DRONE-BASED AI INFRARED HUMAN DETECTION SYSTEM FOR DISASTER RESPONSE: A PROOF-OF-CONCEPT SIMULATION

PRJ381: Milestone 2 – Project Design

By Group PRJ381-34:
Suné Susan du Raan (578640),
Stephanus Jacobus Mathee (578381),
Simphiwe Nkgau (577176),
and Steven Ayensu (577157)

Table of Contents

Introduction

This document focuses on the design phase of the project "Drone-Based AI Infrared Human Detection System for Disaster Response - A Proof-of-Concept Simulation." The planning phase has been completed, and this stage aims to translate the established plans into a detailed design blueprint.

The project aims to address the urgent societal need for efficient and rapid disaster response through technology. Utilising drone technology, artificial intelligence, and infrared sensing capabilities, the project aspires to improve the speed and accuracy of locating individuals in disaster-stricken areas. The focus of the project is to create a proof-of-concept simulation that mimics the drone's perspective, using pre-recorded video footage and an AI model trained on thermal images of humans.

The objective of this document is to outline in detail the approach for fulfilling the requirements and constraints as identified during the planning phase.

1. Requirements Gathering and Analysis

1.1 Introduction

Purpose

The purpose of the Requirements Gathering and Analysis document section is to provide a comprehensive, structured, and detailed outline of the functional and non-functional requirements of our project: "Drone-Based AI Infrared Human Detection System for Disaster Response - A Proof-of-Concept Simulation." This document serves as a formal agreement between the project stakeholders, including but not limited to the product owner, scrum master, developers, and any future contributors or evaluators, on what the system is expected to achieve.

Scope

The scope of this project is strictly confined to the development of a proof-of-concept software simulation that mimics the core functionalities of a drone system designed for disaster response. The simulation will use pre-recorded video footage to represent a drone's perspective of a disaster zone. The software will apply an AI model trained to detect humans based on their heat signatures in thermal images and will provide the pixel coordinates of the detected individuals. The project will not include the development or integration of an actual drone, infrared camera, GPS system, or flight planning software.

Project Background

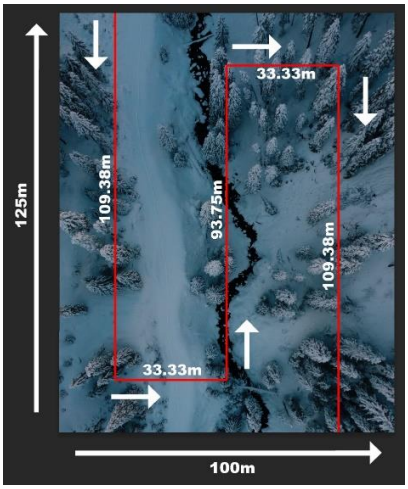
The rapid and efficient response to disasters is a global challenge that necessitates innovative solutions. Traditional methods often require search and rescue teams to operate in hazardous and expansive environments, making the timely detection of individuals in need a daunting task. This project aims to address this issue by leveraging drone technology, artificial intelligence, and infrared sensing capabilities to locate injured or stranded humans in disaster zones.

Assumptions

The project is based on several key assumptions:

- The drone maintains a consistent altitude of 15 meters above ground level.
- The simulated infrared camera has a horizontal field of view of 100 degrees and a vertical field of view of 90 degrees, with a 4:3 aspect ratio.
- The drone navigates over a rectangular region of 12,500 square meters.

- The drone follows an up-and-down flight path, as depicted in the example image below:



- The operational flight area is divided into a grid comprising four rows and three columns.
- The drone takes approximately 126.39 seconds to complete its flight path over the designated area.
- These assumptions have been calculated and made to ensure that the proof-of-concept simulation closely aligns with a practical, real-world scenario.

Intended Audience

This document is intended for the following audiences:

- Product Owner: For decision-making and strategic planning.
- Scrum Master: To manage project workflow and remove impediments.
- Developers: For understanding what is to be developed and tested.
- Quality Assurance Teams: For understanding what needs to be verified.
- Future Contributors: To understand the project requirements and constraints.
- Investors and External Evaluators: To assess the project's viability and potential impact.

Document Conventions

Throughout this document, the following conventions will be used:

Must: Indicates a mandatory requirement.

Should: Indicates a desirable but not mandatory requirement.

May: Indicates an optional requirement.

This document aims to be the definitive source of requirements for the project and will be updated as necessary to reflect any changes or refinements in the project scope or objectives.

1.2 Stakeholders

Overview

The success of the project hinges on the involvement, contribution, and feedback of various stakeholders. Identifying the stakeholders and understanding their roles and responsibilities is crucial for effective requirements gathering, implementation, and eventual deployment of the project.

Identified Stakeholders

Product Owner

Role: The Product Owner is responsible for defining the project vision, prioritizing features, and making decisions about what gets built and in what order.

Responsibilities:

- Backlog management
- Sprint planning
- Stakeholder communication

Interests: Ensuring the project aligns with the original vision and meets the criteria for success.

Scrum Master

Role: The Scrum Master serves as the facilitator for the Agile development team.

Responsibilities:

- Managing Scrum processes
- Resolving impediments
- Facilitating sprint meetings

Interests: Ensuring that the development process runs smoothly and efficiently, and that the team is able to meet its goals within each sprint.

Developers

Role: Developers are responsible for designing and building the proof-of-concept simulation.

Responsibilities:

- Writing code
- Conducting unit tests
- Participating in code reviews

Interests: Developing a robust and efficient simulation that meets the project requirements.

Quality Assurance Team

Role: Responsible for verifying that the developed system meets the defined requirements.

Responsibilities:

- Designing test cases
- Conducting tests
- Reporting bugs and issues

Interests: Ensuring the system is error-free and performs as expected.

Academic Supervisor

Role: Provide academic oversight, guidance, and validation of the project's scientific and technical aspects.

Responsibilities:

- Reviewing methodologies
- Offering technical guidance

- Validating the project's outcomes

Interests: Ensuring the project meets academic standards for research and development.

Potential Investors

Role: Individuals or organizations interested in financially backing the project for future development.

Responsibilities:

- Financial evaluation
- Due diligence

Interests: Determining the project's financial viability and potential return on investment.

End-Users (Disaster Response Teams)

Role: The ultimate beneficiaries of the fully developed and operational product.

Responsibilities:

- Providing user feedback
- Testing the system in simulated or controlled environments

Interests: Having a reliable, efficient, and user-friendly system to aid in disaster response efforts.

Regulatory Authorities

Role: Organizations or governmental bodies that may have jurisdiction over the deployment of drones or AI systems.

Responsibilities:

- Reviewing compliance
- Approving or disapproving the system for operational use

Interests: Ensuring that the system complies with existing laws and regulations concerning drone usage and AI.

Stakeholder Engagement Plan

A stakeholder engagement plan will be developed and maintained by the Scrum Master, detailing the frequency and modes of communication with each stakeholder group. This will include regular updates, sprint reviews, and specialized meetings to address specific concerns or requirements.

By identifying and understanding the roles and interests of each stakeholder, we aim to ensure that the project meets or exceeds the expectations and requirements of all parties involved.

1.3 Project Scope

Overview

The project scope outlines the boundaries, deliverables, and constraints that define the extent of this project. This section provides a detailed description to ensure a shared understanding among all stakeholders of what the project will achieve and what it will not address.

In-Scope

Software Simulation

The primary deliverable is a proof-of-concept software simulation that mimics a drone's capabilities in disaster response scenarios. This simulation will use pre-recorded video footage and apply an AI model trained to detect humans based on thermal imagery.

AI Model

An AI model will be developed and trained to recognize human forms based on their heat signatures in thermal images. The model will be applied to the video footage to identify and locate individuals in need of assistance.

User Interface

A user-friendly interface will be developed to allow users to interact with the simulation. The interface will display the drone's view, provide controls for simulation settings, and show the detected coordinates of individuals.

Grid-based Analysis

The operational area in the simulation will be divided into a grid comprising four rows and three columns, in alignment with Assumption 5. Each grid block will be analysed individually for the presence of humans, and their pixel coordinates will be provided.

Documentation

Comprehensive documentation will be prepared.

Out-of-Scope

Hardware Integration

The project will not involve the procurement, integration, or testing of actual drones, infrared cameras, or GPS systems.

Flight Planning Software

Integration with flight planning software such as DJI GO or DJI Fly is beyond the scope of this project.

Real-world Testing

Due to constraints, the project will not include any real-world testing in actual disaster zones.

GPS Coordinate Mapping

While the simulation will provide pixel coordinates for detected individuals, translating these into real-world GPS coordinates is out-of-scope for this project.

