Software Requirements Specification
for
InfraSight
Infrared Image Object Detection System

September 23, 2025

Contents

1 Introduction

1.1 Project Vision & Goals

InfraSight is a desktop application designed to revolutionize how we process and analyze infrared (thermal) imagery for object detection. The primary vision is to create a powerful yet user-friendly tool that bridges the gap between advanced deep learning technology and practical security applications.

Primary Goals

- Provide real-time object detection capabilities on infrared imagery
- Enable security personnel to monitor areas effectively in low-visibility conditions
- Support researchers in evaluating deep learning model performance on thermal datasets
- Deliver a reliable, standalone solution that requires minimal technical expertise to operate

Secondary Goals

- Maintain high processing speeds for rapid analysis of images
- Ensure system reliability for mission-critical security applications
- Provide intuitive visualization of detection results
- Support multiple image formats for flexibility in deployment

The system addresses a critical need in security infrastructure where traditional visible-light cameras fail to provide adequate surveillance capabilities during nighttime, adverse weather conditions, or in environments where lighting is limited or compromised.

1.2 Key Features Overview

InfraSight incorporates several key features that make it suitable for both research and operational deployment:

Core Detection Capabilities

The system can identify multiple object classes including persons, vehicles (cars), and bicycles within infrared imagery. Each detection includes precise bounding box coordinates, confidence scores, and class labels to provide comprehensive situational awareness.

Image Processing

The application supports static image analysis. Users can load individual infrared images for detailed analysis or process folders of images in batches.

User Interface

A clean, intuitive graphical interface provides easy access to all system functions. The main display shows processed imagery with detection overlays, while control panels allow users to adjust detection sensitivity.

Performance Optimization

The system leverages GPU acceleration through CUDA to achieve high-speed processing, capable of analyzing high-resolution images in seconds on appropriate hardware.

Reliability Features

Comprehensive error handling, logging capabilities, and graceful failure recovery ensure the system remains operational even when encountering unexpected conditions or invalid input data.

1.3 Glossary of Terms

Bounding Box

A rectangular coordinate system that defines the spatial boundaries of a detected object within the image frame. Typically specified using top-left and bottom-right corner coordinates.

Class Label

The categorical identifier assigned to detected objects, such as "Person," "Car," or "Bicycle." These labels correspond to the object classes the deep learning model was trained to recognize.

Confidence Score

A numerical value between 0.0 and 1.0 that indicates the model's certainty level for each detection. Higher values represent greater confidence in the detection accuracy.

CUDA

Compute Unified Device Architecture, NVIDIA's parallel computing platform that enables the application to leverage GPU processing power for accelerated deep learning inference.

Deep Learning (DL)

A subset of machine learning that uses neural networks with multiple layers to learn complex patterns in data, particularly effective for image recognition and object detection tasks.

Inference

The process of using a trained neural network model to analyze new data and generate predictions or classifications without further training.

Infrared (IR)

Electromagnetic radiation with wavelengths longer than visible light, often associated with heat signatures. IR imagery captures thermal emissions from objects rather than reflected visible light.

Thermal Imaging

A technology that captures infrared radiation to create images based on temperature differences, allowing visibility in complete darkness or through obscurants like smoke or fog.

2 User Profiles

2.1 The AI Researcher

Background and Expertise

AI Researchers using InfraSight typically hold advanced degrees in computer science, machine learning, or related technical fields. They possess deep understanding of neural network architectures, training methodologies, and performance evaluation metrics. Their work often involves developing, testing, and comparing different deep learning approaches for computer vision tasks.

Primary Use Cases

Researchers utilize InfraSight primarily for model validation and performance assessment. They need to quickly evaluate how well their trained models perform on diverse infrared datasets, comparing accuracy across different environmental conditions, object sizes, and thermal signatures. The system serves as a rapid prototyping tool that allows them to visualize model outputs without developing custom visualization code.

Technical Requirements

This user group requires access to detailed performance metrics, the ability to adjust confidence thresholds for precision-recall analysis, and clear visualization of detection boundaries. They value processing speed for iterating through large datasets.

Workflow Patterns

Researchers typically work with batch processing of test images, systematic threshold adjustment to generate performance curves, and detailed analysis of false positives and missed detections. They often need to document results for publications or technical reports.

2.2 The Security Operator

Background and Expertise

Security Operators represent the primary operational users of InfraSight in real-world deployment scenarios. They may have limited technical backgrounds but possess extensive training in surveillance protocols, threat assessment, and emergency response procedures. Their expertise lies in interpreting visual information quickly and making critical security decisions.

