



Tunisian Republic
Ministry of Higher Education and Scientific Research
University of Monastir
Faculty of Sciences of Monastir



Computer Science Department

Final Year Internship Project

Thesis submitted for the degree of

Professional Master's in Data Science

**Web Application for UAV Visual Navigation
Using Transformers and SLAM**

Internship completed at:

Avionav - NAVERA Department



Prepared by:

Ms. Hazar Haj Salem

Defended on 12 December 2025, before:

Chair : **Mr. Soufien GDAIM**

Examiner : **Mrs. Ghada Feki**

Company Supervisor : **Mr. Foued El Kamel**

Academic Supervisor : **Ms. Malika Jallouli**

Academic Year: 2024/2025

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

﴿وَقُلْ رَبِّ زِدْنِي عِلْمًا﴾ (طه: 114)

"And say, My Lord, increase me in knowledge." - Surah Taha, 20:114

To my parents, who shaped my heart with love, guided my steps with care, and stood
beside me through every season;

To my siblings, whose cheers lifted me, whose support held me through every low and
every rise, and who have been my first and closest friends since day one;

To my friends, who pushed me to grow, supported me through every challenge, and
shared every step of this journey;

To my professors, who guided me from the very beginning, patiently shared their
knowledge, and inspired me to grow;

With gratitude to Allah, who grants knowledge and strength,

This journey is dedicated to all of you who made it possible.

Acknowledgements

I would like to sincerely thank everyone who supported me during this internship and helped me complete this project.

First and foremost, I am truly grateful to my academic supervisor, **Madame Malika Jallouli**, for her guidance, encouragement, and thoughtful advice throughout this period. Honestly, her support and availability made a big difference, and I often felt reassured knowing I could rely on her expertise whenever I faced difficulties.

I also want to express my appreciation to the teaching staff at **the Faculty of Sciences of Monastir**. Their dedication and quality training provided me with the skills and confidence needed to tackle the challenges of this project and prepared me well for my professional journey.

Finally, I am deeply grateful to **Avionav** for allowing me to complete this internship.

This experience allowed me to apply my theoretical knowledge in a practical environment and gain valuable insights into UAV navigation and real-world challenges.

Abstract

This final internship project was carried out as part of the requirements for the professional master's degree in Data Science at the Faculty of Sciences of Monastir, and it took place over six months at Avionav, within the NAVERA department. The main goal of the project was, I would say, to develop a system to help Unmanned Aerial Vehicles (UAV) navigate by recognizing their surroundings from aerial images, especially in environments where GPS signals are unreliable.

The system integrates deep learning (DL) algorithms, specifically a Swin Transformer model, to classify aerial images of different Tunisian regions, and combines this with Simultaneous Localization And Mapping (SLAM) techniques to improve drone positioning and orientation estimation. Python-based image processing and augmentation techniques were also used to enhance the model's performance and ensure more accurate predictions.

Additionally, a web interface was developed to facilitate interaction with the system. This interface allows users to upload images, visualize classification results, and observe navigation outcomes, providing a practical way to interpret the system's predictions.

Overall, the project demonstrates how combining computer vision, deep learning, and SLAM can contribute to autonomous UAV navigation in complex environments.

Keywords: UAV navigation, DL, Swin Transformer, SLAM, Computer Vision, Image Processing, Web Interface, Autonomous Systems, Positioning, Aerial Images.

Résumé

Ce projet de stage de fin d'études a été réalisé dans le cadre des exigences du master professionnel en Science des Données à la Faculté des Sciences de Monastir, et s'est déroulé sur six mois chez Avionav, au sein du département NAVERA. L'objectif principal du projet était de développer un système d'aide à la navigation des drones en reconnaissant leur environnement à partir d'images aériennes, notamment dans les environnements où les signaux GPS sont peu fiables.

Le système intègre des algorithmes d'apprentissage profond (deep learning), plus précisément un modèle Swin Transformer, pour classifier des images aériennes de différentes régions tunisiennes, et combine cela avec des techniques SLAM pour améliorer l'estimation de la position et de l'orientation du drone. Des techniques de traitement et d'augmentation d'images basées sur Python ont également été utilisées pour améliorer les performances du modèle et garantir des prédictions plus précises.

De plus, une interface web a été développée pour faciliter l'interaction avec le système. Cette interface permet aux utilisateurs de télécharger des images, de visualiser les résultats de classification et d'observer les résultats de navigation, offrant ainsi un moyen pratique d'interpréter les prédictions du système.

Dans l'ensemble, le projet démontre comment la combinaison de la vision par ordinateur, de l'apprentissage profond et du SLAM peut contribuer à la navigation autonome des drones dans des environnements complexes.

Mots-clés : Navigation de drones, Apprentissage profond, Swin Transformer, SLAM, Vision par ordinateur, Traitement d'images, Interface web, Systèmes autonomes, Positionnement, Images aériennes.

Contents

General Introduction	1
1 General Context and Preliminary Study	3
Introduction	3
1.1 Presentation of the Company	3
1.2 Project Context	4
1.3 Problem Statement	5
1.4 Objectives of the Project	5
1.5 Applications and Related Work	6
1.5.1 Drone-STM-RENet	6
1.5.2 AGCosPlace	7
1.5.3 Transformer-Based Feature Segmentation and Region Alignment	8
1.6 Critical Analysis	9
1.7 Proposed Solution	10
1.8 Methodology	11
1.8.1 Phases of the Methodology	12
1.8.2 Project Planning	13
Conclusion	13
2 State of the Art	15
Introduction	15
2.1 Deep Learning	15
2.1.1 How Neural Networks Work	15
2.1.2 Convolutional Neural Networks (CNNs)	16
2.1.3 Transfer Learning	16
2.1.4 Computer Vision	17
2.1.5 Object Detection	17
2.1.6 Transformers	19
2.2 Simultaneous Localization and Mapping (SLAM)	20
2.3 Data Management and Analysis	21
Conclusion	22

3	Requirements Analysis and System Design	23
	Introduction	23
3.1	Identification of Actors	23
3.2	Specification of Requirements	24
3.2.1	Functional Requirements	24
3.2.2	Non-Functional Requirements	25
3.3	Use Case Diagram	25
3.4	Class Diagram	26
3.5	Sequence Diagrams	28
	Conclusion	32
4	Solution Implementation and Analysis	33
	Introduction	33
4.1	Data Acquisition and Preparation	33
4.1.1	Data Collection	33
4.1.2	Preprocessing and Augmentation	35
4.2	Model Implementation	36
4.2.1	Integration of Pretrained Models (Hierarchical Swin Transformer)	36
4.2.2	Model Training and Fine-Tuning	38
4.2.3	SLAM Integration	43
4.3	Web Application Development	43
4.3.1	Web Application Backend	43
4.3.2	Frontend Interface	44
4.4	Evaluation and Metrics	45
	Conclusion	46
5	Technical Implementation and Realization	47
	Introduction	47
5.1	Implementation and Evaluation	47
5.1.1	Development Environment	47
5.1.2	Libraries Used	48
5.1.3	Model Evaluation: Swin Transformer for UAV Navigation	49
5.1.4	SLAM Integration and Observations	52
5.2	Interface Overview	53
5.2.1	Authentication Interface	54
5.2.2	Main Dashboard Interface	55
5.2.3	Trajectory History and Management Interface	64
5.2.4	Administrator Panel	65
	Conclusion	70

CONTENTS

General Conclusion	71
Bibliography	73
Webography	75

List of Figures

1.1	Logo of Avionav	4
1.2	Waterfall Method	12
1.3	Gantt Chart	13
2.1	Architecture and Working of Swin Transformer[16]	19
3.1	Global Use Case Diagram	26
3.2	Global Class Diagram	27
3.3	Setup and Authentication Sequence Diagram	29
3.4	Frame Processing Loop Sequence Diagram	30
3.5	Results and History Management Sequence Diagram	31
4.1	Sample Captured Images	34
4.2	Region-zone mapping structure from the dataset configuration file	35
5.1	Training and validation performance of the hierarchical Swin Transformer model	50
5.2	Example Predictions of the Hierarchical Swin Transformer Model on UAV Images	52
5.3	Example of raw output (Python log)	53
5.4	Authentication Interface	54
5.5	Video Upload Section	55
5.6	Real-Time Processing Section	56
5.7	Performance Metrics Dashboard	57
5.8	Prediction Panel	57
5.9	Processing Console Log	58
5.10	2D Interactive View	59
5.11	3D Interactive View	60
5.12	Detailed Trajectories Windows	62
5.13	Trajectory History and Management Interface	64
5.14	Dataset Manager Interface	66
5.15	Model Trainer Interface	67
5.16	Training History Interface	68

List of Tables

2.1 Comparison of Object Detection Models 19

3.1 Functional Requirements 24

3.2 Non-Functional Requirements 25

4.1 Dataset Distribution Across Tunisian Regions 38

4.2 Model Training Configuration 40

5.1 Development Environment and Technology Stack 48

Acronyms

AR Augmented Reality. 17

CNN Convolutional Neural Network. 4

GPS Global Positioning System. 1, 4

RBAC Role-Based Access Control. 54

RPN Region Proposal Network. 17

SLAM Simultaneous Localization And Mapping. iii, 1, 2, 4

Swin Transformer Shifted Window Transformer. 19

UAV Unmanned Aerial Vehicle. iii, 1, 2, 4, 7

UML Unified Modeling Language. 2, 25

VR Virtual Reality. 17

YOLO You Only Look Once. 17, 18

General Introduction

In the field of aerial navigation, accurately locating drones represents a significant challenge, particularly in environments where GPS signals are unreliable or unavailable. While GPS systems perform well in open areas, they frequently fail in urban environments, dense forests, and mountainous regions, making sole reliance on GPS insufficient for ensuring reliable drone navigation across diverse operational conditions.

Recent advances in artificial intelligence and computer vision have enabled the integration of visual data with deep learning models and real-time positioning techniques, paving the way for more robust and autonomous navigation systems. This methodical approach, grounded in multi-source information analysis, empowers drones to recognize their surroundings and autonomously adjust their position, thereby improving both navigation accuracy and operational reliability.

Following this paradigm, the present project proposes an integrated solution combining aerial image classification through the Swin Transformer model with SLAM techniques to enhance UAV positioning in challenging environments. Conducted at Avionav within the NAVERA department, this work investigates how visual recognition and simultaneous localization and mapping synergistically support autonomous drone navigation, especially in GPS-denied areas. Additionally, a web-based interface was developed to facilitate system interaction, enabling users to upload videos, visualize predictions, and monitor navigation results in real time.

This report presents the development of an innovative web-based system for UAV visual navigation that integrates hierarchical classification and pose estimation capabilities. The document is structured into five chapters, each contributing to the comprehensive understanding and practical realization of the project:

Chapter 1: General Context and Preliminary Study: This chapter establishes the operational framework of aerial navigation challenges, introduces the host organization, and articulates the problem statement. It presents fundamental concepts and

reviews existing approaches, thereby contextualizing the research objectives that motivate this work.

Chapter 2: State of the Art: This chapter examines the theoretical foundations underlying vision-based UAV navigation, reviewing deep learning fundamentals, convolutional and transformer-based architectures, object detection methodologies, and SLAM systems that constitute the technological basis of the proposed solution.

Chapter 3: Requirements Analysis and System Design: This chapter delivers a comprehensive analysis of system requirements and design specifications, detailing functional and non-functional requirements, actor identification, and conceptual modeling through UML diagrams encompassing use cases, class structures, and sequence interactions.

Chapter 4: Solution Implementation and Analysis: This chapter addresses the detailed design of the intelligent navigation system, describing its overall architecture and constituent components. It justifies the selection of the Swin Transformer[1] model and SLAM integration strategy while outlining the methodological pipeline from data collection and preprocessing through model training and evaluation.

Chapter 5: Technical Implementation and Realization: This chapter illustrates the development phase of the web application, presenting the technical environment, development tools, and implementation technologies. It demonstrates the approaches adopted for trajectory visualization and result analysis, substantiated by interface demonstrations and performance evaluation metrics.

Chapter 1

General Context and Preliminary Study

Introduction

This chapter introduces the project and its general context. First, we present the company where the internship took place. We then explain the current solutions in use and the challenges they present. Next, we describe the proposed solution and explain how it helps solve the problem. Finally, we present the methodology followed during the project.

1.1 Presentation of the Company

Company Overview

Avionav is a Tunisian aircraft manufacturing company founded in 2007. It specializes in designing and producing light aircraft for civil and recreational aviation. The company is based in Sousse, Tunisia, and serves both local and international markets. Its planes are known for their lightweight structures, modern designs, and adaptability to various uses, including flight schools, tourism, surveillance, and personal aviation.

Avionav combines Italian aeronautical design expertise with local manufacturing, offering high-performance and fuel-efficient aircraft. The company operates under strict international aviation standards, emphasizing innovation, safety, and precision in its production process.

Through partnerships with aviation professionals and experts, **Avionav** aims to develop high-quality aircraft that are both accessible and technologically advanced.



Figure 1.1: Logo of Avionav

Areas of Activity

The main activities of Avionav include:

- Design and manufacture of light aircraft.
- Assembly and testing of aviation components.
- Research and development in aviation technology.
- Customization of aircraft based on client needs.
- Export of aircraft to international markets.

1.2 Project Context

This project aims to develop a visual navigation module for Unmanned Aerial Vehicle (UAV) that can determine its position without relying on Global Positioning System (GPS). The solution is based on computer vision algorithms, with a focus on Transformer models[2], which have recently demonstrated strong performance in image processing tasks. The goal is to replace traditional Convolutional Neural Network (CNN)-based approaches[3] with a more advanced Transformer architecture to improve the accuracy and robustness of UAV localization, especially at high altitudes.

The system will use aerial images captured by downward-facing cameras to detect unique landmarks, match them to known regions, and estimate the UAV's current position. The use of visual Simultaneous Localization And Mapping (SLAM) techniques will help create a dynamic and local map of the environment in real time.

In addition, the project includes steps for training, optimizing, and evaluating the performance of the Transformer model using real-world aerial datasets. The final deliverable will be an efficient visual navigation module integrated with the UAV's onboard systems, contributing to the NAVERA department's goal of creating a reliable GPS-independent navigation system.

1.3 Problem Statement

UAVs are increasingly used in various fields, including surveillance, mapping, agriculture, and rescue missions. However, their navigation systems still rely heavily on GPS data to determine their position. This dependency becomes a major limitation in environments where GPS signals are weak, blocked, or unavailable, such as inside buildings, under dense foliage, in urban canyons, or during high-altitude flights affected by atmospheric conditions.

In such scenarios, the lack of reliable positioning can lead to navigation errors, mission failure, or even loss of the UAV. Existing alternatives like inertial systems and camera-based localization using CNNs[3] provide some solutions but are often limited in precision, robustness, or real-time performance.

With the recent success of Transformer architectures in image classification and feature extraction[4], a new opportunity emerges to improve the visual positioning systems used in UAVs.

However, integrating these models into a real-time navigation system that works at high altitudes and in complex environments remains a challenging task. Transformer architectures suffer from quadratic computational complexity $O(n^2)$ in their self-attention operations, where n represents the number of image patches[1], making real-time processing demanding for UAV applications[4]. Additionally, high-altitude environments present challenges such as atmospheric effects and sparse visual features that can compromise localization performance[2].

Furthermore, effective navigation requires more than just identifying broad geographical areas. UAVs need to understand location at multiple levels of detail, from recognizing general regions to identifying specific local zones within those regions. This hierarchical understanding of location is essential for making precise navigation decisions in real-world scenarios.

The central problem of this project is, therefore:

How can we improve the accuracy and reliability of UAV localization in GPS-denied environments by using Transformer-based models that provide multi-level geographical understanding?

