A User-Friendly Micro-Autonomous Vehicle with Onboard Structural Degradation Detection for Civil Infrastructure Maintenance (MAV-OCD)

Team RotomBot
Andrew Jairam, Gurpreet Mukker, Ethan Rajah, Arthur Zhuang

ROB498 - Robotics Capstone Design
Final Design Report
April 22, 2025

1. Design Overview

Crack formation in civil structures is problematic; if left unchecked, even small fractures can lead to structural failure [1, 2], posing safety and economic risks. The design of a MAV that executes an automated sweep of a structure using given waypoints with an onboard crack-detection algorithm is proposed to help construction companies, structural engineers, and building maintainers quickly assess and repair potential structural failures. An intuitive approach to providing drone waypoints is presented through the use of speech-to-text recognition, enabling users to easily command the drone towards points of interest using their voice. Structural crack detection is addressed using a fine-tuned neural network approach, providing accurate real-time classifications and fast inference times onboard the drone. Simultaneously, a vision language model (VLM) based output system is proposed that uses the visual input when material degradation is detected to give more information on the type and severity of the degradation. Using this information, an SQL database information storage approach is utilized to compactly and efficiently store the information of the sweep, allowing for deployment of a user-friendly GUI to be constructed. Such a system has great economic value to these stakeholders, allowing for a full sweep of a target infrastructure to be executed cost-effectively with negligible labour costs. The success of the system is evaluated based on the accuracy of structural degradation detection, the clarity and relevance of the descriptions generated by the VLM, and the overall ease of use of the system.

The design philosophy guiding this project is rooted in the belief that modern machine learning frameworks, particularly reasoning-based models, should be integrated into complex autonomous systems to bridge the human interpretability gap often observed in their use. This belief shaped a focused effort on developing a well-integrated MAV system that delivers a straightforward and accessible user experience for issuing commands, visualizing data, and assessing structural integrity in real time. As a result, the obstacle avoidance component of the drone prototype was intentionally deprioritized, as the primary objective was to showcase the application of reasoning models in drone-based inspection tasks. To support this goal, the team workflow was organized into implementation, testing, and integration subsystems, allowing for efficient parallel development and clear division of responsibilities. This structure proved particularly effective for building the crack detection, VLM, and database pipelines, which were designed to integrate seamlessly with the drone and provide real-time feedback during flight.

2. Background & Related Work

2.1. Literature Review

Civil structures are fundamental components of modern society, playing a role in virtually every field. These structures are subject to various naturally occurring pressures such as gravity, winds, weather conditions like rain and snow, and natural disasters. As such, maintaining them is of utmost importance for safety reasons: structural failures can result in dangerous situations for pedestrians and the surrounding environment. Additionally, repair of civil structures is of economic interest to construction companies that are responsible for maintaining them, as small repair jobs are much cheaper and easier to execute in comparison to if the structure reached complete failure, requiring a full rebuild.

Cracks in civil structures are a key indicator of physical damage and can highlight areas susceptible to failure. Most infrastructure develops cracks in its lifetime, due to the inherent properties of materials commonly used, such as wood and concrete [3]. These cracks weaken the structural integrity of components, decreasing the load capacity of the material [4, 5, 6]. If left unaccounted for, cracks can propagate through the structure, resulting in inevitable failure [1, 2], making cracks crucial to catch early on.

Typically, a professional performs manual inspections on components by examining the components visually, and using various tools to identify the severity of cracks formed [7]. Numerous types of cracks have different impacts on the structure, requiring the hired informed professional to make a difficult classification [8]. While for the most part effective, this method is tedious, expensive to execute regularly, time-consuming, and susceptible to human error [9]. This prompts the exploration of automatic crack detection methods. Initial attempts involved sensor-based processing using methods such as thermal imaging, laser scanning, infrared techniques, and radiography [10, 11, 12], however, these methods often suffer from high costs and practical limitations.

Instead, image-based methods have recently been gaining popularity, which are much faster in comparison, less expensive, and more robust to errors [9]. Image processing-based methods first took the forefront of development, relying on an algorithm to identify a crack in an image using methods such as tree structures, genetic programming, image filtering, and feature detection [5, 13, 14, 15, 16]. More recently, machine learning methods have taken over as the preferred method for crack detection with the developments in deep learning and graphics computation [9]. Particularly, Convolutional Neural Networks (CNNs) have shown that cracks can be classified with high accuracy in a variety of environmental settings, due to their advanced visual processing capabilities [4, 17, 18, 19]. To be successful, these methods require a lot of data: for example, Zhang et al. used 30,000 low-resolution images to train their convolutional crack detection model [16]. Fortunately, many open-source crack datasets are available, and data augmentation or fine-tuning can be used to increase performance accuracy in deployed models.

The use of micro-autonomous vehicles (MAVs) in crack detection is promising as they can be piloted to reach inaccessible positions to a human expert, with little risk involved. These systems can use onboard computing systems to execute a deep learning model and return a set of outputs with locations of detected cracks. In literature, MAVs are widely used in the construction field with onboard CNN algorithms [19, 20, 21]. In particular, Li et al. use human-piloted MAVs with high success for outdoor bridge crack detection using Faster R-CNN as the detection model, accounting for dynamic environment settings. While successful, most methods rely on human pilots to operate a MAV [21]. Automating flight is of particular interest, as often, a human operator would have a systematic and static trajectory of locations they would need to check for cracks. Checking can be automated in theory by using waypoints to mark such locations, developing a way to “sweep” for cracks between these checkpoints, and returning info on the locations of potential structural integrity risks. Such a method would allow for an easy and quick search for cracks, with little human operation cost.

2.2. Hardware, Software, and Firmware Considerations

It can be ambiguous to parse through outputted information from a deep learning model: an easy-to-interpret pipeline transforming detected readings to readable outputs is desired to ease the difficulty in using these systems in real settings. As such, a simple, comprehensible output interface is proposed, using VLMs to diagnose detected cracks, and parsing output information into a SQL-based database for easy deployment on a user-friendly interface. Processing outputs through a VLM aids workers who may not be trained to interpret deep learning outputs, such as images with bounding boxes and confidence scores, by providing natural language summarizations on factors such as crack severity, location, and the causes of the positive/negative diagnosis. However, preliminary testing with various open source VLM models demonstrate that VLMs tend to be biased toward predicting that images have cracks, and thus a convolutional neural network classifier is proposed to help distinguish between images with and without cracks, prior to VLM inference being run.

For simple interaction on the worker side, a speech-based input system is proposed, where a user can give voice commands to prompt the drone to move to target inspection sites. This provides an intuitive method for users unfamiliar with the system to execute inspections with repeatability. While inspection sites can have loud ambient noise, modern natural language processing-based speech-to-text transcribers are robust to excess background noise, making this component viable to be used in real settings.

To design the prototype for an autonomous MAV executing crack detection, cheap hardware components are used. The Jetson Nano platform is a cheap and portable processor that features 4 GB of memory and 128 CUDA cores to be used for running machine learning processes in mobile robotics. However, due to the age of the hardware, the latest supported CUDA toolkit version is 10.2, which prevents the onboard use of modern vision-based reasoning models for real-time structural analysis, as libraries such as Transformers require CUDA versions 11.0 or higher. The Jetson's limited available memory presents a significant constraint in achieving high-accuracy predictions, as it restricts the choice of detection networks to smaller, less capable models. This trade-off is necessary to ensure efficient resource allocation and prevent memory overflows that could compromise drone stability or disrupt essential inspection functions. Along with the cheap PX4 Orange Cube flight controller module, onboard software components can take advantage of the ROS2 Foxy and MAVROS packages to communicate with each other and execute control through publishing waypoints. For accurate localization, the prototype used the VICON localization system to track the drone position and obstacle locations. The VICON system is accurate enough to be considered noise-free, and is a good choice for prototyping in the time-scale of this project, as core functionality can be demonstrated without needing to implement a robust localization algorithm, which would take extensive time. The QGroundControl firmware associated with the Orange Cube+ is used to calibrate the orientation of the drone, and motor component directions to ensure flight is achieved.

