Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Synthesis and Evaluation of an Image-Based GNSS-Redundancy System for UAV Navigation

by

Sameer Shaboodien

25002783

Report presented in partial fulfilment of the requirements of the module Project (E) 448 for the degree Baccalaureus in Engineering (Electrical and Electronic) in the Faculty of Engineering at Stellenbosch University.

Supervisor: Prof. RP Theart

01 November 2024

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Studentenommer / *25002783*	Handtekening / 
Voorletters en van / *S Shaboodien*	Datum / *27/10/24*

Abstract

Unmanned Aerial Vehicles (UAVs) rely on the Global Navigation Satellite System (GNSS) for precise navigation, but vulnerabilities such as jamming and spoofing can jeopardize mission success. This project developed an image-based redundancy navigation system to ensure reliable UAV navigation following GNSS signal loss. The system optimizes the positional inference pipeline by integrating advanced feature extraction, matching, and planar transformation techniques, achieving radial localization errors below 2% of the UAV's displacement from a reference image. Designed for real-time operation, the system provides estimated position and heading within two seconds, enabling pilots to maintain effective control. Comprehensive testing across diverse environments demonstrated the system's high accuracy, robustness, and generalizability without the need for environment-specific tuning. It remained effective in challenging operational conditions including variations in lighting, image information overlap, camera tilt, and resolution. This image-based redundancy navigation system offers a practical and dependable redundancy solution for enhancing UAV operational safety and supporting national security.

Contents

List of Figures

List of Tables

Nomenclature

Variables and Functions

$RMSE$	Root Mean Square Error, representing the average magnitude of errors. In this study, the usage of this metric mainly refers to the radial error in Latitude, Longitude estimate, in metres, averaged across the dataset.
d_{radial}	Radial error distance, measuring the distance between estimated and true positions in meters.
x, y	Coordinates of a point in an image or on the ground plane.
θ	Rotation angle between consecutive images in degrees or radians, used for orientation alignment.
T	Transformation matrix representing the planar transformation between images.
H	Homography matrix, specifically mapping points from one image plane to another.
GPS	Global Positioning System, providing geolocation and time information for navigation.
$GNSS$	Global Navigation Satellite System, a generic term for satellite-based navigation systems.
MAE	Mean Absolute Error, measuring the average localization error without directionality.
MI	Mutual Information, measuring the similarity or overlap between images, often used in image matching.
Lat-Lon Estimation	Estimation of Latitude and Longitude; If Referencing an error, it is the radial one.

Key Terms and Concepts

Features	Unique keypoints in an image along with their descriptors, used for identifying and uniquely matching points across different images for localization.
Affine Transformation	A planar transformation that preserves points, straight lines, and planes, including translation, scaling, rotation, and shearing.
Descriptors	Numerical values describing the characteristics of keypoints in an image, allowing for effective matching between different images.
Feature Detectors (or Extractors)	Processes for identifying distinct features in an image, which are invariant to changes in scale, rotation, and illumination.
Global Matching	Process of finding similarities and determining pose between entire images, considering full-image context rather than isolated keypoints.
GPS (Global Positioning System)	A satellite navigation system providing geolocation and time information. Vulnerable to jamming and spoofing, posing reliability issues for UAV navigation.
Homography	A planar transformation mapping points from one image plane to another in 2D space. Used in image processing to describe transformations like rotation, translation, scaling, and shearing.
Homography Estimation	The process of calculating the homography matrix that defines the transformation between two images for alignment and reliable localization.
Keypoints	Specific points in an image used to identify features. Typically areas of strong contrast or distinct patterns, enabling reliable matching across different images.
Local Matching	Matching keypoints between images by comparing descriptors, focusing on individual feature descriptors to establish precise localization.
Mutual Information	A metric quantifying the information shared between two images, aiding in assessing the quality of feature matching and overlap similarity.
Planar Transforms	General transformations applied to a plane in 2D space, such as affine transformations, homography, scaling, and shearing, which are crucial for image alignment.

Scaling	A transformation that changes the size of an image or its features without altering its shape, normalizing feature sizes across different images.
Shearing	A type of affine transformation that slants the shape of an object in an image. Shearing changes angles between lines while keeping parallel lines intact.
Localization	The process of determining the position of an object or feature within a given space, used in the context of GNSS and image-based navigation.

Acronyms and Abbreviations

GPS	Global Positioning System
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
MI	Mutual Information
UAV	Unmanned Aerial Vehicle
GNSS	Global Navigation Satellite System
DOF	Degrees of Freedom

Chapter 1

Introduction

1.1. Background

Unmanned Aerial Vehicles (UAVs) have become indispensable tools in various sectors, including military operations, surveillance, reconnaissance, and intelligence gathering. In South Africa, UAVs play a crucial role in border monitoring and supporting military missions by providing persistent aerial observation [1]. Their capability to operate in hazardous or inaccessible areas enhances operational effectiveness and safety.

Despite their widespread use, UAVs predominantly rely on Global Navigation Satellite Systems (GNSS) such as the Global Positioning System (GPS) for navigation and positioning. GNSS operates by utilizing a constellation of satellites that transmit precise time-stamped signals to Earth-based receivers. Each receiver calculates its position by measuring the time delay of signals from multiple satellites, requiring data from at least four satellites to determine its three-dimensional location. While GNSS provides essential Positioning, Navigation, and Timing (PNT) information globally, its reliance on weak, line-of-sight satellite signals introduces significant vulnerabilities [2].

In recent years, the vulnerabilities of GNSS to jamming and spoofing have become increasingly pronounced. With the advent of more accessible jamming and spoofing technology, intentional GNSS interference is no longer a complex undertaking [3]. Instances of GNSS jamming and spoofing are particularly prevalent in conflict zones, where adversaries exploit GNSS weaknesses to disrupt or mislead UAV operations. The issue is far from hypothetical; over 1100 daily incidents of aircraft GNSS spoofing alone have been reported worldwide, underscoring the urgency of addressing these vulnerabilities [3].

The increasing prevalence of GNSS disruptions has direct implications for national security. Loss of GNSS signals can lead to an inability to locate the UAV, compromising control and potentially resulting in the UAV crashing or being captured. This situation emphasizes the critical need for a safety measure that can ensure the safe return of the UAV following GNSS signal loss [2].