1.4 Objectives of the Project

Main Objective:

Develop a visual navigation system capable of enabling UAVs to determine their location without GPS by integrating aerial image recognition, SLAM techniques, and real-time

positioning, allowing drones to navigate autonomously in challenging environments.

Specific Objectives:

- Build a hierarchical classification model using the Swin Transformer architecture that recognizes both broad geographical regions and specific local zones within those regions, providing multi-level location understanding for precise UAV positioning.
- Design a real-time processing pipeline that integrates deep learning-based visual recognition with SLAM techniques to improve positioning accuracy and enable autonomous navigation in GPS-denied environments.
- Develop an interactive web interface to visualize hierarchical predictions, monitor navigation trajectories, and provide actionable insights for system evaluation and real-world deployment.

1.5 Applications and Related Work

Researching existing solutions is a key step in developing this project. It helps identify the strengths and weaknesses of current technologies and inspires the design of new approaches. In this section, we explore some relevant studies and architectures used in UAV navigation without GPS:

1.5.1 Drone-STM-RENet

The paper "Drone Navigation Using Region and Edge Exploitation-Based Deep CNN" (Arshad et al.2022) introduces Drone-STM-RENet, a CNN designed for autonomous UAV navigation in urban environments using a single forward-facing camera. It integrates region and edge information to enhance obstacle detection, steering angle prediction, and collision avoidance. Trained on datasets from car and bicycle driving scenarios, the model enables UAVs to learn safe navigation behaviors through imitation, achieving robust performance in complex settings without relying on complex sensors or pre-existing maps.

• Advantages:

- Improved Scene Understanding: By leveraging edge and region features, the model enhances obstacle detection and path planning, making it robust to lighting variations and complex textures.
- Single-Camera Efficiency: Eliminates the need for multiple sensors or pre-built maps, reducing hardware complexity and costs.

- Real-Time Capability: Demonstrates strong performance in steering angle prediction and collision avoidance, suitable for real-time autonomous flight.
- Adaptability: Effective in dynamic urban environments, with potential applications in surveillance, delivery, and exploration.

• **Limitations:**

- Suboptimal Trajectories: Generated paths may not always be the most efficient, leading to potential detours.
- Potential Divergence: The UAV may deviate from intended trajectories, increasing collision risks.
- Limited Generalization: Performance in non-urban or unstructured environments is untested, potentially limiting applicability.
- Computational Complexity: Processing regional and edge information increases computational demands, which may challenge resource-constrained UAV platforms.

1.5.2 AGCosPlace

The paper "AGCosPlace: A UAV Visual Positioning Algorithm Based on Transformer" (Arshad et al.2023) presents AGCosPlace, a Transformer-based algorithm for Unmanned Aerial Vehicle (UAV) visual positioning, aimed at improving navigation accuracy in GPS-denied environments. By processing visual data from onboard cameras, the algorithm leverages Transformer models to interpret complex visual cues, enabling precise localization and navigation for applications like search and rescue, surveillance, and environmental monitoring.

• **Advantages:**

- GPS-Independent Navigation: Reduces reliance on GPS, making it ideal for environments where signals are unreliable or unavailable (indoors or urban canyons).
- Enhanced Visual Processing: Transformer models improve the interpretation of complex visual data, leading to better scene understanding and localization accuracy.
- Versatile Applications: Suitable for critical missions requiring high autonomy, such as disaster response or infrastructure inspection.
- Robustness: Potentially resilient to environmental variations due to the Transformer's ability to model long-range dependencies in visual data.

• Limitations:

- High Computational Demand: Transformer models require significant processing power, which may strain onboard hardware and affect real-time performance.
- Environmental Sensitivity: Performance may degrade under extreme lighting conditions or visual obstructions, requiring further validation.
- Generalization Concerns: The algorithm's effectiveness across diverse terrains (rural or forested areas) needs more extensive testing.
- Training Complexity: Training Transformer models requires large, diverse datasets, which may pose challenges for deployment in varied scenarios.

1.5.3 Transformer-Based Feature Segmentation and Region Alignment

The paper "A Transformer-Based Feature Segmentation and Region Alignment Method for UAV-View Geo-Localization" (Dai et al.2022) introduces a Transformer-based method for UAV geo-localization, matching UAV-captured images with satellite references to estimate precise locations. It leverages Transformer models for feature segmentation and region alignment, achieving robust localization in complex environments. Evaluated on datasets like University-1652, it outperforms traditional CNN-based approaches, enabling applications in urban mapping and surveillance.

• Advantages:

- High Localization Accuracy: The Transformer's ability to model long-range dependencies enhances feature matching, leading to precise geo-localization even in complex scenes.
- Robustness to Viewpoint Variations: Effectively handles perspective differences between UAV and satellite imagery, improving performance in diverse environments.
- End-to-End Framework: Integrates feature segmentation and region alignment into a single trainable model, streamlining the localization process.
- Versatile Applications: Ideal for urban mapping, surveillance, disaster response, and autonomous navigation in GPS-denied settings.

• Limitations:

- **Computational Intensity:** Transformer models demand significant processing power, potentially hindering real-time performance on resource-constrained UAVs.
- **Dataset Dependency:** Requires high-quality, diverse datasets of paired UAV-satellite images for effective training, which may limit applicability in under-represented environments.
- **Environmental Sensitivity:** Performance may degrade in extreme conditions (heavy cloud cover, low visibility) that obscure visual features.
- **Scalability Challenges:** Real-world deployment may face constraints due to computational requirements and the need for extensive fine-tuning across varied scenarios.

1.6 Critical Analysis

The review of existing solutions reveals significant progress in visual navigation for UAVs, yet several limitations persist that justify the need for a new approach tailored to our specific context.

Limitations of CNN-based approaches: While systems like Drone-STM-RENet demonstrate effective obstacle avoidance and real-time performance, they suffer from suboptimal trajectory generation and limited generalization beyond urban environments. Their reliance on edge and region detection increases computational demands without guaranteeing robust performance across diverse terrains, particularly at high altitudes where visual features become less distinct.

Challenges with Transformer-based methods: Recent approaches such as AG-CosPlace and region-alignment models show promising improvements in localization accuracy through their ability to capture long-range dependencies in visual data. However, their high computational requirements pose significant challenges for deployment on resource-constrained UAV platforms. Additionally, these methods require extensive, high-quality training datasets that may not be readily available for all geographical regions, limiting their practical applicability.

Integration and deployment gaps: Most existing solutions focus either on obstacle avoidance or geo-localization independently, without providing a complete navigation pipeline. The integration of visual recognition with SLAM techniques remains underexplored, particularly for high-altitude scenarios where GPS alternatives are most needed.

Furthermore, few approaches address the practical concerns of model optimization, real-time inference, and lightweight deployment on actual UAV hardware.

Regional specificity: Current methods are predominantly trained and tested on datasets from urban environments in Europe, Asia, or North America. The effectiveness of these models in different geographical contexts, such as the diverse landscapes of Tunisia with its coastal, urban, and arid regions, remains uncertain and requires validation.

These limitations highlight the need for a solution that combines the strengths of Transformer-based feature extraction with practical considerations for real-time performance, regional adaptability, and integration into a complete navigation system. Our proposed approach aims to address these gaps by developing a lightweight, region-specific model optimized for high-altitude UAV navigation in GPS-denied environments.

1.7 Proposed Solution

To address the limitations discussed earlier, our project proposes a visual navigation system specifically designed for high-altitude UAV localization when GPS is unavailable or unreliable. The main idea is to use a Transformer model instead of traditional CNN architectures. This choice allows us to better understand spatial relationships in aerial images and more accurately distinguish between different geographical regions.

What makes our solution different: Our solution takes a different direction compared to existing methods by focusing on several key aspects:

- **Hierarchical recognition:** Instead of just identifying regions, our system recognizes both the general region AND the specific local zone within it. This two level approach provides much more precise location information.
- **Dual prediction heads:** The model has two separate output layers, one for regions and one for zones, allowing it to learn features at different geographical scales.
- **Custom training data:** We built our dataset using aerial images from different Tunisian regions, with each region divided into multiple zones. This ensures the system works well in our specific geographical environment.
- **Integration with SLAM:** The system works together with SLAM techniques to not only recognize regions but also build maps and track position dynamically in real-time.

Technical improvements to ensure reliability: To make sure our model works consistently and doesn't suffer from common problems like overfitting or poor performance on new data, we included several important techniques:

- **Data augmentation:** We use Albumentations to create variations of our training images changing lighting, rotating images, adjusting colors so the model learns to handle different conditions.
- **Better generalization:** We apply label smoothing[5] and dropout techniques[6] that help the model work well even on images it hasn't seen before.
- **Smart training process:** Early stopping prevents the model from training too long and memorizing data, while learning rate scheduling helps it learn efficiently.
- **Balanced training:** Weighted random sampling ensures all regions are represented fairly during training, preventing the model from becoming biased toward more common regions.

How it works in practice: The system takes UAV flight video as input and processes it frame-by-frame, with each frame analyzed individually to predict the corresponding region or location. This allows the UAV to determine where it is throughout its flight path using only visual information, without relying on GPS signals. The approach is designed to be both practical and scalable, making it suitable for real-world applications such as infrastructure inspection, emergency response operations, mapping missions, and surveillance tasks in GPS-denied environments.

By combining advanced Transformer technology with practical optimizations and region-specific training, our solution aims to provide reliable navigation support for UAVs operating in challenging environments where GPS-based systems are unavailable or unreliable.

1.8 Methodology

To ensure the success of our project and meet the technical expectations of the NAVERA department, we adopted the waterfall methodology [11], a structured, sequential approach to project management. This methodology was selected due to its linear and disciplined process, which is ideal for projects with well-defined requirements and minimal expected changes. Below are the key reasons for choosing the waterfall methodology:

- **Sequential Structure:** The waterfall model follows a clear, step-by-step process where each phase is completed before the next begins, ensuring a disciplined approach to development and clear visibility of progress.

- **Comprehensive Documentation:** Each phase produces detailed documentation, which supports traceability, facilitates technical validation, and ensures alignment with project objectives.
- **Rigorous Review Process:** Formal reviews at the end of each phase allow for early identification and resolution of issues, maintaining high quality throughout the project lifecycle.
- **Predictable Planning:** The structured nature of the waterfall model enables accurate estimation of timelines and resource requirements, supporting efficient project management and cost control.

The waterfall methodology is well-suited for our research-oriented project, providing a clear, systematic, and technically robust framework for implementation.

1.8.1 Phases of the Methodology

The project was executed through distinct phases, as outlined in the waterfall model[6]. The figure below illustrates these phases:

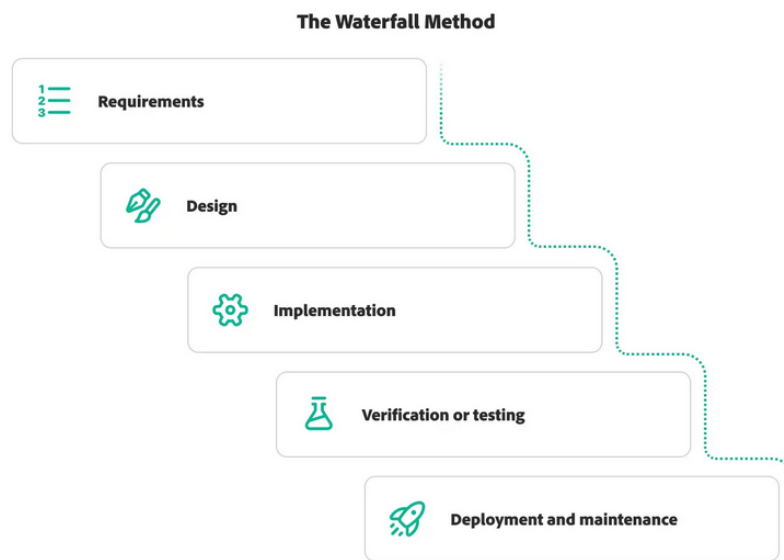


Figure 1.2: Waterfall Method

- **Requirements Analysis:** This initial phase is all about defining the project's goals. We work to understand all the project requirements upfront, including any constraints, costs, and timelines. The aim is to create a clear plan that everyone can agree on before we start building anything.

- **System Design:** Once we’ve defined the requirements, we move on to designing the technical solution. This involves creating a detailed blueprint of the system’s architecture. We decide how all the different parts will work together, what technologies we’ll use, and how the data will flow.
- **Implementation:** With the design complete, we get to the actual coding. This is where the developers build the software based on the plans we’ve created. Since we’ve already done all the careful planning, this phase is focused on execution.
- **Testing:** Before we release the product, it needs to be thoroughly tested. Our team verifies the software to ensure it functions correctly and meets all the original requirements. This step is crucial for providing a high-quality product for the end-user.
- **Deployment:** The final stage is releasing the finished software to the customer. After it’s launched, we move into the maintenance phase. This involves ongoing support, fixing any issues that come up, and releasing updates to keep the software running smoothly.

1.8.2 Project Planning

To ensure effective execution, we developed a detailed project schedule that distributes tasks across the internship period. The Gantt chart below outlines the timeline and dependencies of the project phases.

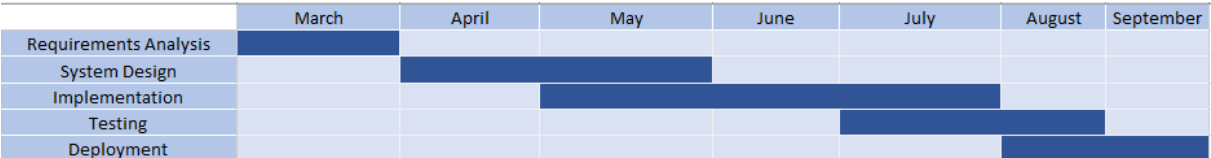


Figure 1.3: Gantt Chart

Conclusion

This first chapter served as an introduction to the overall framework of the project. We began with a presentation of the host company and its fields of activity, followed by a detailed explanation of the project context and its main objectives. We then addressed the problem statement, carried out a study of existing solutions, and presented the proposed approach designed to overcome their limitations. Finally, we introduced the chosen methodology, the waterfall model, and outlined the phases of project execution and planning.

In the next chapter, we will focus on specifying the system requirements, both functional and non-functional, and provide a conceptual overview of the system's architecture and components.

Chapter 2

State of the Art

Introduction

This chapter presents the current technologies and research related to drone navigation, visual recognition, and deep learning. Understanding these techniques helps design a robust visual navigation system that can estimate the drone's position without GPS.

We focus on deep learning models for image analysis, SLAM for mapping and localization, and tools for managing and analyzing drone data.

2.1 Deep Learning

Deep learning is a branch of machine learning, which itself is part of artificial intelligence. It is widely used for visual recognition and has proven very effective in analyzing images. For example, a deep learning model can identify objects, landmarks, or regions from aerial images. It can also extract features from complex scenes to help drones understand their environment. Deep learning relies on artificial neural networks[4], inspired by the human brain, to model and solve complex problems.

2.1.1 How Neural Networks Work

Neural networks are made up of layers of interconnected neurons that process input data to produce outputs. Each neuron receives signals, applies weights, combines them, and passes the result to the next layer.

As the network gets deeper:

- Early layers detect simple features like edges or colors.

- Deeper layers combine these to recognize complex shapes, like buildings, roads, or vegetation in aerial images.

Networks learn through backpropagation [7], a gradient-based optimization algorithm that adjusts their weights to minimize the difference between predictions and actual results. The basic weight update rule is:

$$w_{t+1} = w_t - \eta \frac{\partial L}{\partial w_t} \quad (2.1)$$

Where w_t represents the network weights at iteration t , η is the learning rate, and $\frac{\partial L}{\partial w_t}$ is the gradient of the loss function L with respect to the weights. Through this iterative process, the model progressively improves its predictions on new, unseen data.