The IMX219-160 monocular colour camera is used to capture adequate-quality 1920x1080 resolution images for crack detection and VLM inference. Due to observed distortion from the camera outputs, a camera calibration process, made available with OpenCV, is used to undistort the images before performing detection and VLM inference. This ensures that the best quality outputs from the available hardware are sent to the detection and reasoning models for inference. For processing a camera feed to capture images for inferencing, an OpenCV and GStreamer-based pipeline is used, as this is standard for operations involving live camera feeds, and OpenCV can integrate with CUDA to take advantage of the onboard graphics acceleration.

The design of software components must take into account the aforementioned compute limitations of the Jetson Nano, where using lightweight machine learning models and non-memory-intensive processes are key considerations. For detection, the convolution-based Darknet YOLO framework fine-tuned on a crack dataset is chosen. This is the best method as the smallest Darknet YOLO model is lightweight enough and compatible dependency-wise to use with the Jetson Nano, with open-source pre-trained models readily available. As an industry-standard architecture, YOLO is easy to work with due to the vast documentation available. Additionally, the VLM and SQL database are run offboard on a separate computer to address the hardware compute and memory limitations present for performing real-time analysis of structural defects during flight. The MLX formatted 2.2 billion parameter SmolVLM2 developed by HuggingFace is chosen as the VLM for the design because of its lightweight nature and robust performance on complex multimodal tasks [22]. This choice aims to minimize offboard compute requirements, while aiming to maintain high fidelity analysis capabilities from the reasoning model. The VLM is not fine tuned due to time required to do so, thus limiting the descriptive potential that can be achieved from the language model. Instead, an iterative prompt engineering process is utilized to generate consistent and informative descriptions of the cracks. Moreover, the MLX version of the model is chosen to support Apple Silicon for fast inference capabilities. Similarly, an MLX formatted OpenAI Whisper speech-to-text model with 1.55 billion parameters is chosen for transcribing speech waypoint commands to text for parsing due to its high accuracy and efficient inference performance on standard consumer laptops. Since the speech input consists of short waypoint commands, transcription is nearly instantaneous, enabling fast and seamless integration into the database pipeline. Communications between the database and the drone can be executed through API calls between the two systems, making this approach viable. The backend for the API is built as a Python-based Flask application, which is industry-standard for SQL solutions.

3. Design Objectives & Solution Details

3.1. High-Level Objectives

The key high-level objectives pertaining to the design of a fully autonomous MAV inspection system that can perceive and describe cracks on civil structures are summarized in this section as the following:

- **Enable early intervention** by identifying potential structural failures before they become critical.
- **Minimize human labor and risk** by ensuring contact-free inspections in unsafe environments.
- **Prioritize critical detections** to reduce review time and assist structural engineers in addressing the most urgent issues first.
- **Improve inspection interpretability and efficiency** by bridging detection outputs with clear, explainable descriptions, leveraging modern reasoning models to enhance crack description and prioritization.
- **Ensure safe deployment** by restricting drone operation to cleared inspection zones and incorporating basic obstacle avoidance features.

- **Simplify the inspection analysis process** by providing intuitive methods of interacting with the MAV system and accessing organized outputs on a structured database.
- **Design a low-cost solution** that can operate effectively on limited hardware while maintaining high accuracy performance.

While an ideal inspection solution should be able to identify and provide recommendations for various types of structural defects such as cracks and corrosion, the solution space is rescoped to focus on structural failures resulting from cracks to simplify the detection optimization process on the limited hardware available for a prototype. Nonetheless, the design solution must promote early intervention of potential structure failures, reducing the future risk of harming individuals who may work or live in these structures at a point of failure and aim to reduce the manual labour and risk associated with individuals checking for cracks in unsafe environments. For this reason, the drone must be able to perform its navigation and checks contact free, so that overseers are only responsible for deploying and recovering the drone on site. With minimal human risk and labour, the design should reduce the cost to survey civil structures for cracks, providing more opportunities for corporations and governments to inspect more buildings.

The design also aims to provide high accuracy in detecting structural defects among various lighting, background, and orientation conditions, while providing real-time accurate descriptions of the severity of the detected cracks. This approach must be as accurate or better than current inspection approaches to ensure that the quality of the assessment is not degraded with the design.

While the drone must have effective control and navigation through the implemented path planning and mapping algorithms, there is still potential risk for human injury if the drone is flown in areas with high density of people. The drone should be overseen by a team of inspectors who will be responsible for deploying and recovering the drone, particularly for battery replacements if the site is large. The site should be emptied of people prior to deploying the drone, leaving only the operation team in the area for recovery. This also aims to ensure privacy of people who may live or work inside the inspected buildings as the drone would need to take pictures of the site and store them in the database for review if defects are detected. Furthermore, the data collected and VLM descriptions should only be used to aid structural engineers in their reviews of the structural integrity. The VLM output alone should not be used to make decisions on whether particular cracks should be addressed.

The original proposal involved demonstrating the objective of achieving a low-cost solution by integrating both the crack detection and VLM on the same hardware. However, due to the constraints presented in **Section 2.2**, this was unachievable, resulting in a reliance of an offboard system to handle the reasoning model processing.

3.2. Functional Requirements

Table 1 presents the core functional requirements of a prospective autonomous MAV inspection system, developed to address the high-level objectives described in **Section 3.1**. These requirements define the essential capabilities the system must possess to perform reliable, efficient, and safe inspections of civil structures. Each requirement reflects a specific functional aspect, ranging from autonomous navigation and crack detection to intuitive user interaction and data logging, ensuring the system can operate effectively in real-world environments. Together, these requirements guided prototype implementation decisions and aided in verifying that the system fulfilled its intended purpose.

Table 1: Functional requirements for an autonomous MAV design performing inspections on civil infrastructure.

Function	Requirement Description
Autonomous Navigation	The drone must autonomously navigate a predefined or dynamically mapped inspection area, maintaining stable flight and avoiding static obstacles.
Speech Command Transcription	The system must interpret short spoken waypoint commands and convert them into navigation instructions, using onboard or offboard inference.
Crack Detection	The system must identify surface cracks on structural elements during flight.
Real-time VLM Descriptions	The system must use a vision-language model to generate textual descriptions of each detected crack, including relevant severity information and recommended actions.
Drone Database Monitoring	Each crack detection must be recorded in an SQL database along with the time, location, captured image, and descriptive output.
User Interface Access	Users must be able to access and browse the logged database entries, with entries clearly defining the severity of cracks.
Onboard Data Management	The system must manage data collection in a way that preserves sufficient computational and memory resources for critical ROS processes, ensuring stable operation during inspections.

3.3. Design Constraints

The design is constrained in physical implementation, as well as in performance to ensure it provides competition within the MAV surveying market. For this reason, the popular DJI Mavic 3 Enterprise drone is used as a reference for developing the constraints as it is the industry standard for inspection operations [23]. **Table 2** below summarizes the key constraints associated with the problem.

Table 2: Key drone constraints for effective crack detection.