Various alternative navigation methods have been explored to mitigate complete reliance on Global Navigation Satellite Systems (GNSS). Odometry-based solutions, or inertial measurement units (IMUs) estimate UAV displacement through inertial measurements from accelerometers and gyroscopes; however, they suffer from significant drift over relatively short distances, rendering the location estimates of even high-end IMUs unusable for

typical missions [4]. Quantum sensing navigation employs cold atom inertial sensors to achieve superior precision compared to traditional IMUs, but it is often prohibitively expensive and bulky [5]. Radio Frequency (RF) communication systems can triangulate UAV positions using ground beacons or cellular towers, yet their effectiveness is limited in remote areas and by the Earth's curvature, restricting their operational range [6]. Light Detection and Ranging (LIDAR) systems map the ground by emitting laser pulses and measuring their return times to create detailed 3D environmental maps, but they are power-intensive and emit detectable signals, which are undesirable for stealth operations prevalent in military contexts [7].

Image-based navigation presents a viable solution that addresses the limitations of the aforementioned methods. This technique involves capturing images of the landscape and comparing them to previously captured reference images using planar transformations. These transformations align two images of the same planar surface based on their features, enabling the estimation of the UAV's orientation and position relative to the reference image. The coordinates and heading of the reference image can then be used to determine the UAV's absolute location and heading. Reference images can be captured during the UAV's outbound path, allowing it to navigate back to base after GNSS denial by retracing its steps. Alternatively, these images can be pre-acquired and stored in a database for real-time navigation [8].

However, it remains unclear whether image-based systems can effectively operate in the context of UAV navigation. Specifically, it is uncertain whether they can generalize to various terrains and operational conditions, such as varying light levels, while maintaining their accuracy. This study aims to develop a working pipeline for this context and test it on real-world data, thereby addressing these uncertainties and evaluating the viability of image-based navigation as a redundancy measure to GNSS.

1.2. Problem Statement

The increasing frequency of GNSS disruptions due to jamming and spoofing poses a significant threat to UAV operations [2]. Current alternatives to GNSS navigation, such as quantum sensing, odometry, RF communication, and LIDAR, have significant drawbacks: they are either too expensive and bulky, reduce stealth capabilities, or suffer from drift over time [4–7]. To address these drawbacks and enhance the safety and effectiveness of national military missions, a comprehensive synthesis and evaluation of an image-based redundancy navigation system is required. Existing studies on image-based UAV navigation lack detailed system setups and fail to evaluate the system practicality across diverse environments and challenging conditions [9, 10].

1.3. Aim

The primary aim of this project is to synthesize and evaluate an image-based navigation system for UAVs that estimates their global position in GNSS-denied environments. The approach will leverage images and telemetry data captured prior to GNSS denial, enabling the UAV to navigate back to base by following its outbound path.

1.4. Objectives

1. **Develop a Localization Pipeline:** Create a pipeline that achieves the highest possible position estimation accuracy while maintaining real-time performance and generalizability.

2. **Identify the Best Techniques:** Select, integrate, and optimize feature extraction, matching, and planar transformation estimation techniques.

3. **Evaluate the Method Across Environments:** Assess the system's performance in various terrains and operational conditions to ensure robustness and adaptability.

4. **Evaluate the Method Under Stressful Conditions:** Test the system's reliability and accuracy under challenging conditions such as low light levels, high-speed movement, and partial occlusions.

1.5. Requirements

Accuracy: Achieve a maximum single-image radial location estimation error of less than 10% of the UAV's displacement between the current and reference image across all datasets.

Real-Time Performance: Ensure a maximum processing time of less than 2 seconds for position and heading estimation following GNSS signal loss across all datasets.

Processing of reference images during the outbound journey, while GNSS signal is available, will have a maximum allowed processing time of 5 seconds. Since the navigator does not need to respond at this point, and the landscape changes relatively slowly, a higher capture rate would only result in excessive, nearly identical reference images.

Adaptability: Maintain the accuracy and time constraints across diverse environments without requiring manual parameter adjustments for each environment.

1.6. Scope

This project is scoped to ensure feasibility within the given timeframe and resources by making several assumptions and acknowledging limitations. It assumes that the UAV's downward-facing camera remains perfectly aligned and that the UAV's altitude is constant, minimizing perspective distortion and scale changes, respectively. Images are expected to

be free from occlusions and significant dynamic movement. The system relies on onboard sensors like IMUs, coupled with acceptable turning rates, to turn around and recapture its path after GNSS signal loss, ensuring over 60% image overlap between current and reference images. The study focuses on assessing the system's viability using established methods and parameter sets, excluding exhaustive optimization and untrained machine learning methods. The system outputs the estimated position and heading information for navigation assistance based on simulated video footage; it does not integrate with real-world systems.

1.7. Data Provisioning

An agreement was established with an external party to provide real-world flight data from their UAV operations for use prior to the start of this study. However, the data was not provided within the project's timeframe. To proceed, Google Earth data [11] was utilized. It offers free access to high-resolution aerial imagery with 3D terrain features and therefore real-world perspective changes. It also provides latitude and longitude (Lat-Lon) coordinates and heading, closely approximating real-world data.

1.8. Structure of the Report

This report is structured to provide a comprehensive overview of the research undertaken to develop the image-based GNSS redundancy navigation system for UAVs. Chapter 2 reviews the studies related to vision-based navigation solutions and discusses their contextual limitations. Chapter 3 details the methodology, including feature extraction, matching, similarity computation, and planar transformations. Chapter 4 provides a comparative analysis of different methods applicable to each stage of the pipeline. Chapter 5 presents the evaluation of the developed pipeline against the objectives and various operational challenges. Chapter 6 summarizes the project's outcomes and outlines recommendations for future work.

Chapter 2

Literature Review

2.1. Related Work

Autonomous navigation for Unmanned Aerial Vehicles (UAVs) has been extensively studied, with prominent methods like Simultaneous Localization and Mapping (SLAM) providing comprehensive mapping and localization capabilities. SLAM enables UAVs to construct detailed maps of their environment while simultaneously tracking their position within these maps [8]. Typically, SLAM systems integrate data from multiple sensors—such as cameras, LiDAR, and Inertial Measurement Units (IMUs)—to achieve accurate localization, making them particularly effective in structured, indoor environments like warehouses. However, building and maintaining high-resolution maps in SLAM is computationally intensive, requiring significant processing power and memory [8]. This computational burden poses challenges for real-time applications in resource-limited UAVs. Additionally, SLAM systems are sensitive to map distortions and inaccuracies, which can degrade localization reliability [8]. Moreover, in the context of UAV navigation for return flights, constructing an entire map of the outbound path may be unnecessary, making SLAM an excessive solution in terms of memory consumption and computational resources for such applications.

