# MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

# SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

*for*

# Autonomous Drone Landing Scene Recognition Using Transfer Learning

**Prepared by**


| S. No | Roll Number | Student Name |
|---|---|---|
| 1. | 21N31A6657 | Gudla Likhitha |
| 2. | 21N31A6663 | Jakkula Keerthi |
| 3. | 22N35A6601 | Burugulapelly Vivek |


| | |
|---|---|
| **Supervisor:** | **Mrs. G. Deepthi** |
| **Designation:** | **Assistant Professor** |
| **Department:** | **School of Computational Intelligence, MRCET** |
| **Batch ID:** | **21CIMP2A01** |
| **Date:** | **09-04-2025** |
| **Supervisor Sign. & Date** | |

## Department of Computational Intelligence

**Title of the Project: Autonomous Drone Landing Scene Recognition using Transfer Learning**

# Content



# Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| 1.0 | Gudla Likhitha | Primary Revision giving an overall view of the project and document. | 07/04/2025 |

# 1 Introduction

This section details the proposed system, " Autonomous Drone Landing Scene Recognition Using Transfer Learning" an innovative content creation system designed to address the shortcomings of existing multimedia production methods. By leveraging the power of Generative AI, this system aims to provide a more efficient, scalable, and accessible approach to transforming text into high-quality video content. This section will outline the core features, functionalities, and design principles of the proposed system, providing a comprehensive overview of how this AI-driven platform will achieve its objectives.

## 1.1 Document Purpose

This Software Requirements Specification (SRS) document provides a comprehensive outline of the requirements, functionalities, and design aspects of the "Autonomous Drone Landing Scene Recognition Using Transfer Learning" project. The purpose of this document is to serve as a guideline for development teams, stakeholders, and researchers involved in the project. It ensures that all necessary specifications, including software, hardware, functional, and non-functional requirements, are well-defined to achieve precise autonomous drone landings. The document also aims to:

- Define the system's objectives and constraints.
- Detail the technical specifications required for implementation.
- Identify the scope and limitations of the system.
- Establish performance, safety, and security benchmarks.

## 1.2 Project/Product Scope

The project focuses on the development of an AI-based drone landing system that utilizes deep learning techniques, particularly transfer learning, to landing precision. The system is designed to address the challenges of autonomous landings by leveraging scene recognition models trained on diverse environmental datasets. Key features include:

- Automated landing zone detection using a fine-tuned Convolutional Neural Network (CNN).
- Real-time obstacle avoidance through sensor fusion.
- Adaptability to various altitudes and environmental conditions.
- Improved safety measures through predictive control algorithms.
- Integration with existing drone navigation systems to enhance functionality.

## 1.3 Existing System

Current autonomous drone landing systems rely on **basic image processing techniques** or **manual intervention**, leading to several limitations:

1. **Basic Computer Vision Methods:** Many existing solutions use traditional image processing techniques that lack adaptability to complex environments.
2. **Manual Drone Control:** Pilots need to guide drones manually, which increases operational costs and risks.
3. **Limited Scene Understanding:** Current systems struggle to differentiate between safe and unsafe landing zones in real-time.
4. **Increased Landing Time:** Inefficient landing protocols lead to extended flight durations, draining battery life.
5. **Low Precision in Varying Conditions:** Environmental factors like lighting, wind, and terrain variations reduce the reliability of current landing methods.

## 1.4 Problems with Existing System

- **Poor Generalization:** Current systems cannot generalize well across different terrains, lighting conditions, or altitudes.
- **High Dependency on GPS:** Some systems heavily rely on GPS-based navigation, which fails in GPS-denied environments.
- **Manual Supervision Required:** Most solutions require human intervention, reducing autonomy.
- **Inefficient Obstacle Detection:** Limited integration of sensors leads to higher chances of collisions.
- **Slower Decision Making:** Traditional landing approaches take more time to compute safe landing zones.