Constraints

Financial Constraints

Limited funding restricts the scope to a software-based proof-of-concept, excluding hardware and real-world testing.

Time Constraints

The project has a fixed timeline, within which the proof-of-concept must be developed, tested, and documented.

Technical Constraints

The AI model's accuracy is dependent on the quality of the training data and computational resources available for model training.

Assumptions

The project relies on the assumptions outlined in the Introduction section, including drone altitude, camera field of view, operational area dimensions, flight path, and grid structure.

1.4 Functional Requirements

The functional requirements of the project are broken down into various subsystems for easier understanding and implementation. Each requirement is tagged with an identifier for traceability.

Data Collection Subsystem

Load Pre-recorded Video (FR-DC-001)

The system must be able to load pre-recorded thermal video footage from a local disk. The footage should mimic the drone's point of view and must be in a 4:3 aspect ratio.

Frame Extraction (FR-DC-002)

The system should automatically extract a single frame for analysis when the simulated drone is in the middle of each block of the grid. This corresponds to 12 frames in total for a full flight path from Block 1 to Block 12.

Image Analysis Subsystem

AI Model Integration (FR-IA-001)

The system must integrate a model trained to detect human heat signatures in thermal images.

Human Detection (FR-IA-002)

Upon processing each frame, the system should identify if a human is present based on their heat signature. The system should utilize the AI model for this purpose.

Coordinate Mapping (FR-IA-003)

When a human is detected, the system should calculate the pixel coordinates of the individual detected. These coordinates should be mapped in relation to the entire operational area (map).

Navigation Simulation Subsystem

Grid Formation (FR-NS-001)

The system should simulate a rectangular operational flight area divided into a grid comprising four rows and three columns. Each block within this grid should have specific dimensions as defined in the project assumptions.

Reporting Subsystem

Coordinate Logging (FR-RP-001)

The system should log the pixel coordinates of detected humans for each block where a human is detected. These logs should be timestamped.

Report Generation (FR-RP-002)

The system must generate a final report summarizing the human detections, complete with the pixel coordinates and corresponding blocks.

User Interface Subsystem

Real-Time Analysis Display (FR-UI-001)

The interface should display the footage analysis, indicating blocks where humans are detected.

Report Access (FR-UI-002)

The interface must provide an option to view and download the image and coordinates.

System Validation and Testing Subsystem

Unit Tests (FR-SV-001)

Each functional module should be accompanied by unit tests to validate that each module performs its defined function correctly.

Integration Tests (FR-SV-002)

After integrating the modules, end-to-end integration tests should be carried out to ensure seamless functionality.

Performance Testing (FR-SV-003)

The system should undergo performance testing to ensure that it can handle the load and perform tasks within acceptable time limits.

1.5 Non-Functional Requirements

The non-functional requirements specify the quality attributes that the project must adhere to. Each non-functional requirement is tagged with a unique identifier for traceability.

Performance Requirements

Load Time (NFR-PR-001)

The pre-recorded thermal video footage must load within 5 seconds after the user initiates the load command from the user interface.

Usability Requirements

User Interface (NFR-US-001)

The user interface should be intuitive and require no more than 10 minutes of training for a person familiar with basic computer operations.

Documentation (NFR-US-002)

The system should come with comprehensive user documentation detailing how to operate the software, interpret the results, and troubleshoot basic issues.

Reliability Requirements

Error Handling (NFR-RL-001)

The system must provide clear error messages for common issues such as file loading errors, AI model failures, or data inconsistencies.

Scalability Requirements

Extensibility (NFR-SA-001)

The system should be designed in a modular fashion to allow for future extensions, such as integration with actual drone hardware or more advanced AI models.

Data Volume (NFR-SA-002)

The system should be capable of handling a video size of up to 2GB without performance degradation.

Availability Requirements

Uptime (NFR-AV-001)

The system should be available for operation at least 98% of the time, excluding scheduled maintenance periods.

Legal and Compliance Requirements

License Compliance (NFR-LC-001)

All software libraries and tools integrated into the system must have licenses that permit their use in academic research projects.

Data Privacy (NFR-LC-002)

The system must comply with data privacy laws and regulations relevant to the area in which it will be deployed or demonstrated.

Environmental Requirements

Power Consumption (NFR-EN-001)

While this is a software simulation, considerations should be made for the power consumption of the eventual hardware. The system should be optimized for low CPU and memory usage to mimic efficient power usage in a real-world drone.

1.6 Data Requirements

In this section, the types of data that project will require, generate, and manage are analysed. This includes input data, output data, and data formats. Each data requirement is tagged with an identifier for traceability.

Input Data

Pre-recorded Thermal Video Footage (DR-ID-001)

Format: The footage must be in a compatible video format, such as MP4, and should have an aspect ratio of 4:3.

Resolution: High-definition (720p or 1080p).

Duration: Should be long enough to mimic the average 2.11-minute flight time over the operational area.

Storage: Should be stored in a local disk and easily accessible by the system.

AI Model for Human Detection (DR-ID-002)

Format: The AI model should be in a format compatible with the human detection library being used, such as a .h5 file for TensorFlow.

Attributes: The model should be capable of identifying heat signatures corresponding to human forms.

Output Data

Extracted Frames (DR-OD-001)

Format: Frames must be extracted as individual image files, preferably in JPEG or PNG format.

Naming Convention: Files should be named in a manner that reflects the block number from which they were extracted.

Human Detection Reports (DR-OD-002)

Format: Reports should be in a text-based format, such as .txt or JSON, and should be easily readable.

Content: Must include the timestamp, block number, and pixel coordinates of detected humans.

Log Files (DR-OD-003)

Format: Text-based (.txt or .log).

Content: Should contain system activities, errors, and other significant events for debugging and auditing purposes.

Metadata

Grid Information (DR-MD-001)

Format: Text-based or JSON.

Content: Must include the dimensions of each block, the number of rows and columns, and the total area of the operational flight zone.

Simulation Parameters (DR-MD-002)

Format: Text-based or JSON.

Content: Must include all the simulation parameters like drone speed, altitude, and the total length of the flight path.

Data Storage

Local Disk (DR-DS-001)

Capacity: Must have enough storage space to store the pre-recorded footage, extracted frames, AI model, and generated reports.

Accessibility: Data must be easily accessible by the system and should adhere to standard file/folder hierarchies for easy navigation.

Temporary Data (DR-DS-002)

Storage: The system should have enough memory to hold temporary data, such as variables and intermediate computations, during the simulation.

1.7 System Constraints

This section outlines the constraints under which the project will be developed and operated. These constraints are critical in shaping the design, development, and deployment of the system.

Hardware Constraints

No Real Drone Hardware (SC-HW-001)

Due to budget limitations, the project will not include the use of actual drones, infrared cameras, or onboard computing systems. All functionalities will be simulated in software.

Local Computing Resources (SC-HW-002)

The system is constrained by the computing resources of the machine where it will be run. It is assumed that a standard laptop or desktop will be used for development and demonstration purposes.

Software Constraints

No Integration with Flight Planning Software (SC-SW-001)

The project will not integrate with any existing flight planning software like DJI GO or DJI Fly. The drone's flight path will be simulated within the system itself.

Operating System (SC-SW-002)

The system is expected to run on Windows 10 or later versions, given that the development team has expertise in this environment.

Data Constraints

Pre-recorded Video Footage (SC-DT-001)

The system will rely solely on pre-recorded thermal video footage to simulate the drone's point of view. Real-time data collection is outside the scope of this project.

Frame Resolution (SC-DT-002)

The system will operate on the assumption that the thermal video footage has a fixed 4:3 aspect ratio, which is common for general-use thermal cameras.

Performance Constraints

Processing Time (SC-PF-001)

Given the need to analyse 12 frames for a complete flight path, the system should be able to process each frame and perform human detection within 10 seconds to simulate real-time operation.

Regulatory Constraints

Data Privacy (SC-RG-001)

Although the system uses pre-recorded thermal video footage, any resemblance to actual persons is coincidental, and the system must adhere to data privacy laws and guidelines.

Environmental Constraints

Simulated Operational Area (SC-EN-001)

The system is designed to simulate a drone flying over a specific rectangular area of 12,500 square meters. Altering the area dimensions would require reconfiguration and recalibration of the system.

Fixed Altitude (SC-EN-002)

The system will simulate the drone flying at a constant altitude of 15 meters above ground level, as per Assumption 1. Variations in altitude are not accounted for in this project.

1.8 Risks and Assumptions

In order to effectively plan and execute this project, it's essential to acknowledge the potential risks and underlying assumptions that have been made during the planning phase. Identifying these elements upfront allows for proactive risk mitigation and ensures that the project is built on a solid foundation.