Optical flow is another technique for motion estimation, which calculates the apparent motion of brightness patterns in the image plane between consecutive frames [9]. Optical flow relies on the brightness constancy assumption and small, smooth movements, which can be violated in cases of abrupt UAV movements, rotations, or scale changes, leading to unreliable motion estimates. Consequently, optical flow is generally more suitable for short-term odometry rather than long-distance reverse path tracking. Additionally, computing dense optical flow across entire frames is computationally intensive, rendering it less practical for real-time applications on resource-constrained UAVs [12].

Several studies have explored image-based solutions for UAV navigation, utilizing feature matching techniques to estimate the UAV's location.

For instance, Zhang et al. [10] introduced LoFTRS, a deep learning-based image matching method with semantic constraints to improve matching accuracy. LoFTRS provides refined feature correspondences that strengthen subsequent UAV localization tasks. While this study offers valuable insights into potential pipelines and highlights the benefits of efficient feature extractors, it lacks a detailed implementation, comparative

analysis of methods, and practical testing across diverse environments.

Another study by Sim et al. [9] proposed an image-based location estimation approach that combines relative positioning techniques through aerial image sequences to improve short-term odometry, enhancing UAV navigation reliability over short distances. Their method emphasizes the importance of constant global correction by normalizing to a global reference frame. However, their approach does not consider the use of fixed reference images for navigation, which limits its applicability for reference-image-based navigation in GNSS-denied environments.

However, these existing methods do not provide detailed descriptions of the complete navigation pipeline, including the selection of specific methods and parameter choices. Moreover, they do not assess the system's ability to generalize across diverse environments or test its robustness under typical operational challenges faced during UAV missions.

To address these gaps, this study presents a comprehensive image-based navigation pipeline for UAVs. It includes in-depth comparisons of different methods, emphasizes environmental adaptability, and undergoes rigorous testing under adverse conditions. This approach aims to provide a reliable and scalable framework for addressing real-world UAV navigation challenges in GNSS-denied environments.

2.2. Fundamental Concepts and Techniques

This section provides a comprehensive overview of the fundamental concepts and techniques that underpin the proposed image-based UAV navigation system. By delving into feature extraction, matching, and planar transformations, it establishes the theoretical foundation essential for the subsequent system design and implementation. The discussion emphasizes the critical role of feature-based methods over direct approaches, setting the stage for understanding the chosen methodologies.

2.2.1. Feature Detectors

Feature extraction is a cornerstone of image-based UAV navigation, enabling the estimation of transformations such as rotation and translation between consecutive images. Feature detectors identify **keypoints**—distinct, repeatable points within an image—and generate **descriptors** that encapsulate information about the local image region surrounding each keypoint. These keypoints and descriptors, or simply features, are essential for accurate matching across multiple frames, allowing the system to track movement while maintaining invariance to changes in scale, rotation, and illumination. An example of feature detection and matching, the latter explained in the subsequent section, is illustrated in Figure 5.6. This image shows the top 50 matches between two images after rotational alignment.

ORB (Oriented FAST and Rotated BRIEF)

ORB combines the **FAST** [13] keypoint detector with the **BRIEF** [14] descriptor, enhanced for rotation invariance. **FAST** rapidly identifies keypoints by analyzing pixel intensity differences in a circular region around each candidate point. Once detected, **BRIEF** encodes the local image patch into a binary string through intensity comparisons. ORB introduces rotational invariance by aligning keypoints based on their dominant orientation before descriptor computation. This enhancement makes ORB both extremely fast and robust to scale and in-plane rotation, although it may struggle with repetitive textures or complex lighting variations [15].

AKAZE (Accelerated-KAZE)

AKAZE constructs a nonlinear scale space using diffusion-based filtering, capturing finer image details more effectively than linear methods. It detects keypoints by assessing local contrast with a specialized adaptive filter, enabling the identification of subtle features that simpler detectors might miss. The **Modified Local Difference Binary (MLDB)** [16] descriptor encodes the neighborhood of each keypoint into a binary vector based on pixel intensity differences. While AKAZE is both fast and compact, its performance can be sensitive to detection thresholds across different environments, potentially affecting its robustness in varied operational contexts [17]. However, a notable strength of AKAZE is its rotational invariance [15].

SuperPoint with LightGlue

SuperPoint is a deep learning-based keypoint detector and descriptor that leverages convolutional neural networks (CNNs) to identify and describe keypoints in a single forward pass. Pre-trained on extensive image datasets, SuperPoint excels at recognizing stable and distinctive keypoints under varied conditions [18]. However, its performance may degrade on datasets significantly different from its training data. Pairing SuperPoint with **LightGlue**, a machine-learning-based matcher, enhances matching accuracy through advanced graph-based techniques that were recognized at the 2023 International Conference on Computer Vision [19]. Despite their high accuracy, SuperPoint and LightGlue are computationally intensive and require GPU acceleration for real-time applications. However, their improved performance justifies their inclusion in this study, even though they may not achieve real-time performance when tested on a CPU alone [18].

2.2.2. Feature Matching

Feature matching establishes correspondences between keypoints in different images based on descriptor similarity. After identifying these correspondences, ambiguities and low-

quality matches are removed, as detailed in Section 2.2.5.

Each matcher generates a list of potential matches along with their similarity scores, quantified using a descriptor-space distance metric. These scores are instrumental in subsequent filtering processes.

Feature matching involves two primary components: the choice of matching technique to acquire potential matches and the search technique that determines which of these matches to retain.

Types of Feature Matching Techniques

The following match acquisition techniques are commonly employed in feature-based navigation systems:

Brute-Force Matcher (BFMatcher): The Brute-Force Matcher is a simple, exhaustive matcher that matches each feature in one image with every feature in the second image, ensuring the best possible match based on descriptor similarity. While this guarantees high accuracy, it is computationally expensive, especially with large numbers of keypoints, making it less suitable for real-time applications without optimization [20].