## 1.5 Proposed System

The proposed system introduces a deep learning-based approach for autonomous drone landing scene recognition. Using transfer learning, the system improves landing precision, even in dynamic and unpredictable conditions. The enhancements include:

1. **Deep Learning for Scene Recognition**: A CNN-based model is fine-tuned with landing zone images to classify suitable and unsuitable landing areas accurately.
2. **Real-Time Processing:** The system utilizes real-time inference for identifying landing zones within seconds.
3. **Obstacle Detection & Avoidance:** Integration of LiDAR, ultrasonic, and infrared sensors ensures enhanced safety by detecting obstacles in the landing path.
4. **Autonomous Control Algorithms:** The system implements predictive control models to adjust drone movements for smooth and precise landings.
5. **Adaptive Landing Mechanism:** The drone dynamically selects the safest landing site based on terrain, weather conditions, and obstacles.
6. **Self-Learning Capabilities:** The system continuously learns from past landing experiences, improving accuracy over time.
7. **Weather Condition Adaptability:** It incorporates weather prediction models to ensure safe landings in diverse environmental conditions.
8. **Integration with IoT and Cloud Platforms:** Enables real-time data sharing and analysis, facilitating remote monitoring and control.

## 1.6 Advantages of Proposed Systems

1. **Higher Accuracy in Landing Zone Recognition:**
    - The use of pre-trained deep learning models enhances the accuracy of landing zone detection.
    - Transfer learning enables adaptation to new environments with minimal retraining.
2. **Faster and More Efficient Landings:**
    - The system processes images and sensor data in real-time, allowing drones to land quickly and safely.
    - Optimized control algorithms minimize landing delays, reducing battery consumption.
3. **Enhanced Security & Safety Measures:**
    - Advanced obstacle avoidance mechanisms prevent mid-air and ground collisions.
    - The drone can recalculate landing sites dynamically if obstacles are detected.
    - Secure data transmission protocols ensure safe communication between drone and ground control.
4. **Reduced Dependency on GPS:**
    - The system works efficiently even in GPS-denied areas, such as indoor environments or remote locations.
    - Image-based landing site detection eliminates reliance on satellite signals.
5. **Improved Adaptability and Robustness:**
    - The model is trained on various environmental conditions, ensuring adaptability to different terrains, altitudes, and lighting scenarios.
    - Can be deployed on multiple drone models with minimal modifications.
6. **Scalability and Future Expansion Possibilities:**
    - The system's modular architecture allows for integration with advanced AI techniques, such as reinforcement learning.
    - Can be expanded to multi-drone coordination systems, enabling synchronized landings.
    - Integration with 5G technology for enhanced real-time communication and data processing.

# 2 Overall Description

## 2.1 Feasibility Study

A feasibility study evaluates the technical, economic, legal, operational, and scheduling aspects of the proposed Autonomous Drone Landing Scene Recognition system. The goal is to assess whether the system can be successfully implemented within given constraints.

This evaluation covered four key areas: technical, economic, operational, and scheduling feasibility.

- **Technical Feasibility**
    - The project is based on deep learning and computer vision techniques, requiring high computational power for model training.
    - Transfer learning is used to fine-tune a pre-trained CNN-based model for landing scene recognition, reducing the need for extensive labelled datasets.
    - The drone will integrate multiple sensors (camera, LiDAR, ultrasonic, and infrared) to enable real-time scene recognition and obstacle detection.
    - Requires a powerful onboard processing unit such as an NVIDIA Jetson Nano, Raspberry Pi 4, or an equivalent embedded AI computing system to process real-time data.
    - The system will communicate with ground control stations for continuous monitoring and control, ensuring high reliability.
- **Economic Feasibility**
    - **T**he cost of implementing the system is moderate, as it relies on off-the-shelf hardware components.
    - Using open-source AI frameworks (e.g., TensorFlow, PyTorch) reduces software development costs.
    - Reduces the need for manual drone operation, leading to long-term cost savings for industries using drones in agriculture, military, and disaster management.
    - The ROI (Return on Investment) is high due to increased efficiency, safety, and reliability in drone landings.
- **Operational Feasibility**
    - Can be **integrated into existing drone navigation systems** without major modifications.
    - Requires **minimal human supervision** for initial setup and maintenance.
    - Can function in **varied environmental conditions**, ensuring widespread usability.

**Scheduling Feasibility**

- The system is expected to be developed within **6-8 months**, including model training, hardware integration, and testing.
- Deployment can be achieved in **phases**, starting with **limited prototypes** before full-scale implementation.

## 2.2 Product Functionality

The **Autonomous Drone Landing Scene Recognition System** will have the following core functionalities:

**1. Scene Recognition and Landing Decision**

- Utilizes CNN-based deep learning models to classify landing zones as safe or unsafe.
- Real-time image and sensor fusion for accurate landing decisions.
- Multi-class classification: Determines land suitability based on terrain type, obstacles, and surface stability.