Risks

Inaccuracy of AI Model (Risk ID: RSK-001)

Description: The AI model may not be completely accurate in detecting human heat signatures, leading to false positives or negatives.

Impact: High

Mitigation: Rigorous training and validation of the AI model using diverse datasets. Consider implementing a fail-safe mechanism for manual review.

Processing Speed (Risk ID: RSK-002)

Description: The system may experience slowdowns, affecting real-time analysis and reporting.

Impact: Medium

Mitigation: Optimize code for performance and conduct performance testing to identify bottlenecks.

Data Integrity (Risk ID: RSK-003)

Description: The pre-recorded thermal video footage may be corrupted or may not accurately represent a disaster scenario.

Impact: Medium

Mitigation: Ensure the quality and integrity of the pre-recorded videos used for simulation.

User Interface Usability (Risk ID: RSK-004)

Description: The user interface may not be intuitive, affecting the user's ability to effectively interact with the system.

Impact: Low

Mitigation: User testing and feedback loops to iteratively improve UI/UX.

Resource Limitation (Risk ID: RSK-005)

Description: Limited financial and time resources may affect the scope and quality of the project.

Impact: High

Mitigation: Prioritize critical features and functionalities, and be prepared to scale down less critical aspects if needed.

Assumptions

Consistent Altitude (Assumption ID: ASM-001)

The drone is presumed to maintain a consistent altitude of 15 meters above the ground level. This is crucial for the AI model's accuracy in detecting human heat signatures.

Camera Specifications (Assumption ID: ASM-002)

The thermal camera used for simulation has a horizontal field of view of 100 degrees and a vertical field of view of 90 degrees. The footage is in a 4:3 aspect ratio.

Operational Area and Navigation (Assumption ID: ASM-003)

The drone will consistently navigate over a rectangular region with an area of 12500 square meters, following a predetermined flight path facilitated by established flight planning software.

Zig-Zag Flight Path (Assumption ID: ASM-004)

The drone is presumed to consistently follow an up-and-down zig-zag flight path, crucial for frame extraction and human detection.

Grid Structure (Assumption ID: ASM-005)

The operational flight area will be subdivided into a grid with four rows and three columns. The drone initiates its flight path from Block 1 and continues sequentially to Block 12.

Flight Duration (Assumption ID: ASM-006)

It is calculated that the drone will take approximately 126.39 seconds to record footage of the entire area, flying at a speed of 3 meters per second and covering a total path length of 379.16 meters.

1.8 Requirement Traceability Matrix

The Requirement Traceability Matrix (RTM) is a tool that ensures each requirement is linked back to its source. This provides a way to check that all requirements are met in the development

process and allows for easier identification of the impact of potential changes. Below is the Requirement Traceability Matrix for the project:

Requirement ID	Requirement Description	Source	Subsystem	Priority
FR-DC-001	Load pre-recorded thermal video footage from a local disk.	Project Description	Data Collection	High
FR-DC-002	Extract a single frame for analysis at the middle of each block.	Assumption 5	Data Collection	High
FR-IA-001	Integrate a machine learning model for human heat signature detection.	Project Description	Image Analysis	High
FR-IA-002	Identify human presence based on heat signature in each frame.	Project Description	Image Analysis	High
FR-IA-003	Calculate the pixel coordinates of detected human.	Assumption 6	Image Analysis	High
FR-NS-001	Simulate a rectangular operational flight area divided into a grid.	Assumption 3	Navigation Simulation	Medium
FR-NS-002	Simulate the drone's zig-zag flight path.	Assumption 4	Navigation Simulation	Medium
FR-RP-001	Log the pixel coordinates of detected humans for each block.	Project Description	Reporting	Medium
FR-RP-002	Generate a final summary	Project Description	Reporting	Medium

	report of human detections.			
FR-UI-001	Provide functionality for video playback.	Project Description	User Interface	Low
FR-UI-002	Display ongoing analysis in real-time.	Project Description	User Interface	Low
FR-UI-003	Allow manual navigation through the frames.	Project Description	User Interface	Low
FR-UI-004	Provide an option to view and download generated reports.	Project Description	User Interface	Low
FR-SV-001	Include unit tests for each functional module.	Project Description	System Validation	High
FR-SV-002	Conduct end-to-end integration tests.	Project Description	System Validation	High
FR-SV-003	Perform system performance testing.	Project Description	System Validation	Medium

Legend

- Requirement ID: The unique identifier for each requirement.
- Requirement Description: A brief description of the requirement.
- Source: Where the requirement originated from, either from the project description or project assumptions.
- Subsystem: The subsystem that will implement this requirement.
- Priority: The level of importance of the requirement (High, Medium, Low).

1.9 Appendix

This section of the Requirements Gathering and Analysis document section serves as an ancillary space to provide supplementary information, calculations, and clarifications that contribute to the understanding and implementation of the project.

A.1 Grid Dimensions Calculations

A.1.1 Area of Operational Flight Zone

As per Assumption 3, the operational flight area, referred to as the "map," is a rectangular region with an area of 12,500 square meters. This corresponds to dimensions of 125 meters in length (vertical) and 100 meters in width (horizontal).

A.1.2 Grid Blocks

The grid is divided into four rows and three columns, as depicted in Assumption 5. Each block's dimensions are as follows:

Horizontal length: 33.33 meters

Vertical length: 31.25 meters

Total area of one block: 1041.56 square meters

A.2 Flight Path Calculations

A.2.1 Total Flight Path Length

The total length of the flight path is calculated to be 379.16 meters as per Assumption 6.

A.2.2 Time Required for Complete Flight

Given a drone speed of 3 meters per second and a total flight path of 379.16 meters, the time required to complete the flight is approximately 126.39 seconds or 2.11 minutes.

A.3 Field of View

As per Assumption 2, the thermal camera has a horizontal field of view of 100 degrees and a vertical field of view of 90 degrees. The footage will be taken at an aspect ratio of 4:3.

A.4 Assumptions and Constraints Mapping

A mapping of how each assumption directly correlates with the system constraints and functional requirements:

- Assumption 1: Correlates with System Constraint SC-EN-002
- Assumption 2: Correlates with System Constraint SC-DT-002
- Assumption 3: Correlates with Functional Requirement FR-NS-001 and System Constraint SC-EN-001
- Assumption 4: Correlates with Functional Requirement FR-NS-002
- Assumption 5: Correlates with Functional Requirement FR-DC-002
- Assumption 6: Correlates with Functional Requirement FR-IA-003

A.5 Glossary of Terms

- Thermal Footage: Video footage captured through an infrared camera that shows heat signatures.
- AI Model: Artificial Intelligence model trained to recognize human heat signatures.
- Pixel Coordinates: X and Y coordinates in a digital image.
- Operational Flight Area: The area over which the drone is programmed to fly.
- Grid Block: A smaller, distinct section within the operational flight area.

2. System Design

This section serves as a comprehensive blueprint that outlines the technical architecture, components, and data flow of the project software. This section explains the system's inner

workings, ensuring that it aligns with the needs of all stakeholders, including the product owner, scrum master, developers, and potential end-users such as disaster response teams.

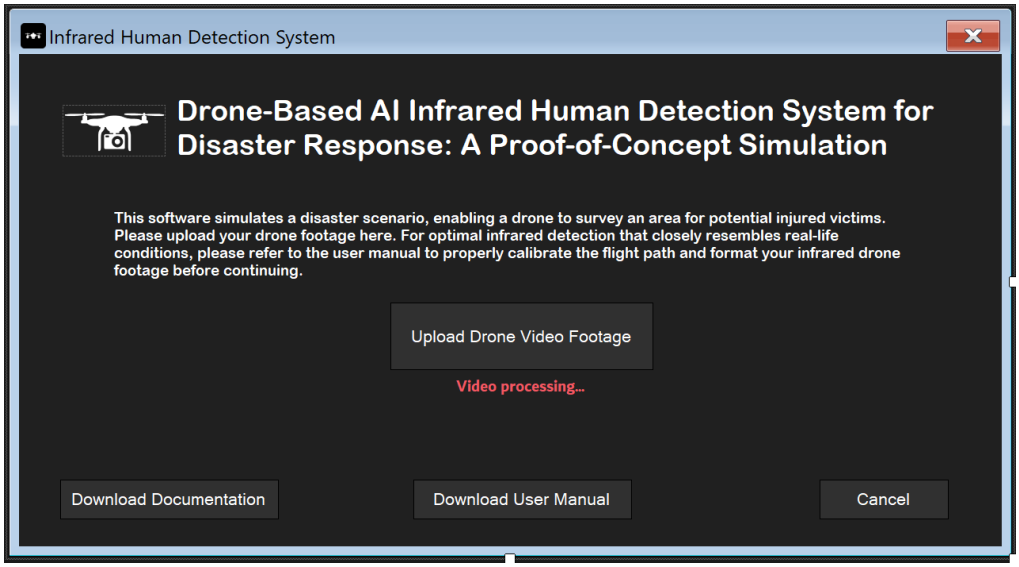
2.1 System Architecture

The system architecture is modular and is divided into three primary packages, each serving a specific function:

