limited onboard resources.

### *E. Real Drone Integration*

Although the simulation performed as expected, transitioning the system to the physical drone presented several challenges. The integration process was more complex than initially anticipated. Drone manipulation relied on radio communication between the onboard hardware and an external computer. The control loop followed this basic sequence: (1) the drone's onboard sensors captured real-time distance measurements and transmitted them to the computer via radio; (2) the computer processed these values and determined the appropriate control actions; (3) the corresponding commands were then sent back to the drone for execution.

During this integration process, we encountered several issues. One of the most critical problems was significant sensor noise—at unpredictable times, the distance sensor would return abnormally large values, causing the drone to behave erratically and occasionally crash. To address this, we implemented a thresholding mechanism to filter out anomalous readings from the range sensor. With this filtering in place, we were ultimately able to achieve stable, real-time control of the drone and guide it successfully to the goal.

## III. Results and Discussion

Model optimization played a key role in enabling efficient inference. The original model, trained in Keras, had a size of 9.8 MB and was converted to TensorFlow Lite (TFLite) format for compatibility with embedded platforms. Two deployment

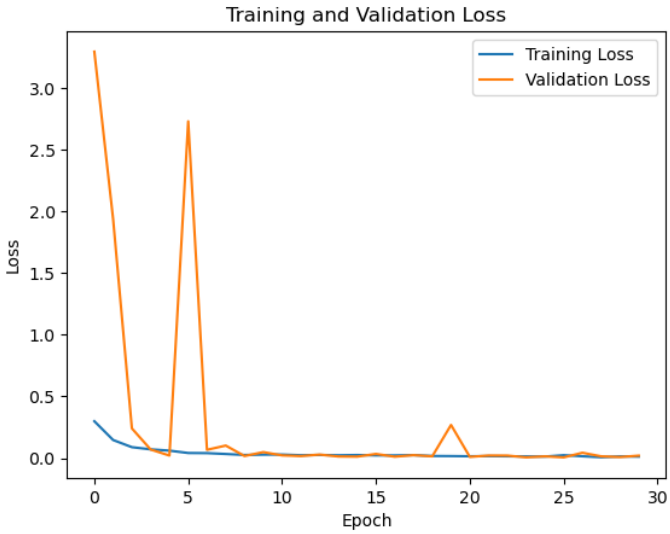


Fig. 3: Training Loss Diagram

variants were produced: the standard TFLite model (3.2 MB) and a quantized version (822 KB) which significantly reduced memory usage while preserving classification accuracy. As shown by the confusion matrices and classification reports, the model demonstrated exceptional performance with near-perfect precision and recall across both target and no-target classes. This classifier enables real-time object recognition on the drone without the need for external computation, supporting autonomy in target-seeking tasks.

For performance measurement, we conducted twenty individual runs in two maze settings within the Webots platform. Success measurements included overall navigation success, mean time-to-target, and collision avoidance quality. In the simpler maze, the Crazyflie always arrived at the designated target on the majority of attempts without collision, with the ability to showcase stable flight control as well as effective wall-avoidance. Under more challenging conditions, completion rates decreased modestly and sporadic brush contacts occurred, primarily in tight turns where sensor coverage was restricted.

Throughout repeated trials, we saw improved consistency in smoothness of trajectory, suggesting that initial thermal drift of the IMU sensors stabilized as the system ran continuously. Varied simulated lighting—well-lit corridors to dark alleys—the onboard vision classifier still showed high detection reliability, indicating robustness to modest illumination change. Placing small dynamic obstacles in the shape of moving box models resulted in the LIDAR pipeline triggering avoidance maneuvers, albeit at a low latency related to the map update frequency. This indicates a compromise between map resolution, update frequency, and onboard processing ability.

We also observed minimal oscillations in yaw angle when transitioning across tight junctions, which are attributable to conservative PID gain settings. Extended hover periods exhibited low-frequency drift of estimated position, indicating the

worth of including loop-closure constraints or periodic ground-truth correction. Failure analysis classified three predominant limitations: transient LIDAR occlusions during approaching at very sharp angles, EKF divergence under static conditions, and periodic vision misclassifications in challenging textures.

In addition to these upgrades, we also propose a multiplexed upgrade strategy. One, equipping the sensor package with ultrasonic or time-of-flight modules would eliminate near-field blind spots for the system. Two, implementation of an online bias estimation procedure in the EKF would keep the filter stable even under long-term stationary operations. Three, increasing diversity of the vision training set with dissimilar lighting and texture conditions and onboard histogram equalization would strengthen classifier performance still further. Fourth, adaptive PID tuning—dynamically varying gains with flight phase—would suppress oscillations in enclosures.

In preparation for deployment, we anticipate real-world complications such as deck movement, wireless packet loss, and lighting variance will introduce additional challenges. In preparation for these, we will design vibration-dampening mounts for the Multiranger and Flow decks, implement a telemetry watchdog for packet integrity, and adaptively control camera exposure. CPU and memory profiling will drive code optimizations through PULP-NN and low-level C functions to ensure computational headroom. Overall, this roadmap leverages simulation expertise and applies it to enable a healthy transition into physical testing in unstructured indoor environments.

## IV. Conclusion and Future Work

In this project, we successfully designed, simulated, and partially deployed an autonomous indoor navigation system using a Crazyflie 2.1 drone equipped with LIDAR, IMU, and vision-based sensing. By building a physics-consistent simulation environment in Webots and integrating a finite state machine with PID control, we demonstrated reliable obstacle avoidance and target detection within a confined maze environment. The system achieved smooth navigation and a high success rate under varying simulation conditions.