**2. Obstacle Detection and Avoidance**

- Uses LiDAR, ultrasonic, and infrared sensors to detect obstacles in the landing zone.
- Implements collision avoidance algorithms to adjust landing trajectory.
- Failsafe mechanisms: If obstacles are detected, the drone repositions to find a safer landing site.

**3. Autonomous Landing System**

- Adaptive landing mechanism dynamically selects the best possible landing location.
- Predictive control models ensure smooth, stable landings.
- Can operate in GPS-denied environments by relying on image-based landing detection.

**4. Remote Monitoring and Control**

- Real-time telemetry and video feed transmission to ground control stations.
- Allows manual override in emergency situations.
- Cloud-based system for logging flight data and analytics.

**5. Environmental Adaptability**

- Adapts to varying weather conditions (wind, rain, low visibility).
- Works in both urban and rural landscapes, detecting dynamic obstacles.

## 2.3 Design and Implementation Constraints

A **design constraint** is a limitation or restriction imposed on the system's design, whereas an **implementation constraint** refers to restrictions on how the system is developed or deployed. The **Autonomous Drone Landing Scene Recognition System** has several constraints that must be considered for successful deployment.

**Hardware Constraints**

1. **Processing Power**
    - The drone’s onboard processing unit (e.g., NVIDIA Jetson Nano, Raspberry Pi 4) must be capable of executing deep learning models in real-time.
    - High-performance edge computing is required to process camera feeds, sensor data, and machine learning algorithms simultaneously.
    - The processor should support parallel computations and be optimized for low power consumption.
2. **Battery Life and Power Consumption**
    - AI-driven computations consume high power, reducing flight time.
    - The drone’s battery must balance power distribution between flight control, camera processing, AI inference, and communication systems.
    - Efficient power management algorithms must be implemented to ensure optimal energy usage.
3. **Sensor Limitations**
    - The system relies on multiple sensors (camera, LiDAR, ultrasonic, infrared) to assess landing conditions.
    - Limited weight-carrying capacity of drones restricts the number of onboard sensors.
    - Sensor accuracy may degrade in adverse weather conditions, requiring advanced sensor fusion techniques.
4. **Camera Quality and Night Vision**
    - The drone must have a high-resolution camera for accurate landing scene recognition.
    - Low-light conditions may reduce image clarity, requiring infrared or thermal cameras for night landings.

- Motion blur due to drone movement must be minimized using stabilization algorithms.

**Software Constraints**

1. **Model Accuracy and Performance**
   - The CNN-based deep learning model must achieve high accuracy while maintaining real-time processing speed.
   - Requires robust training datasets to ensure generalization across various terrains, lighting, and weather conditions.
   - Overfitting is a challenge; model training must include data augmentation and regularization techniques.
2. **Real-Time Processing Speed**
   - The AI model should process frames at ≥ 30 FPS to enable real-time decision-making.
   - Latency in image classification should not exceed 100 ms to avoid delays in drone landing.
   - Edge AI optimization (e.g., TensorRT, OpenVINO) is required to enhance inference speed.
3. **Computational Load and Memory Usage**
   - The system must balance memory allocation between deep learning models and drone flight control software.
   - Onboard storage capacity may be limited, requiring cloud-based data offloading.
   - Models must be compressed (e.g., via quantization, pruning) to reduce memory footprint.
4. **Autonomous Navigation Algorithms**
   - The drone must use reinforcement learning or heuristic algorithms for landing site selection.
   - Obstacle avoidance should function independently of external inputs when GPS signals are weak.
   - Decision-making must be fault-tolerant to handle sensor malfunctions or partial failures.

**Environmental Constraints**

1. **Weather Conditions**
    - The system must operate reliably in variable weather conditions like wind, rain, fog, and snow.
    - Wind speeds exceeding 20 km/h can affect drone stability, requiring adaptive control mechanisms.
    - Rain can obscure the camera lens, requiring water-resistant sensors.
2. **Terrain Variability**
    - The drone must accurately identify safe landing zones in urban, rural, mountainous, and coastal environments.
    - Sloped or uneven surfaces should be detected using depth estimation algorithms.
3. **Radio Frequency (RF) Interference**
    - Urban environments with high RF interference can disrupt drone GPS and communication signals.
    - A backup sensor-based navigation system should be implemented in case of signal loss.

**Regulatory Constraints**

1. **Aviation Safety and Compliance**
    - Must comply with regulations from DGCA (India), FAA (USA), EASA (Europe) for autonomous drones.
    - Flight restrictions in military zones, airports, and urban areas must be enforced via geofencing.
2. **Data Privacy and Security**
    - Images and videos captured must be encrypted to prevent unauthorized access.
    - GDPR and similar data protection laws may restrict how data is stored and used.