Neural networks have become fundamental tools in computer vision [1, 8], autonomous navigation [3, 4], and robotics because of their ability to learn hierarchical representations and complex patterns from raw sensory data such as images.

2.1.2 Convolutional Neural Networks (CNNs)

CNNs are a type of neural network designed for images. They are commonly used for image classification, object detection, and segmentation.

Key parts of CNNs:

- **Convolutional layers:** Detect small patterns like edges and textures.
- **Pooling layers:** Reduce the size of the data while keeping important features.
- **Fully connected layers:** Combine features to make the final prediction.

CNNs are trained using backpropagation. Early layers detect simple patterns, and deeper layers identify complex objects. They are inspired by the visual system in animals and are very effective for image-based tasks.

2.1.3 Transfer Learning

Transfer learning uses a model that has already been trained on a large dataset and adapts it to a new task. This is useful when you don't have a lot of training data.

Two main methods:

- **Fine-tuning:** Adjust the weights of a pre-trained model to fit new data.
- **Feature extraction:** Use a pre-trained model to extract features and train a new classifier on them.

For this project, we fine-tuned a pre-trained Swin Transformer[1] on aerial images of Tunisian regions. This allowed the model to learn important features of each region while using knowledge from large image datasets.

2.1.4 Computer Vision

Computer vision allows computers to understand images. For drone navigation, it is used to:

- **Classify images**[3]: Identify the region or type of landscape in an image.
- **Detect objects**[9, 9, 8]: Locate landmarks, roads, or buildings.
- **Understand scenes**[4, 2, 1]: Help the drone navigate safely by recognizing the environment.

Computer vision is widely used in autonomous vehicles, robotics, and AR/VR[9][10]. In our project, it is key to making the drone "see" its environment.

2.1.5 Object Detection

Object detection is a crucial task in computer vision that allows machines to identify and locate important features in images. In our project, detecting landmarks in aerial images is essential for determining the drone's position and enabling autonomous navigation. By recognizing objects like roads, buildings, fields, and other landmarks, the drone can understand its environment more accurately and reduce reliance on GPS signals.

Modern object detection approaches are mainly based on deep learning. Two well-known methods are Faster R-CNN [1] and You Only Look Once (YOLO)[10]. Both are built on CNN architectures, but they have different trade-offs between speed and accuracy. While these models are effective, recent Transformer-based approaches like Swin Transformer and DETR[8] have shown superior performance in capturing global context and fine details in images, which makes them very promising for drone applications.

1. **Faster R-CNN :**

Faster R-CNN is a two-stage object detector introduced in 2015 by Shaoqing Ren and colleagues. It improves both speed and accuracy by combining a Region Proposal Network (RPN)[14] with a CNN-based detector.

Key components of Faster R-CNN:

- **RPN (Region Proposal Network):** Generates candidate regions in the image that might contain objects.

- **Shared Convolutional Layers:** These layers are used both by the RPN and the detection network to reduce computation.
- **RoI Pooling:** Aligns proposed regions into a fixed size for further processing.
- **Detection Network:** Classifies each proposed region and refines the bounding box coordinates.

Faster R-CNN achieves high detection accuracy but is computationally intensive, making it slower for real-time applications.

2. YOLO :

You Only Look Once (YOLO) is a single-stage detector that predicts object locations and classes in one pass. It divides an image into a grid and predicts bounding boxes and confidence scores for each cell.

Key features of YOLO:

- **Fast and efficient:** Can process images at very high speed, suitable for real-time detection.
- **Limitations:** The entire network is trained to minimize prediction errors, making it highly streamlined.

Although YOLO is popular for real-time tasks, it cannot fully capture long-range relationships in complex aerial scenes, which is critical for our drone's visual navigation.

3. Transformers for Object Detection:

Swin Transformers are Transformer-based models designed for computer vision tasks. They divide images into small patches and use self-attention to model both local and global relationships in the image..

Advantages for our project:

- **High Accuracy:** They can detect small or partially occluded landmarks more reliably than CNNs.
- **Global Context Awareness:** Useful for aerial images where objects may be far apart.
- **Robustness:** Effective under varying lighting conditions, angles, and environments.

Using a Swin Transformer allows our drone to detect landmarks reliably, improving navigation and localization without relying solely on GPS.

4. Model Comparison

Table 2.1: Comparison of Object Detection Models

Model	Type	Speed	Accuracy	Best Use
Faster R-CNN	Two-stage CNN	Slow (~10 FPS)	High	High-precision detection
YOLOv8	Single-stage CNN	Very Fast (40+ FPS)	Good	Real-time detection
Swin Transformer	Transformer	Moderate (~15 FPS)	High	Complex aerial scenes

Choice for our project: We selected Swin Transformers because their ability to capture global context and fine details makes them ideal for aerial landmark detection, providing a balance of accuracy and reliability for drone navigation.

2.1.6 Transformers

Transformers are a type of deep learning model that have revolutionized various fields, including natural language processing and computer vision. In computer vision, the Shifted Window Transformer (Swin Transformer) has emerged as a powerful architecture.

What is the Swin Transformer?

The Swin Transformer is a vision transformer model designed to process images efficiently. Unlike traditional convolutional neural networks (CNNs), which use fixed-size filters to scan images, the Swin Transformer divides an image into smaller patches and processes them using self-attention mechanisms. This approach allows the model to capture both local and global features of the image.

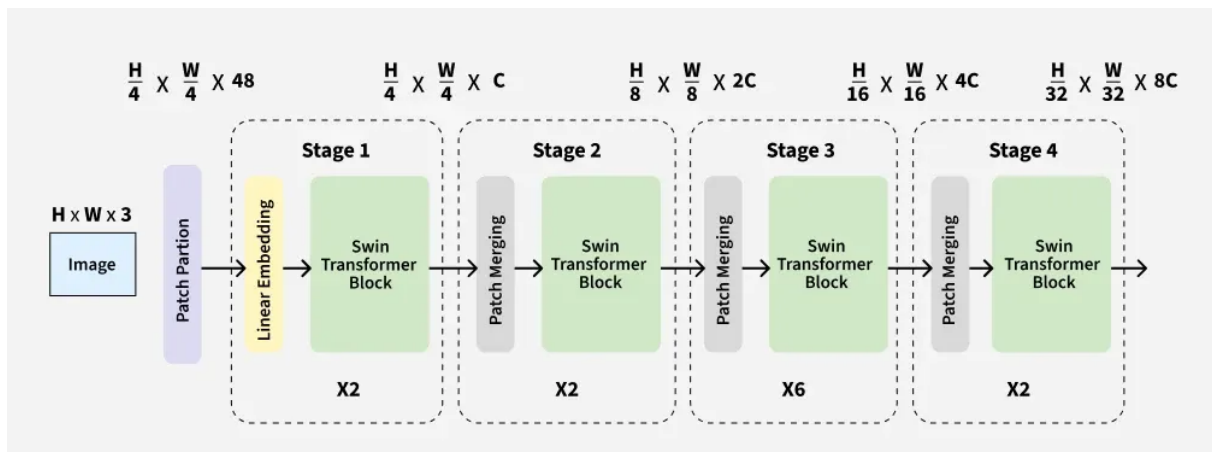


Figure 2.1: Architecture and Working of Swin Transformer[16]

Key Features of the Swin Transformer

1. **Hierarchical Structure:** The Swin Transformer processes images at multiple scales, gradually merging patches to capture features at different levels of abstraction.
2. **Shifted Windowing:** Instead of applying self-attention to the entire image, the model divides the image into non-overlapping windows. In each layer, these windows are shifted to allow information exchange between neighboring patches, enhancing the model's ability to capture global context.
3. **Efficient Computation:** By limiting self-attention to local windows and shifting them across layers, the Swin Transformer reduces computational complexity, making it scalable to high-resolution images.

Applications in Computer Vision

The Swin Transformer has demonstrated state-of-the-art performance in various computer vision tasks, including:

- **Image Classification:** Assigning labels to images based on their content.
- **Object Detection:** Identifying and locating objects within images.
- **Semantic Segmentation:** Classifying each pixel in an image into predefined categories.
- **Instance Segmentation:** Detecting objects and delineating their boundaries at the pixel level.

Its versatility and efficiency make the Swin Transformer a valuable tool in modern computer vision applications

2.2 Simultaneous Localization and Mapping (SLAM)

SLAM is a core technique for drone navigation when GPS is unavailable or unreliable. It allows a drone to simultaneously create a map of its surroundings and figure out its own position within that map. This makes it possible for drones to navigate in complex or unknown environments, such as urban areas, forests, or disaster zones.

Our SLAM pipeline works in three main steps:

1. **Feature Extraction:** The drone identifies distinctive points in images, such as edges, corners, or patterns. These points are easy to recognize in multiple frames, which helps track the drone's movement over time.

2. **Mapping:** Using the detected features, the drone combines consecutive images to gradually build a detailed map of the environment. This map is continuously updated as the drone explores new areas.
3. **Localization:** When the drone captures a new image, it compares the detected features with the existing map to estimate its exact position. This step ensures that the drone knows where it is, even without GPS signals.

By integrating SLAM with image classification using the Swin Transformer, the drone gains an extra layer of intelligence. The Swin Transformer can recognize landmarks, regions, and terrain patterns, helping the drone navigate more accurately. This combination allows reliable autonomous flight in areas where GPS signals are weak, blocked, or completely absent.

Using this approach, the drone does not rely solely on GPS, making it more robust, precise, and capable of operating in challenging environments where traditional navigation systems might fail.

2.3 Data Management and Analysis

Handling the images collected from Google Earth [5] is essential for training and improving the drone's visual navigation system. Proper management of these images ensures the system can reliably recognize landmarks and regions during flight.

1. **Structured Storage:** All images are organized into a dataset grouped by region or landmark. This organization makes it easy to access specific locations when training the model or testing its performance.
2. **Preprocessing and Augmentation:** Before using the images for training, they are preprocessed, resized, normalized, or adjusted to reduce noise. Augmentation techniques like rotation, flipping, and color changes are applied to increase dataset diversity. This helps the model handle different perspectives, lighting conditions, and angles, making it more robust when applied to real-world drone imagery.
3. **Analysis and Evaluation:** The dataset is regularly analyzed and visualized to monitor how well the model recognizes different regions and landmarks. This analysis helps identify gaps in the dataset, guide additional image collection, and refine training strategies.

By carefully managing images from Google Earth, the system can learn to accurately identify landmarks and regions, providing a strong foundation for autonomous drone navigation.

Conclusion

In this chapter, we reviewed key technologies for drone visual navigation. We explained deep learning, including CNNs and transfer learning, and explored object detection, highlighting models like Faster R-CNN, YOLO, and especially Transformers for their superior ability to capture complex patterns. We also covered machine learning approaches and dataset preparation, including images from Google Earth.

Overall, these methods provide a solid foundation for building an accurate and reliable autonomous drone navigation system.

Chapter 3

Requirements Analysis and System Design

Introduction

In this chapter, we analyse the functional and non-functional requirements of our Visual Navigation with Transformers. The system is designed to leverage Transformer architectures to improve visual localisation accuracy, replacing traditional CNN-based approaches. We also present the conceptual design through use cases and diagrams to model the interactions between different components of the system. This analysis clarifies the project requirements and provides a clear foundation for building and implementing the system.

3.1 Identification of Actors

The main actors who interact with our visual navigation web application are:

- **User(Operator):** Uploads UAV flight videos and uses the web interface to monitor real-time processing, view hierarchical predictions (region and zone), visualize flight trajectories in 2D/3D, check statistics, and save analysis results to history for future reference.
- **Admin (System Administrator):** Manages system maintenance and improvement by updating the training dataset with new aerial images, retraining the hierarchical Swin Transformer model, and monitoring overall system performance and accuracy.
- **System (Backend Processing):** Authenticates users, extracts video frames, processes each frame through hierarchical classification (region + zone prediction) and

SLAM pose estimation (position + orientation), fuses the results, streams real-time updates to the interface, and stores analysis data in the database.

3.2 Specification of Requirements

It is essential to clearly define both functional and non-functional requirements in order to effectively meet the operational goals of our project. These requirements guide the design and development of the visual navigation system and ensure that the final solution is reliable, efficient, and well-adapted to the intended use.

3.2.1 Functional Requirements

Functional requirements represent the essential actions and services that the system must provide to users. Our solution focuses on core capabilities: secure user authentication, video upload and frame processing, hierarchical geographic classification using the Swin Transformer model, SLAM-based pose estimation for trajectory reconstruction, real-time result streaming, interactive 2D/3D visualization, comprehensive flight statistics, historical analysis management, and administrative tools for dataset expansion and model retraining.

Table 3.1: Functional Requirements

Functionality	Description
User Authentication	Secure login via Firebase with role-based access (user/admin).
Video Upload & Processing	Upload UAV videos, extract frames, and preprocess (224×224 resize, normalization).
Hierarchical Classification	Dual-head Swin Transformer predicts region and zone with confidence scores.
SLAM Pose Estimation	Calculate UAV position (x, y, z) and orientation (roll, pitch, yaw) per frame.
Real-Time Results	Stream classification and pose data to UI during processing.
Trajectory Visualization	Interactive 2D/3D flight path display with geographic labels.
Statistics Dashboard	Show flight metrics: distance, duration, confidence scores, visited locations.
History Management	Save and retrieve past flight analyses via Firebase.
Admin Tools	Upload training images and retrain the classification model.

3.2.2 Non-Functional Requirements

Non-functional requirements define the quality, performance, and security criteria that the system must meet, independent of the proposed functionalities. For our solution, they primarily concern:

Table 3.2: Non-Functional Requirements

Requirement	Description
Performance	Frame processing <1s, model inference <100ms for near real-time analysis.
Reliability	Handle complete sessions without crashes, manage corrupted frames gracefully.
Security	HTTPS communication, encrypted authentication, user data isolation.
Availability	24/7 accessible web application with minimal downtime.
Scalability	Support expanding datasets without performance degradation.
Usability	Responsive UI, cross-browser compatible, clear error messages.

3.3 Use Case Diagram

Use case diagrams help us visualize how different actors interact with the system. Based on the Unified Modeling Language (UML) [7], these diagrams show the main functionalities and clarify who does what. They provide a clear overview of how the system works in practice.

Global Use Case Diagram

The global use case diagram 3.1 provides a complete visual overview of how users and admins interact with the UAV Visual Navigation System. It shows the main functionalities available to each actor and how different use cases connect and depend on each other through include and extend relationships.

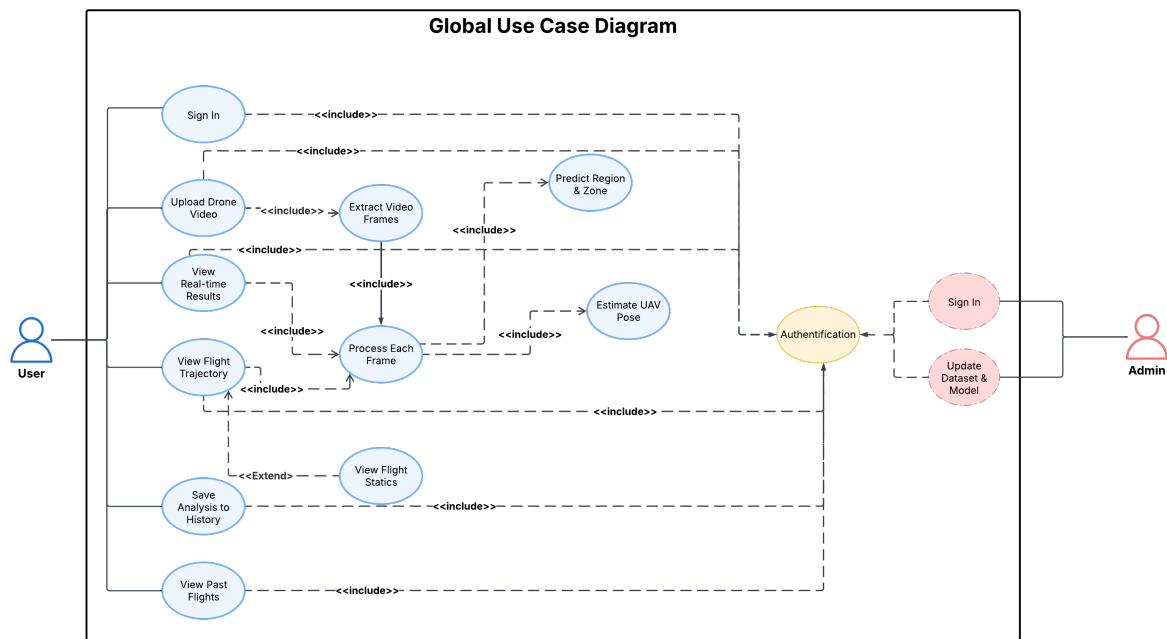


Figure 3.1: Global Use Case Diagram

3.4 Class Diagram

The class diagram is a crucial means of describing an object-oriented system. It provides a static view of the application, displaying the different classes, their attributes, and their methods. This type of diagram also helps to understand the relationships and dependencies between classes.