Fast Library for Approximate Nearest Neighbours (FLANN): FLANN accelerates the nearest neighbour search in high-dimensional descriptor spaces using algorithms such as KD-trees or hierarchical clustering, adapting dynamically to the dataset. This approximate matching approach offers significant speed improvements with minimal loss in accuracy, making it ideal for real-time applications with extensive datasets [21].

LightGlue: Explained in Section 2.2.1.

The following search techniques are commonly used to filter matches and retain only the most reliable correspondences:

Radius Search: This method retains matches within a specified distance in descriptor space, effectively filtering out weaker matches. However, it does not guarantee a fixed number of matches per keypoint, leading to inconsistent results [22].

K-Nearest Neighbours (KNN) Matching: KNN matching retains the top K matches for each keypoint, allowing the application of post-filtering techniques such as Lowe's ratio test to eliminate ambiguous matches [22].

Vanilla Matching: Vanilla matching returns the single best match for each keypoint based on the closest descriptor distance. It is a subset of KNN matching with K=1, offering simplicity and ease of implementation [22].

2.2.3. Image Similarity Computation

Image similarity computation is a pivotal component of UAV navigation systems that rely on reference images for accurate localization and pose estimation. Effective similarity

measures ensure efficient processing of extensive image datasets and facilitate precise transformation estimations, which are essential for reliable navigation.

2.2.3.1 Proximity-Based Techniques

To achieve efficient, real-time performance, the search space is reduced to images within the proximity of UAV's last known location, filtering images within a static or dynamic radius. While this method is highly efficient, it does not account for potential deviations from the expected flight path or the presence of poor-quality reference images. This limitation implies that this measure cannot be used as the sole basis for image similarity computation, necessitating the integration of additional techniques for comprehensive assessment.

2.2.3.2 Global Matching Techniques

To ensure images are evaluated for similarity and ensure they are free from significant distortion, global matching, or direct methods are employed. Direct methods estimate planar transformations by comparing entire image pixel intensities and minimizing differences through optimization techniques like gradient descent. Because they consider the entire image context, these methods are well-suited for similarity comparisons; however, they are not ideal for precise transformation estimation due to their sensitivity to noise and illumination changes [23]. The following methods are global matching techniques commonly used in image similarity computation:

Cross-Correlation

Cross-correlation measures similarity by sliding one image over another and computing the sum of pixel-wise multiplications at each position. The peak value signifies the best alignment, and its magnitude indicates the confidence level of the similarity. Higher confidence values reflect greater similarity between the images. While straightforward to implement, cross-correlation is sensitive to noise and illumination changes, which can compromise the reliability of the similarity measure [24].

Histograms

Histogram comparison assesses similarity by analyzing the distribution of pixel intensities within each image. Typically, each image's histogram is divided into 256 intensity bins for 8-bit images, and similarity is quantified using metrics such as Chi-Square or Bhattacharyya distance. This method emphasizes global color and brightness distributions but neglects spatial information, making it less effective for nuanced structural differences [25].

Structural Similarity Index (SSIM)

SSIM evaluates similarity by decomposing images into luminance, contrast, and structure components. It computes local statistics within small windows and integrates them into a single similarity score that mirrors perceived image quality. SSIM effectively captures structural information like edges and textures, aligning closely with human visual

perception. Although slightly more computationally expensive than the former methods, it is robust to varied conditions [26].

Local Detectors Conversion Although not inherently a global matching technique, local feature matching can be adapted to achieve a global understanding of image similarity. This involves identifying and matching keypoints in both images and assessing the overall number of good matches. However, this approach alone does not ensure an even distribution of matches across the entire image, potentially leading to biased, localized similarity assessments. To mitigate this, a grid matching technique is employed, dividing the image into grids and limiting the number of matches per grid. Although the most computationally intensive, this method enhances robustness against distortions and rotations by ensuring a uniform distribution of matches across the image. To maintain reasonable runtime, this method has to employ a very crude detection and matching layer.

2.2.4. Planar Transformation Estimators

Feature-based methods extract and match keypoints from both reference and real-time images, often incorporating outlier removal stages [23]. By focusing on distinctive features rather than every pixel, these methods excel in handling large viewpoint changes and rotations, enabling more precise transformation inference. This targeted approach enhances computational efficiency and robustness to environmental distortions, though it requires careful management to avoid performance degradation from variation in the number of extracted feature points.

In typical UAV flight scenarios, the primary transformations of interest are rotation and translation. Perspective distortion, caused by three-dimensional structures, is minimal at high altitudes. Similarly, shear distortion, which occurs when the UAV turns or is not parallel to the ground, is negligible when the UAV maintains level flight. Furthermore, scaling is not present as per the scope of this study. The following subsections detail the primary planar transformations employed in the system.

Affine Transformation

Affine transformation captures translation, rotation, scaling, and shear, providing six degrees of freedom. It is represented by a 2×3 matrix that maps points from one plane to another while preserving lines and parallelism. Affine transformations are computed by estimating the affine transformation matrix between two sets of corresponding points using OpenCV's `estimateAffine2D` function [27]. While versatile, the inclusion of scaling and shear introduces unnecessary error points in the UAV case.

Rigid Transformation Estimation (SVD)

The rigid transformation via SVD preserves the shape and size of objects by estimating only rotation and translation, excluding scaling and shear. Represented by a 2×3 matrix, rigid transformation ensures orthogonality in the rotation component. Utilizing Singular Value Decomposition (SVD), this method minimizes the least-squares error between two point sets. The process involves: computing the weighted centroids of both point sets, centering the points by subtracting their respective centroids, calculating the covariance matrix of the centered points, performing SVD on the covariance matrix to derive the rotation matrix, and determining the translation vector based on the centroids. The resulting 2×2 rotation matrix and 2×1 translation vector are combined to form the rigid transformation matrix. Rigid transformation is computationally efficient and well-suited for UAV applications [28].

Partial Affine Transformation

Partial affine transformation simplifies the full affine model by focusing solely on translation, rotation, and limited uniform scaling, offering four degrees of freedom. This transformation is also represented by a 2×3 matrix, similar to the affine transformation but without shearing and with reduced, uniform scaling. This offers similar performance to the prior rigid method, but with minor additional error points due to scaling [27].