Metric	Constraint
Dimensions (Without Propellers)	Maximum of 347.5×283×107.7 mm (L×W×H).
Flight Time	Assuming nominal flight behaviour (indoor environment, battery fully charged), the drone shall be able to fly 5 iterations (take off, navigate to waypoints and analyze data).
Battery Replacement Time	Must have a modular method of easily replacing batteries to minimize downtime between flights. Maximum 30 seconds for replacement time.
Communications	Must have onboard WiFi or LTE communication.
Storage	Minimum 128GB microSD.
Memory	Minimum 4GB onboard RAM.
Hovering Accuracy	Maximum vertical error of 0.1m. Maximum horizontal error of 0.3m.
Detection Accuracy	Minimum of 90% detection accuracy of cracks.
VLM Description Accuracy	Minimum of 90% description accuracy.
Operation Cost	Must be less expensive than the current human-equivalent approaches to the problem.
Image Quality	Minimum 1080p resolution for detailed crack and corrosion analysis. Camera should work in low-light condition.

4. Design Prototype Overview

4.1. High Level System Rundown

The high-level diagram of the prototype is shown in **Figure 1** and consists of two main components. The first of which, the offboard system, consists of a speech-to-text parser, an API web-based SQL database, and a VLM, which are hosted externally of the drone on a MacBook with an M3 compute chip. The SQL database includes entries for job ID, date, time, status of the current request, site-of-interest name, image, VLM description, and a manual inspection flag that is set to “yes” if the VLM classifies the inspection result as severe and is suggesting structural engineers to manually inspect the detection on site. The VLM layout is depicted in **Figure 2** below.

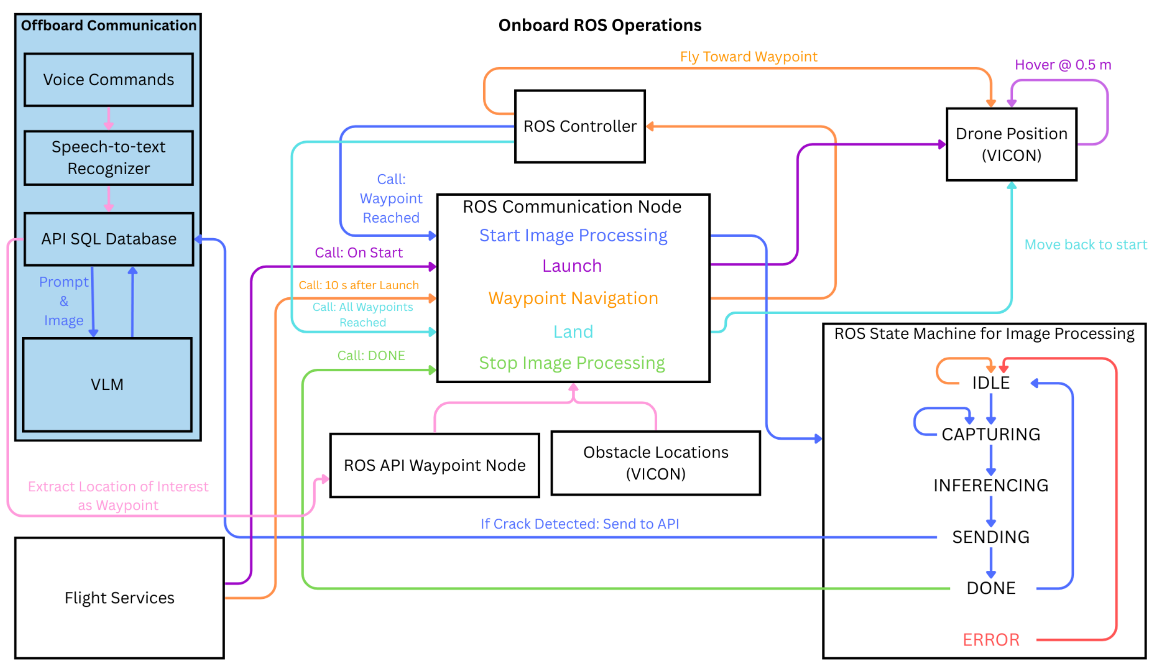
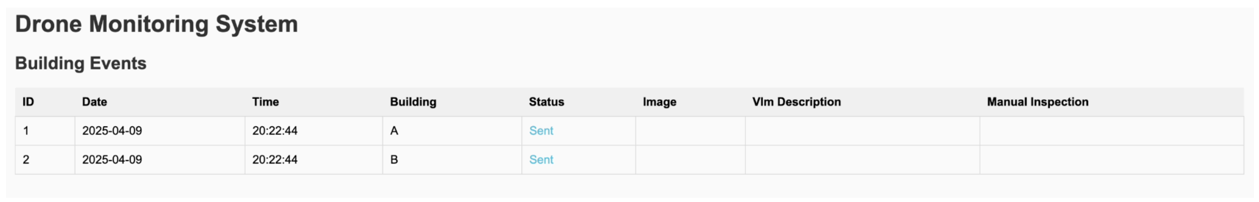


Figure 1: High-Level System Diagram.



Drone Monitoring System

Building Events

ID	Date	Time	Building	Status	Image	Vlm Description	Manual Inspection
1	2025-04-09	20:22:44	A	Sent			
2	2025-04-09	20:22:44	B	Sent			

Figure 2: Snippet of the web-based API frontend with two waypoints, A and B, sent for the drone to receive and inspect.

The second component, the onboard ROS communication system, integrates the drone and obstacle VICON positions, the flight controller, an API node to process sent waypoints, and the state machine to take images using the onboard camera and run inference. The API waypoint node stores the coordinate locations of the keypoints on each obstacle, with the positions given by VICON markers on the obstacles. The controller used to navigate to waypoints operates at 20 Hz and includes a simple obstacle avoidance algorithm inspired by potential fields, where each site, treated as obstacles, are confined in a safety circle of radius 0.8 m. When the drone is not within a safety circle, it flies at most 0.75 m toward the current desired waypoint. Otherwise, a repulsive vector (represented by the red arrow in **Figure 3**) is added to the aforementioned 0.75 m movement vector (represented by the purple vector in **Figure 3**) that originates from the centre of the obstacle's safety circle, and points away from the obstacle. This vector is rescaled by the original magnitude of the movement vector, which is at most 0.75 m. The resulting movement vector is directed away from the obstacle, avoiding a collision. All ROS components are integrated using a central ROS communication node that starts all nodes on different threads, and controls communications using services.

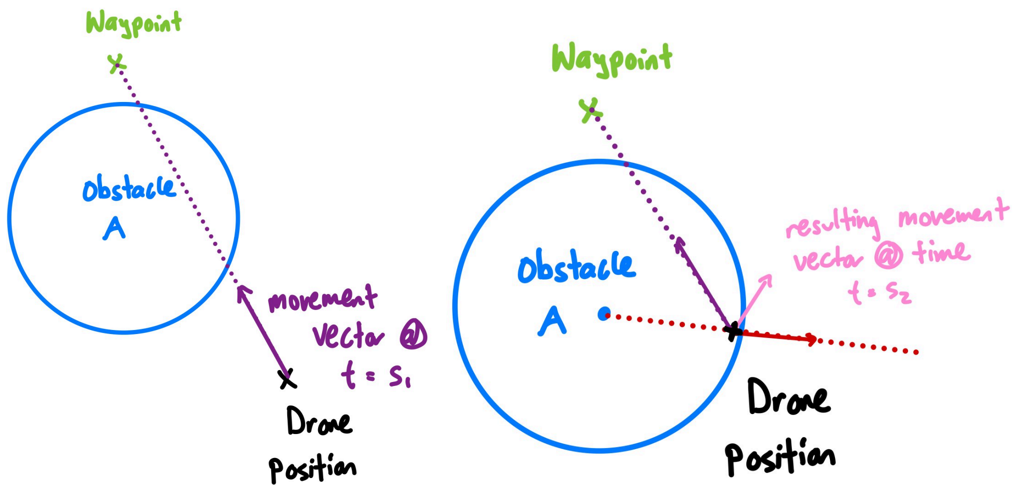

Waypoint
Obstacle
A
movement vector @ t = s_1
Drone Position
Waypoint
Obstacle
A
resulting movement vector @ time t = s_2
Drone Position

Figure 3: (Left): Controller action when drone is not inside an obstacle. (Right): Controller action when drone breaches an obstacle's safety circle.