The following figure shows the class diagram designed for our credit management system.

Global Class Diagram

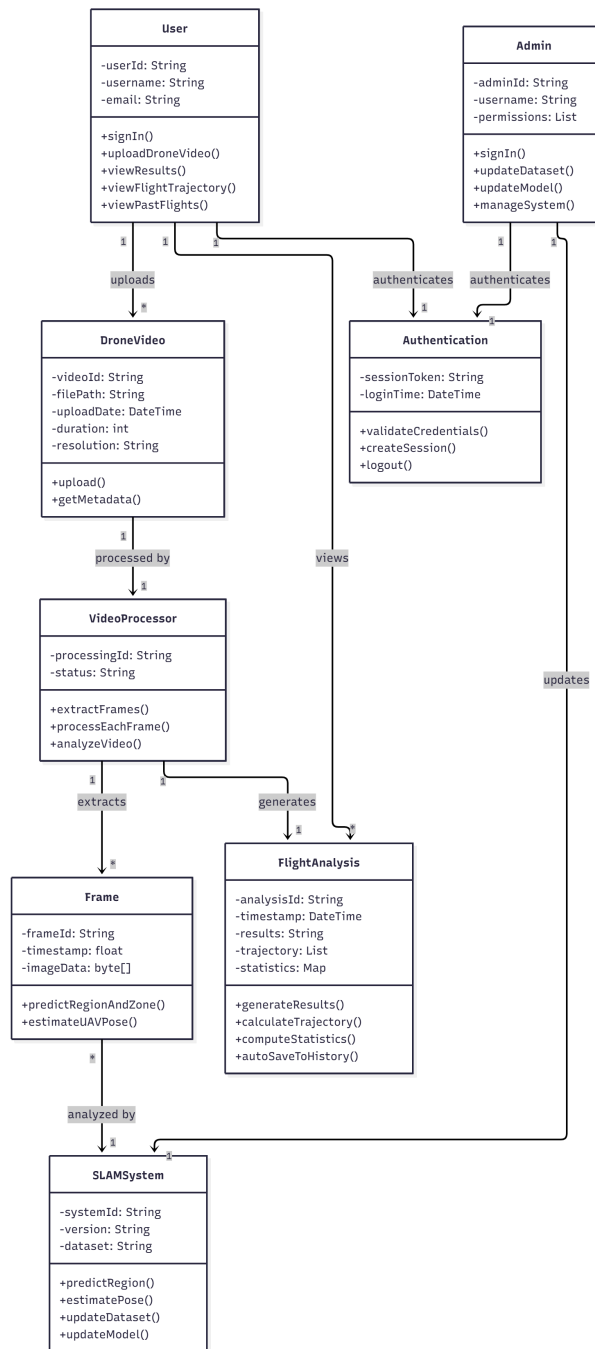


Figure 3.2: Global Class Diagram

Description

The global class diagram illustrates the main components of our UAV analysis system and their relationships:

- **User:** Represents the drone operator with attributes like userID, username, and

email. They upload drone videos, view real-time results, flight trajectories, and access past flight history.

- **Admin:** Represents system administrators who manage authentication, update datasets, and maintain the SLAM model to ensure optimal system performance.
- **Authentication:** Handles secure sign-in functionality for both Users and Admins, managing session tokens and credentials validation.
- **DroneVideo:** Stores uploaded video data with metadata such as videoID, file path, upload date, duration, and resolution.
- **VideoProcessor:** Core processing engine that extracts frames from videos, processes each frame sequentially, and coordinates the overall video analysis workflow.
- **Frame:** Represents individual video frames with attributes like frameID, timestamp, and image data. Each frame is analyzed for region prediction and UAV pose estimation.
- **SLAMSystem:** Manages the SLAM (Simultaneous Localization and Mapping) functionality, handling region prediction, pose estimation, and can be updated by administrators with new datasets and models.
- **FlightAnalysis:** Aggregates and stores analysis results including flight trajectory, real-time statistics, and automatically saves the analysis to history for future reference.

3.5 Sequence Diagrams

Sequence diagrams describe how users interact with the web application over time. They illustrate the sequence of messages exchanged during various processes. These diagrams help us understand the main steps of the application's features, such as registration, login, and credit management.

Setup and Authentication Sequence Diagram

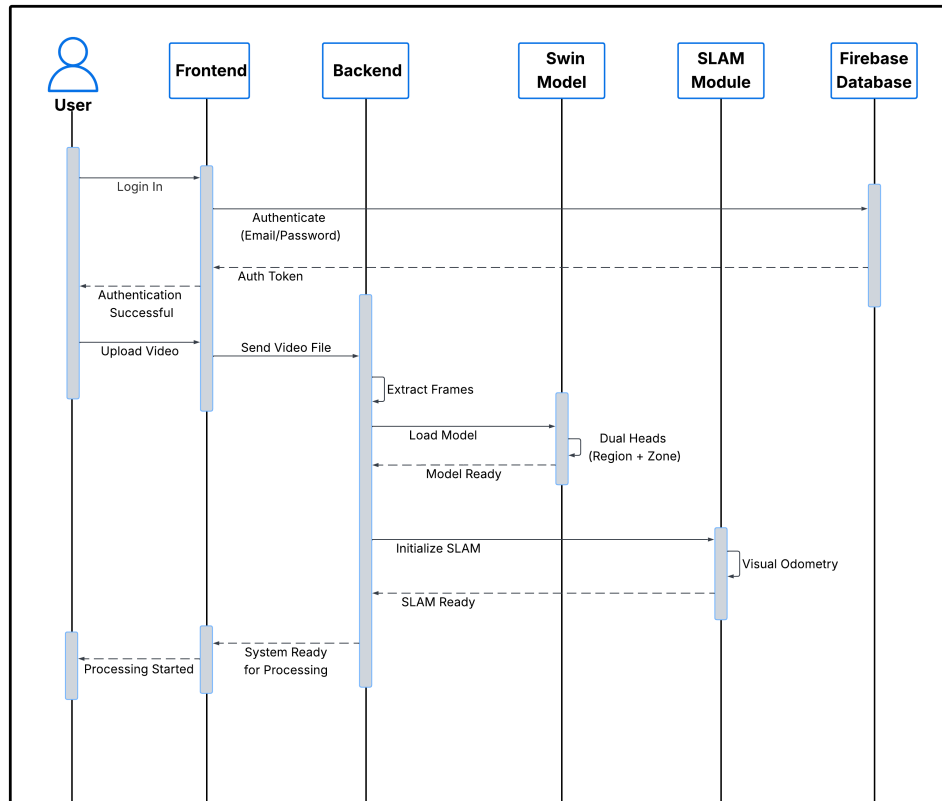


Figure 3.3: Setup and Authentication Sequence Diagram

Description : The diagram 3.3 shows how a user logs in and uploads a video to start the analysis process.

- User logs in through the frontend with their email and password
- Backend checks the credentials with Firebase Database and sends back an authentication token
- User uploads a video file which gets sent to the backend
- Backend breaks down the video into individual frames
- Swin Model starts up with two detection heads to identify regions and zones
- SLAM Module turns on to track camera movement and position
- System tells the frontend everything is ready to start processing

Frame Processing Loop Sequence Diagram

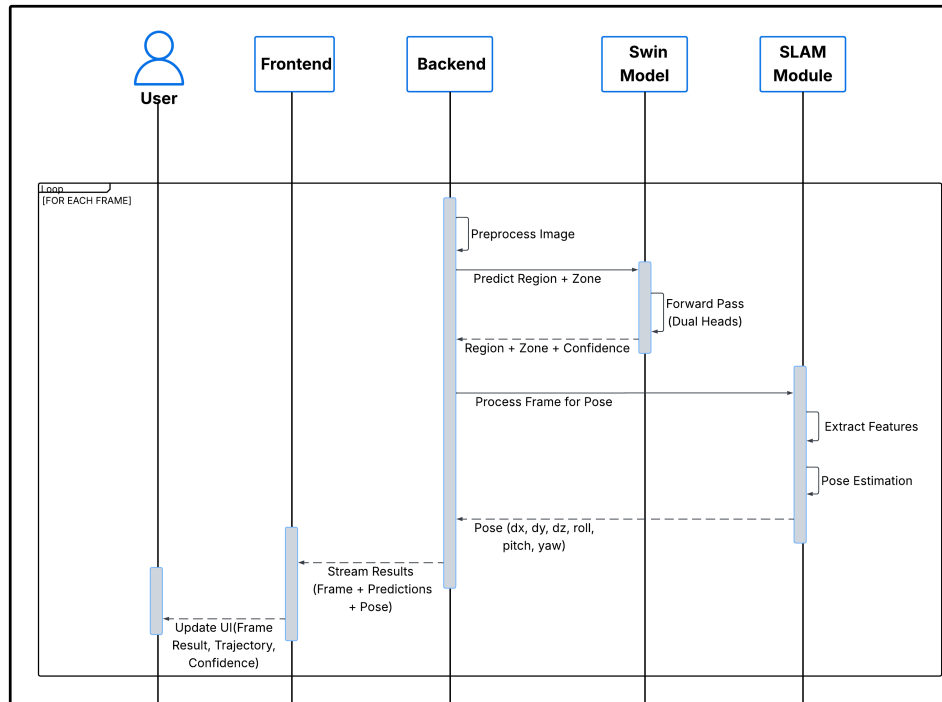


Figure 3.4: Frame Processing Loop Sequence Diagram

Description : The diagram 3.4 shows what happens to each video frame during the analysis.

- Backend goes through each frame one by one in a continuous loop
- Each frame gets cleaned up and prepared before analysis
- Swin Model analyzes the frame to detect the region, zone, and how confident it is
- SLAM Module looks at the frame to figure out camera features and position
- Position data includes movement and rotation in all directions (dx, dy, dz, roll, pitch, yaw)
- Results get sent back to the frontend as they happen
- Frontend updates the screen showing the current frame, flight path, and confidence scores

Results and History Management Sequence Diagram

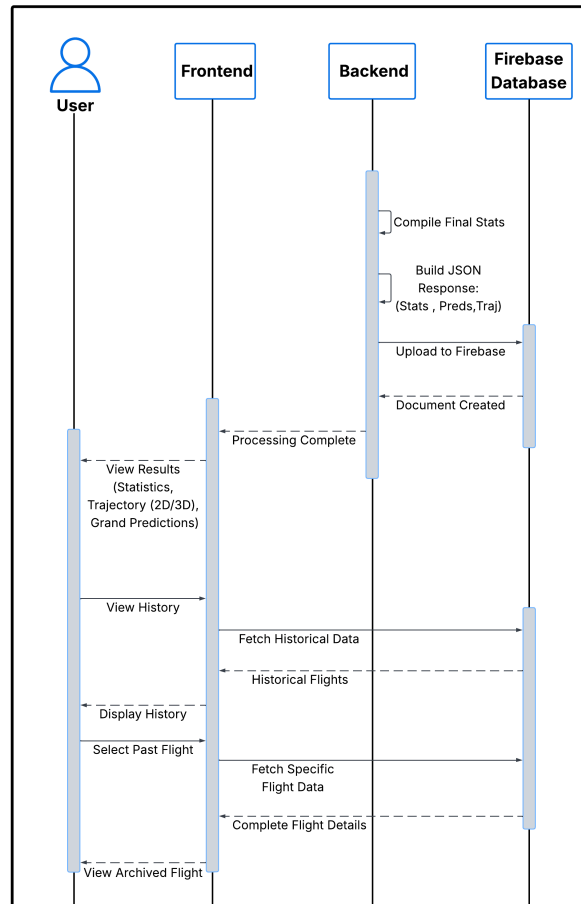


Figure 3.5: Results and History Management Sequence Diagram

Description : The diagram 3.5 shows how results are saved and how users can look at past flights.

- Backend puts together all the final statistics after processing finishes
- System creates a JSON file with all the stats, predictions, and flight path data
- Everything gets uploaded to Firebase Database and saved
- Frontend tells the user that processing is done
- User can see detailed results including charts, 2D/3D flight path, and landing predictions
- User can browse their flight history by pulling old records from Firebase
- Clicking on a past flight loads all the details so they can review it

Conclusion

In conclusion, this chapter has identified and analyzed the functional, non-functional, and technical requirements of our visual navigation system for UAVs. We presented the system's conceptual design using global and detailed use case diagrams, class diagrams, and sequence diagrams, illustrating the interactions between the UAV, the navigation module, and the operators. This formalization of requirements and processes provides a solid foundation for the system's development, ensuring a structured and coherent approach that supports the implementation and optimization of the Transformer-based navigation module.

Chapter 4

Solution Implementation and Analysis

Introduction

This chapter details the methodology used to develop our UAV visual navigation system. The goal is to enable GPS-free localization by combining aerial image recognition with SLAM-based motion estimation. From data acquisition to preprocessing, feature extraction with a Swin Transformer, SLAM integration, and the web-based interface, each stage contributes to accurate and real-time UAV navigation.

4.1 Data Acquisition and Preparation

Data acquisition and preparation represent the foundational phase of any data-driven system, directly influencing the accuracy and reliability of downstream processes. This critical stage encompasses the systematic collection of raw data from various sources, followed by comprehensive preprocessing operations including cleaning, normalization, and augmentation. Proper execution of these tasks ensures that the data fed into analytical models is of sufficient quality, consistency, and format to support robust and meaningful insights.

4.1.1 Data Collection

The training dataset for our UAV visual navigation system is entirely sourced from Google Earth imagery. Rather than relying on actual UAV flight data during the development phase, we adopt a systematic approach by first identifying specific landmarks and regions of interest, then collecting corresponding aerial images from Google Earth. This methodology offers several advantages: it provides access to high-quality, georefer-

enced imagery across diverse geographical areas, eliminates the need for costly and time-consuming initial flight campaigns, and enables the creation of a comprehensive labeled dataset before deployment

Google Earth Images for Training

1. **Region Selection:** Urban, rural, and natural areas are included to enhance dataset diversity.
2. **View Variation:** Different zoom levels and viewpoints are used to simulate multiple UAV altitudes and headings.
3. **Quantity and Balance:** Images are collected to maintain roughly similar counts per region, reducing class imbalance.
4. **Format and Storage:** Images are saved in PNG format and organized per region, supporting supervised training workflows.