Homography Transformation

Homography transformation accounts for translation, rotation, scaling, shear, and perspective distortion, providing eight degrees of freedom. It is represented by a 3×3 matrix and is estimated using OpenCV's `findHomography` function, typically with RANSAC for outlier rejection [29]. While homography offers greater flexibility in modeling complex, typically three-dimensional transformations, its additional degrees of freedom introduce unnecessary errors and computational overhead for UAV-based applications.

2.2.5. Optimization Techniques

Optimization techniques aim to refine the accuracy and reliability of the matched points used for transformation estimation by effectively filtering out erroneous matches and improving transformation accuracy. The following subsections detail the primary optimization methods employed in this system.

Random Sample Consensus (RANSAC) for Planar Transformation

RANSAC is a robust estimation technique used to estimate planar transformations by iteratively selecting random subsets of point correspondences to fit a model and identify

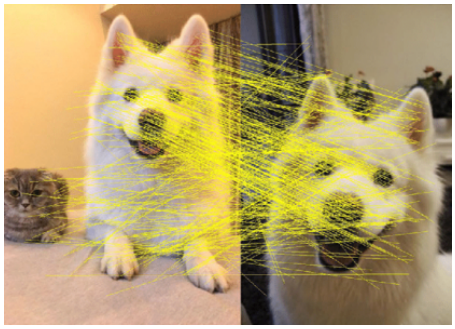
inliers [30]. The process involves randomly selecting a minimal subset of point pairs, estimating the transformation model (e.g., affine or homography) based on the selected subset, determining the number of inliers that fit the estimated model within a predefined threshold, and repeating the process for a set number of iterations or until a sufficient inlier ratio is achieved. This approach is highly effective in datasets with significant outliers, focusing on finding a model that best fits the largest subset of inliers. However, due to its iterative nature and the need to sample repeatedly, RANSAC can result in increased runtime, particularly in larger datasets or when dealing with numerous outliers [30].

Least Median of Squares (LMeds) for Planar Transformation

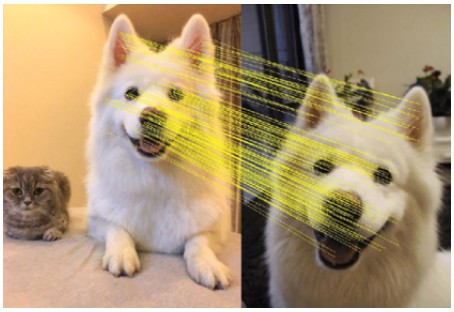
Least Median of Squares (LMeds) is a robust estimation technique used for estimating planar transformations by minimizing the median of the squared residuals between matched points [31]. Unlike RANSAC, which focuses on maximizing the number of inliers, LMedS aims to find a model that minimizes the median error, making it less sensitive to outliers. The process involves calculating the transformation parameters that result in the lowest median of squared residuals across all data points. While LMedS can be more robust in the presence of outliers compared to RANSAC, it can be computationally more intensive and may not perform as well when the percentage of outliers is high.

Lowe's Ratio Test

Lowe's ratio test is a widely used filtering technique used to eliminate ambiguous or false keypoint matches by comparing the distance of the best match to the second-best match [32]. For each keypoint match, the ratio of the distance of the best match to that of the second-best match is calculated, and the match is retained if this ratio is below a predefined threshold. A lower ratio indicates that the best match is significantly better than the alternatives, thereby increasing the likelihood of the match being correct. An example of Lowe's ratio test filtration is shown in Figure 2.1a and Figure 2.1b.



(a) Matched Features Prior to Lowe's Ratio Test. Reproduced from [32]



(b) Matched Features Following Lowes Ratio Test. Reproduced from [32]

N-Match or Absolute Thresholding

N-match thresholding involves setting a threshold that allows only a specific number of matches with the smallest descriptor distances to be retained. Absolute thresholding filters matches based on a fixed distance in descriptor space. Only matches that meet or fall below this predefined distance threshold are retained, ensuring that only sufficiently similar matches are used in the transformation estimation process. These methods primarily suffer from difficulty in setting the threshold, which is often extremely sensitive to dataset variations and method parameters.

Chapter 3

System Design

3.1. High-Level System Design

This chapter outlines the developed methodology for the UAV navigation system, detailing the pipeline and its various components. The system is designed to accurately estimate the UAV's position and heading after GNSS signal is lost. The high-level flow of the system is illustrated in Figure 3.1.

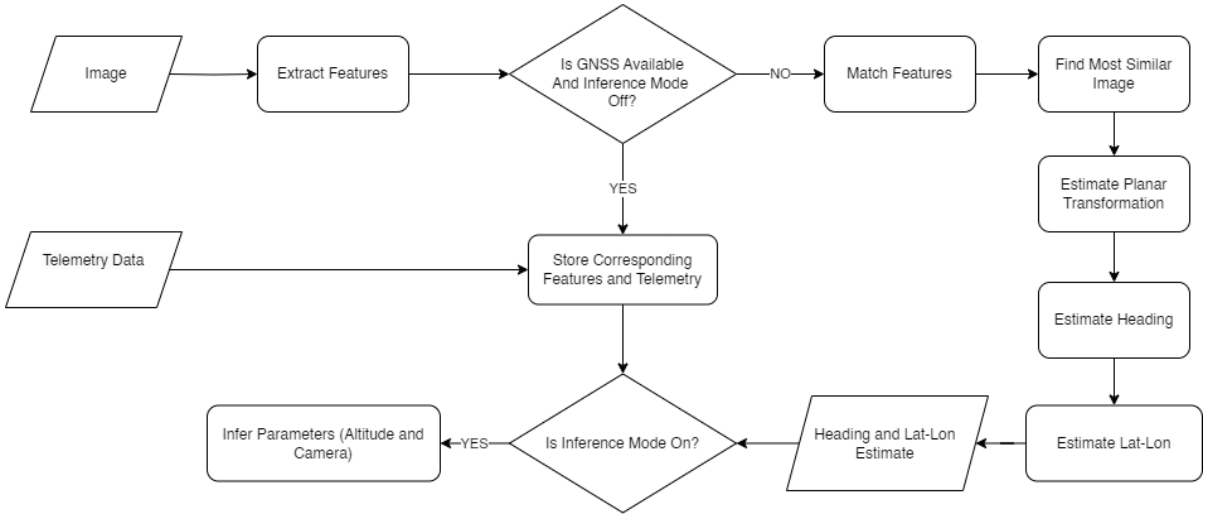


Figure 3.1: High-Level Flow of the System

3.2. Detailed System Pipeline

The system pipeline consists of multiple stages, each designed to perform a specific task in the navigation process. The pipeline is designed to be robust, accurate, and computationally efficient, ensuring reliable navigation in various operational scenarios. The pipeline stages are detailed below.