Primary Use Cases

Operators use InfraSight for analyzing images captured for perimeter monitoring, facility security, and area surveillance during low-visibility conditions. They need to detect unauthorized personnel or identify suspicious activities from static images. The system must provide clear, immediate alerts for potential security breaches.

Operational Requirements

This user group requires a simple, intuitive interface that minimizes training requirements and reduces the possibility of operational errors. The system must provide reliable, consistent performance and clear visual indicators of detected objects. False alarm rates must be minimized to maintain operator attention and system credibility.

Workflow Patterns

Operators typically analyze images captured in response to an event or as part of a periodic review. They need to quickly load an image, view the detection results, and make a decision based on the visual information.

3 Functional Requirements: System Features

3.1 Feature: Media Input

3.1.1 FR-1: Loading an Image

Description

The system shall provide functionality for users to load individual infrared images for analysis. This feature serves as the foundation for single-image object detection tasks and detailed static analysis.

User Interaction

Users initiate image loading by clicking the ”Load Image” button located in the control panel. This action opens a standard operating system file dialog that allows navigation through the local file system to select the desired image file.

Supported Formats

The system accepts common image formats including PNG (.png), JPEG (.jpg), and JPEG (.jpeg) files. These formats provide sufficient quality for infrared imagery while maintaining reasonable file sizes for processing efficiency.

Processing Workflow

Upon file selection, the system performs format validation to ensure compatibility. Valid files are loaded into system memory and immediately processed through the object detection pipeline. The original image is preserved while detection results are overlaid for display.

User Feedback

The selected image appears in the main display panel with any detected objects highlighted through bounding boxes and labels. The image filename is shown in the status bar to confirm successful loading.

Error Conditions

Invalid file formats trigger user-friendly error messages explaining the supported formats. Corrupted files or access permission issues are handled gracefully with appropriate error notifications.

3.1.2 FR-2: Error Handling for Invalid Files

File Format Validation

The system performs immediate validation of file extensions upon selection. Unsupported formats are rejected before attempting to load file contents, preventing system errors and providing clear user feedback.

Corruption Detection

File integrity checks identify corrupted media files that cannot be properly decoded. The system provides specific error messages indicating file corruption and suggests trying alternative files.

Access Permission Handling

Files with insufficient read permissions generate appropriate error messages explaining access restrictions. The system guides users toward selecting files from accessible locations.

Memory and Resource Constraints

Large files that exceed system memory capabilities are handled gracefully with warnings about potential performance impacts. Users are informed when files may cause processing delays or resource limitations.

Recovery Procedures

All error conditions allow users to attempt loading different files without requiring application restart. The system maintains stable operation even after encountering invalid inputs.

3.2 Feature: Detection Engine

3.2.1 FR-3: Core Object Detection

Description

The detection engine represents the core functionality of InfraSight, responsible for identifying and localizing objects within infrared imagery using deep learning models. This component processes input images and generates detection results including object locations, classifications, and confidence scores.

Supported Object Classes

The system can detect three primary object classes relevant to security applications:

- **Person:** Human figures in various poses and orientations
- **Car:** Motor vehicles including sedans, SUVs, and trucks
- **Bicycle:** Two-wheeled vehicles and cyclists

Detection Process

Input images are preprocessed and fed to the deep learning model for inference. The model analyzes thermal signatures and shapes to identify objects and generate bounding box coordinates. Each detection includes spatial coordinates (x, y, width, height), object class label, and confidence score.

Processing Pipeline

Image preprocessing includes format conversion, resizing to model input requirements, and normalization. Post-processing filters detections based on confidence thresholds and applies non-maximum suppression to eliminate duplicate detections of the same object.

Performance

The detection engine utilizes GPU acceleration through CUDA to achieve high processing speeds. Optimization techniques include efficient memory management for large images.

Output Generation

Detection results are formatted for visualization, including bounding box coordinates for overlay rendering, text labels for object identification, and confidence scores for user assessment of detection reliability.

3.2.2 FR-4: Adjusting the Confidence Threshold

Description

The confidence threshold feature allows users to control the sensitivity of object detection by filtering results based on the model's certainty level. This capability enables adaptation to different operational requirements and environmental conditions.

Threshold Range

The confidence threshold operates on a scale from 0.0 to 1.0, where 0.0 shows all detections regardless of certainty, and 1.0 shows only detections with maximum confidence. The default setting is typically 0.5 to balance detection completeness with false positive reduction.