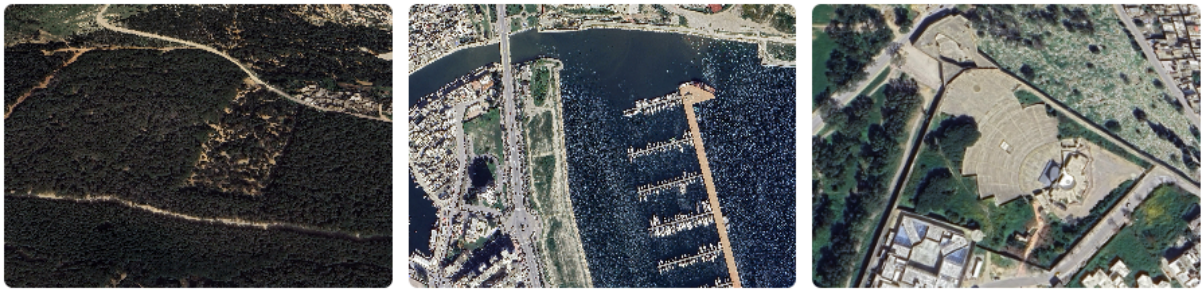


Figure 4.1: Sample Captured Images

Thus, this stage is more than just taking pictures. Capturing high-quality and diverse images sets the groundwork for pre-processing, feature extraction with Transformers, and accurate SLAM-based navigation.

Hierarchical Organization

Unlike typical classification datasets, our data is organized in a two-level hierarchy:

1. **Region Level:** Images are first grouped into folders by region. Each region represents a broad geographical area.
2. **Zone Level:** Within each region, images are labeled with specific zone names through filename prefixes. These zones represent local areas with distinct visual characteristics.

This structure is defined in a configuration file that maps each region to its possible zones. For example:

```
{
  "jendouba":
  [
    "Agricultural jendouba",
    "Ain Draham buildings",
    "Ain Draham forests",
    "Fernana",
    "Ghardimaou",
    "Jandouba Mountain",
    "Oued Medjerda",
    "Ruines el Hajra ez Zarga",
    "bulla regia",
    "centre ville"
  ],
  "Monastir":
  [
    "Airport",
    "Mausolee Bourguiba",
    "Medina",
    "Olive fields",
    "Ribat monastir",
    "Route vers Ksar Hellal",
    "Sahline Sebkha",
    "marina"
  ],
}
```

Figure 4.2: Region-zone mapping structure from the dataset configuration file

4.1.2 Preprocessing and Augmentation

Once collected, images require careful preprocessing to ensure that minor artifacts such as noise, uneven lighting, or motion distortions do not affect downstream recognition. This step is crucial for both model inference and any future SLAM-based localization.

Preprocessing Steps

1. **Noise Reduction and Edge Preservation:** Basic filters, like Gaussian blur, reduce unwanted noise, while edges are preserved because they often correspond to landmarks the system relies on.
2. **Normalization and Resizing:** The images are resized to a fixed resolution and normalized to a standard range. This helps the model treat all images fairly, reducing bias from scale or brightness differences.
3. **Label Normalization:** To handle real-world naming inconsistencies (accents, capitalization, typos), we apply automatic normalization that removes accents, converts

to lowercase, and uses fuzzy matching between filenames and expected zone names. This ensures robust label assignment even when file naming isn't perfectly consistent.

4. **Validation:** After preprocessing, images are checked for quality. Frames with severe blur, excessive noise, or other defects are removed to maintain dataset integrity. This step ensures that both the model and SLAM system receive only reliable input.

Augmentation Techniques

To improve model generalization and make the system robust to real-world flight variability, several augmentation techniques are applied:

- **Geometric Transformations:** Random horizontal and vertical flips simulate viewpoint changes.
- **Photometric Adjustments:** Random modifications in brightness, contrast, and saturation mimic different lighting conditions.
- **Blur and Occlusion Simulation:** Gaussian blur and coarse dropout are applied to imitate mild motion blur or partial occlusions, which can occur during UAV flight.
- **Consistency for Model Input:** All augmented images are converted into tensors with standardized shapes and channels, ready for inference.

Together, preprocessing and augmentation prepare the images for robust feature extraction. They ensure that the web application receives clean, consistent inputs, allowing the Swin Transformer to produce accurate region predictions in real time. Furthermore, these steps indirectly support SLAM-based localization by reducing the risk of feature-tracking errors caused by noisy or inconsistent frames.

4.2 Model Implementation

4.2.1 Integration of Pretrained Models (Hierarchical Swin Transformer)

After images are collected and preprocessed, the next step is to extract features and make predictions using a specially adapted Swin Transformer. Unlike standard classification models that predict a single class, our model needs to predict two things simultaneously: the broad region and the specific local zone.

Why Hierarchical Architecture?

Aerial images contain features at multiple scales:

- Large-scale features: Roads, coastlines, urban density patterns that identify regions
- Fine-grained features: Building types, vegetation patterns, infrastructure that identify zones

A single classification head cannot effectively capture both levels. Therefore, we use a hierarchical architecture with shared feature extraction but separate prediction heads.

Model Architecture

1. **Shared Backbone:** We use a pretrained Swin-Base Transformer as the feature extractor. This backbone processes the input image through multiple attention layers, learning both local and global visual patterns. The pretrained weights come from ImageNet, giving us a strong starting point.
2. **Dual Classification Heads:** Instead of a single output layer, we add two separate classification heads:
 - **Region Head:** A fully connected layer that predicts which of the 19 Tunisian regions the image belongs to. This head learns to recognize broad geographical patterns.
 - **Zone Head:** A separate, fully connected layer that predicts the specific local zone within the region. The number of possible zones varies by region, so this head must handle all zone classes across all regions.
3. **Regularization:** Both heads include dropout layers ($p=0.3$) to prevent overfitting, especially important given our moderate dataset size.

How Predictions Work

1. **Input Preparation:** A preprocessed image (224×224 pixels) is converted to a tensor and passed to the model.
2. **Feature Extraction:** The Swin Transformer backbone processes the image, producing a rich feature representation that captures both coarse and fine details.
3. **Parallel Prediction:** The feature representation is fed to BOTH classification heads simultaneously:
 - Region head outputs probabilities for 19 regions

- Zone head outputs probabilities for all possible zones

4. Output Handling: The model returns:

- Predicted region with confidence score
- Predicted zone with confidence score
- Both predictions are logged and sent to the frontend for visualization

By using this dual-head architecture, the system provides operators with multi-level location information, knowing both region and zone gives much more precise navigation guidance than knowing just the region alone.

4.2.2 Model Training and Fine-Tuning

While we start with a pretrained Swin Transformer, fine-tuning on our specific Tunisian aerial image dataset is essential for accurate region recognition. This section describes the training process, configuration, and techniques used to adapt the model to our navigation task.

Dataset Overview

Our training dataset consists of aerial images from 19 different Tunisian regions collected from Google Earth. The dataset is split into training and testing sets with the following distribution:

Table 4.1: Dataset Distribution Across Tunisian Regions

Region	Train	Test	Region	Train	Test
Beja	161	40	Kairouan	127	32
Bizerte	287	72	Kasserine	82	21
Jendouba	246	62	Kebili	87	21
Kef	162	40	Mahdia	79	20
Medenine	108	27	Monastir	118	29
Sfax	55	14	Siliana	114	28
Sousse	118	30	Tataouine	188	47
Djerba	91	23	Gabes	160	40
Gafsa	126	32	Tozeur	247	61
Tunis	104	26			
Total	2,960	725			

The dataset exhibits class imbalance, with some regions like Bizerte having 287 training images while others like Sfax have only 55. This imbalance is addressed through weighted sampling and augmentation techniques described below.

Data Preprocessing and Augmentation

To improve model robustness and handle the class imbalance, we apply extensive data augmentation using the Albumentations library:

- **Geometric Transformations:**

- Resize to 256×256 pixels
- Random crop to 224×224 pixels (Swin Transformer input size)
- Horizontal flip with probability 0.5
- Vertical flip with probability 0.3

- **Photometric Adjustments:**

- Random brightness and contrast adjustment ($\pm 20\%$, $p=0.5$)
- Simulates different times of day and weather conditions

- **Blur and Occlusion Simulation:**

- Gaussian blur with kernel size 3-5 ($p=0.3$) to simulate motion blur
- Coarse dropout: randomly drops 8 rectangular patches (8-16 pixels) with probability 0.3 to simulate partial occlusions

- **Normalization:**

- Applied ImageNet statistics (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
- Ensures compatibility with pretrained Swin Transformer weights

These augmentation techniques artificially expand the training set and force the model to learn invariant features that generalize well across different viewing conditions.

Training Configuration

The following configuration was used for fine-tuning the Swin Transformer:

Table 4.2: Model Training Configuration

Parameter	Value
Base Model	Swin-Base (microsoft/swin-base-patch4-window7-224)
Model Parameters	88M parameters
Architecture	Hierarchical dual-head classifier
Number of Classes	19 Tunisian regions
Number of Zone Classes	Varies by region (total across all regions)
Training Samples	2,960 images
Validation Samples	725 images
Training Epochs	20 epochs (with early stopping)
Batch Size	32 images per batch
Image Input Size	224×224 pixels
Optimizer	AdamW
Learning Rate	3×10^{-5}
Weight Decay	1×10^{-4}
Loss Function	Hierarchical weighted loss (30% region + 70% zone)
Learning Rate Scheduler	ReduceLROnPlateau (patience=2, factor=0.5)
Early Stopping Patience	17 epochs
Dropout Rate	0.3 (before classification layer)
Hardware	Google Colab Tesla T4 GPU (16GB VRAM)

Model Architecture Adaptation

The pretrained Swin-Base model is adapted for our 19-class classification task:

1. **Load Pretrained Weights:** The model is initialized with weights pretrained on ImageNet-1K, providing strong general-purpose visual features.
2. **Replace Classification Head:** The original 1000-class classifier is replaced with a new head consisting of:
 - Dropout layer ($p=0.3$) for regularization
 - Linear layer mapping to 19 output classes (Tunisian regions)
3. **Fine-Tune All Layers:** Unlike some transfer learning approaches that freeze the backbone, we fine-tune all layers with a relatively low learning rate (3×10^{-5}). This allows the model to adapt its feature representations to aerial imagery characteristics while preserving useful pretrained knowledge.

Hierarchical Loss Function

Training a hierarchical model requires a specialized loss function that balances learning at both levels:

$$\text{Loss} = 0.3 \times \text{Region_Loss} + 0.7 \times \text{Zone_Loss} \quad (4.1)$$

Why this weighting?

- Region prediction is relatively easier (only 19 classes, visually distinct areas)
- Zone prediction is harder (many more classes, subtler visual differences)
- For navigation, knowing the exact zone is more important than just knowing the region

Both losses utilize cross-entropy with label smoothing ($\varepsilon = 0.1$) to prevent overconfidence and enhance generalization.

During training:

1. Each image is processed through the backbone
2. Region head predicts region $\rightarrow$ calculate region loss
3. Zone head predicts zone $\rightarrow$ calculate zone loss
4. Losses are combined with 30/70 weighting
5. Backpropagation updates all layers to minimize the combined loss

This approach ensures the model learns features useful for both coarse (region) and fine-grained (zone) classification simultaneously.

Addressing Class Imbalance

Given the significant class imbalance in our dataset (ranging from 55 to 287 samples per region), we implement weighted random sampling:

- **Weight Calculation:** For each training sample, a weight is computed as the inverse of its class frequency:

$$w_i = \frac{1}{n_{c_i}} \quad (4.2)$$

where n_{c_i} is the number of samples in the class of sample i .

- **Weighted Random Sampler:** During each epoch, samples are drawn with replacement using these weights, ensuring that underrepresented regions (like Sfax with 55 images) appear as frequently as overrepresented ones (like Bizerte with 287 images).

- **Label Smoothing:** Cross-entropy loss with label smoothing (factor=0.1) prevents the model from becoming overconfident, especially important for imbalanced datasets.

Training Process and Regularization

The training process incorporates several techniques to prevent overfitting and ensure stable convergence:

1. **Dropout Regularization:** A dropout layer ($p=0.3$) is added before the final classification layer to reduce co-adaptation between neurons and improve generalization.
2. **Weight Decay:** L2 regularization (weight decay= 1×10^{-4}) is applied through the AdamW optimizer to penalize large weights.
3. **Learning Rate Scheduling:** ReduceLROnPlateau scheduler monitors validation loss and reduces the learning rate by factor 0.5 when no improvement is seen for 2 consecutive epochs. This enables fine-grained optimization as training progresses.
4. **Early Stopping:** Training automatically stops if validation loss doesn't improve for 7 consecutive epochs, preventing the model from overfitting to the training set.
5. **Best Model Checkpoint:** The model weights producing the lowest validation loss are saved and used for final deployment.

Training Monitoring

Throughout training, the following metrics are tracked for each epoch:

- **Training Loss:** Cross-entropy loss on the training set
- **Validation Loss:** Cross-entropy loss on the held-out test set
- **Validation Accuracy:** Classification accuracy on the test set
- **Learning Rate:** Monitored to observe scheduler adjustments

These metrics guide training decisions and provide early indicators of overfitting or convergence issues. The complete training results and performance analysis are presented in Chapter 5 (Implementation and Results).

The fine-tuned model successfully adapts the pretrained Swin Transformer to aerial imagery of Tunisian regions, achieving strong validation performance despite the moderate dataset size and class imbalance. The trained model is then deployed in the FastAPI backend for real-time inference during UAV flights.

4.2.3 SLAM Integration

While the current system primarily focuses on UAV image processing through the web module, SLAM is an essential component for full GPS-free navigation. Even though it was not fully implemented in this stage, it is important to describe its role and how it would interact with the web-based processing hub.

Role of SLAM in the system:

1. **Pose Estimation:** SLAM estimates the UAV's motion between consecutive frames by tracking visual features or using direct photometric methods. This provides relative position information when GPS is unavailable.
2. **Map Construction:** As the UAV moves, SLAM builds a local map of the environment. These maps can help recognize previously visited regions and improve navigation accuracy.
3. **Integration with Swin Transformer Outputs:** The region predictions from the Swin Transformer serve as high-level priors for SLAM. For instance, when a predicted region matches a previously visited area, the system can attempt loop closure, correcting accumulated drift in the UAV's pose estimate.
4. **Data Fusion:** The SLAM module would combine odometry and region recognition cues to produce a more stable and globally consistent position estimate.

In the current implementation, the web module provides region predictions that could later support a SLAM backend. Even though SLAM is not yet fully integrated, these predictions already offer valuable cues for future pose estimation and loop closure. This suggests that, once combined, SLAM could significantly improve UAV localization accuracy and reduce drift, highlighting the potential benefits of fusing visual recognition with motion estimation.

4.3 Web Application Development

4.3.1 Web Application Backend

The backend serves as the central processing hub for the UAV visual navigation system, managing both region recognition and SLAM-based pose estimation. FastAPI is used for its efficiency, ease of development, and ability to handle asynchronous requests, which is essential when processing multiple images and real-time pose data simultaneously.

Key responsibilities of the backend include:

1. **Image Reception and Storage:** The backend receives images from UAV flights and stores them in a structured directory format by flight session or region. This organization supports both model inference and SLAM processing, which relies on the temporal order of frames.
2. **Preprocessing and Model Inference:** Images are preprocessed, resized, normalized, and lightly cleaned before being passed to the Swin Transformer. The model outputs region predictions with confidence scores, which are then fused with SLAM to refine UAV pose estimates.
3. **SLAM Processing:** Visual odometry and map-building computations are performed in real time. Region predictions from the model act as priors for loop closure, improving localization accuracy and reducing drift.
4. **Result Logging and Monitoring:** Both prediction outputs and SLAM pose estimates are logged with timestamps and metadata. This ensures full traceability and enables operators to monitor system performance continuously.
5. **Task Management:** Images and SLAM computations are queued to prevent bottlenecks, ensuring reliable sequential processing during high-throughput UAV flights.