The default state of the drone in the state machine is the "IDLE" state, where no inferencing functionality is run. When starting up the drone and flying, the "IDLE" state is active to prevent inferencing from occurring. The "CAPTURING" state, which is triggered by the "Start Image Processing" service call, captures five images at 1 Hz, undistorted using the camera calibration matrix and selects the best image by using a Laplacian variance metric. This chooses the least blurry image of the five by selecting the highest laplacian variance, and saves it for inference. Once done capturing and selecting, the "INFERENCING" state is triggered, where a red bounding box is cropped out of the best image using an OpenCV pixel-wise method, and passed to the Darknet model to detect bounding boxes with confidence scores for "crack" and "no crack" labels. If one bounding box classifying "crack" with over a confidence threshold of 0.3 is detected, the overall result of the classification is true, and otherwise the result is false. This threshold was tuned manually to achieve the best performance. Once a classification has been extracted, the "SENDING" state is triggered, where if the classification is true, the image and a flag to prompt the VLM for a description are sent to the offboard API. Otherwise, no image is sent, and a message of "No Crack Detected for this Waypoint" is written into the VLM field. The VLM is not called if no crack is detected to prevent the VLM from hallucinating that a crack is in the image, which was a common result in testing. When the request is sent, the "DONE" state is triggered, and the VLM generates an output independent of the onboard ROS system. This state calls the "Stop Image Processing" service, which communicates to the drone that inferencing is done, and the drone can proceed to the next waypoint. The "ERROR" state is for safety purposes only, being called only when a process in one of the aforementioned states throws an unexpected error. This simply calls the "DONE" process, allowing the drone to skip the current waypoint and proceed to the next waypoint.

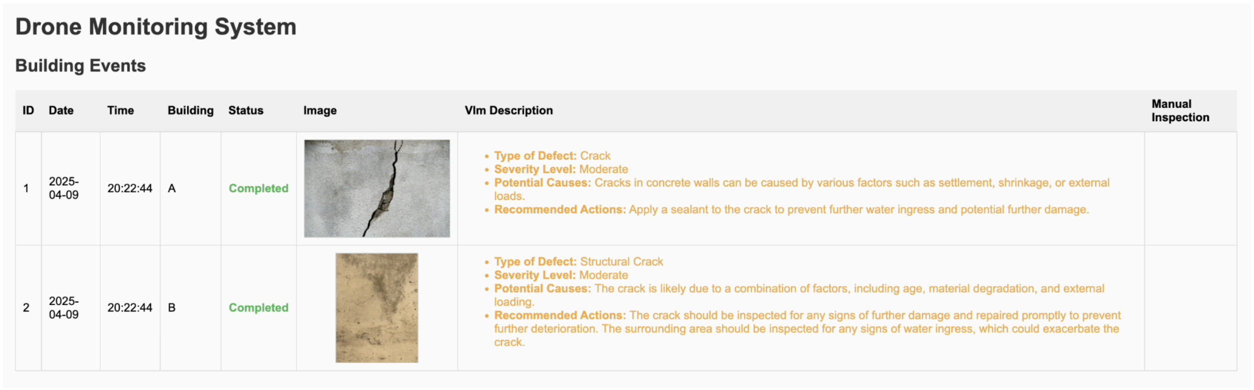
To kick off inspections, a user inputs a voice command on the offboard system describing where to inspect: for example, "Performing inspection on building A and building B". The Whisper speech-to-text model transcribes the voice snippet and parses the extracted text for the site locations; in this example, "A", and "B" would be extracted. The desired locations are processed as entries in the web-based API SQL database with the status field marked as "Sent". The onboard API waypoint node picks up any requests marked "Sent", retrieves the coordinates for each site of interest, and publishes the coordinates as waypoints to a waypoint topic. The status field for each processed request is marked by the node as “En Route”. To begin the flight, the central ROS node is started, which instantiates the controller and the state machine, loads the YOLO Darknet model, and captures both waypoint names and coordinates using a dictionary from the earlier API waypoint node process. A user manually sending the “Launch” ROS service causes the drone to take off at a low height of 0.3 m. Manually sending the “Waypoint Navigation” ROS service kicks off the flight, where the drone flies to the first waypoint using the control mechanism detailed earlier. When the target waypoint is reached, the communication node autonomously sends the “Start Image Processing” service to run inferencing and send the result to the API. The “Stop Image Processing” service is triggered when the process is done, causing the drone to fly to the next waypoint and repeat the process. Once all waypoints are visited, the “Land” service is autonomously called, causing the drone to land in the home position, corresponding to the origin of the VICON frame.

Results can be interpreted by viewing the SQL database on the web client it is deployed on, where images and VLM descriptions for each inspection are logged. The status of each request is flagged as “Completed” after each waypoint is reached. An example output for test structural defect images is shown in **Figure 4.** The input prompt to the VLM is experimentally tuned based on the quality of the output descriptions and the model’s adherence to the prompt structure. The prompt that provided the most informative responses from the perspective of a structural analyst is given in **Table 3**, where post processing is performed to structure the output into a consistent format that is acceptable for database presentation and readability.

“You are an expert in structural analysis. Make sure to follow the format strictly. If the image is not a structural crack, state 'None' in each section. Otherwise, carefully analyze the structural crack image and provide insights in the following structured format. Give bullet responses for each section:

- *Type of Defect: Describe the defect or anomaly present in the image*
- *Severity Level: Estimate the severity as a Minor, Moderate, or Severe class*
- *Potential Causes: Explain what could have led to the defect*
- *Recommended Actions: Suggest appropriate mitigation steps”*

Table 3: Prompt used to generate structured outputs from the VLM for crack analysis.



Drone Monitoring System

Building Events

ID	Date	Time	Building	Status	Image	Vlm Description	Manual Inspection
1	2025-04-09	20:22:44	A	Completed		• **Type of Defect:** Crack • **Severity Level:** Moderate • **Potential Causes:** Cracks in concrete walls can be caused by various factors such as settlement, shrinkage, or external loads. • **Recommended Actions:** Apply a sealant to the crack to prevent further water ingress and potential further damage.	
2	2025-04-09	20:22:44	B	Completed		• **Type of Defect:** Structural Crack • **Severity Level:** Moderate • **Potential Causes:** The crack is likely due to a combination of factors, including age, material degradation, and external loading. • **Recommended Actions:** The crack should be inspected for any signs of further damage and repaired promptly to prevent further deterioration. The surrounding area should be inspected for any signs of water ingress, which could exacerbate the crack.	

Figure 4: Snippet of web-based API frontend after completed inspection.