User Interface

A slider control in the main interface allows real-time threshold adjustment. The current threshold value is displayed numerically alongside the slider for precise setting. Changes take effect immediately, updating the displayed detection results without requiring re-processing.

Impact on Detection Results

Higher thresholds reduce false positives but may miss valid objects with lower confidence scores. Lower thresholds increase detection completeness but may include more false alarms. Users can optimize settings based on specific operational needs.

Operational Guidelines

For security applications, lower thresholds may be preferred to avoid missing potential threats. Research applications may benefit from higher thresholds to focus on high-confidence detections for accuracy evaluation.

3.3 Feature: User Interface & Visualization

3.3.1 UI-1: Main Display

Description

The main display serves as the central focal point of the InfraSight interface, presenting processed infrared imagery with detection overlays in a clear, organized layout that supports both research analysis and operational monitoring requirements.

Layout Organization

The interface follows a logical organization with the main media display occupying the largest central area for optimal visibility. A control panel is positioned on the right side for easy access to primary functions. The bottom status bar provides real-time system information and feedback.

Media Display Panel

The central display panel presents loaded images with integrated detection visualization. The display automatically scales content to fit available space while maintaining aspect ratio. High-resolution images can be panned and zoomed for detailed examination.

Visual Design Principles

The interface uses a dark theme appropriate for security monitoring environments, reducing eye strain during extended operation periods. High contrast elements ensure visibility under various ambient lighting conditions. Color coding distinguishes different object classes and system states.

Responsive Design

The interface adapts to different screen sizes and resolutions while maintaining usability. Minimum window dimensions ensure all controls remain accessible and readable. The layout scales appropriately for high-DPI displays.

Accessibility Considerations

Text sizes and color contrasts meet accessibility standards for users with visual impairments. Keyboard shortcuts provide alternative navigation methods. Error messages and status updates are clearly visible and descriptive.

3.3.2 UI-2: Control Panel Layout

Description

The control panel provides organized access to all system functions through clearly labeled buttons, sliders, and controls that enable efficient operation without cluttering the main display area.

Media Loading Controls

A "Load Image" button provides clear access to the file loading function. The button is sized appropriately for easy targeting and includes a descriptive label. Loading progress indicators appear during file operations.

Confidence Threshold Control

A prominent slider allows real-time confidence threshold adjustment with immediate visual feedback. The current threshold value is displayed numerically with precision to two decimal places. Min/max labels indicate the available range.

System Information Display

The control panel includes areas for displaying current system status, processing time, and active configuration settings. Information updates dynamically during operation.

3.3.3 UI-3: Displaying Detection Results

Description

Detection results are visualized through intuitive graphical overlays that clearly indicate detected objects while preserving the underlying infrared imagery for context and manual verification.

Bounding Box Visualization

Detected objects are highlighted with colored rectangular bounding boxes drawn around their perimeters. Different object classes use distinct colors for immediate identification: green for persons, blue for cars, yellow for bicycles.

Label and Score Display

Each detection includes a text label showing the object class name and confidence score. Labels are positioned near the top-left corner of bounding boxes to minimize visual obstruction. Font sizes are optimized for readability at typical viewing distances.

Visual Styling

Bounding boxes use semi-transparent fills with solid borders to highlight objects without completely obscuring underlying details. Line thickness and opacity are calibrated for visibility across different infrared image characteristics.

Multiple Object Handling

When multiple objects are detected in a single image, each receives individual visualization elements. Overlapping detections are handled through z-order management to ensure all objects remain visible.

4 Non-Functional Requirements

4.1 Performance

NFR-PERF-1: Processing Speed

The system must be highly responsive. Specifically, InfraSight shall process a 1024x768 resolution infrared image in under 2 seconds on target hardware configurations. This ensures a fluid user experience for single-image analysis.

NFR-PERF-2: Startup Time

Application initialization time directly impacts operational readiness. The system shall complete startup procedures and become ready for user input within 5 seconds of launch. This includes loading necessary libraries, initializing the detection model, and presenting the user interface.

NFR-PERF-3: Memory Utilization

Efficient memory management is critical for stable long-term operation. The system shall not consume more than 2 GB of system RAM during operation to maintain system stability and allow concurrent operation of other applications.

NFR-PERF-4: GPU Resource Management

The system requires dedicated GPU memory for deep learning inference. Minimum requirements include 6 GB of dedicated video memory (VRAM) for processing high-resolution imagery and maintaining model data in GPU memory for optimal processing speed.