## 2.4 Assumptions and Dependencies

The **Autonomous Drone Landing Scene Recognition System** is based on certain **assumptions** and is dependent on **external factors** for proper functionality.

**Assumptions**

1. **Sufficient Hardware Availability**
    - The system assumes high-performance embedded AI processors are available for real-time execution.
    - The cost of hardware components remains affordable for commercial deployment.
2. **Reliable Weather Conditions**
    - The drone is assumed to operate in moderate weather conditions.
    - Severe weather (e.g., storms, snowfall) is beyond the scope of this version of the system.
3. **Regulatory Approvals Will Be Obtained**
    - The drone is assumed to receive flight permissions for test and deployment phases.
    - The government will approve autonomous drone operations for non-military use.
4. **Training Data Is Diverse and Sufficient**
    - The AI model is assumed to be trained on a broad dataset covering different terrains, lighting conditions, and obstacles.
    - The system assumes no bias in dataset representation.
5. **Drone Connectivity Remains Stable**
    - Assumes constant GPS, Wi-Fi, or LTE connectivity for real-time monitoring.
    - Backup mechanisms are in place in case of temporary communication failures.

**Dependencies**

1. **Deep Learning Frameworks**
    - The system depends on TensorFlow, PyTorch, and OpenCV for AI model training and deployment.
    - Any change in these frameworks (e.g., version updates, deprecated features) may impact functionality.
2. **Hardware Components**
    - Requires LiDAR, ultrasonic, infrared, and high-resolution cameras to function optimally.
    - A failure in any sensor may affect landing accuracy and safety.

3. **Regulatory and Legal Compliance–**
    - The project depends on government approvals for test flights in public areas.
    - Failure to meet safety requirements can lead to project delays or cancellations.
4. **Communication Networks**
    - Requires low-latency networks for transmitting video feeds and telemetry data.

# 3 Functional Requirements

## 3.1 Software Requirement Specifications

The Autonomous Drone Landing Scene Recognition System requires the following software components:

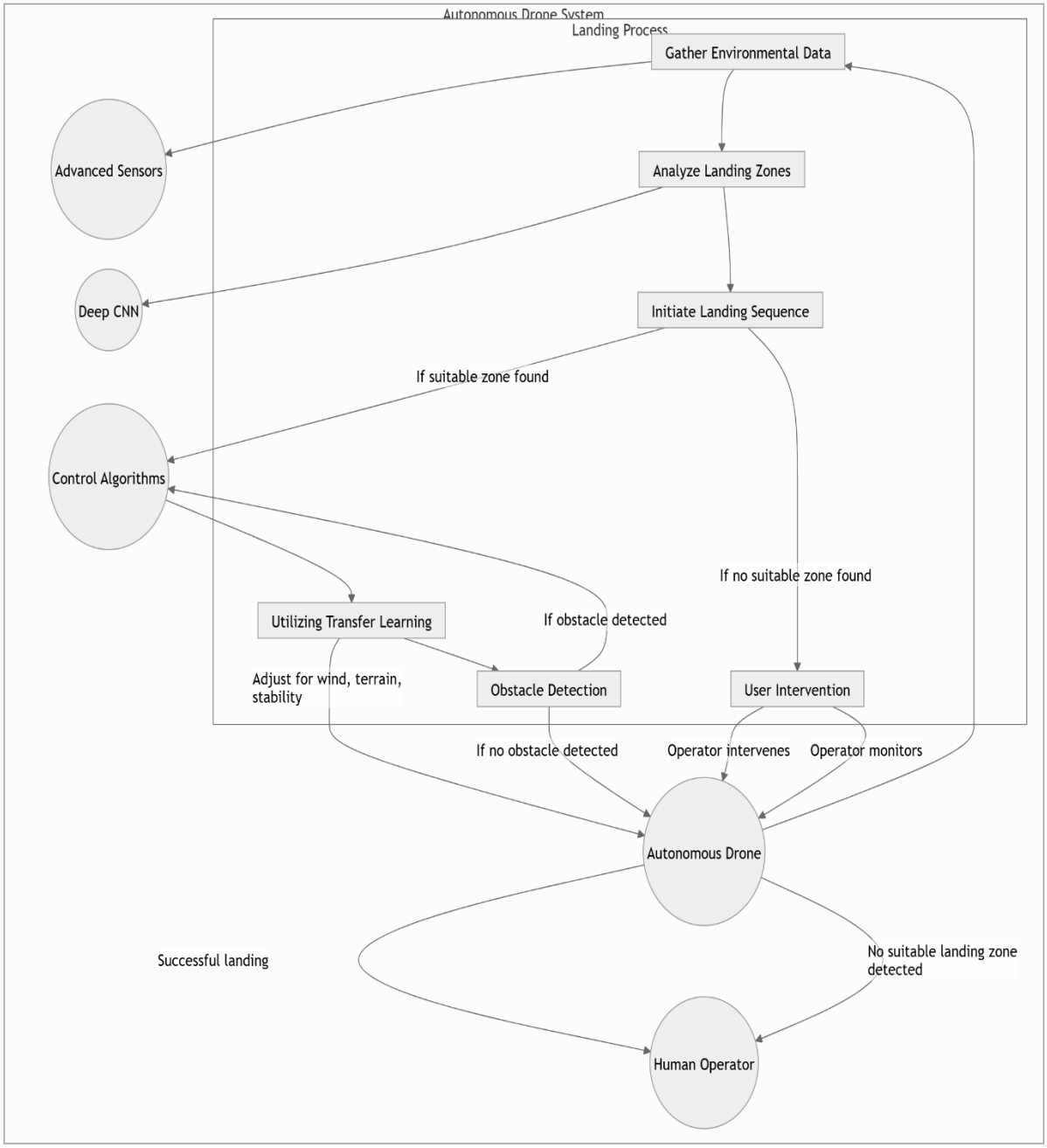
- **Operating System**: Windows 10/11, macOS, or Linux.
- **Programming Language**: Python 3.x
- **Web Framework**: Flask/Django (for UI and API integrations)
- **Cloud & Storage**: AWS S3, Google Drive, Local Storage
- **Version Control System**: Git/GitHub
- **Deployment**: Docker, Kubernetes
- **Data Processing & Visualization**: OpenCV, Matplotlib, Pandas, NumPy