4.2. Evaluation of Proposed System Design

The two main design goals are robust perception and interpretable description. The prototype achieves the latter exceptionally, where any cracks perceived from the drone on the offboard side are neatly organized in the easy-to-use web-based frontend. The API SQL backend at minimum is adequate to be rolled out on a real system, with only stylization needed to deploy the application on a visually appealing frontend. In terms of perception, the solution can successfully detect and classify cracks at target sites, but has to be given the exact location of where to look using a red bounding box. For prototype testing, this red bounding box is necessary because the flight arena where testing is done is not reminiscent of a realistic inspection site, causing the Darknet inference model to often get confused with other surroundings. This would not be functional for the real system, where cracks are not guaranteed to show up at one specific location. Future work with the detection model should involve contacting civil engineering firms to collect a wider variety of structural defect image data for training to ensure high accuracy inference results over a variety of situations, particularly when in the presence of distractive feature surroundings.

The use of a VLM to describe cracks addresses the objective of improving inspection efficiency as it provides the ability for structural engineers to review the drone surveying data in order of greater severity, as determined by the VLM, therefore reducing the average time required to have an engineer review a potentially critical crack among the other less severe detections. Moreover, the VLM acts as a bridge between the drone's visual perception of cracks and human interpretability, helping individuals better understand the reasons for further investigation, prior to consulting experienced structural engineers with the resulting data. However, the VLM tends to produce biased outputs, primarily classifying cracks as “moderate” in severity and often repeating aspects of previous recommendations. Future work requires fine-tuning the VLM on a dataset of structural crack images, queries and answer labels annotated by structural engineers to significantly reduce the likelihood of hallucinations and repetition when providing feedback on the detected crack.

As mentioned in **Section 2.2**, the use of a speech-to-text model to transcribe waypoint commands for drone flight addresses the aim of simplifying the process of interacting with the design solution, particularly for users unfamiliar with the system from a technical perspectives, such as onsite managers and foreman, as it removes the need for specialized inputs or interfaces and enables fast, natural communication with the MAV. This, along with the structured classification and recommendations provided by the VLM, ensure that user interactions are as simple as possible, shifting operational focus on reviewing the detection results in real time on an organized database. The simplifications presented within the solution aim to encourage multiple inspections to be conducted within a given time period, depending on the observations described by the VLM and structural experts. The prototype currently requires manually running backend commands to perform speech-to-text inference and send the resulting waypoint text to the SQL database. For a complete real-world implementation, the speech-to-text model would need to be integrated directly within the website to ensure frontend support for commanding and interacting with the MAV through a simple interface with transcription and database updates handled seamlessly in the backend.

In terms of the objective of minimizing human labour and risk, intervention is minimized in tests since the prototype only requires input of key waypoints and voice prompts whenever an inspection is necessary, and risks are mitigated through the obstacle avoidance system. However, a key pain point exists that waypoints and obstacles must be manually mapped by workers and inserted into the onboard ROS API node. This is inconvenient in the first tests for every new set of waypoints, but successive tests are much easier to execute. The objectives of enabling early intervention, prioritizing critical detections, and ensuring human interpretability are also achieved in the proposed design due to the integrated API solution, where the speech-to-text component is confirmed to be robust to loud background noise and the VLM output descriptions are color classified based on detected severity. Specifically, minor classifications are output in blue, moderate classifications are output in orange, and severe classifications are output in red to provide users with a clear visualization regarding which locations should be prioritized during further inspection.

To scale up the prototype for deployment, a better frontend solution to improve the visuals of the website, such as using React and CSS styling, is necessary, as mentioned, to make the solution more visually appealing and interactive to use. Additionally, a different localization system is needed since the reliance on a VICON system for localization in an outdoor environment is not practical and technically possible. SLAM or A* on a pre-mapped layout of the site with the aid of GPS could be two potential alternatives, where more robust obstacle avoidance can also be achieved as a side effect of better mapping.

The main component to improve is the detection method, where a more robust architecture is needed to eliminate the need for red bounding boxes to direct the model as to where exactly to look. Using larger YOLO models with more parameters, or more robust architectures such as vision transformers, can be two ideas to improve detection if more computationally powerful hardware is used in the real design. Instead of hovering in place when capturing images, it would be ideal to sweep around a site to cover more surface area, and run detection on a set of images at every waypoint. Inferencing asynchronously during flight can be implemented to reduce flight times, where bigger models and more images to infer on can become a bottleneck causing the drone to loiter at a site longer than needed.

5. Results & Evaluation

5.1 Testing Setup

The buildings are simulated using columns from the MY580 area. Cracks are represented by printed images on paper, which are glued onto cardboard boxes. A red bounding box is drawn around each crack, as shown in **Figure 5**. Two waypoints are tested to establish the baseline performance of the entire system.

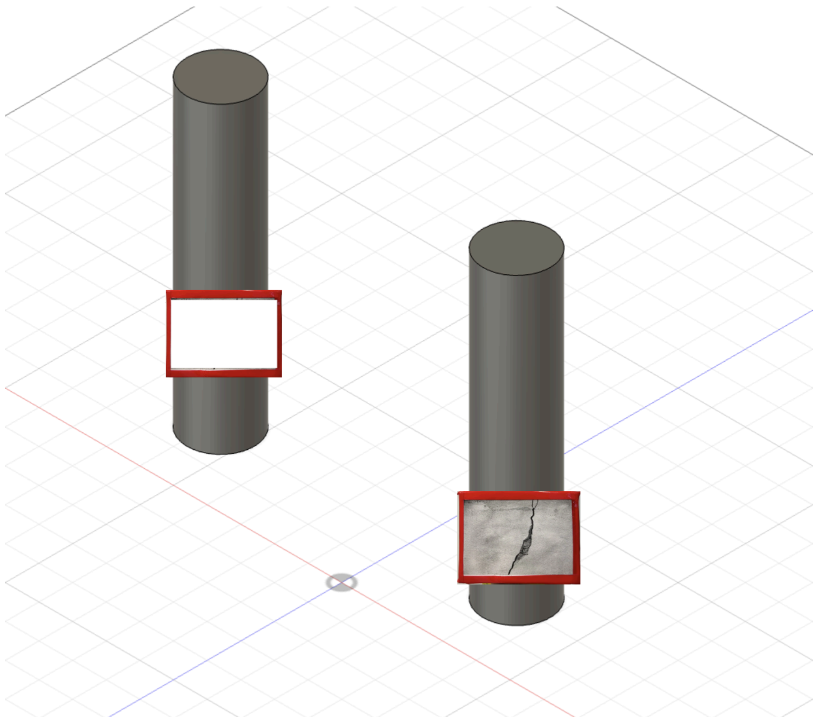


Figure 5: Testing setup for the system functionality verifications

5.2 Verification Against Requirements

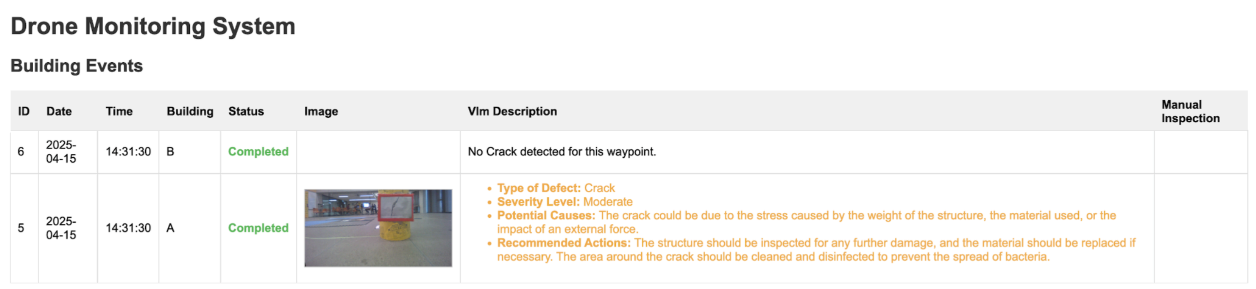
The technical requirements were based on the DJI Mavic 3 Enterprise, as referenced in **Table 2**. Given the constraints of limited hardware resources and time, certain requirements and verification processes have been adapted to better align with our prototype testing, as detailed in **Table 4** below.