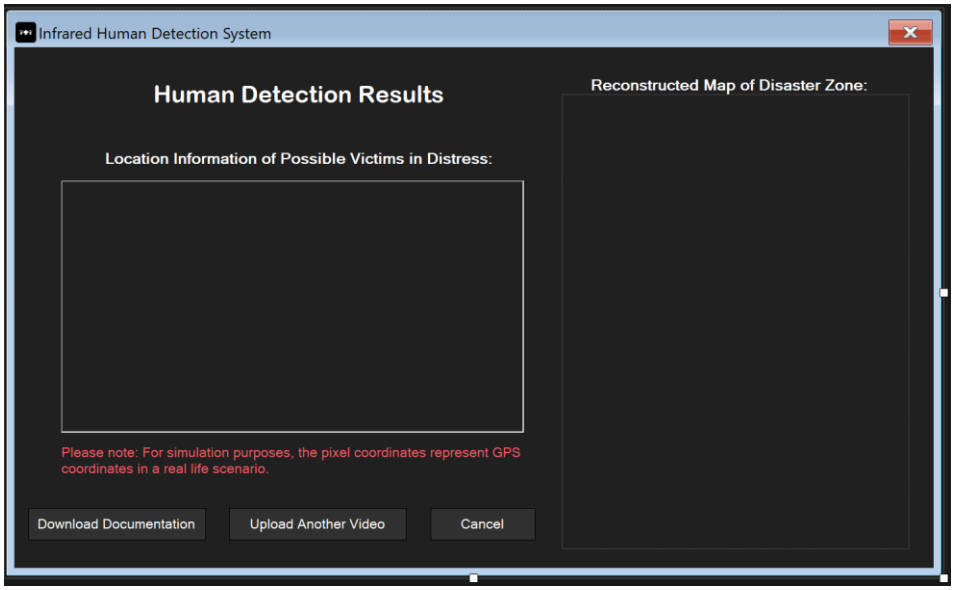
User Interface (UI) Package

This package is the first point of interaction between the user and the system. It is designed to be intuitive and user-friendly. The UI package includes several forms:

Main Form: Allows the user to initiate the simulation:



Results Display Form: Showcases the coordinates of detected humans:



Core Logic Package

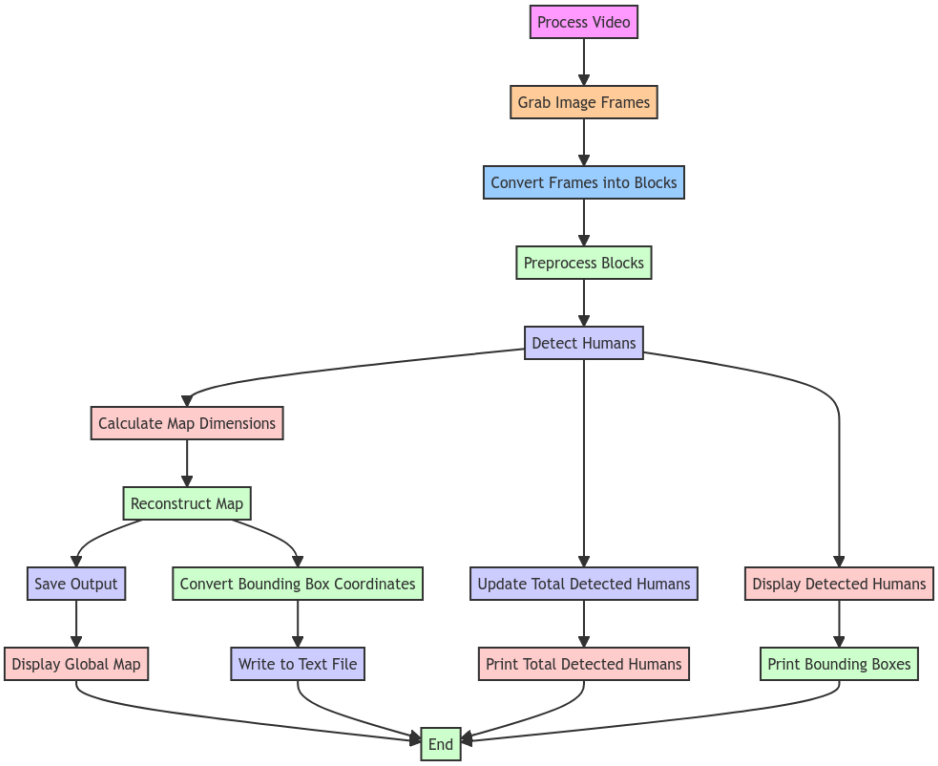
This is the computational engine of the system. It houses several critical components:

AI Model for Human Detection: Utilizes an algorithm that identifies humans based on their heat signatures.

Data Processor: Takes the output from the AI model and processes it to calculate the pixel coordinates of detected humans.

Coordinate Mapper: Maps the pixel coordinates to the simulated environment, which could be translated to real-world GPS coordinates in a fully developed system.

Core Logic Diagram:



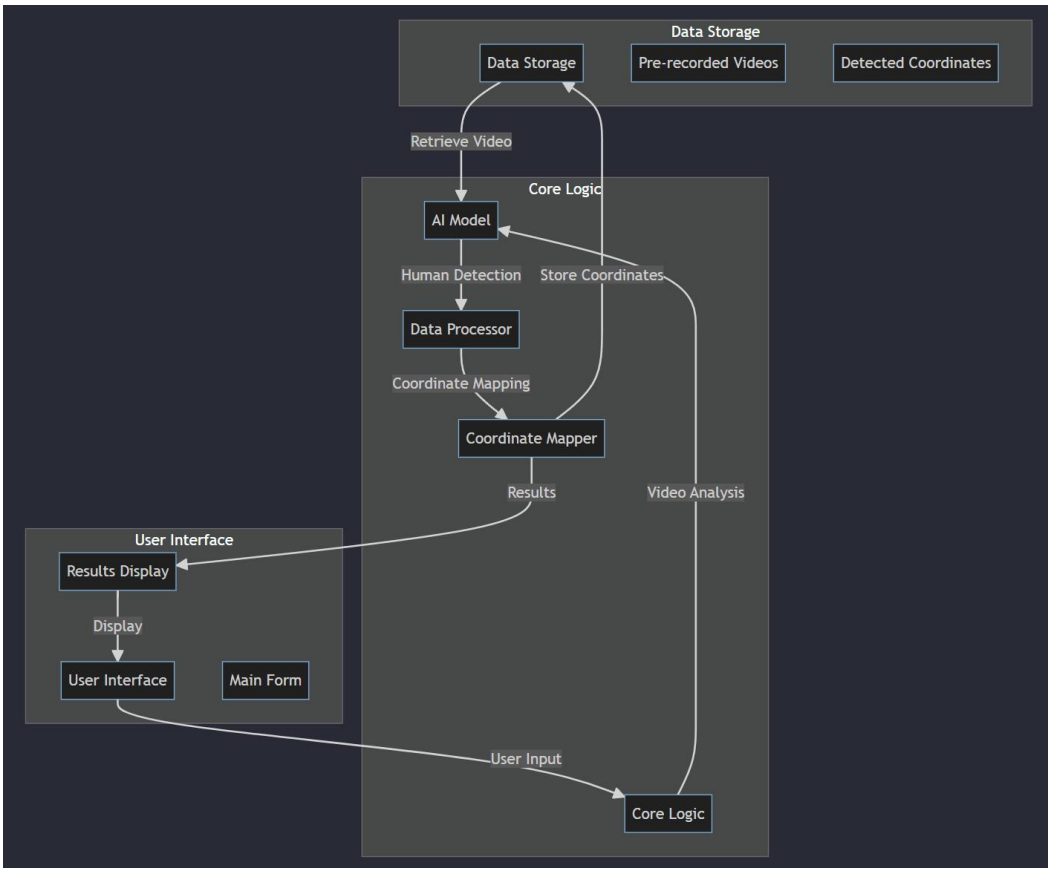
Data Storage Package

This package is responsible for all data persistence operations. It stores:

Pre-recorded Videos: Simulated drone footage of a disaster zone.

Detected Coordinates: The coordinates of detected humans for future reference or analysis.