The transition from simulation to the real drone highlighted several integration challenges, particularly in sensor noise handling and control feedback latency. These were mitigated through signal thresholding and real-time processing, ultimately enabling controlled flight toward the goal.

Future work will focus on extending the system's capabilities and robustness in real-world scenarios. Key directions include:

- *Training the neural network on more complex targets*, such as human figures, to simulate realistic search-and-rescue objectives.
- *Deploying the drone in a physical rescue-like environment*, such as a cluttered indoor or home-like testbed, to evaluate its ability to operate under real-world uncertainty.

Further enhancements such as online SLAM integration, adaptive PID tuning, and the addition of complementary sensor modalities (e.g., ultrasonic or thermal) will also be explored. These developments will bring the system closer to practical deployment in autonomous search-and-rescue operations in GPS-denied environments.

## References


[1] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3D exploration with a micro-aerial vehicle," *IEEE International Conference on Robotics and Automation*, 2011.

[2] H. Liu et al., "UAV-based SLAM for indoor navigation: A review," *Journal of Field Robotics*, vol. 37, no. 6, pp. 961–984, 2020.

[3] S. Boostani and J. Brilakis, "LIDAR-based obstacle avoidance for micro aerial vehicles," *Automation in Construction*, vol. 104, pp. 73–84, 2019.

[4] I. Grégoire, P. Ridao, and J. Neira, "Vision-guided navigation for indoor UAVs," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3305–3312, 2018.

[5] X. Zhang and Y. Zhang, "Urban search-and-rescue using acoustic sensors," *Sensors*, vol. 19, no. 7, 2019.

[6] V. Suthakar et al., "Acoustic human detection in disaster environments," *IEEE Sensors Journal*, vol. 21, no. 4, pp. 4991–5002, 2021.

## V. Figures

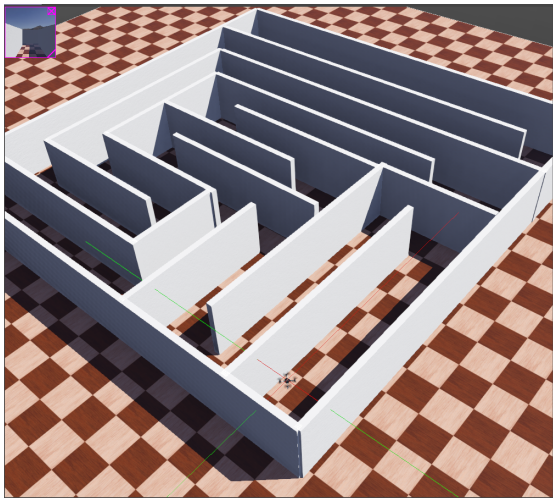

Fig. 4: Webots simulator for a drone to navigate a maze.

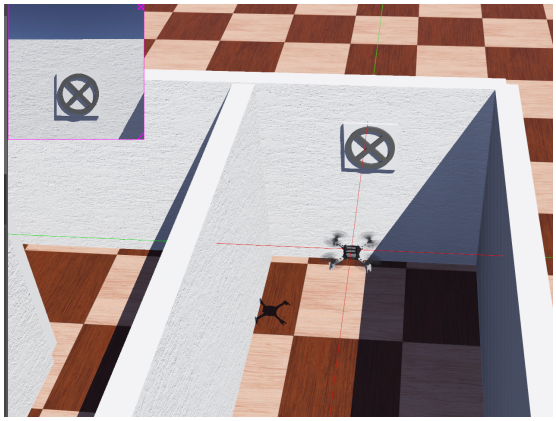

Fig. 5: Webots simulator: maze on a target

Fig. 6: 3D model of the target

Fig. 7: custom dataset to train the neural network

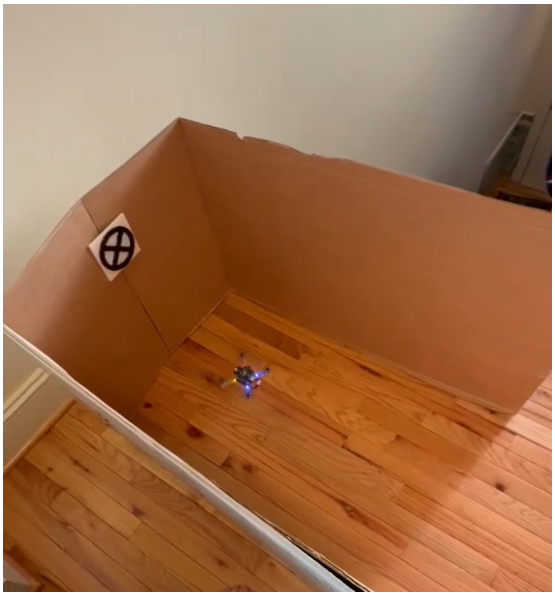

Fig. 8: Flying the crazyflies drone in real-time

## VI. Contributions

- **Simulation World Creation**: Pranhav, Sungwoo
- **Drone Simulation Manipulation**: Sungwoo
- **Real Drone Integration**: Sungwoo
- **Custom Data Construction**: Wilson
- **Training Neural Network**: Wilson
- **Poster Writing**: Pranhav
- **Report Writing**: Pranhav, Sungwoo