1. **Image (Input):** The process begins with capturing a live image from the UAV's downward-facing camera. This real-time visual input provides essential information about the UAV's environment, forming the basis for position estimation.

2. **Extract and Store Features:** Keypoints and descriptors are extracted from the current image to aid in the matching process. This extraction occurs in two layers:

the coarse layer, used in the second stage of image match search space reduction, and the dense layer, used during the precise transformation stage. The system also extracts features when GNSS is available to reduce computational load during critical phases when GNSS is lost.

3. **Store Corresponding Features and Telemetry:** Extracted features (keypoints and descriptors) along with their corresponding telemetry data (GNSS position and heading) are stored for future reference. Stored features facilitate relative transformation inference when GNSS is unavailable, while telemetry data assists in converting relative transformations to real-world coordinates and headings.

4. **Infer Parameters (Altitude and Camera):** To be able to convert from a pixel translation to a metre value, a conversion factor is required. This conversion factor is dependent on the altitude of the UAV and the camera's focal length. When GNSS signal is available, this stage uses the complete pipeline to estimate the UAV's Lat-Lon coordinates using a placeholder factor value. These estimates are accumulated and compared, using only the first 5 images to balance overall efficiency and parameter accuracy, with the ground truth Lat-Lon coordinates via linear regression to infer the conversion factor. This is necessary in the scope of this study, which assumes constant and unknown altitude and camera parameters. These parameters are both represented within a single factor, the pixel to metre ratio.

5. **Match Features:** Features between the current image and reference images are matched to ensure that comparisons are based on mutual features. This involves matching of both the coarse and dense layers of features.

   The matching process is optimized to ensure robustness against noise and outliers, with a focus on computational efficiency. The techniques include usage of match acquisition techniques, search techniques, as well as subsequent optimization techniques to refine the matches.

6. **Find Most Similar Image:** The stage involves two layers of potential image match search space reduction until a single image is chosen for the current image to accurately infer its relative transformation against.

   The first is reduction based on proximity to the last known Lat-lon Coordinates. This stage outputs 5 matches.

   Thereafter, a more precise global matching technique is applied to identify the most similar match from the reduced image search space. This requires initial rotational alignment using the coarse layer of matched features and subsequent global matching techniques to identify the best match based on similarity scores. The crude layer is

chosen as global matching techniques were proven in the testing phase to be robust against minor rotational inaccuracies, allowing for efficiency prioritization.

7. **Estimate Planar Transformation:** After identifying the best match, the system performs a precise estimation of both rotation and translation between the input and reference images using the dense match layer.

   The first step involves estimating the rotation between the images, followed by aligning the images based on this rotation.

   Thereafter, the system recomputes the dense layer of features and matches on the aligned images. The reason for the recomputation prior to translation estimation is due to improved accuracy; aligned point clouds allow for improved translation estimation by way of less parameters to estimate. The amount of mutual information also remains constant when applying alignment to an image.

   Finally, the system estimates the translation between the images using the refined dense layer.

   This rotation and translation estimation are outputted to the next stages for conversion from relative to absolute conversion of the UAV's heading and position.

8. **Estimate Heading:** The internal angle between the current and reference images is added to the reference image's heading to determine the UAV's current heading.

9. **Estimate Lat-Lon:**

   The estimated translation at this stage is a pixel value representing the displacement between aligned images. To convert this to real-world latitude and longitude coordinates, the vector must be scaled in magnitude and rotated to align with the global coordinate system. This process consists of the following steps:

   The translation vector, initially relative to the internal image coordinate system, is rotated by the heading of the reference image, aligning it with the global latitude-longitude system without altering its pixel magnitude.

   Next, the estimated pixel displacement is multiplied by the inferred pixel-to-metre conversion factor, resulting in a translation vector in metres with East and North components.

   Then, this vector is converted to a latitude and longitude displacement. To account for the Earth's oblate spheroid shape, the equations below are used, which consider the latitude-dependent distance per degree of longitude:

   $\Delta\text{Lon} = \frac{\Delta\text{East}_\text{m}}{111320 \cdot \cos(\text{Lat}_\text{ref})}, \quad \Delta\text{Lat} = \frac{\Delta\text{North}_\text{m}}{111320}$

   where the East and North components represent the eastward and northward displacement vector components in metres, aligned to the global coordinate system.

The constant 111320 converts degrees of latitude to metres, with longitude scaled by $\cos(\text{Lat}_{\text{ref}})$ to reflect latitude-dependent longitudinal distance.

Finally, the calculated displacements in latitude and longitude are added to the reference image's known coordinates, yielding the UAV's estimated absolute position.

10. **Heading and Lat-Lon Estimate:** The systems heading and Lat-Lon estimates are outputted to the user interface for real-time monitoring and in practice, navigation.

3.3. Dynamic Methods and Techniques

Testing showed that the degree of environmental variation often leads to situations where a static pipeline will either not find a sufficient number of matches or take too long. To maintain generalizability across datasets (terrains), adaptive methods were implemented to adjust the pipeline without prior knowledge of the environment.

The first adaptation was made to AKAZE, which did not have a built-in method to adjust the number of keypoints detected. This method sampled the first image in the dataset and altered its threshold iteratively until a set number of keypoints were found. This was done once per dataset to maintain computational efficiency, but in practice, it may be done per n images or per time interval.

Secondly, Lowe's ratio employed an adjusting threshold that increased in leniency until a set number of matches or a percentage of available keypoints were found. This was done per image.

3.4. Software

This sections details a snippet from the main loop showing the key software flow.