System Architecture Diagram:



2.2 Data Flow

The data flow is designed to ensure seamless operation:

1. User Input Phase: The user interacts with the Main Form where a video is provided for analysis.
2. Video Processing Phase: Once the simulation starts, the AI model fetches the video from the Data Storage package. Frame-by-frame analysis is performed to identify potential human subjects based on their heat signatures.
3. Human Detection Phase: The AI model scans each frame to identify humans. This data is then passed to the Data Processor for further analysis.
4. Coordinate Mapping Phase: The Data Processor calculates the pixel coordinates of the detected humans and maps them to the simulated environment. This is a crucial step as it translates the AI model's findings into actionable data.
5. Results Display and Storage Phase: The mapped coordinates are displayed on the Results Display Form. They are also stored in the Data Storage package for future reference, analysis, or to be used in reports.

2.3 User Interface Design

The User Interface is designed with the end-user in mind:

1. Main Form: This is the dashboard where the user can initiate the simulation.
2. Results Display Form: This form is the culmination of the simulation. It not only displays the coordinates of detected humans but also provides options to export the data.

2.4 Assumptions and Constraints

The system design is rooted in several assumptions and constraints, as outlined in the project description. These include the drone's altitude, the camera's field of view, and the area of operation. These assumptions are integrated into the AI model to ensure its accuracy and reliability. For example, the AI model is trained to detect humans based on the assumption that the drone is flying at a consistent altitude of 15 meters.

3. Development and Coding

Please find all the application code in the folder named “Code”. Note that as of the end of Milestone 2, the application is not yet complete and is subject to change. This code merely serves as a demonstration of the software development process thus far.

4. Testing

Please find the HTML version of the Jupyter notebook in which testing was conducted, along with results, evaluation metrics, and performance visualisation here:

Infrared Human Detection System Tes

5. Deployment

5.1 Test Plan

Introduction

The purpose of this Test Plan is to outline the strategy, objectives, resources, schedule, and scope of the testing activities for the "Drone-Based AI Infrared Human Detection System for Disaster Response: A Proof-of-Concept Simulation" project. This document serves as a guideline for all members involved in the testing process, including the product owner, scrum master, developers, and quality assurance teams.

The project aims to develop a proof-of-concept software system that mimics the core functionalities of a drone equipped with an infrared camera for human detection in disaster zones. While the ultimate goal is to create a drone with real-world capabilities, the focus of this academic endeavour is on software simulation due to financial and resource constraints.

The software system is developed to identify human heat signatures from pre-recorded videos simulating a drone's perspective of a disaster-stricken area. Given the critical nature of the application in potential life-saving scenarios, it is imperative to conduct comprehensive testing to ensure the system's accuracy, reliability, and efficiency.

This Test Plan will detail the different types of testing to be performed, the test environment setup, the test data requirements, and the schedule for each testing phase. The plan is designed to be agile, allowing for flexibility and adaptability, in line with the Agile methodology adopted for project management.

In summary, this Test Plan aims to:

1. Validate that the system accurately identifies human heat signatures in various simulated scenarios.
2. Ensure the software is reliable and can efficiently process video data.

3. Confirm that the system can provide the coordinates of detected humans for potential use in real-world rescue operations.
4. Assess the usability of the software interface.
5. Facilitate the identification of defects, if any, to be addressed in subsequent development sprints.

Test Objectives

The primary objectives of testing for the project are multi-faceted and aligned with both the technical requirements and the overarching goals of the project. Below are the key objectives we aim to achieve through a robust testing strategy:

1. Functional Correctness:

Algorithm Validation: To ensure that the machine learning algorithm employed for detecting human heat signatures is accurate and reliable. This includes testing the model's performance metrics such as precision, recall, and F1-score.

Data Handling: To validate that the system can efficiently load, process, and display pre-recorded videos and thermal images as intended. This will include tests for file format compatibility, loading speed, and data integrity.

2. Performance Efficiency:

Processing Speed: To evaluate how quickly the system can analyze a video feed and identify human heat signatures. This is crucial in disaster response scenarios where time is of the essence.

Resource Utilization: To assess how efficiently the system uses computational resources. This includes monitoring CPU and memory usage during the execution of the algorithm.

3. Usability:

User Interface: To test the intuitiveness, responsiveness, and ease of use of the system's interface. This will involve usability tests that gauge how straightforward it is for a user to navigate through the system and interpret results.

Error Messages and Help: To verify that the system provides meaningful error messages and help guides. This is to ensure that users can troubleshoot issues or understand functionalities without external assistance.

4. Reliability:

Error Handling: To ensure that the system can gracefully handle unexpected situations such as missing files, incompatible formats, or corrupted data.

System Stability: To assess if the system can run for extended periods without crashing or causing memory leaks, particularly when processing large video files.

5. Scalability and Extensibility:

Modular Testing: To validate that individual components of the system can function independently and can be easily integrated with other components. This is important for future extensions of the project.

Load Testing: To evaluate the system's performance under varying loads, especially concerning the size and duration of the video feeds it needs to analyze.

6. Compliance and Security:

Code Standards: To ensure that the codebase adheres to industry standards and best practices for maintainability and readability.

Data Security: Although not a primary focus for this proof-of-concept, we aim to ensure that any data handled by the system is processed and stored securely, complying with relevant data protection laws.

By achieving these objectives, the testing phase aims to confirm that the proof-of-concept system is not just technically sound but also user-friendly, reliable, and ready for potential future development and real-world application. This comprehensive approach ensures that we maintain the highest standards of quality, thereby fulfilling the project's vision and objectives.

Test Resources

The testing phase for the project will require a coordinated effort involving multiple resources. This section outlines the various resources needed to accomplish the test objectives effectively and efficiently.

1. Human Resources:

Quality Assurance (QA) Team: All team members will be responsible for designing, executing, and evaluating test cases. They will also be in charge of monitoring test results, logging defects, and ensuring that all test objectives are met.

Developers: The development team will work closely with the QA team to resolve any issues or defects found during testing. They will also assist in setting up the test environment and may contribute to writing unit tests.

Scrum Master: Responsible for overseeing the entire testing process within the Agile framework, ensuring that testing activities align with project milestones and timelines.

Product Owner: Involved in defining acceptance criteria for test cases and ensuring that the test objectives align with the project goals and customer requirements.

2. Software Tools:

Testing Frameworks: Tools such as PyTest for Python will be used for unit testing and integration testing.

Performance Testing Tools: Software to evaluate system performance under varying loads.

Version Control: Git will be used for code versioning, enabling seamless collaboration between the development and QA teams.

Defect Tracking: Tools like Jira or Bugzilla to manage, track, and document issues and defects discovered during testing.

Continuous Integration Tools: GitLab CI/CD to automate the testing process as new code is integrated.

3. Hardware:

Test Environment: A dedicated test environment that mimics the production environment as closely as possible. This includes the same operating system, database, and network configurations.

Computational Resources: Adequate CPU, memory, and storage resources to perform intensive tasks, such as load testing and performance benchmarking.

4. Test Data:

Sample Videos and Images: Pre-recorded video and image files that simulate various disaster scenarios. These will be used to test the system's accuracy in detecting human heat signatures.

Synthetic Data: Artificially generated data to test edge cases that might not be covered by real-world data.

5. Documentation:

Test Plan: This comprehensive document outlines the testing strategy, objectives, and schedule.

Test Cases: Detailed descriptions of individual test cases, including their expected outcomes and acceptance criteria.

Test Logs: Records of test executions, issues discovered, and any deviations from the planned test activities.

Defect Reports: Detailed reports of defects discovered during testing, including steps to reproduce the issues and their severity levels.

6. Timeline:

Testing Schedule: A well-defined timeline that aligns with the project's Agile sprints, accounting for preparation, execution, evaluation, and re-testing phases.

By allocating and utilizing these resources effectively, we aim to ensure a thorough, well-coordinated testing effort that aligns with the project's objectives and constraints. This will enable us to deliver a robust, reliable, and efficient system capable of meeting both the technical and user-experience goals set forth in the project scope.

Test Environment Setup

The test environment is a critical component for the effective execution of our testing strategy for the project. Here's how we plan to set up the test environment:

1. Software Configuration:

Operating System: The test environment will be set up on Microsoft Windows.

Python Environment: A separate Python virtual environment will be created to isolate the project dependencies. All the libraries mentioned in the "Install Libraries" section of the Jupyter notebook, such as OpenCV and Keras, will be installed.

Jupyter Notebook: The notebook will be run on a Jupyter server, replicating the development environment.

2. Hardware Requirements:

Computing Power: A machine with adequate CPU and memory resources will be set up. The specifics will depend on the computational requirements of the human detection model and video processing tasks.