Overall, the backend integrates image processing, region recognition, and SLAM pose estimation into a unified hub, providing reliable, real-time navigation data while allowing the UAV to focus on safe flight operations.

4.3.2 Frontend Interface

The frontend presents UAV positions, SLAM-based pose estimates, and model predictions to operators in an interactive way. React handles dynamic content, while Leaflet provides real-time map visualization.

Main features include:

1. **Real-Time Map Visualization:** UAV positions, predicted regions, and SLAM pose estimates are displayed on an interactive map.
2. **Prediction Display and Confidence Indicators:** Each UAV image shows the predicted region and confidence score, while SLAM provides an additional metric of localization certainty.
3. **Flight Monitoring and Management:** Operators can review flight paths, access stored images, and manage sessions. Historical data visualization allows performance assessment over multiple missions.

4. **Alerts and Notifications:** Low-confidence predictions, unusual outputs, or inconsistencies in pose estimation are highlighted for operator attention.

Together, the backend and frontend provide a fully integrated system where UAV images, model predictions, and SLAM pose estimates are processed, fused, and visualized in real time, ensuring accurate navigation and operator oversight.

4.4 Evaluation and Metrics

Evaluating a visual navigation system is never just a mechanical step. It feels more like a process of deciding which aspects of performance truly matter and then measuring them in a way that reflects real-world navigation. In our case, the evaluation looks at two complementary dimensions: recognition performance using the Swin Transformer and localization stability using the SLAM module.

1. **Classification Accuracy**

The first measure is straightforward: we check how often the system predicts the correct region or landmark. This is reported as accuracy on a held-out validation set, but accuracy alone can be misleading. Some regions are visually more complex or less represented in the dataset, so we also examine **precision, recall, and F1-score per region**, as well as a **confusion matrix**. The confusion matrix is particularly insightful because it highlights which regions are frequently confused with others, guiding future dataset improvements or augmentation strategies.

2. **Localization Accuracy**

Since the ultimate goal is reliable navigation, it is not enough for the system to simply classify images correctly. We also look at how well the integrated SLAM module estimates the UAV's trajectory. Pose error is typically reported as absolute trajectory error (ATE), which compares the estimated UAV path to a ground-truth trajectory (for example, from GPS or motion capture during testing). Another useful metric is relative pose error (RPE), which captures how much drift accumulates between consecutive frames. Together, these measures tell us whether the system can maintain a stable, accurate estimate of the UAV's position over time.

3. **Qualitative and Visual Analysis**

Ultimately, there is value in simply examining the results. Numbers are essential, but they do not always tell the whole story. Visualizing a confusion matrix can quickly reveal patterns of misclassification that raw metrics might hide. Similarly, plotting estimated trajectories against ground truth makes it immediately clear where SLAM is working well and where it is drifting off course. When needed, drift

plots help distinguish between slow, systematic errors (which might be acceptable or correctable) and sudden, erratic failures (which are riskier for navigation).

In short, evaluation is not merely a final checklist but a continuous process of measuring, interpreting, and iterating. By combining classification accuracy with trajectory-level metrics and visual analysis, we obtain a balanced view of how well the system works and where it can still be improved.

Conclusion

In conclusion, this chapter has presented a methodology for UAV visual navigation, combining careful image collection, preprocessing, and augmentation with feature extraction using a pretrained Swin Transformer. While SLAM integration remains in progress, the system's backend and frontend ensure real-time processing, prediction, and visualization. Overall, this approach supports accurate, GPS-free navigation and provides a solid foundation for further improvements and integration of motion-based localization.

Chapter 5

Technical Implementation and Realization

Introduction

In this final chapter, we carefully present how the proposed UAV visual navigation system was implemented and evaluated. The goal here is not only to describe the technical steps, but also to reflect on how the design choices shaped the final system. We explain the software tools and libraries that made the system possible, the structure of the back-end and front-end, and how SLAM and deep learning were integrated. Finally, we discuss how we tested the solution and measured its performance.

5.1 Implementation and Evaluation

5.1.1 Development Environment

The development of this project required a carefully selected technology stack combining modern frameworks, powerful libraries, and efficient tools. Table 5.1 presents the main technologies used throughout the implementation phase, detailing their specific purposes and the reasons behind their selection.

Table 5.1: Development Environment and Technology Stack

Category	Tool	Usage and Justification
Primary Language	 Python	Reference language for data science and machine learning. Used for data processing, model implementation, API integration, and interface development.
Backend Framework	 FastAPI	Modern web framework for building APIs. Chosen for its speed, automatic documentation generation, and seamless integration with machine learning models.
Frontend Framework	 React	JavaScript library for building interactive user interfaces. Provides components for image upload, prediction visualization, and real-time display of navigation results.
Development IDE	 Visual Studio Code	Complete integrated development environment with advanced debugging, Git support, Python package management, and profiling tools. Facilitates collaborative development and code maintenance.
Training Platform	 Google Colab	Cloud-based Python environment with free GPU access. Used for training and fine-tuning the Swin Transformer model on aerial image datasets.
Data Collection	 Google Earth	Platform for collecting high-resolution aerial images of Tunisian regions to build the training dataset.
Version Control	 Git & GitHub	Tools for version control and source code management. Enable smooth team collaboration with precise real-time change tracking and branch management.

5.1.2 Libraries Used

Python Libraries

- **torch**: Deep learning framework for loading and running the Swin Transformer model.
- **torchvision**: Provides pre-trained models and image transformations for computer vision tasks.
- **opencv-python**: Computer vision library for image preprocessing, feature extrac-

tion, and SLAM integration.

- **numpy**: Numerical computing library for array operations throughout the image processing pipeline.
- **albumentations**: Image augmentation library for applying transformations during training to improve model robustness.
- **fastapi**: Web framework for building the REST API.
- **uvicorn**: ASGI server for running the FastAPI application.
- **pillow**: Python Imaging Library for image loading and basic manipulations.
- **joblib**: Tool for model serialization and parallel processing operations.

C++ Libraries

- **RTAB-Map**: Core SLAM library providing visual odometry, loop closure detection, and mapping capabilities.
- **OpenCV**: Used for feature detection (ORB, SIFT) and image processing within the SLAM module.
- **Eigen**: Linear algebra library for pose estimation and geometric computations.
- **CMake**: Cross-platform build system for compiling C++ modules and managing Python integration.

5.1.3 Model Evaluation: Swin Transformer for UAV Navigation

The evaluation of our hierarchical Swin Transformer model, trained for both region and zone classification, shows strong performance across both tasks. Figure 5.1 presents four key metrics over 20 epochs: combined loss, region accuracy, zone accuracy, and learning rate schedule.

Training Progression:

The combined loss curves show steady convergence from 4.5 to near 1.0, with training and validation losses remaining close throughout, indicating healthy generalization without overfitting. The model reached its best performance at epoch 19, marked by the red dashed line.

Hierarchical Accuracy:

Since we're training two separate classification heads, we track two accuracies:

- **Region Classification:** The model rapidly distinguished between the 19 Tunisian regions, reaching over 95% accuracy by epoch 10 and plateauing near 99-100% by epoch 20. This quick convergence makes sense regions have distinct large-scale visual features like urban density and coastal patterns.
- **Zone Classification:** Fine-grained zone prediction was more challenging, but still achieved 90-95% validation accuracy by epoch 19. The model successfully learned subtle differences between zones within the same region, such as downtown versus industrial areas.

The learning rate remained constant at 3×10^{-5} , indicating the initial configuration was well-tuned and required no adjustments.

Multi-Level Confidence:

This hierarchical approach provides operators with reliable positioning at multiple scales. When both accuracies are high, the system offers precise location data. When zone confidence is lower, high region accuracy still provides dependable coarse positioning.

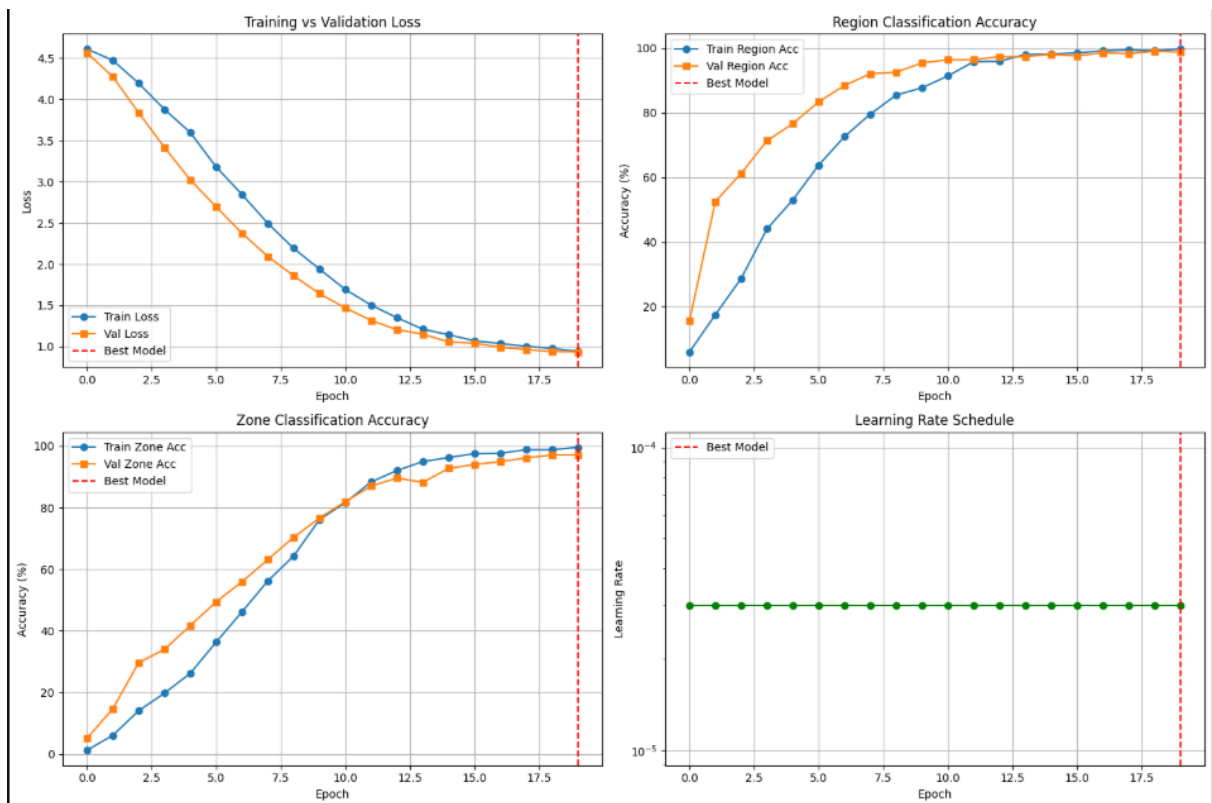


Figure 5.1: Training and validation performance of the hierarchical Swin Transformer model

Test Results on UAV Aerial Images

After training, we tested the model on new UAV images not included in the training dataset:

- **Hierarchical Prediction Accuracy:** The model correctly classified regions in over 99% of test images and achieved 90%+ zone-level accuracy, even with complex patterns or partial views. This confirms the dual-head architecture effectively learns features at both geographical scales.
- **Inference Performance:** Processing was highly efficient preprocessing took approximately 1079.6 ms per image, while dual-head inference required only 63.2 ms. The hierarchical architecture adds minimal computational overhead, making it suitable for near real-time predictions.
- **Robustness to Variations:** Both region and zone predictions maintained reliability under different angles, lighting conditions, and minor motion blur. Notably, even when zone confidence dropped below 80%, region predictions typically remained above 95%, providing a fallback positioning level.

The figure 5.2 shows hierarchical predictions from an aerial view of Beja region.

The model simultaneously predicts both geographical levels:

- **Region:** Beja (88.5% confidence) broad geographical area
- **Zone:** Ain Gham (62.1% confidence) specific local area within Beja

This dual-level output provides operators with multiscale location information. The higher region confidence (88.5%) compared to zone confidence (62.1%) is typical, region classification is generally easier due to more distinct large-scale features. Even when zone confidence is moderate, the high region confidence provides reliable coarse positioning. These hierarchical predictions can be combined with SLAM outputs to enhance GPS-free navigation accuracy.



Figure 5.2: Example Predictions of the Hierarchical Swin Transformer Model on UAV Images

In summary, the evaluation confirms that our hierarchical Swin Transformer is well-suited for UAV visual navigation. The model provides a strong foundation for GPS-free positioning with multi-level location understanding, offering both broad regional context and precise zone identification.

5.1.4 SLAM Integration and Observations

The SLAM module in our system uses RtabmapWrapper Python binding to figure out where the UAV is and which direction it's facing, just by looking at images. It works together with the Swin Transformer, so we can navigate and analyze at the same time.

Here's how it works: when we upload an image, the backend converts it into a format the SLAM module can read, then calls `process()` to get the pose data. This data tells us where the UAV is (x, y, z coordinates) and how it's tilted (roll, pitch, yaw angles). The frontend receives this information as JSON, along with the predicted region, so operators can see both the location and the classification at once.

The pose estimates change smoothly from one frame to the next, showing how the UAV is moving. The module doesn't need GPS at all - it only uses what it sees in the images and optical flow, which makes it reliable for GPS-free navigation. How accurate it is depends on things like image quality, how many features are in the scene, and lighting. In areas with not much texture or poor lighting, the estimates might be noisier, but looking at multiple frames in a row gives us stable trajectories for consistent navigation. Overall, combining visual pose estimation from SLAM with AI-based region predictions

really improves autonomous UAV navigation. The system gives us accurate positioning, real-time feedback, and a solid foundation for flying without GPS.

1. SLAM Output Capture and UAV Trajectory

In this section, we show the results from the SLAM module integrated with our UAV system. We want to capture the UAV's position (dx, dy, dz) and orientation (roll, pitch, yaw) for each frame and visualize its path.

(a) Capturing SLAM Output

Here we present the final SLAM results captured during a UAV flight. The main goal is to see the UAV's position in both 2D and 3D space using the SLAM module, without relying on GPS. The pose includes dx, dy, and dz (in meters), along with roll, pitch, and yaw angles.

The raw SLAM output for frames 575–588 is shown below:

```
[575] Pose: (-0.429, 3.582, 3.150), dx=-0.429, dy=3.582
[576] Pose: (-0.433, 3.576, 3.155), dx=-0.433, dy=3.576
[577] Pose: (-0.398, 3.544, 3.161), dx=-0.398, dy=3.544
[578] Pose: (-0.392, 3.581, 3.166), dx=-0.392, dy=3.581
[579] Pose: (-0.340, 3.448, 3.172), dx=-0.340, dy=3.448
[580] Pose: (-0.300, 3.369, 3.177), dx=-0.300, dy=3.369
[581] Pose: (-0.305, 3.136, 3.183), dx=-0.305, dy=3.136
[582] Pose: (-0.268, 2.956, 3.188), dx=-0.268, dy=2.956
[583] Pose: (-0.268, 2.956, 3.194), dx=-0.308, dy=2.887
[584] Pose: (-0.224, 2.721, 3.199), dx=-0.224, dy=2.721
[585] Pose: (-0.270, 2.608, 3.205), dx=-0.270, dy=2.608
[586] Pose: (-0.270, 2.608, 3.210), dx=-0.303, dy=2.537
[587] Pose: (-0.251, 2.339, 3.216), dx=-0.251, dy=2.339
[588] Pose: (-0.251, 2.339, 3.221), dx=-0.232, dy=2.240
```

Figure 5.3: Example of raw output (Python log)

5.2 Interface Overview

In this section, we show the main interfaces of our UAV visual navigation system. We designed these interfaces to be simple and practical, so users can easily interact with the system and see results in real time.