The full code may be found at https://github.com/Samshabz/Skripsie

```
# Setup Lines...

# Phase 1: GNSS is Available and Inference Mode is On
for i in range(1, inference_images + 1):
    navigator.add_image(i, directory)
navigator.estim_pos(inference_images, Inference_Mode_On=True)
navigator.find_pixel_to_metre_factor()

# Phase 2: GNSS is Available and Inference Mode is Off
for i in range(inference_images+1, total_images + 1):
    navigator.add_image(i, directory)
```

```
# Phase 3: GNSS is Unavailable
navigator.estim_pos(total_images, Inference_Mode_On=False)

# Debug Lines...
```

3.5. Testing Shortlist

The following methods and techniques were tested, with some exclusions based on empirical results:

Feature Detectors: AKAZE, SuperPoint (with LightGlue matcher), and ORB. Note, the SuperPoint detector is used in conjunction with the LightGlue matcher to enhance performance.

Feature Matchers: FLANN and BruteForce. KNN.

Search Techniques: KNN with $K = 2$. Empirical tests showed a value of 1 to be ineffective since it excluded Lowe's ratio test, while values above 2 introduced excessive computational overheads. Further, radius search was seen to be unreliable due to the varying density of keypoints across datasets.

Planar Transformation Estimation: Homography, Affine, Partial Affine (Rigid) transformations using OpenCV, and SVD-based rigid transformations for rotation and translation estimations. Rotational and translational estimates were initially tested separately, due to the possibility of different responses to different prior stages and methods. However, it was seen that inter-method comparisons subtended equivalent conclusions; the methods were tested for visual brevity using a combined transform.

Global Image Similarity Measures: SSIM, histogram matching, local retrofit, and cross-correlation.

Optimization Techniques: Standard Deviation Filtering, LMEDS, RANSAC, Lowe's ratio test, n-Match thresholding, and absolute thresholding for match refinement. Empirical tests indicated that cross-checking was not applicable due to the excessive computational costs involved. KNN.

Chapter 4

Comparison of Methods

This chapter compares the performance of various methods used in the UAV navigation system, focusing on accuracy, runtime, and robustness. The evaluation includes feature detectors, local feature matchers, rotational and translational estimators, and optimization techniques. Each method is tested across multiple datasets to assess generalization and suitability for real-world UAV applications.

4.1. Testing Setup

This section outlines the framework used to evaluate the performance of the proposed UAV navigation methods. The evaluation focuses on three primary metrics: accuracy, runtime, and robustness. These metrics are critical to ensure that the navigation system can reliably operate under diverse and challenging conditions, reflecting real-world scenarios where the UAV may encounter varying environmental factors.

Note that understanding the testing setup is important for interpreting the subsequent results section.

4.1.1. Aim of Testing

The aim of these tests is to compare different methods, rather than to evaluate the overall system performance. The tests were conducted under intermediary development pipeline methods that are not listed, and not under optimized runtime or accuracy settings for the entire system; however, when testing methods within a particular stage, all other stages and parameters were held constant to ensure fair comparisons. Additionally, non-testing parameters and methods were set to optimal or near-optimal values to ensure that each method was evaluated under favorable conditions. This choice ensures that comparisons were still fair without having to iteratively retest as improvements were made. Therefore, the results presented are not indicative of the overall system performance, and no conclusions about the system's overall performance should be drawn from this chapter.

4.1.2. Datasets

Five distinct datasets were created to rigorously evaluate the methods' generalization and performance across diverse environments. These datasets were captured using Google Earth and involved both translational and rotational movements, simulating typical UAV

navigation tasks. Although the primary transformations were translation and rotation, there were subtle perspective and scale distortions introduced by the 3D rendering in Google Earth. The magnitude of these distortions is implicitly addressed by the degrees of freedom of the transformation estimation methods used, negating the need to quantify these distortions explicitly.

To maximize the utility of the datasets, the same images are used for both the GNSS-available and GNSS-denied stages. During the GNSS-available stage, all 15 images are streamed and stored along with their features and telemetry data. The first 5 images are used to infer the fixed pixel-to-meter factor, which is related to the camera focal length and UAV altitude. Subsequently, during the GNSS-denied stage, the 15 images have their location and heading estimated without GNSS data. Importantly, each image in the GNSS-denied stage recomputes its features and does not rely on any ground truth telemetry or use itself as a reference image, ensuring a fair test. Additionally, the images are spaced sufficiently to introduce challenges in the datasets.

The datasets have various properties to ensure that the tests are challenging yet feasible, and to mimic real-world data. The datasets were captured at altitudes corresponding to ground heights of 5–7 km, using images with a resolution of 1920×972 pixels. The specific resolution was chosen to eliminate image text that could cause false positive matches. The radial movement between frames varies between 300 and 700 pixels, depending on the image and the best match found. This corresponds to around 50% to 80% overlap between frames. Although the accuracy of the distance in metres is dependent on the translation size between frames, the method that yields the lowest meter error also produces the lowest overall error. Therefore, this variability does not impact the comparison of methods based on meter measurements. Each dataset consists of 15 images, balancing testing time with sufficient evaluation of the methods' performance.

The datasets used in this analysis encompass a range of terrains and motion types. The CITY1 and CITY2 datasets were captured in Cape Town, with CITY1 incorporating both rotational and translational changes between frames, whereas CITY2 focuses solely on translational movements and is taken at a significantly lower altitude, as per Table 5.3. The ROCKY dataset, collected in the semi-arid Karoo region, features rugged terrain with sparse vegetation and includes both translations and rotations. The DESERT and AMAZON datasets, captured in the Sahara Desert and Amazon Rainforest respectively, are marked by sparse, repetitive patterns that pose challenges for feature extraction and matching. These datasets involve both translations and rotations and are difficult even for human observers to distinguish frame-to-frame differences.

Examples of the datasets are shown below:

Figure 4.1: Examples of the CITY1 and CITY2 Datasets.

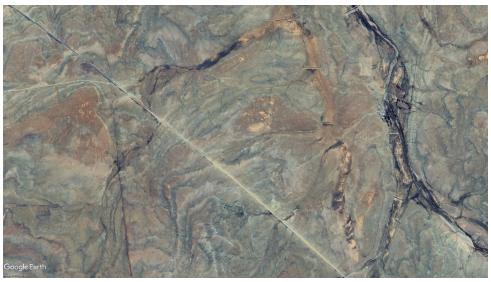


Figure 4.2: Example of the ROCKY Dataset.

Figure 4.3: Example of the DESERT Dataset.