Storage: Sufficient disk space to store test videos, logs, and other test data.

3. Test Data:

Video and Image Files: Pre-recorded video files and thermal images will be used to simulate a drone's perspective of a disaster zone. These files will be prepared based on the assumptions outlined in the documentation, including drone altitude, field of view, and flight path.

Ground Truth Data: Data files containing "ground truth" information will be used for comparing the system's outputs, aligning with the metrics such as precision, recall, and F1 score specified in the notebook.

4. Network Configuration:

Local Server: The Jupyter notebook will be hosted on a local server, ensuring that all team members can access it for testing purposes.

Firewall & Security: Necessary firewall rules will be set up to ensure that the test environment is secure but accessible to the testing team.

5. Monitoring Tools:

Performance Monitoring: Tools will be set up to monitor system performance metrics like CPU and memory usage during test execution.

Logging: Proper logging mechanisms will be set up to capture all events, errors, and outputs during test execution. These logs will be essential for debugging and validating test cases.

6. Backup and Recovery:

Data Backup: All test data, configurations, and databases will be backed up to prevent any loss during testing.

Recovery Plan: A recovery plan will be put in place to restore the test environment in case of any failures or crashes.

By meticulously setting up the test environment to mimic the production setting, we aim to ensure that the test results are as accurate and reliable as possible, thereby facilitating a smooth transition from development to deployment.

Test Schedule

This schedule is structured to fit within the project's Agile development framework and is designed to be rigorous yet flexible to adapt to any changes or challenges that may arise:

August 26 - Test Planning and Environment Setup:

Morning Session (9:00 AM - 12:00 PM): Finalization of the Test Plan document, ensuring that all test objectives, resources, and scope are clearly defined.

Afternoon Session (1:00 PM - 5:00 PM): Setup of the test environment, including software installation, hardware configuration, and network setup.

August 27 - Functional Testing:

Morning Session (9:00 AM - 12:00 PM): Execution of functional test cases focusing on data handling capabilities, including video and image loading, processing, and display.

Afternoon Session (1:00 PM - 5:00 PM): Execution of functional test cases that validate the human heat signature detection functionality.

August 28 - Performance and Reliability Testing:

Morning Session (9:00 AM - 12:00 PM): Conducting performance tests to assess system speed and resource utilization.

Afternoon Session (1:00 PM - 5:00 PM): Conducting reliability tests to evaluate system stability and error handling mechanisms.

August 29 - Usability and Security Testing:

Morning Session (9:00 AM - 12:00 PM): Usability tests to assess the user interface's effectiveness, responsiveness, and ease of navigation.

Afternoon Session (1:00 PM - 5:00 PM): Security tests to ensure data protection and compliance with relevant security standards.

August 30 - Review and Finalization:

Morning Session (9:00 AM - 12:00 PM): Review of all test logs, defect reports, and other documentation to ensure that all test objectives have been met.

Afternoon Session (1:00 PM - 5:00 PM): Finalization of the Test Closure Report, detailing the test activities, results, and any recommendations for future development cycles.

By adhering to this test schedule, we aim to conduct a thorough, well-coordinated testing process that aligns with the project's objectives and constraints.

Features to be Tested

This section outlines the specific features of the project that will be subjected to testing. The test cases have been designed to cover all essential functionalities and components of the system.

1. Script Execution

The system's primary script, Human_Detection_Script.py, will be tested to ensure that it executes without errors and produces the expected output when invoked programmatically.

2. Dependency Management

The installation and proper functioning of all required libraries and dependencies, such as OpenCV and Keras, will be verified. The system should automatically flag any missing dependencies and guide the user through the installation process.

3. Video File Handling

The system must correctly handle the input of video file paths, check for the existence of these files, and calculate their size. Appropriate error messages should be displayed for invalid or missing files.

4. Video Processing

The system's capability to process videos and match the duration of the output video with the input will be tested. Any deviations must fall within acceptable limits as defined in the system requirements.

5. Time-Stamp Calculation

The function responsible for calculating time stamps in the system will be tested for accuracy. This function should correctly compute time stamps based on variables like total time and speed.

6. Human Detection Algorithm

The core feature, the human detection algorithm, will be rigorously tested using a variety of pre-recorded videos under different conditions. The algorithm must accurately identify human figures based on thermal imaging data.

7. Proximity Levels

The system should correctly identify and categorize detected humans based on their proximity to the drone. Different alert levels should be triggered based on these proximity levels.

8. Coordinate Mapping and Location Data

The system's ability to map detected coordinates to real-world locations will be verified. This feature should seamlessly integrate with external JSON files containing mapping data.

9. Error Handling and Logging

All forms of error handling will be tested, including file errors, processing errors, and algorithmic errors. The system should log these errors and provide intelligible error messages to the user.

10. Output Verification

The system must produce outputs in a format compliant with the specifications. This includes the visual display of processed videos, logging of human detection instances, and any alert mechanisms.

11. User Interface

Although primarily a back-end system, any user interface components, such as command-line options or configuration files, will be tested for usability and error resilience.

Features Not to be Tested

This section delineates the components and features of the project that are explicitly out of scope for this round of testing. The decision to exclude these features from testing is based on various factors such as current project priorities, resource constraints, financial constraints, and the developmental stage of certain components.

1. C# Windows Form User Interface

The user interface for the system, designed as a C# Windows Form, is beyond the scope of this testing phase. Although it serves as an important component for user interaction, it is very simplistic and not the main focus of this project.

2. Hardware Integration

As this project is a proof-of-concept simulation, it does not involve the actual drone hardware or infrared camera equipment. Therefore, tests related to hardware-software integration, real-time data capture, and onboard computing will not be conducted.

3. Network Communication

The system is designed to function offline for this version, and as such, features related to network communication, data synchronization, and remote control of drones will not be tested.

4. Battery Life Optimization

Since the project is focused on software simulation, elements like battery life optimization for the drone are not applicable and will not be subjected to testing.

5. Real-world GPS Coordinate Mapping

Although the system simulates coordinate mapping, the testing for real-world GPS coordinate integration and location accuracy is not within the scope of this test plan.

6. Multi-threading and Concurrency

The current version of the software is designed to operate as a single-threaded application. Therefore, aspects related to multi-threading and concurrent processing are not part of this test cycle.

7. Third-party Software and Services

Any dependencies or services not directly part of this system, such as database servers or cloud storage solutions, will not be evaluated during this testing phase.

8. Long-term Performance and Scalability

Tests focused on the long-term performance, stress testing, and scalability of the system will not be performed at this stage.

Test Strategy

The test strategy for the project outlines the overall approach and methodologies that will be employed to ensure comprehensive and effective testing. This strategy is designed to align with the project's objectives, constraints, and timelines.

1. Test Levels

Unit Testing: Each individual module, such as the video processing and human detection algorithms, will undergo unit tests to ensure they function as intended in isolation.

Integration Testing: Post unit testing, integrated modules will be tested to validate their interactions and dependencies, particularly focusing on data flow and error handling.

System Testing: The entire software system will be tested in an environment that closely mimics the production setting. This phase will assess the system's overall functionality and performance.

2. Test Types

Functional Testing: The primary focus will be on functional tests to verify that each feature behaves according to the defined specifications and requirements.

Performance Testing: Key performance indicators such as processing speed, algorithm accuracy, and resource utilization will be measured.

Usability Testing: Though the primary user interface (C# Windows Form) is out of scope, any command-line interfaces or configuration files will be tested for user-friendliness and error resilience.

Boundary Testing: Edge cases, especially involving video and data inputs, will be rigorously tested to ensure system robustness.

3. Test Automation and Tools

Automation will be employed wherever possible to speed up the testing process and improve efficiency. Python-based automation frameworks like pytest will be used for this purpose.

For performance testing, specialized tools such as JMeter may be employed to simulate various scenarios.

4. Test Environment

A controlled test environment will be set up that mimics the production environment as closely as possible. This includes the same operating system, Python version, and library dependencies.

A repository of pre-recorded videos and thermal images will be maintained for testing purposes.

5. Test Data

Test data will consist of both synthetic pre-recorded videos and thermal images. These will be used to validate the accuracy and reliability of the human detection algorithm.

6. Test Schedule

Given the Agile nature of the project, testing will be conducted in sprints, aligned with the development cycles. Each sprint will end with a dedicated testing phase to validate the features developed during that sprint.

7. Risk Management

Potential risks, such as delays in development, lack of resources, or discovery of critical bugs, will be identified upfront. Contingency plans will be in place to mitigate these risks effectively.

8. Reporting and Documentation

Detailed test reports will be generated at the end of each testing phase, summarizing the test results, discovered defects, and any deviations from the expected outcomes.