## 3.2 Hardware Requirements Specifications

The platform's AI models and video generation processes require:

- **Processor**: Intel Core i7/i9 or AMD Ryzen 7/9 (or higher)
- **RAM**: Minimum **16GB** (32GB recommended for AI model training)
- **Storage**: 500GB SSD (1TB+ preferred for large media files)
- **GPU**: NVIDIA RTX 3060 or higher (for AI-based image processing)
- **Camera**: High-resolution camera module for scene recognition
- **Internet**: High-speed internet for real-time cloud processing
- **Drone Hardware**: Compatible with ROS-based flight controllers.

## 3.3 Use Case Model



### 3.3.1 Use Case #1 (Game Play – U1)

**Author** – Team AI Video Generation

**Purpose** – This use case diagram provides a high-level overview of how an autonomous drone interacts with various system components to perform an accurate and safe landing using AI, advanced sensors, and control algorithms.

**Requirements Traceability:**

- **R1**: Environmental data collection through sensors
- **R3**: Deep learning-based landing zone analysis
- **R6**: Transfer learning for adaptability to new environments
- **R10**: Obstacle detection and path adjustment
- **R15**: Human operator override mechanism

**Priority** – High

**Preconditions** – User has uploaded text content (document, script, or raw text).

**Postconditions** Drone lands successfully or human operator takes control if no suitable landing zone is detected

**Actors**

- **Autonomous Drone**
- **Advanced Sensors**
- **Deep CNN (Convolutional Neural Network)**
- **Control Algorithms**
- **Human Operator**