5.2.1 Authentication Interface

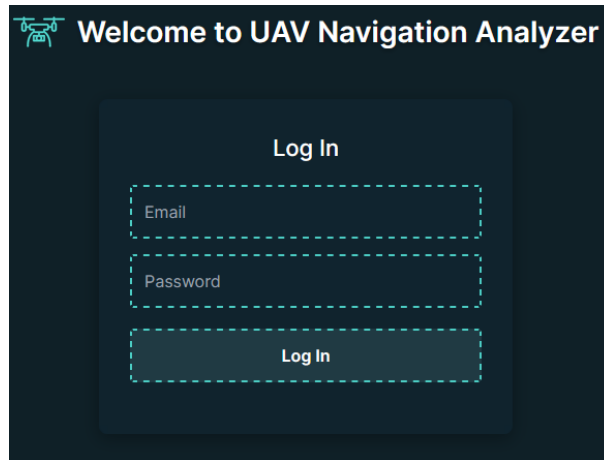


Figure 5.4: Authentication Interface

The system implements a secure authentication mechanism with Role-Based Access Control (RBAC), providing two distinct user privilege levels through a clean, minimalist login interface.

User Roles and Access Levels

The authentication system distinguishes between two types of accounts:

1. Standard User Role:

Regular users can access the complete UAV visual navigation analysis toolkit, including:

- Video upload and processing
- Real-time trajectory visualization
- Flight path analysis and metrics
- Classification results monitoring
- Historical trajectory review and comparison

2. Administrator Role:

Administrator accounts inherit all user-level functionalities while gaining exclusive elevated privileges:

- All Standard User Features: Complete access to analysis, visualization, and history tools

- Dataset Management: Ability to upload new training datasets for system enhancement
- Model Training: Rights to initiate and manage machine learning model training procedures
- Training History Access: Ability to view and monitor training session records

Authentication Process

The login interface provides:

- Secure email/password authentication
- Role-based session management
- Automatic privilege assignment based on account type
- Consistent dark-themed aesthetic matching the application's technical monitoring environment

5.2.2 Main Dashboard Interface

The main dashboard serves as the central control panel for UAV video analysis, presenting three primary functional sections in a modular, card-based layout optimized for technical monitoring.

Video Upload Section

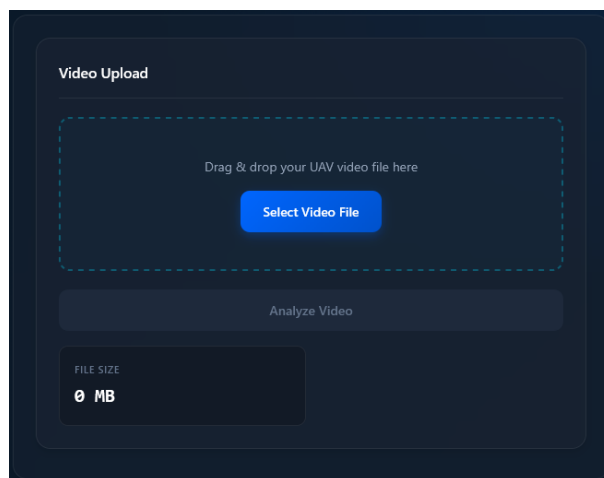


Figure 5.5: Video Upload Section

The section 5.5 enables users to initiate analysis by uploading UAV flight recordings:

- Drag-and-drop interface for intuitive file selection
- Supported formats: MP4, AVI, MOV video files
- File size display: Real-time indication of uploaded video size
- Analyze Video button: Initiates the classification and trajectory analysis pipeline

Real-Time Processing Section

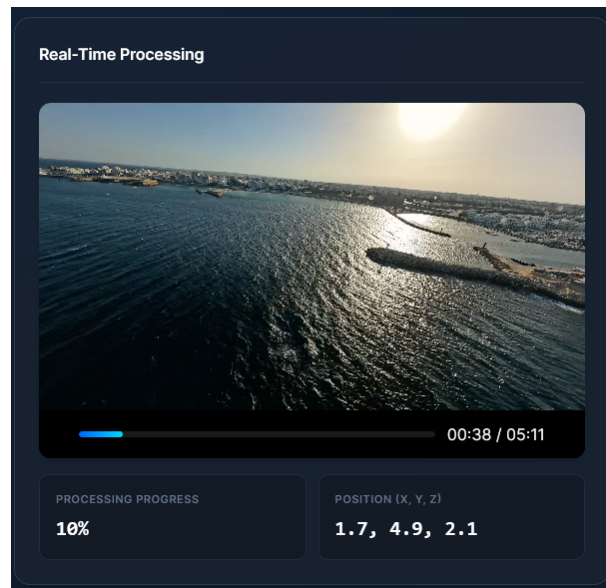


Figure 5.6: Real-Time Processing Section

Once a video is uploaded and being analyzed, the Real-Time Processing Section 5.6 displays the aerial images being processed along with important contextual information:

- Video Visualization: Shows the video being processed with a time progress indicator
- Processing Progress: Percentage of how much has been analyzed
- Current Position: Coordinates (X, Y, Z) estimated by SLAM for the current frame

This interface updates dynamically during processing, allowing users to watch the analysis progress in real-time.

Performance Metrics Dashboard

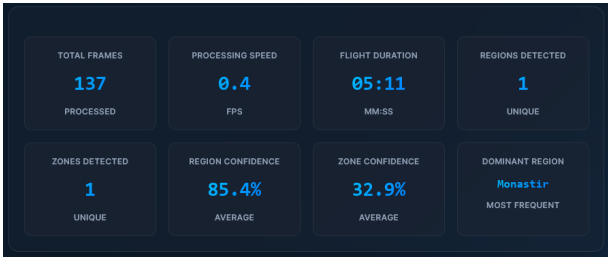


Figure 5.7: Performance Metrics Dashboard

The metrics dashboard 5.7 presents eight key performance indicators in a grid layout, providing instant visibility into processing performance and classification quality:

Processing Statistics:

- Total Frames Processed: Running count of analyzed video frames
- Processing Speed: Real-time FPS measurement indicating system performance
- Flight Duration: Calculated total flight time in MM:SS format

Detection Metrics:

- Regions Detected: Number of unique geographical regions identified
- Zones Detected: Count of distinct zones within regions
- Region Confidence: Average confidence level for region predictions
- Zone Confidence: Average confidence level for zone-specific predictions
- Dominant Region: Most frequently identified location throughout the flight

Prediction Panel

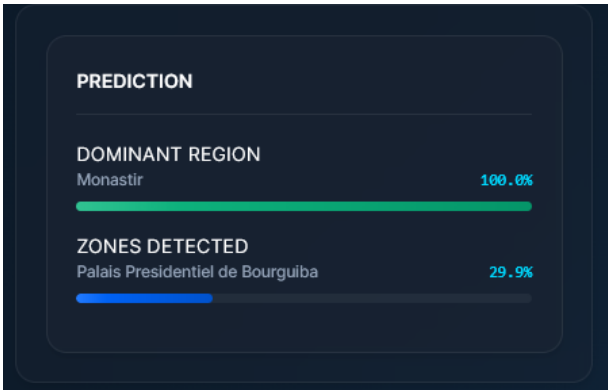


Figure 5.8: Prediction Panel

The prediction panel 5.8 presents current classification results with visual confidence indicators:

- Dominant Region: Displays the primary identified geographical region with a horizontal progress bar showing prediction strength
- Zones Detected: Lists specific landmarks or zones within the region, each with its corresponding confidence percentage displayed as a progress bar

Processing Console Log

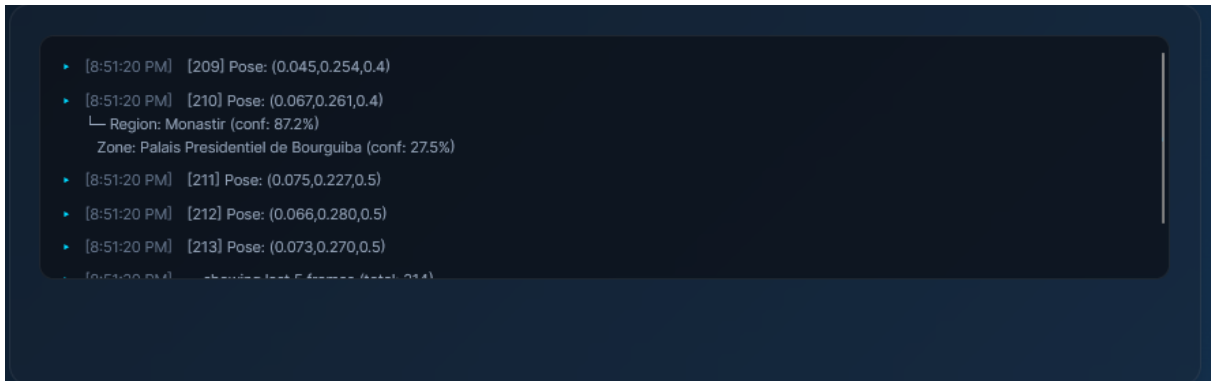


Figure 5.9: Processing Console Log

The prediction panel 5.9 provides a scrollable, frame-by-frame technical record of the analysis process in real-time.

Each log entry contains:

- Timestamp: Precise time of frame processing in [HH:MM:SS PM] format
- Pose Estimation: 3D coordinates representing the estimated UAV position (e.g., "Pose: (0.065,0.192,0.8)")
- Region Classification: Identified geographical region with confidence percentage
- Zone Classification: Specific landmark detection within the region with confidence level

This chronological log maintains a complete processing history, enabling operators to:

- Trace the classification decision-making process for each frame
- Identify patterns in region/zone recognition over time
- Review confidence levels throughout the flight
- Debug potential anomalies or misclassifications
- Validate system performance against known flight paths

Trajectory Visualization Section

The system provides two complementary visualization modes for analyzing UAV flight paths, available both in the main dashboard (5.10, 5.11) and as expandable full-screen views (5.12). Users can switch between modes and access detailed analysis through the "View in new tab" option.

Flight Data Metrics All visualization modes display consistent flight information:

- Total Distance: Cumulative path length traveled by the UAV
- Data Points: Number of SLAM pose estimates recorded during the flight
- Status: Real-time processing indicator showing "Live" during active analysis
- View Mode: Current visualization type (2D or 3D)

2D Trajectory View



Figure 5.10: 2D Interactive View

The 2D view presents a horizontal projection showing UAV movement along the X (East) and Y (North) axes.

Visual Elements:

- Color coded Path: The trajectory line transitions from red (start) through orange/yellow (middle) to green (end), providing temporal context at a glance
- Start Point: Green circle marking trajectory origin
- End Point: Yellow/orange circle at final position
- Cardinal Grid: N, S, E, W directional markers with X-Axis (East, red) and Y-Axis (North, green) reference lines
- Grid Background: Provides scale reference for distance estimation

Hierarchical Location Labels: Geographic predictions appear as text annotations along the path:

- Region labels (e.g., "Monastir")
- Zone labels (e.g., "Palais Presidentiel de Bourguiba")
- Labels positioned near their corresponding trajectory segments

Interactive Controls:

- Drag to pan: Move the view to explore different areas
- Scroll to zoom: Adjust magnification level
- Auto-scaled on load: Automatically fits the entire trajectory in view

Best Used For:

- Quick assessment of horizontal coverage area
- Measuring straight-line distances between geographic points
- Identifying region and zone boundary transitions
- Rapid overview of flight patterns

3D Interactive View

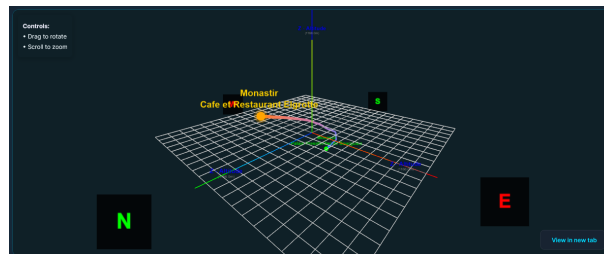


Figure 5.11: 3D Interactive View

The 3D view provides a perspective representation that includes altitude variations, offering complete spatial understanding of the flight trajectory.

Visual Elements:

- 3D Grid Floor: Provides depth perception and ground plane reference
- **Multi-Colored Flight Path:**
 - Green segments: Initial flight phase
 - Purple/magenta segments: Mid-flight trajectory

- Red/orange segments: Approach and final positioning

- **Coordinate System:**

- X-Axis (East): Red axis marker labeled "E"
- Y-Axis (North): Green axis marker labeled "N"
- Z-Axis (Altitude): Blue vertical axis labeled "Z (Altitude)"
- Cardinal Directions: N, S, E, W positioned around grid perimeter

- Start Point: Green sphere at trajectory origin
- End Point: Orange sphere marking final UAV position

Hierarchical Location Annotations:

Geographic predictions are rendered as 3D text labels floating above their corresponding positions:

- Region displayed in yellow
- Zone displayed in yellow/orange
- Labels maintain readability as the camera rotates

Interactive Controls:

- Drag to rotate camera: Orbit around the trajectory for different viewing angles
- Scroll to zoom in/out: Adjust viewing distance
- Auto-scaled to fit view: Automatically frames the entire flight path

Best Used For:

- Analyzing altitude profiles and vertical maneuvers
- Understanding three-dimensional flight behavior
- Validating terrain clearance and obstacle avoidance
- Visualizing takeoff, landing, and approach patterns
- Comprehensive spatial analysis including elevation changes

Full-Screen Detailed View

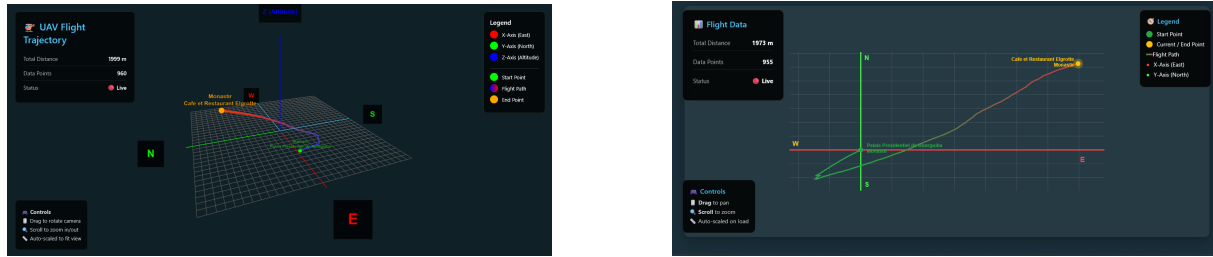


Figure 5.12: Detailed Trajectories Windows

By clicking "View in new tab", an expanded full-screen window opens with enhanced visualization area and additional analytical capabilities.