These reports will be reviewed in sprint retrospectives to inform future development and testing efforts.

Entry and Exit Criteria

This section outlines the criteria that must be met for the test activities to commence (Entry Criteria) and to be considered complete (Exit Criteria). These criteria serve as guidelines to ensure that the testing process is systematic, effective, and aligned with the project objectives.

Entry Criteria

Test Plan Approval: The comprehensive test plan must be reviewed and approved by all relevant stakeholders, including the Product Owner, Scrum Master, and project supervisor.

Environment Setup: A dedicated test environment, mimicking the production settings, must be fully configured and operational. This includes the installation of all required software, libraries, and dependencies.

Availability of Test Data: A well-defined set of test data, including pre-recorded videos and thermal images, must be available and accessible to the testing team.

Code Readiness: The features slated for testing must be fully developed, code-reviewed, and merged into the main codebase. Partially completed features will not be accepted into the testing phase.

Documentation: All features must be accompanied by adequate documentation, outlining their expected behaviour, constraints, and known limitations.

Resource Allocation: The testing team must have sufficient resources, both human and computational, to carry out the planned tests within the stipulated timeline.

Exit Criteria

Test Case Execution: All planned test cases must be executed. Skipped or incomplete test cases will keep the testing phase open.

Defect Resolution: All critical and high-severity defects must be resolved and retested successfully. Low-severity defects should be documented for future resolution.

Performance Benchmarks: The system must meet or exceed all defined performance benchmarks, including processing speed and algorithmic accuracy.

Test Report Completion: Comprehensive test reports, summarizing the test results, defects, and any deviations from the expected outcomes, must be completed and reviewed.

Stakeholder Approval: The test results and reports must be reviewed and approved by key stakeholders, ensuring that the system meets the established quality standards.

Documentation Update: All test-related documentation, including defect reports and performance metrics, must be updated and archived for future reference.

Post-Testing Review: A post-testing review meeting must be conducted to discuss the lessons learned, challenges faced, and potential areas for improvement in future testing cycles.

Test Deliverables

This section outlines the various artifacts, documents, and other outputs that will be produced and maintained throughout the testing process for the project. These deliverables serve as a record of the testing activities and provide valuable insights for stakeholders and future development cycles.

1. Test Plan Document

A comprehensive Test Plan document detailing the testing objectives, scope, strategy, environment, schedule, and resource allocation will be produced and maintained. This document serves as the blueprint for all testing activities.

2. Test Cases and Scripts

Detailed test cases, along with automated test scripts where applicable, will be created for each feature to be tested. These will be stored in a version-controlled repository accessible to all team members.

3. Test Data Sets

A curated collection of test data, including pre-recorded videos and thermal images, will be prepared and stored securely. This data set will be used to validate the system's functionality and performance.

4. Traceability Matrix

A traceability matrix will be developed to map each test case to its corresponding requirement or feature. This ensures that all requirements are covered by the test cases and facilitates impact analysis in case of changes.

5. Defect Reports

All defects discovered during testing will be documented in a standardized defect report format. Each report will include details such as defect severity, steps to reproduce, expected and actual results, and any relevant screenshots.

6. Test Execution Logs

Logs capturing the details of test execution, including test cases executed, the sequence of execution, and the time taken, will be generated automatically by the test automation framework.

7. Test Summary Report

A comprehensive test summary report will be produced at the end of each testing phase. This report will include an overview of the testing activities, metrics, defects discovered and resolved, deviations from the plan, and lessons learned.

8. Performance Metrics

Detailed performance metrics, such as algorithmic accuracy and processing speed, will be captured and documented. This data is crucial for evaluating whether the system meets its performance requirements.

9. User Documentation

Though not a direct output of the testing phase, updated user documentation, including any identified limitations or caveats discovered during testing, will be produced.

10. Archival Package

An archival package containing all test deliverables, including test cases, scripts, data sets, reports, and documentation, will be prepared for long-term storage and future reference.

Risks and Contingencies

This section identifies the potential risks that could adversely impact the testing phase of the project. Alongside each identified risk, appropriate contingency plans are outlined to mitigate the impact and ensure the smooth execution of the testing cycle.

1. Resource Constraints

Risk: Due to limited manpower and computational resources, the testing phase may experience delays.

Contingency: Consider overtime work schedules if required.

2. Software Dependencies

Risk: Updates or changes in external libraries and frameworks could introduce breaking changes or incompatibilities.

Contingency: Maintain a stable, version-controlled environment for testing and keep backup copies of all required dependencies.

3. Data Limitations

Risk: Inadequate or faulty test data may compromise the quality of the testing.

Contingency: Maintain a repository of quality-assured test data, including video files and thermal images, which can be used for different testing scenarios.

4. Algorithmic Complexity

Risk: The algorithm's complexity may result in excessively long processing times during testing.

Contingency: Conduct preliminary performance testing to gauge the algorithm's efficiency and consider using reduced data sets for initial test cycles.

5. Unidentified Bugs and Errors

Risk: New bugs or errors could emerge during the testing phase, requiring unplanned debugging and fixing.

Contingency: Allocate buffer time within the testing schedule specifically for debugging and issue resolution.

6. Environmental Issues

Risk: Unforeseen environmental issues, such as power outages or system crashes, could disrupt the testing process.

Contingency: Implement regular data backup procedures and establish protocols for resuming testing after interruptions.

7. Test Case Inadequacy

Risk: The current set of test cases may not cover all possible scenarios, leading to gaps in the testing.

Contingency: Conduct an interim review halfway through the testing phase to identify any gaps in test coverage and update the test cases accordingly.

8. Non-Cooperation of Stakeholders

Risk: Lack of timely inputs or approvals from stakeholders could stall the testing process.

Contingency: Maintain open channels of communication with all stakeholders and establish predefined timelines for feedback and approvals.

9. Documentation Delays

Risk: Delays in updating test documentation could affect the overall testing schedule.

Contingency: Assign dedicated resources for real-time documentation updates and consider using automated tools for test logging and reporting.

10. Compliance and Security Concerns

Risk: The project may face compliance issues related to data privacy and information security.

Contingency: Consult with legal and compliance teams in advance to ensure that all testing activities adhere to relevant laws and regulations.

Test Data Requirements

This section outlines the specific data requirements needed for comprehensive testing of the project. The quality, variety, and relevance of test data are crucial factors in ensuring that the system is robust, accurate, and ready for deployment.

1. Video Files for Human Detection

A collection of pre-recorded videos simulating various disaster scenarios, such as earthquakes, floods, and fires, is required. These videos should contain human figures in different states (standing, lying down, moving) and at varying distances from the camera to test the accuracy of the human detection algorithm.

2. Thermal Image Data

A set of thermal images in different formats (JPEG, PNG, TIFF) is required to test the system's compatibility with various file types. These images should contain human heat signatures under different conditions like daylight and low light.

3. Artificially Generated Data

To test the system's robustness against false positives, a series of artificially generated videos and images that do not contain human figures will be used. These may include animals, vehicles, and other objects that could potentially be mistaken for humans.

4. Location Mapping Data

A set of JSON files containing real-world mapping data will be needed to test the system's ability to convert detected coordinates into actual geographic locations. These files should be in a format compatible with the system's location mapping function.

5. Error and Exception Data

Files that are corrupted, incomplete, or otherwise problematic will be used to test the system's error-handling capabilities. This includes video files with missing metadata, thermal images with skewed colour mappings, and JSON files with incorrect formatting.

6. Large Data Sets

To test the system's performance and scalability, large video files exceeding 1GB in size and thermal images with high resolutions will be required. These large files will help in assessing whether the system can handle big data without crashing or slowing down significantly.

7. Timestamp and Speed Variables

A series of numerical values will be needed to test the time-stamp calculation function. This includes a range of total time values (in seconds) and speed values (in meters per second) to ensure the function's accuracy across various scenarios.

8. Output Format Templates

Pre-defined templates for expected output formats, such as text logs, visual alerts, and coordinate mapping files, will be required. These templates will serve as the baseline for comparing the actual outputs generated during testing.

9. Configuration Files

A variety of configuration files with different settings will be needed to test how the system behaves under various configurations. This includes settings related to video processing speed, detection sensitivity, and location mapping accuracy.

Individual Test Case Details

This section elaborates on the individual test cases that will be executed to validate the various features of the "Drone-Based AI Infrared Human Detection System for Disaster Response: A Proof-of-Concept Simulation" project. Each test case is designed to verify specific functionalities and requirements, ensuring comprehensive coverage of the system's features.

Test Case 1: Verify Script Execution

Objective: To ensure that the main script (Human_Detection_Script.py) runs successfully without errors.

Input: Invoke the script via Python's subprocess library.