**Extends** – N/A

**Flow of Events**

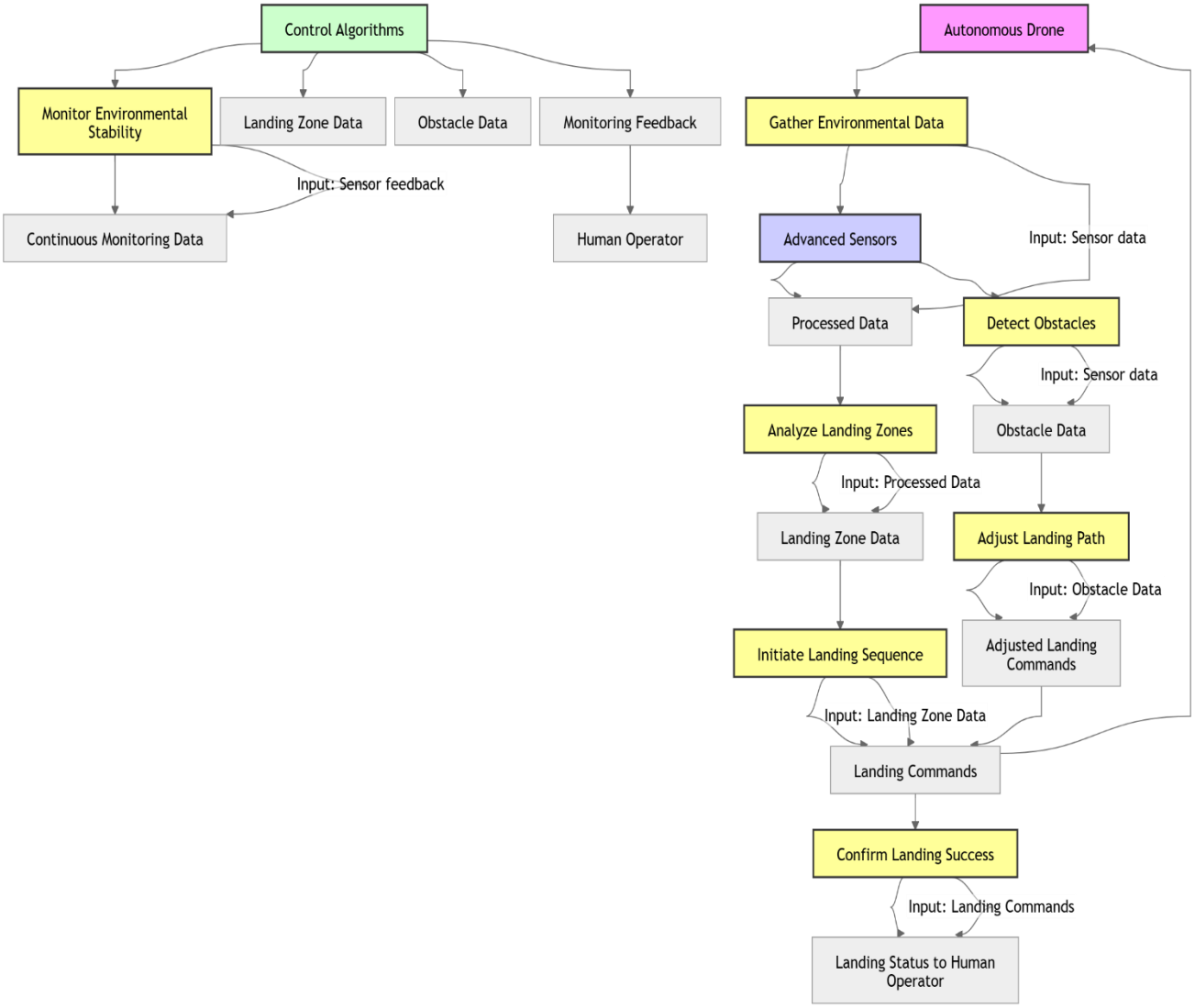
**Main Flow (Basic Flow)**:

1. Autonomous Drone initiates the landing sequence by gathering environmental data.
2. Advanced Sensors provide real-time input to the drone.
3. Deep CNN processes the data to analyse landing zones.
4. If a suitable landing zone is found, the drone proceeds with the landing sequence.
5. Control Algorithms adjust for wind, terrain, and ensure stability using transfer learning.
6. The drone checks for obstacles.
7. If no obstacles are detected, the Autonomous Drone proceeds with the landing.
8. Upon successful landing, the system terminates the sequence.

**Alternative Flow**

1. If **no suitable landing zone is found**, the system requests **user intervention**.
    - **Human Operator** either monitors or takes control for manual landing.
2. If **obstacles are detected**, control algorithms adjust the path accordingly.