Table 4: Technical verification plan and results

Metrics	Verification methods	Results
Dimensions (Without Propellers)	Measure with tapes and calipers	Target requirement was met. The Z height is slightly over DJI Mavic 3 enterprise's design, but it could easily be fixed if we put the cube inside the frame. Actual dimension: 210mm×230mm×140mm
Battery rundown test	Use mavros battery percentage ROS topic to check battery capacity left after each flight.	Target requirement was met. Each flight drains around 17% of the battery, 5 iterations should draw 85% of the battery.
Battery Replacement Time	Test start: unplug battery after switching networks and saving progress. Test end: user replaces battery and drone boots up. Time will be calculated as the difference of the above time.	Target requirement was met. Most of the time was spent on tightening the velcro straps. Replacement time around 15 seconds
Communications	The drone will be able to communicate with SQL server using WIFI, blank entries on the web UI or maximum retries exceeded error on the terminal will indicate a failure.	Target requirement was met. All results being relayed through API to SQL database correctly as shown in **Figure 6** below.
Storage	No out-of-storage warning was observed. Typical out-of-storage behavior includes the drone automatically descending or entering a stable hover due to the Jetson system rebooting.	Target requirement was met. No flight discontinuities were observed during testing.
Memory	No out-of-memory (OOM) warning was observed. Typical signs of OOM include the drone automatically descending or entering a stable hover as a result of the Jetson system rebooting.	Target requirement was met. No flight discontinuous observed during testing
Obstacle Avoidance & Hovering Accuracy	VICON data was used to plot the drone's 3D trajectory during autonomous flight.	Target requirement was met. As shown in **Figure 7**, the obstacle avoidance algorithm functions as

	Obstacle avoidance was visually confirmed, with no collisions observed during flight. Hovering accuracy was validated by observing the drone stabilizing at each waypoint and analyzing the X, Y, and Z deviations on the plotter.	intended—the potential field vector effectively directs the drone away from the defined safety radius. Once the drone stabilizes at a waypoint, the horizontal drift remains under 0.1 m, and the vertical drift is also less than 0.1 m.
Detection Accuracy	Due to limited testing time, detection inference was performed using the IMX219 camera to capture images of various crack samples taped onto a cardboard box. A total of 20 images were tested—16 contained cracks of varying severity, while the remaining 4 depicted no cracks (either blank walls or regular concrete surfaces).	Target requirement was met. 19 images out of 20 are identified correctly (95% accuracy). One image that is missed is due to camera resolution making the cracks blurry. In future, training with the deployment camera would be ideal.
VLM Description Accuracy	After experimentally tuning the algorithm using a training set, we selected 10 test images—7 containing cracks ranging from multiple major cracks (5 or more) to a single minuscule crack, and 3 showing no cracks (plain white or concrete walls). We then verified whether the VLM outputs aligned with the expected responses based on the prompts.	Target requirement was met. All 10 images were correctly described in accordance with our guidelines, although some results exhibited repetitive phrasing and similar outputs.
Image Quality at various conditions	The model must be capable of detecting and analyzing images under all lighting conditions in the MY580 environment. Two conditions were tested: (1) with the MY580 ceiling lights on, and (2) with the motion-sensing lights off. For the second condition, we waited several minutes for the motion-sensing lights to automatically turn off before	Target requirement was met. The algorithm produced consistent outputs under both lighting conditions—when the MY580 motion-sensing light was on and when it was off.

	conducting inference tests.	



Drone Monitoring System

Building Events

ID	Date	Time	Building	Status	Image	Vlm Description	Manual Inspection
6	2025-04-15	14:31:30	B	Completed		No Crack detected for this waypoint.	
5	2025-04-15	14:31:30	A	Completed		• **Type of Defect:** Crack • **Severity Level:** Moderate • **Potential Causes:** The crack could be due to the stress caused by the weight of the structure, the material used, or the impact of an external force. • **Recommended Actions:** The structure should be inspected for any further damage, and the material should be replaced if necessary. The area around the crack should be cleaned and disinfected to prevent the spread of bacteria.	

Figure 6: Testing setup web UI result from in-flight communication.

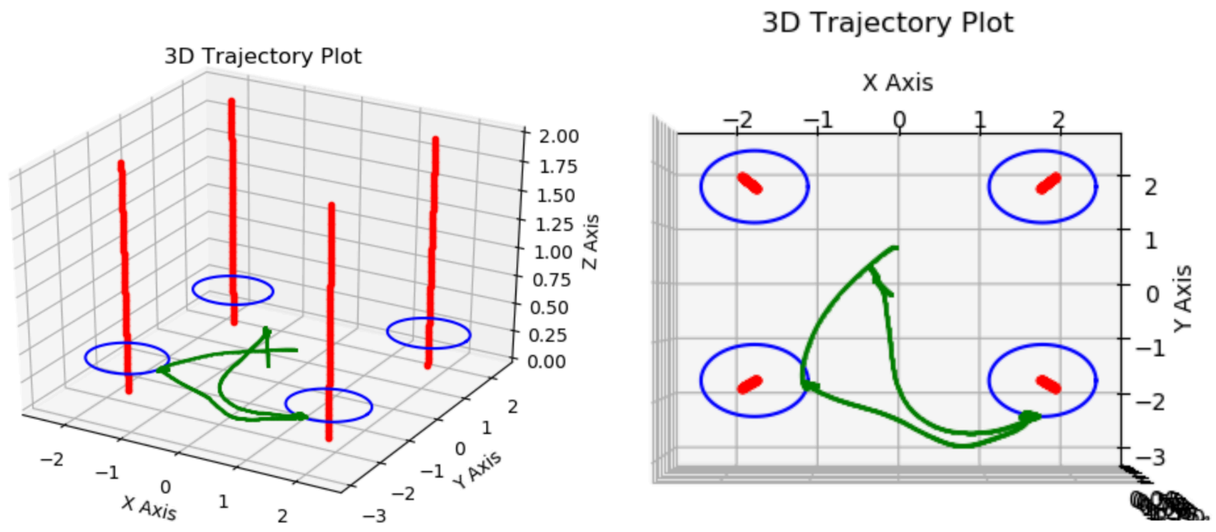


Figure 7: Obstacle avoidance potential field algorithm results

5.3 Validation against requirements

To take this capstone project one step further, we consulted three external stakeholders: a nuclear engineer specializing in nuclear dam design, a mechanical engineer with experience in robotic systems, and a materials engineer focused on structural design.

To effectively evaluate whether we have built the right system, we focused on three key aspects: VLM result interpretability, operational cost, and overall system usability. Custom metric tables were developed to assess VLM interpretability and system usability, as shown in **Table 5** and **Table 6** below.