NFR-PERF-5: Scalability

The system should gracefully handle higher resolutions with proportional performance adjustments. Processing time may increase with larger images, but the system must remain responsive and stable.

NFR-PERF-6: Benchmark Hardware

Performance requirements are based on systems equipped with NVIDIA GPUs supporting CUDA Compute Capability 7.0 or higher, representing modern hardware suitable for deep learning applications.

4.2 Reliability

NFR-REL-1: Operational Stability

InfraSight must demonstrate exceptional reliability. The system shall be able to process a batch of 500 images consecutively without experiencing crashes or requiring restart procedures.

NFR-REL-2: Error Recovery

When errors occur, the system must handle them gracefully without terminating unexpectedly. All critical errors shall be logged to a local file (error.log) with detailed timestamps, error descriptions, and system state information to facilitate troubleshooting and system improvement.

NFR-REL-3: Input Validation

Robust input validation prevents system failures caused by malformed or unexpected data. The system shall validate all image files, user inputs, and configuration parameters before processing, providing clear error messages for invalid inputs while maintaining system stability.

NFR-REL-4: Resource Monitoring

The system shall monitor its own resource usage and provide warnings when approaching memory or processing limits. Automatic cleanup procedures shall prevent resource exhaustion during extended operation periods.

NFR-REL-5: Graceful Degradation

When operating under resource constraints or suboptimal conditions, the system shall continue functioning with reduced performance rather than failing completely. Users shall be informed of performance limitations and possible mitigation steps.

NFR-REL-6: Data Integrity

All processing operations must maintain data integrity, ensuring that detection results accurately correspond to input imagery. The system shall include validation checks to verify processing consistency and accuracy.

4.3 Security

NFR-SEC-1: Network Isolation

InfraSight operates as a completely offline application to prevent security vulnerabilities associated with network communications. The system shall not create any external network connections, ensuring that sensitive imagery and detection data remain within the local environment.

NFR-SEC-2: Data Protection

All user data, including loaded images and detection results, shall be processed in memory without creating unnecessary persistent copies. Any temporary files created during processing must be securely deleted upon completion.

NFR-SEC-3: File System Security

The application shall only access files explicitly selected by users through standard file dialogs. No automatic scanning or accessing of file system contents beyond user-specified locations is permitted.

NFR-SEC-4: Model Security

Deep learning models loaded by the system shall be validated for integrity and compatibility. The system shall not execute arbitrary code or load models from untrusted sources without user verification.

NFR-SEC-5: Audit Trail

All system operations that access or modify data shall be logged for security audit purposes. Logs include file access patterns, processing operations, and any security-relevant events without storing sensitive image content.

NFR-SEC-6: Privacy Compliance

When deployed for surveillance applications, the system design supports compliance with privacy regulations by limiting data retention, preventing unauthorized data access, and maintaining user control over data processing.

4.4 Usability

NFR-USA-1: Learning Curve

The interface shall be intuitive enough for security operators to become proficient within minimal training periods. Complex technical concepts are abstracted behind simple controls that focus on operational tasks rather than technical configuration.

NFR-USA-2: Visual Clarity

All interface elements use clear labeling, appropriate sizing, and logical grouping to support quick comprehension and efficient operation. Detection results are presented with sufficient visual contrast and detail for accurate interpretation.

NFR-USA-3: Error Prevention

The interface design minimizes opportunities for user errors through clear feedback, confirmation dialogs for significant actions, and input validation that prevents invalid operations before they can cause problems.

NFR-USA-4: Operational Efficiency

Common operations are accessible through minimal clicks or keystrokes. Threshold adjustments and file loading operations can be performed quickly without navigating complex menu structures.

NFR-USA-5: Status Communication

The system continuously communicates its current state, processing progress, and any relevant warnings through status indicators, progress bars, and informational messages that keep users informed without overwhelming them.

NFR-USA-6: Help and Documentation

Built-in tooltips and contextual help provide guidance for interface elements and system functions without requiring separate documentation resources during normal operation.

4.5 Maintainability

NFR-MNT-1: Code Documentation

All software components include comprehensive documentation to support future maintenance and enhancement. Functions, classes, and modules include detailed docstrings explaining purpose, parameters, return values, and usage examples.

NFR-MNT-2: Modular Architecture

The system architecture separates concerns into distinct modules for user interface, detection engine, image processing, and configuration management. This separation facilitates independent maintenance and testing of individual components.