### 3.3.2 Data Flow Diagram



# 4 Other Non-functional Requirements

## 4.1 Performance Requirements

To ensure accurate and real-time scene recognition for autonomous drone landing, the following performance metrics must be achieved:

- **P1. Inference Latency:** The system must identify a landing zone within **200 milliseconds** using the transfer learning model on onboard edge devices (e.g., Jetson Nano, Raspberry Pi 4).
- **P2. Model Accuracy:** The recognition system must achieve at least **95% accuracy** on test data for classifying landing vs. non-landing zones.
- **P3. Frame Processing Rate:** The system must process at least **10 frames per second (FPS)** in real-time video feeds.
- **P4. Model Load Time:** The pre-trained model (e.g., MobileNetV2, ResNet50) must be loaded and initialized within **3 seconds** during boot-up.
- **P5. Environmental Adaptation Time:** The model should adapt to changing lighting conditions within **5 frames** using image normalization techniques.
- **P6. Memory Footprint:** The model size should not exceed **100 MB** to ensure deployment on resource-constrained edge devices.
- **P7. Obstacle Detection Delay:** Scene detection and landing zone validation must not exceed **1 second** in cluttered environments.
- **P8. Heat Management:** The system should not exceed **75°C** during continuous real-time inference to prevent hardware failure.
- **P9. Data Logging Efficiency:** Telemetry and detection results must be logged at **1-second intervals** with minimal performance degradation.
- **P10. Communication Delay:** Wireless transmission of scene recognition data to a ground control station must be under **500 milliseconds**.

## 4.2 Safety and Security Requirements

Due to the critical nature of autonomous drone operations, the following safety and security standards must be followed:

- **S1. Obstacle Avoidance Fallback:** In case of scene misidentification, the drone must abort landing and return to a safe hover state.
- **S2. Fail-Safe Handling:** The system should default to manual override or return-to-home if GPS signal is lost or inference fails.

- **S3. Model Tampering Protection:** The deployed model must be validated via checksum or digital signature to avoid tampering.
- **S4. Secure Firmware Updates:** Updates to the model and firmware must be encrypted and securely delivered (e.g., using TLS).
- **S5. Input Data Sanitization:** Sensor and camera input must be verified for anomalies to prevent adversarial attacks or spoofing.
- **S6. Privacy Compliance:** No personal or sensitive data (e.g., facial data) should be stored or transmitted from the drone.
- **S7. Airspace Compliance:** Scene recognition and autonomous actions must comply with **DGCA** and **FAA** regulations.
- **S8. Secure Communication:** All communication between drone, cloud, and GCS must use **end-to-end encryption (e.g., AES-256)**.
- **S9. Hardware Access Control:** Access to the onboard compute module must require admin credentials or biometric authentication.
- **S10. Logging and Monitoring:** All recognition outputs, anomalies, and system events must be logged for auditing and debugging.

## 4.3 Software Quality Attributes

The following attributes are critical for ensuring the platform's usability, maintainability, and scalability.

### 4.3.1. Usability

**Requirement:** The drone system must offer a minimal UI for drone operators to monitor recognition outputs.

**Implementation:**

- Real-time preview of camera with highlighted safe zones.
- Visual indicators for "Landing Safe", "Unsafe", or "Obstructed".
- Simplified dashboard for pre-launch checks and logs.

**Verification:** Usability tested through drone pilot feedback and controlled field trials.

### 4.3.2 Maintainability

**Requirement:** The software must support easy updates and maintenance for models and scene detection logic.

**Implementation:**

- Modular pipeline for model swapping and retraining.
- Configuration files for drone-specific tuning.

- Source control (Git) for tracking changes**.**

**Verification:** Assessed via update simulation, Git logs, and code documentation reviews

### 4.3.3 Adaptability **(Design for Change)**

**Requirement:** The system must allow new landing zone scenarios (e.g., rooftop, grassland) to be added with minimal effort.

**Implementation:**

- Plug-and-play architecture for additional training data and retraining pipelines.
- Support for domain adaptation techniques in transfer learning.
- Extendable class labelling system.

**Verification:** Time taken to adapt the model to new terrains and successful field validation**.**

#### 4.3.4 Reliability

**Requirement:** The scene recognition must be reliable and consistent across various weather and terrain conditions.

**Implementation:**

Training with a diverse dataset (sunny, cloudy, dusk).

Real-time anomaly detection for inference confidence.

System stress testing under edge device constraints.

**Verification:** Reliability scored through simulation benchmarks and real-world drone flight evaluations.

# 5 Other Non-functional Requirements

This section outlines additional requirements not covered in the previous sections, which are essential for the complete development and deployment of "Autonomous Drone Landing Scene Recognition Using Transfer Learning"

## 5.1 Database Requirements (If Applicable)

- **R1. Flight Log Storage:** Logs must be stored locally and optionally uploaded to cloud storage (e.g., Firebase, AWS S3).
- **R2. Model Metadata Tracking:** Store version, training dataset, accuracy, and timestamp for every deployed model.w
- **R3. Log Encryption:** Logs and metadata must be encrypted at rest.