Table 5: VLM interpretability rubric

Usability Rubric					
Metrics Category	Metric details	Inadequate	Adequate	Good	Great
VLM interpretability	**Structural-Aware Semantic Parsing** *(Does the model produce relevant, structured, and domain-appropriate content for each prompt section?)*	Output is irrelevant, incomplete, or ignores structure. Uses generic or non-engineering terms.	Follows structure but with shallow or partially inaccurate content. Some bullet points are vague or missing.	Accurately fills each section with appropriate structural context; mostly relevant and well-phrased.	Fully structured, domain-specific, and precise. All bullet points are relevant and aligned with civil/structural engineering expectations.
	Visual-Textual Grounding *(Are the text responses clearly linked to actual visual features in the image, such as crack width, location, or direction?)*	Text has no link to image content. Descriptions are imagined or mismatched.	Some loose correspondence to visible features, but lacks clarity or specificity.	Most textual claims relate to visible cracks. Mentions of direction, location, or size are reasonably inferred.	Highly specific references to observable features. Crack properties directly justify each description.
	Interpretability of Recommended Actions *Are the suggested actions logical, proportional to severity, and interpretable for structural maintenance?*	Recommendations are vague, irrelevant, or over/under reactive. No actionable insight.	Actions are generic (e.g., “monitor” or “repair”) but show some connection to the issue.	Actions are appropriate, reasonably scaled to severity, and interpretable by practitioners.	Actions are clear, actionable, and technically justified (e.g., sealing, structural reinforcement, further inspection). Tailored to defect type and severity.

Table 6: System usability rubric

System Usability	**Mission Setup & Deployment** How easy is it to prepare and initiate a flight mission?	Setup process is confusing or error-prone; significant technical intervention needed.	Setup is possible with effort; instructions or documentation are necessary.	Setup is smooth with clear steps and basic instructions; low risk of error.	Setup is fast, intuitive, and reliable; users can deploy with minimal training or support.
	UI Clarity & Pre-Flight Feedback *How clearly does the system communicate readiness, mission details, and potential issues?*	UI lacks clarity; critical system states or errors are hard to interpret.	Key info is available but may be buried or inconsistently presented.	UI presents important info clearly; users can interpret pre-flight status confidently.	All system states are clear and well-visualized; users have full situational awareness before launch.
	Autonomous Flight Confidence *How confident are users in the drone completing the mission correctly once launched?*	Users frequently experience mission failures or unexpected behavior.	Missions usually complete but users remain unsure or anxious during flight.	Users trust the system for most missions; behavior is generally consistent.	Users have full confidence in the system's autonomy; flight behavior is predictable, safe, and robust across scenarios.

Overall, VLM interpretability and system usability were rated as good on average. Some of the comments from the three stakeholders are summarized below.

Nuclear Engineer:

"The VLM demonstrates good structural parsing, offering organized outputs; however, interpretability could be improved by incorporating more context-specific insights related to safety and system integrity. Crack identification is generally reliable, though interpretability would benefit from additional details about the location or criticality of affected areas. While the recommended actions are clear, they remain somewhat generic—more specific, actionable guidance would increase their practical value."

"System usability is strong overall. The setup process is intuitive and supports quick deployment, though it could be enhanced by including detailed operational checks relevant to high-stakes environments. The user interface clearly conveys mission information, but integrating safety-related warnings and checks would improve its effectiveness. Confidence in autonomous operation is solid."

Materials Engineer:

"The VLM outputs are well-structured and relevant, aligning with general engineering expectations. Still, the inclusion of material-specific insights—such as degradation mechanisms or fatigue indicators—would further enhance utility. Visual descriptions are accurate in identifying surface defects, though deeper characterization of material properties, like depth or corrosion potential, would improve applicability. Recommended actions are logical and proportionate, but they would benefit from incorporating materials-based interventions to strengthen practical relevance."

"System usability is strong, with robust deployment and clear integration of key functionalities. However, incorporating procedures for monitoring material degradation during setup would make the system more practical for material-sensitive applications. The user interface is effective in conveying essential mission data. Confidence in autonomous operation is high, and expanding validation under varied environmental or surface conditions would further reinforce system reliability."

Mechanical Engineer:

“The VLM produces structured and relevant outputs, though further refinement could enhance the precision of technical insights. Descriptions of physical defects are generally accurate and visually consistent, but adding more detail on their mechanical implications—such as potential impact on functionality—would improve interpretability. While recommended actions are appropriately scaled to severity and understandable, they could be made more practical with task-specific suggestions. System usability is strong overall. The setup and deployment process is smooth and mechanically well-integrated, though slight improvements in handling and maintenance procedures could boost efficiency. The user interface communicates essential mission data clearly, supporting user confidence. Autonomous flight appears reliable, and performance consistency suggests sound system behavior; however, broader validation in varied operational contexts could further strengthen trust.”

Overall, external stakeholder feedback has been positive. However, several improvements are needed to make the VLM more task-specific. In particular, descriptions should be less repetitive and more precise regarding the location and orientation of cracks. These enhancements can be achieved by fine-tuning the LLM with domain-specific image datasets.

At the system level, usability could be further improved by enhancing the web UI to display more drone status information—such as battery levels—and by streamlining the launch process to reduce reliance on manual service calls.

Finally, operational cost calculations were conducted. The total cost of the drone is estimated at approximately $1,500 CAD based on the hardware bill of materials. This figure could be further reduced through economies of scale as more units are produced. In contrast, traditional structural inspections by a professional engineer cost around $220 CAD per hour. Assuming each critical site requires biannual inspections, with an average of four sites and approximately five hours of engineering time per site (including on-site assessment and report preparation), the total annual cost could reach up to $8,800 CAD.

In comparison, our drone-based solution offers a highly cost-effective alternative, significantly reducing reliance on expensive manual inspections. It not only lowers operational expenses but also enables more frequent and scalable monitoring. This opens the door for proactive maintenance strategies and earlier detection of structural issues, ultimately improving safety and asset longevity.

Finally, operational cost calculations were conducted. The total cost of the drone is estimated at approximately $1,500 CAD based on hardware bill of materials. This figure could be further reduced through economies of scale as more units are produced. In contrast, traditional structural inspections by a professional engineer cost around $220 CAD per hour [24]. Assuming each critical site requires biannual inspections, with an average of four sites and approximately five hours of engineering time per site (including on-site assessment and report preparation), the total annual cost could reach up to $8,800 CAD.

In comparison, our drone-based solution offers a highly cost-effective alternative, significantly reducing reliance on expensive manual inspections. It not only lowers operational expenses but also enables more frequent and scalable monitoring. This opens the door for proactive maintenance strategies and earlier detection of structural issues, ultimately improving safety and asset longevity.

6. Lessons Learned

In this project, unique technical challenges were encountered. Namely, the fickle nature of drone software and hardware often posed issues, including package build failures, camera disconnections causing the camera pipeline to not start, and drone crashes caused by faulty wiring. These issues can occur both when code changes are made or when impact damage is incurred, and even when no changes are made: resulting in a need to reboot the system. Safety is a primary concern in this project, as crashes incur monetary and temporal cost, and should take these small issues that can arise mid-flight into account. Thus, it is essential to discuss and adopt rigorous development practices from the beginning of development.

One important strategy employed in this project was developing a framework to test the full functionality of the system without flying by publishing fake data to the VICON and waypoint topics. Debugging without flying is crucial to avoid sustaining crashes, which can cause substantial time to be lost in setting up a new platform. This also significantly streamlines debugging in downtimes when the flight area is not available. Additionally, a debugging approach of forming a list of possible causes, and working to test the proposed issues one-by-one proved to be effective for this project and goes hand-in-hand with the developed dummy data testing framework. Nothing should be ruled out prematurely, as problems are often interconnected. For example, a loose cable could cause a cascading failure that leads to a software component malfunction. Maintaining version control is critical for saving functioning software components, in case a crash is sustained, or performance cannot be recovered when changes are made. When actually flying, it is important to have a safety protocol that can land the drone instantly, ready to further prevent damage to the drone by collision.