NFR-MNT-3: Configuration Management

System settings and parameters are managed through configuration files and clear interfaces that allow adjustment without modifying source code. Version control and change tracking support systematic updates and rollback capabilities.

NFR-MNT-4: Testing Framework

Automated testing procedures validate system functionality and detect regressions during maintenance updates. Test suites cover critical functions including detection accuracy, performance benchmarks, and error handling scenarios.

NFR-MNT-5: Logging and Diagnostics

Comprehensive logging captures system behavior, performance metrics, and error conditions to support troubleshooting and system optimization efforts. Log levels allow filtering between normal operation monitoring and detailed debugging information.

NFR-MNT-6: Update Procedures

The system supports systematic updates to detection models, software components, and configuration parameters while maintaining compatibility with existing operational procedures and data formats.

5 Technical Constraints & Dependencies

5.1 Required Hardware

CON-HW-1: Graphics Processing Unit (GPU)

InfraSight requires an NVIDIA graphics card with CUDA Compute Capability 7.0 or higher for deep learning acceleration. This requirement ensures compatibility with modern deep learning frameworks and provides sufficient processing power for object detection. Examples of suitable GPUs include the RTX 2060 and newer models in the RTX series, or equivalent Quadro cards for professional deployments.

CON-HW-2: System Memory (RAM)

Minimum system memory requirements include 8 GB of RAM for basic operation, with 16 GB recommended for optimal performance during intensive processing tasks. The additional memory supports large image processing, model loading, and operating system requirements while maintaining responsive performance.

CON-HW-3: Video Memory (VRAM)

The system requires a minimum of 6 GB of dedicated video memory to store deep learning models and process high-resolution imagery efficiently. Insufficient VRAM leads to reduced processing speeds or inability to handle large images.

CON-HW-4: Central Processing Unit (CPU)

A modern multi-core processor (minimum quad-core) is required for efficient data preprocessing, user interface responsiveness, and coordination between GPU and system components. Intel Core i5 or AMD Ryzen 5 processors represent the minimum performance tier for satisfactory operation.

CON-HW-5: Storage Requirements

The application requires approximately 2 GB of disk space for installation, including the software executable, required libraries, and deep learning models. Additional storage space is needed for user image files and system logs based on operational requirements.

CON-HW-6: Display Requirements

A minimum screen resolution of 1280x720 pixels ensures proper interface layout and readable text. Higher resolutions (1920x1080 or greater) provide improved visualization of detection results and more comfortable operation.

5.2 Required Software & Libraries

CON-SW-1: Operating System Compatibility

InfraSight supports deployment on Windows 10/11 and Ubuntu 20.04 LTS operating systems. These platforms provide necessary driver support for NVIDIA hardware and compatibility with required software libraries.

CON-SW-2: NVIDIA Driver and CUDA

The system depends on NVIDIA graphics drivers that support CUDA 11.x or newer. Proper driver installation is critical for GPU acceleration functionality. The CUDA runtime environment must be available for the deep learning framework to access GPU resources.

CON-SW-3: Python Runtime Environment

The application is built on Python 3.8 or newer versions to leverage modern language features and library compatibility. The Python environment includes the interpreter and standard library components required for application execution.

CON-SW-4: Deep Learning Framework

PyTorch version 1.10 or newer provides the foundation for deep learning model execution and GPU acceleration. This framework includes optimized routines for neural network inference and CUDA integration.

CON-SW-5: Computer Vision Library

OpenCV version 4.5 or newer handles image processing operations including format conversion, resizing, and display functions. This library provides efficient implementations of common computer vision operations.

CON-SW-6: Additional Python Dependencies

Supporting libraries include NumPy for numerical operations, Pillow for image format handling, and tkinter for graphical user interface components. These dependencies are typically bundled with the application distribution to ensure compatibility and simplify deployment.

CON-SW-7: Packaging and Distribution

The application is distributed as a standalone executable that includes all necessary Python dependencies, eliminating requirements for system-wide Python installations or complex dependency management on target systems.

CON-SW-8: Model Dependencies

The system requires pre-trained deep learning models in PyTorch format (.pth or .pt files) that are compatible with the specified framework version. Model files must be available locally and properly configured for the supported object detection classes.

CON-SW-9: Library Version Compatibility

All software dependencies are tested for mutual compatibility to prevent conflicts during deployment. Version specifications ensure that the application functions correctly across the supported range of library versions while providing stability and security updates.