## 5.2 Internationalization Requirements (If Applicable)

- **R4. Multilingual Labels (optional):** UI should optionally support English and regional languages.
- **R5. Coordinate Formats:** Support multiple GPS coordinate formats (DMS, Decimal).
- **R6. Unicode Compliance:** Ensure all labels and logs use UTF-8 encoding.

## 5.3 Legal Requirements

- **R7. Airspace Regulation Compliance:** Drone usage and AI-based automation must follow **DGCA/FAA** rules.
- **R8. No-Person Detection Clause:** Model should avoid detecting or storing human features unless explicitly authorized.

- **R9. Data Retention Policy:** All video/image data must be deleted after a fixed retention period (e.g., 30 days).

## 5.4 Reuse Objectives

- **R10. Model Portability:** Allow the trained model to be deployed across different drone platforms with minimal tuning.

- **R11. Dataset Reusability:** Training data should be reusable for other perception-based drone applications.

- **R12. Modular Components:** Reuse image preprocessing, inference engine, and telemetry logging components in related projects.

## 5.5 Development Environment Requirements

- **R13. Programming Stack:** Python (TensorFlow/PyTorch, OpenCV), ROS, Flask (optional UI), C++ for firmware-level control.

- **R14. DevOps Tools:** GitHub, Docker for containerized deployment, CI/CD pipeline for model updates.

- **R15. Simulation Tools:** Use Gazebo, AirSim, or DroneKit for testing before real-world deployment.

## 5.6 Documentation Requirements

- **R16. Technical Docs:** Document model architecture, deployment scripts, and flight control integration.

- **R17. Deployment Guide:** Step-by-step installation and setup guide for edge devices.

- **R18. Field Test Report Template:** For each real-world trial, generate standardized test outcome reports.

- **R19. Troubleshooting FAQ:** List of common deployment and recognition issues with fixes.

- **R20. Version History:** Track changes in model, firmware, and configs across project iterations.

# 6 References

[1] LI B Q, HU X H Effective distributed convolutional neural network architecture for remote sensing images target classification with a pre-training approach Journal of Systems Engineering and Electronics, 2019, 30 (2): 238- 244.

[2] XIA G S, BAI X, DING J, et al DOTA: a large-scale dataset for object detection in aerial images Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, 3974-3983.

[3] CHENG G, XIE X X, HAN J W, et al Remote sensing image scene classification meets deep learning: challenges, methods, benchmarks, and opportunities IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2020, 13, 3735- 3756.

[4] ANAND A K, BARMAN S, PRAKASH N S, et al Vision based automatic landing of unmanned aerial vehicle Proc. of the International Conference on Information Technology and Applied Mathematics, 2019, 102- 113.

[5] TIAN C, HUANG C B An algorithm for unmanned aerial vehicle landing scene recognition based on deep learning and computational verbs Proc. of the IEEE International Conference on Civil Aviation Safety and Information Technology, 2019, 180- 184.

[6] LU C W, TSOUGENIS E, TANG C Kim proving object recognition with the l-channel Pattern Recognition, 2016, 49, 187- 197.

[7] ZHOU B L, LAPEDRIZA A, KHOSLA A, et al Places: a 10 million image database for scene recognition IEEE Trans. on Pattern Analysis and Machine Intelligence, 2018, 40 (6): 1452- 1464.

[8] XIAO J X, HAYS J, EHINGER K A, et al SUN database: large-scale scene recognition from abbey to zoo Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, 3485- 3492.

[9] XIAO J X, EHINGER K A, HAYS J, et al SUN database: exploring a large collection of scene categories International Journal of Computer Vision, 2016, 119 (1): 3- 22.

[10] PATTERSON G, XU C, SU H, et al The SUN attribute database: beyond categories for deeper scene understanding International Journal of Computer Vision, 2014, 108 (1/2): 59- 81.

# SRS DOCUMENT REVIEW

## CERTIFICATION

This Software Requirement Specification (SRS) Document is reviewed and certified to proceed for the project development by the Departmental Review Committee (DRC).

| | |
|---|---|
| **Date of SRS Submitted:** | |
| **Date of Review:** | |
| **Supervisor Comments:** | |
| **Supervisor Sign. & Date.** | |
| **Coordinator Sign. & Date** | |
| **HOD Sign. & Date** | |
| **Dept. Stamp** | |