In the design of software components, the Jetson platform’s compute limitations were a key factor to design around, where low compute cost functionality was generally preferred to prevent the drone from overloading mid-flight. This includes optimizing code and choosing low-cost frameworks such as convolutional neural networks. Additionally, while chasing a fully robust system is ideal, designing systems that consider every minute detail is often unnecessary, and can become a bottleneck. For example, taking the onboard frame transform between the VICON marker and the Cube flight controller was an avenue pursued in this project to get millimetre precision, but ultimately ignoring this detail would yield similar performance since the scale of the waypoint navigation was much larger than the millimetre precision obtained. Thus, pursuing the simplest solutions to problems tends to be more achievable under time constraints, and can yield similarly robust solutions in comparison to the performance of systems taking every small detail into account.

For future teams taking on a similar design challenge, the main takeaways from the above discussion are to develop a testing framework to ensure that components are functioning, develop safety procedures to prevent damage during flight, and define which problems matter, and ignore smaller issues to find the simplest solutions to problems. Additionally, starting development of software early is important as debugging takes up a substantial amount of time. Parallelizing team work can help with doing this, as some team members can focus on developing discussed components, while others move forward in designing future systems to be developed.

7. References

[1] Wang, W.; Deng, L.; Shao, X. Fatigue design of steel bridges considering the effect of dynamic vehicle loading and overloaded trucks. J. Bridge Eng. 2016, 21, 04016048. [Google Scholar] [CrossRef]

[2] Zheng, K.; Zhou, S.; Zhang, Y.; Wei, Y.; Wang, J.; Wang, Y.; Qin, X. Simplified evaluation of shear stiffness degradation of diagonally cracked reinforced concrete beams. Materials 2023, 16, 4752. [Google Scholar] [CrossRef]

[3] Xu, Y.; Bao, Y.; Chen, J.; Zuo, W.; Li, H. Surface fatigue crack identification in steel box girder of bridges by a deep fusion convolutional neural network based on consumer-grade camera images. Struct. Health Monit. 2019, 18, 653–674. [Google Scholar] [CrossRef]

[4] Zhang, L.; Yang, F.; Zhang, Y.D.; Zhu, Y.J. Road crack detection using a deep convolutional neural network. In Proceedings of the International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3708–3712. [Google Scholar]

[5] Zou, Q.; Cao, Y.; Li, Q.; Mao, Q.; Wang, S. CrackTree: Automatic crack detection from pavement images. Pattern Recognit. Lett. 2012, 33, 227–238. [Google Scholar] [CrossRef]

[6] Dung, C.V.; Anh, L.D. Autonomous concrete crack detection using deep fully convolutional neural network. Autom. Constr. 2019, 99, 52–58. [Google Scholar] [CrossRef]

[7] Oliveira, H.; Correia, P.L. Automatic road cracks detection and characterization. IEEE Trans. Intell. Transp. Syst. 2012, 14, 155–168. [Google Scholar] [CrossRef]

[8] Liu, Y.; Fan, J.; Nie, J.; Kong, S.; Qi, Y. Review and prospect of digital-image-based crack detection of structure surface. China Civ. Eng. J. 2021, 54, 79–98. [Google Scholar]

[9] H. S. Munawar, A. W. A. Hammad, A. Haddad, C. A. P. Soares, and S. T. Waller, "Image-based Crack Detection Methods: A Review," MDPI, https://www.mdpi.com/2412-3811/6/8/115 (accessed Feb. 5, 2025).

[10] Gavilán, M.; Balcones, D.; Marcos, O.; Llorca, D.F.; Sotelo, M.A.; Parra, I.; Ocaña, M.; Aliseda, P.; Yarza, P.; Amírola, A. Adaptive road crack detection system by pavement classification. Sensors 2011, 11, 9628–9657. [Google Scholar] [CrossRef] [PubMed]

[11] Zou, Q.; Zhang, Z.; Li, Q.; Qi, X.; Wang, Q.; Wang, S. Deepcrack: Learning hierarchical convolutional features for crack detection. IEEE Trans. Image Process. 2018, 28, 1498–1512. [Google Scholar] [CrossRef] [PubMed]

[12] Nishikawa, T.; Yoshida, J.; Sugiyama, T.; Fujino, Y. Concrete crack detection by multiple sequential image filtering. Comput. Civ. Infrastruct. Eng. 2012, 27, 29–47. [Google Scholar] [CrossRef]

[13] Fernández, A.C.; Rodríguez-Lozano, F.J.; Villatoro, R.; Olivares, J.; Palomares, J.M. Efficient pavement crack detection and classification. EURASIP J. Image Video Process. 2017, 2017, 1–11. [Google Scholar]

[14] Liu, X.H.; Danczyk, A. Optimal sensor locations for freeway bottleneck identification. Comput. Aided Civ. Infrastruct. Eng. 2009, 24, 535–550. [Google Scholar] [CrossRef]

[15] Yeum, C.M.; Dyke, S.J. Vision—Based automated crack detection for bridge inspection. Comput. Aided Civ. Infrastruct. Eng. 2015, 30, 759–770. [Google Scholar] [CrossRef]

[16] Zhang, A.; Wang, K.C.P.; Li, B.; Yang, E.; Dai, X.; Peng, Y.; Fei, Y.; Liu, Y.; Li, J.Q.; Chen, C. Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. Comput. Aided Civ. Infrastruct. Eng. 2017, 32, 805–819. [Google Scholar] [CrossRef]

[17] Bang, S.; Park, S.; Kim, H.; Kim, H. Encoder-decoder network for pixel-level road crack detection in black-box images. Comput. Aided Civ. Infrastruct. Eng. 2019, 34, 713–727. [Google Scholar] [CrossRef]

[18] Pauly, L.; Peel, H.; Luo, S.; Hogg, D.; Fuentes, D.H.A.R. Deeper networks for pavement crack detection. In Proceedings of the 34th ISARC, Taipei, Taiwan, 28 June–1 July 2017; pp. 479–485. [Google Scholar]

[19] Li a b et al., “Automatic bridge crack detection using unmanned aerial vehicle and faster R-CNN,” Construction and Building Materials, https://www.sciencedirect.com/science/article/pii/S0950061822033153 (accessed Feb. 5, 2025).

[20] Yu, L., Yang, E., Luo, C. *et al.* AMCD: an accurate deep learning-based metallic corrosion detector for MAV-based real-time visual inspection. *J Ambient Intell Human Comput* 14, 8087–8098 (2023). https://doi.org/10.1007/s12652-021-03580-4

[21] Pan, P., Hu, K., Huang, X., Ying, W., Xie, X., Ma, Y., ... & Kang, H. (2024). Developing Smart MAVs for Autonomous Inspection in GPS-denied Constructions. *arXiv preprint arXiv:2408.06030*.

[22] Marafioti, A., Zohar, O., Farré, M., Noyan, M., Bakouch, E., Cuenca, P., Zakka, C., Allal, L. B., Lozhkov, A., Tazi, N., Srivastav, V., Lochner, J., Larcher, H., Morlon, M., Tunstall, L., von Werra, L., & Wolf, T. (2025). SmolVLM: Redefining small and efficient multimodal models. ArXiv Preprint ArXiv:2504.05299.

[23] DJI. Mavic 3 Enterprise. Enterprise DJI. Available at: https://enterprise.dji.com/mavic-3-enterprise. [Accessed: 7 Feb 2025].

[24] HomeGuide, "How much does a structural engineer cost?" *HomeGuide*, [Online]. Available: https://homeguide.com/costs/structural-engineer-cost. [Accessed: Apr. 20, 2025].