Enhanced Features:

- Larger Viewing Area: Maximized screen space for detailed trajectory examination
- Comprehensive Legends: Color-coded indicators for all trajectory elements (start point, flight path, end point, coordinate axes)
- Precise Metrics: More visible display of total distance, data points, and processing status
- Improved Label Readability: Hierarchical geographic annotations with better spacing and contrast

Same Interactive Controls:

- All pan, zoom, and rotation controls function identically to the dashboard views
- Seamless switching between 2D and 3D modes within the full-screen interface
- Real-time updates continue during live flight processing

Trajectory Enrichment with Hierarchical Classification

Both visualization modes integrate SLAM-derived positions with hierarchical geographic predictions:

- Spatial-Semantic Fusion: Each trajectory point combines geometric coordinates (X, Y, Z from SLAM) with semantic location labels (Region + Zone from the hierarchical classifier)
- Dynamic Labeling: As the UAV moves through different areas, region and zone labels update to reflect the most recent predictions

- Multi-Level Context: Operators see both broad geographical context (e.g., "Monastir" region) and precise local detail (e.g., "Cafe et Restaurant Elgrotte" zone)

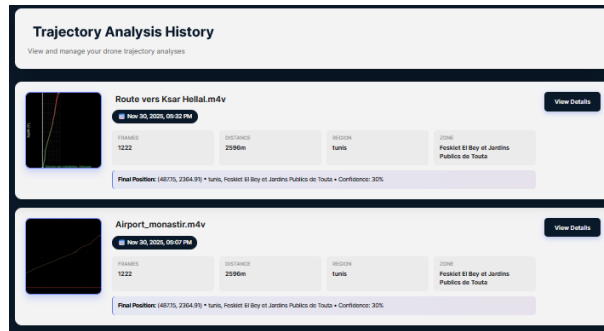
Analytical Capabilities

These visualization tools enable operators to:

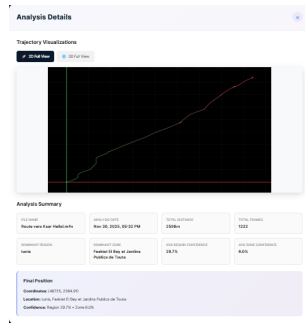
- Validate Navigation Performance: Compare intended vs. actual flight paths
- Assess Geographic Coverage: Verify which regions and zones were successfully visited
- Analyze Movement Patterns: Examine altitude changes, direction transitions, and velocity variations
- Identify Anomalies: Detect unexpected deviations, erratic movements, or navigation errors
- Evaluate Classification Consistency: Check if hierarchical predictions remain stable along trajectory segments or show frequent transitions
- Support Mission Planning: Use historical trajectories to plan future flights and optimize coverage

By combining geometric trajectory data from SLAM with semantic location information from the hierarchical classifier, the system provides operators with complete understanding of both where the UAV flew (coordinates) and what locations it recognized (regions and zones), enabling effective GPS-free autonomous navigation.

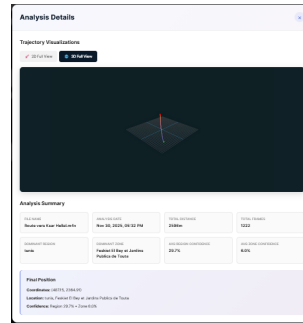
5.2.3 Trajectory History and Management Interface



(a) Trajectory History List



(b) 2D Trajectory View



(c) 3D Trajectory View

Figure 5.13: Trajectory History and Management Interface

The system maintains a comprehensive history of all UAV trajectory analyses (5.13a), enabling users to review, compare, and track flight patterns over time. The trajectory history interface provides:

Historical Record Display Each trajectory entry in the history includes:

- File Name: The source video file
- Timestamp: Precise date and time of analysis
- Key Metrics Summary
 - Total frames processed
 - Distance traveled
 - Dominant region identification
 - Dominant zone with detailed location
 - Final position coordinates with confidence levels

Visual Trajectory Thumbnails

Each history entry displays a thumbnail visualization of the trajectory path (5.13b, 5.13c), allowing users to quickly identify and distinguish between different flights through visual patterns. The thumbnails show the 2D and 3D trajectories' representation with color-coded start and end points.

Analysis Access

Users can access detailed analysis for any historical trajectory through the "View Details" button, which opens the complete trajectory visualization and analysis summary.

Metadata Tracking

The system tracks additional metadata for each trajectory:

- Visited regions and zones
- High-confidence prediction changes
- Final position with coordinate precision

This historical tracking capability enables users to:

- Monitor UAV flight patterns over time
- Compare multiple trajectories for operational analysis
- Maintain audit trails for regulatory compliance
- Identify recurring routes or zones of interest

5.2.4 Administrator Panel

The Administrator Panel provides exclusive management capabilities for users with elevated privileges, enabling complete control over the system's machine learning pipeline. This interface is accessible only to administrator accounts and consists of three integrated modules for comprehensive system management.

Dataset Manager

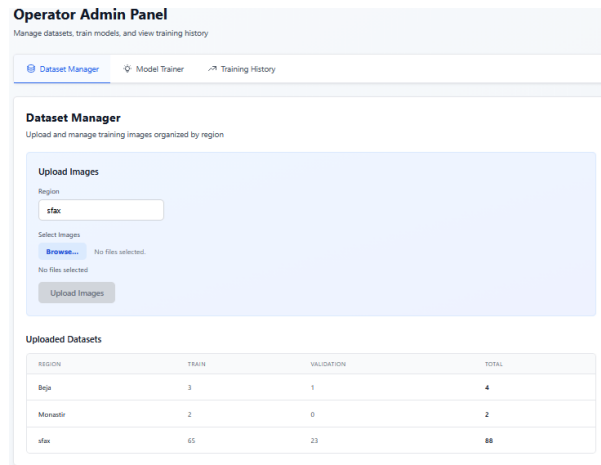


Figure 5.14: Dataset Manager Interface

The Dataset Manager module enables administrators to upload and organize training images for enhancing the system’s classification capabilities.

Upload Images Section:

The upload interface provides a streamlined workflow for adding new training data:

- **Region Selection:** Dropdown menu for specifying the geographical region associated with the images
- **File Browser:** "Browse..." button for selecting multiple image files from local storage
- **Selection Indicator:** Real-time feedback showing the number of files selected or "No files selected" status
- **Upload Control:** "Upload Images" button to initiate the transfer and organization of selected images into the training dataset

Uploaded Datasets Table:

A comprehensive table displays the current state of all regional datasets with the following columns:

- **REGION:** Geographic location identifier for the dataset
- **TRAIN:** Number of images allocated for model training
- **VALIDATION:** Count of images reserved for validation during training
- **TOTAL:** Combined total of training and validation images per region

This table allows administrators to quickly assess dataset balance across regions and identify areas requiring additional training data.

Model Trainer

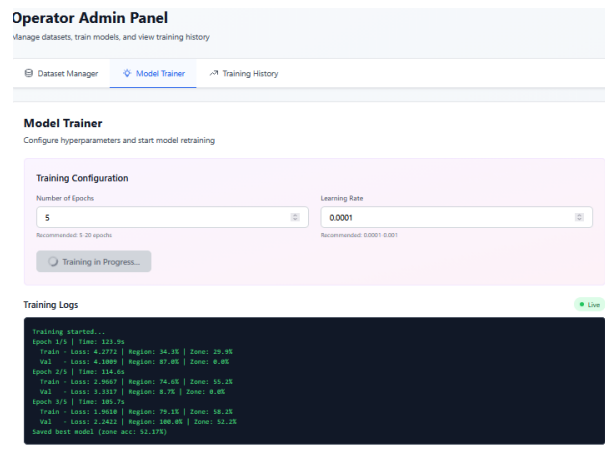


Figure 5.15: Model Trainer Interface

The Model Trainer module provides administrators with controls to initiate and configure machine learning model training sessions for the UAV region classification system.

Training Configuration Panel:

Administrators can customize key training hyperparameters through an intuitive interface:

- Number of Epochs: Numeric input field specifying the total training iterations
- Learning Rate: Decimal input field controlling the optimization step size
- Training Control Button: "Training in Progress..." button that initiates training when clicked and displays current status during active sessions

The configuration interface provides sensible default values and recommendations to help administrators select appropriate parameters for optimal model performance.

Training Logs Console:

A real-time console display with a dark terminal aesthetic provides live feedback during the training process, marked with a green "Live" indicator.

The console log displays:

- Session Initialization: "Training started..." message with timestamp
- Epoch Progress: Real-time updates for each training epoch showing:

- Epoch number and elapsed time
- Training metrics: Loss value, Region accuracy, Zone accuracy
- Validation metrics: Loss value, Region accuracy, Zone accuracy
- Model Checkpointing: Automatic saving notifications when validation performance improves

Real-Time Monitoring Features:

The training logs enable administrators to:

- Monitor training convergence in real-time
- Identify overfitting through train/validation metric divergence
- Assess both region-level and zone-level classification performance
- Track processing time per epoch for resource planning
- Verify automatic model checkpoint saves for best-performing iterations

The live console provides complete transparency into the training process, allowing administrators to make informed decisions about when to stop training or adjust parameters for subsequent sessions.

Training History

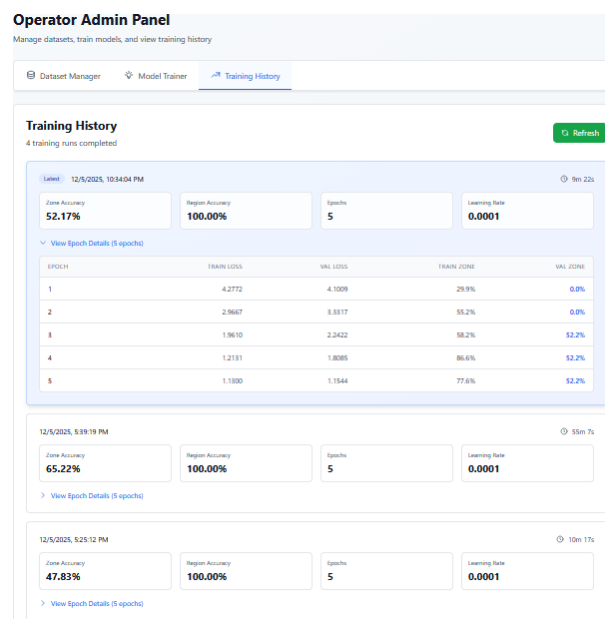


Figure 5.16: Training History Interface

The Training History module maintains a comprehensive record of all completed training sessions, enabling administrators to track model evolution and compare performance metrics.

Overview Panel:

The interface displays:

- Total Completed Runs: Count of all finished training sessions
- Refresh Button: Updates the display with the most recent training data

Training Session Records:

Each training session is presented as an expandable card showing:

Session Summary Cards:

- Timestamp: Date and time of completion
- Duration: Total elapsed time
- Zone Accuracy: Final zone-level classification accuracy
- Region Accuracy: Final region-level classification accuracy
- Epochs: Number of training iterations
- Learning Rate: Learning rate used during training

Epoch Details Table: Clicking "View Epoch Details" reveals a detailed breakdown with columns:

- EPOCH: Sequential epoch number
- TRAIN LOSS: Training loss value showing model error
- VAL LOSS: Validation loss indicating generalization performance
- TRAIN ZONE: Training accuracy for zone classification
- VAL ZONE: Validation accuracy for zone classification

This historical view enables administrators to compare training sessions, analyze convergence patterns, and select the best-performing model for deployment based on validation accuracy metrics.

Conclusion

In conclusion, this chapter presented the implementation of our UAV visual navigation system. The Swin Transformer was successfully trained and evaluated for aerial image classification, while the SLAM module provided GPS-free pose estimation. Together, these components, along with the web interfaces for video upload, real-time processing, and result visualization, form a functional system for UAV navigation. Overall, the work demonstrates a practical approach to combining deep learning and visual odometry, providing a solid foundation for future improvements.

General Conclusion

Completing this internship project at **Avionav**, within the NAVERA department, has been, I would say, a very interesting and instructive experience. It allowed me to explore, step by step, the application of deep learning, computer vision, and SLAM technologies in the context of UAV navigation. Honestly, at first, the task seemed quite complex, especially because integrating image-based localization with drone systems is not trivial. But gradually, through experimentation and testing, the process became clearer, and the results started to make sense

The main objective of this project was to design a system capable of recognizing locations from aerial images and assisting UAVs in navigating without relying solely on GPS. By using a **Swin Transformer model**, we tried to classify images from different Tunisian regions, and although the accuracy was not perfect, it showed promising potential for real-world applications. Integrating **SLAM** into the system added another layer of reliability, allowing the drone to better estimate its position and orientation based on its surroundings.

Regarding the interfaces, developing the web application was, I think, quite useful to visualize and manage the UAV navigation results. The image upload interface, as well as the analysis results interface, allowed for a smooth interaction with the system, making the output easier to interpret. While the system is functional, there is, of course, room for improvement, especially in increasing the robustness of the classification model and optimizing the SLAM performance.

Looking ahead, I believe that continuous refinement of the model, along with expanding the dataset to cover more regions and scenarios, would significantly enhance navigation accuracy. Also, exploring more advanced fusion techniques between image recognition and SLAM could help achieve more precise positioning, which is crucial for high-altitude and autonomous drone navigation.

Overall, this project at **Avionav** was not only an opportunity to apply theoretical

knowledge in a practical context, but it also taught me how to handle real-world challenges in UAV navigation. The skills gained in deep learning, computer vision, and system integration will, without a doubt, be valuable for my future career. At the same time, this experience highlighted the potential impact of advanced technologies in improving autonomous navigation systems and, perhaps, opens the door to more innovative solutions in the field of UAVs.

Bibliography

- [1] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021. doi: 10.1109/ICCV48922.2021.00986. URL <https://arxiv.org/abs/2103.14030>.
- [2] Mengxi Dai, Jianming Hu, Jian Zhuang, and Enhui Zheng. A transformer-based feature segmentation and region alignment method for uav-view geo-localization. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(7):4376–4389, 2022. doi: 10.1109/TCSVT.2021.3135013. URL <https://ieeexplore.ieee.org/document/9648201>.
- [3] Muhammad Aamir Arshad, Sana Ullah Khan, Sheraz Qamar, Muhammad Waqas Khan, Iqbal Murtza, Jeonghwan Gwak, and Arbab Khan. Drone navigation using region and edge exploitation-based deep cnn. *IEEE Access*, 10:95441–95450, 2022. doi: 10.1109/ACCESS.2022.3204876. URL <https://ieeexplore.ieee.org/document/9878336>.
- [4] Yuan Guo, Yi Zhou, and Feng Yang. Agcosplace: A uav visual positioning algorithm based on transformer. *Drones*, 7(8):498, 2023. doi: 10.3390/drones7080498. URL <https://www.mdpi.com/2504-446X/7/8/498>.
- [5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. doi: 10.1109/CVPR.2016.308. URL <https://arxiv.org/abs/1512.00567>.
- [6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.

- [7] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. doi: 10.1038/323533a0.
- [8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. European Conference on Computer Vision (ECCV)*, pages 213–229, 2020. doi: 10.1007/978-3-030-58452-8_13. URL <https://arxiv.org/abs/2005.12872>.
- [9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pages 91–99, 2015. URL <https://arxiv.org/abs/1506.01497>.
- [10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. doi: 10.1109/CVPR.2016.91. URL <https://arxiv.org/abs/1506.02640>.

Webography

- [1] **Google Earth.**
<https://earth.google.com/web/>
- [2] **UML.**
https://en.wikipedia.org/wiki/Unified_Modeling_Language
- [3] **Neural network.**
[https://en.wikipedia.org/wiki/Neural_network_\(machine_learning\)](https://en.wikipedia.org/wiki/Neural_network_(machine_learning))
- [4] **Augmented reality.**
https://en.wikipedia.org/wiki/Augmented_reality
- [5] **Virtual reality.**
https://en.wikipedia.org/wiki/Virtual_reality
- [6] **Waterfall methodology.**
<https://business.adobe.com/blog/basics/waterfall>
- [7] **Region Proposal Network (RPN).**
<https://www.geeksforgeeks.org/computer-vision/region-proposal-network-rpn-in-object-detection/>
- [8] **Swin Transformer.**
https://huggingface.co/docs/transformers/en/model_doc/swin
- [9] **What is hierarchical data?.**
<https://www.tibco.com/glossary/what-is-hierarchical-data>
- [10] **SLAM (Simultaneous Localization and Mapping) .**
<https://www.mathworks.com/discovery/slam.html>
- [11] **Python.**
<https://www.python.org/>
- [12] **FastAPI.**
<https://fastapi.tiangolo.com/>

[13] **torchvision.**

<https://docs.pytorch.org/vision/stable/index.html>

[14] **RTAB-Map.**

<https://introlab.github.io/rtabmap/>

[15] **OpenCV.**

<https://opencv.org/>

[16] **Architecture and Working of Swin Transformer.**

<https://www.geeksforgeeks.org/computer-vision/swin-transformer/>