Expected Result: The script should execute and produce the initial output logs indicating successful initialization.

Actual Result: To be recorded during testing.

Test Case 2: Validate Dependency Management

Objective: To verify that all required libraries are installed and functional.

Input: List of required libraries, including OpenCV and Keras.

Expected Result: The system should automatically check for dependencies and confirm their presence or guide the user through the installation process.

Actual Result: To be recorded during testing.

Test Case 3: Check Video File Path Handling

Objective: To validate that the system can handle input video file paths correctly.

Input: A set of valid and invalid video file paths.

Expected Result: The system should validate the file paths and proceed with processing only for valid ones, displaying appropriate error messages for invalid paths.

Actual Result: To be recorded during testing.

Test Case 4: Test Video Processing Duration

Objective: To confirm that the processed video has a duration that matches the input video.

Input: A pre-recorded video with a known duration.

Expected Result: The duration of the processed video should be approximately equal to that of the input video, within an acceptable margin of error.

Actual Result: To be recorded during testing.

Test Case 5: Validate Time-Stamp Calculation

Objective: To ensure that the time-stamp calculation function returns accurate values.

Input: A set of numerical values for total time and speed.

Expected Result: The function should return time-stamps that match the expected calculations based on the input values.

Actual Result: To be recorded during testing.

Test Case 6: Verify Human Detection

Objective: To validate the accuracy of the human detection algorithm.

Input: A video containing human figures in various states and distances.

Expected Result: The algorithm should correctly identify all human figures present in the video.

Actual Result: To be recorded during testing.

Test Case 7: Test Error Handling

Objective: To evaluate the system's ability to handle errors effectively.

Input: A set of files designed to trigger various errors, such as missing files and incorrect formats.

Expected Result: The system should display appropriate error messages and log these errors for future reference.

Actual Result: To be recorded during testing.

Test Case 8: Assess Output Verification

Objective: To confirm that the system outputs results in the expected formats.

Input: A set of pre-defined templates for expected output formats.

Expected Result: The actual outputs should match the templates in structure and content.

Actual Result: To be recorded during testing.

5.2 Test Cases for Python Script: Human_Detection_Script.py

Please find the Test Cases Excel sheet here:

Test Cases.xlsx

6. Maintenance and Support

6.1 Deployment Script

Disclaimer

The deployment script provided in this documentation is intended as a rough example and subject to change. The final deployment script will be developed upon the completion of the application to ensure it aligns with the final product.

Introduction

As a part of the Agile development process, one of the critical steps is creating a deployment script that packages the software into an installer. This installer ensures that the end-user can easily install the application on their machine without manual configuration.

Pre-Deployment Preparation

Final Compilation

The application will be compiled in 'Release' mode to optimize performance for end-users.

Tool Selection

After reviewing various installer creation tools, Inno Setup will be chosen for its flexibility and comprehensive feature set. Inno Setup will be downloaded and installed from the official website.

Writing the Deployment Script

Script Initialization

A new Inno Setup script file with an .iss extension will be initiated. This script will serve as the blueprint for the installer.

Basic Metadata

The script will specify essential metadata such as the application's name, version, and the default directory for installation:

```
[Setup]
AppName=Infrared Drone Human Detection System
AppVersion=1.0
DefaultDirName={pf}\Infrared Drone Human Detection System
```

Installer Settings

Additional settings governing the installer's behaviour, such as output directory, compression methods, and others, will also be defined:

```
OutputDir=Output
OutputBaseFilename=setup
Compression=lzma
SolidCompression=yes
```

File and Directory Instructions

The directories to be created and the files to be copied during installation will be specified:

```
[Dirs]
Name: "{app}\Docs"; Flags: uninsneveruninstall
Name: "{app}\Dependencies"; Flags: uninsneveruninstall

[Files]
Source: "Path\To\Release\*"; DestDir: "{app}"; Flags: ignoreversion recursesubdirs createallsubdirs
```

Icons and Post-install Actions

Code will be added to create icons in the Start Menu and on the desktop. Post-installation actions like launching the application will also be specified:

```
[Icons]
Name: "{group}\Infrared Drone Human Detection System"; Filename: "{app}\YourMainExecutable.exe"
Name: "{commondesktop}\Infrared Drone Human Detection System"; Filename: "{app}\YourMainExecutable.exe"

[Run]
Filename: "{app}\YourMainExecutable.exe"; Description: "Launch Application"; Flags: nowait postinstall skipifsilent
```

Compilation

Upon completion of the script, it will be compiled using Inno Setup, resulting in a setup.exe file. This executable will serve as the final installer.

Testing and Validation

The installer will be tested on multiple systems to ensure its correct operation and validate that it installs the application as intended.

6.2 Installation Guide

Please find the installation guide document here:

Installation Guide.pdf

6.3 User Manual

Please find the user manual document here:

User Manual.pdf

6.4 Bug Reports

There are currently no bugs to report on. However, consider the following bug reports structure:

Title:

Short, descriptive title that summarizes the issue.

Bug ID:

Unique identifier for tracking purposes.

Date Reported:

Date when the bug was initially discovered.

Reported By:

Name of the person/team who discovered the bug.

Environment:

Operating System, Browser Version, Device Information, etc. where the bug was found.

Steps to Reproduce:

1. ***Step 1***
2. ***Step 2***

3. *Step 3*
4. ...

Expected Result:
Describe what should happen when the steps are followed.

Actual Result:
Describe what actually happens when the steps are followed.

Attachments:
Include any relevant attachments like screenshots, log files, etc.

Priority:
Critical, High, Medium, Low

Status:
New, In Progress, Resolved, Closed, etc.

Assigned To:
Name of the person/team responsible for fixing the bug.

Description:
Additional details that describe the bug, its impact, and possible causes.

Sample Bug Report for the Drone-Based AI Infrared Human Detection System:

Bug Report 1:

- **Title:** Application crashes when uploading a large video file.
- **Bug ID:** 101
- **Date Reported:** 2023-08-30
- **Reported By:** Developer 2
- **Environment:** Windows 11, C# Windows Forms
- **Steps to Reproduce:**
    1. Open the application.
    2. Click on "Upload Video".
    3. Select a video file larger than 2GB.
- **Expected Result:** The application should either upload the video or show an error message.
- **Actual Result:** The application crashes.
- **Attachments:** crash_log.txt
- **Priority:** Critical
- **Status:** New
- **Assigned To:** Developer 1
- **Description:** Application cannot handle video files larger than 2GB, leading to crashes.

6.5 Change Requests

There are currently no change requests to report on. However, consider the following change requests structure:

Change Request ID:

Unique Identifier for the Change Request

Project:

Name of the Project for which the change is requested

Date Submitted:

The date when the Change Request was submitted

Submitted By:

Name of the person/team submitting the Change Request

Stakeholder(s):

Name of the person or group who has a stake in the proposed change

Description:

Detailed description of the change, why it is necessary, and its expected impact on the project

Justification:

A rationale for why the change is necessary, including any data or metrics

Proposed Change:

Detailed specification of what exactly will be changed

Impact Analysis:

Analysis of how the change will impact the existing system and the project as a whole, including risks and dependencies

Estimated Time:

How long will it take to implement the change?

Priority:

Is this change Critical, High, Medium, or Low priority?

Approval Status:

Pending, Approved, Rejected

Approved By:

Who has the authority to approve this change request?

Comments:

Any additional comments or information that should be considered

Sample Change Request for the Project:

Change Request 1:

- **Change Request ID:** CR-001
- **Project:** Drone-Based AI Infrared Human Detection System
- **Date Submitted:** 2023-08-30

- **Submitted By:** Product Owner
- **Stakeholder(s):** Development Team, QA Team
- **Description:** Add a feature to allow users to upload multiple video files for processing.
- **Justification:** Users have requested the ability to process multiple videos simultaneously.
- **Proposed Change:** Modify the "Upload Video" functionality to support multi-file selection and processing.
- **Impact Analysis:** This will require changes in the backend to handle multiple files and may increase the time for processing.
- **Estimated Time:** 2 weeks
- **Priority:** Medium
- **Approval Status:** Pending
- **Comments:** This will be a useful feature but requires a thorough impact analysis.

Conclusion

The design phase of the project has been comprehensively detailed in this document. The completion of this phase sets the stage for Milestone 3: the implementation phase, where the theoretical frameworks and plans will be put into practice. The designs and strategies laid out here aim to serve as a comprehensive guide for all stakeholders involved, ensuring alignment and facilitating effective communication.

This document is subject to updates and revisions to accommodate any changes in project scope, requirements, or other unforeseen variables. Stakeholders will be notified of such updates, ensuring transparency and alignment throughout the project lifecycle.