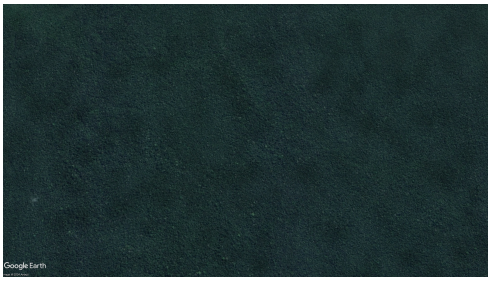


Figure 4.4: Example of the AMAZON Dataset.

4.1.3. Testing Structure

Each method is subjected to rigorous testing based on the following criteria:

Accuracy: Evaluated using the radial Root Mean Square Error (RMSE) of Lat-Lon estimations, in meters. This metric provides a clear indication of the method's accuracy in estimating the UAV's position. The accuracy is tested at the end of the pipeline, as errors in prior stages propagate to the final Lat-Lon error. The radial error is given as the mean of the radial errors for all images in the dataset.

Runtime: The runtime of the entire dataset is used for comparison. This includes the time to compute all 15 images for both the with GNSS signal and without GNSS signal phases. Intermediary stages propagate increases in time, and the runtime per line is not necessarily indicative of better performance; hence, the entire runtime is used for comparison. For instance, a method might filter out less keypoints, thereby running faster, but the subsequent stage will take longer due to the increased number of keypoints.

Robustness: This criterion evaluates the method's performance under variation in its parameters, assessing how difficult the method is to tune and how it performs with crudely chosen parameters across datasets. A scoring system from 1-5 is used, where 5 indicates a wide range of parameters offer near-perfect performance, and 1 indicates no static threshold works across datasets. 5 indicates a wide range of parameters offer near optimal performance across datasets; 4 indicates a large range of parameters offer good performance; 3 indicates a small range of parameters offer good performance; 2 indicates a small range of parameters offer usable performance; and 1 indicates that no

static threshold works across datasets.

4.2. Feature Detectors

This section presents the evaluation results of three feature detectors: ORB, AKAZE (dynamic keypoint targeting implementation), and SuperPoint with LightGlue. Feature detectors were applied to extracting a crude feature layer for rotational estimation, as well as a dense layer for rotational and translational estimation. Initially, all 3 transformations were tested independently for each detector, however, it was seen that the inter-method comparison conclusions were equivalent for each stage. Therefore, to maintain visual brevity, the detectors are applied to all stages and compared once, with the understanding of equivalent comparative independent responses to each stage.

Accuracy and Runtime

Figure 4.5 and Figure 4.6 present the radial error and runtime values for each feature detector across different datasets. AKAZE, utilizing dynamic keypoint targeting, demonstrated the highest accuracy across all datasets while maintaining reasonable runtime. SuperPoint recorded the highest radial errors, particularly in the AMAZON datset, where its training set did not generalize well to the dense, repetitive jungle environment.

Meanwhile, ORB proved to be the most efficient detector, making it suitable for applications requiring fast processing. SuperPoint demonstrated the longest runtimes across all datasets, highlighting its limited applicability for time-sensitive applications unless optimized with GPU acceleration.

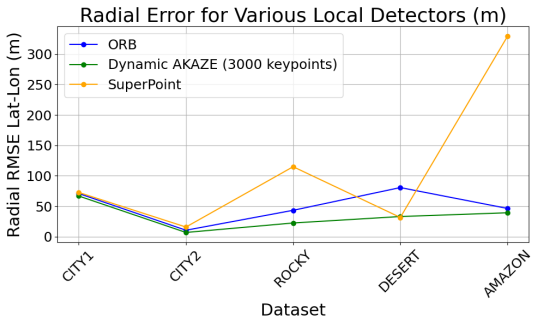


Figure 4.5: Radial Error for Various Feature Detectors.

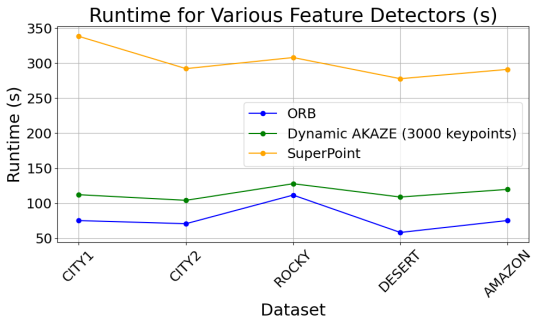


Figure 4.6: Runtime for Various Feature Detectors.

Robustness

All three methods were able to maintain their keypoint targets across environments; however, the quality of those keypoints. SuperPoint consistently achieved equivalent performance across multiple keypoint targets, earning it a robustness score of 5. AKAZE, with its dynamic targeting, was able to generalize well across datasets and only significantly dropped in performance when using below 1000 keypoints, earning a robustness score of 4.

ORB, while efficient, required precise tuning to achieve consistently good performance across datasets and was highly sensitive to the target parameter, earning a robustness score of 3.

Final Selection of Feature Detectors

Coarse Layer (Initial Detection): ORB was chosen for the crude, rotational estimation layer due to its balance of accuracy and efficiency. A range of 3000 - 8000 keypoints maintained reasonable accuracy and runtime, largely due to the invariance of the image similarity estimators to rotational inaccuracies. 3000 keypoints was chosen to prioritize speed and maintain FLANN runtime as per Figure 4.9.

Dense Layer (Refined Detection): Dynamic AKAZE was selected for the dense layer due to its consistent performance and robustness and applied to both rotational and translational estimation stages. A range of 3000-5000 keypoints maintained consistent runtime and accuracy. 3000 keypoints was chosen to balance runtime, accuracy, and maintain FLANN performance as per 4.9.

4.3. Local Feature Matchers

This section evaluates two prominent local matchers, BFMatcher and FLANN; LightGLue was implicitly tested in the feature detectors section with SuperPoint.

Accuracy and Runtime Evaluation

Figure 4.7 presents the radial error in Lat-Lon values for BFMatcher and FLANN across different datasets. The results indicate that while BFMatcher achieves slightly better accuracy in certain cases, FLANN remains highly competitive with only marginally higher RMSE values.

Due to their similar performance, the accuracy was evaluated at varying keypoint targets across datasets to see if their is a specific point of accuracy convergence in the two methods. The difference between accuracy in BFMatcher and FLANN, shown in 4.9, indicates that as the number of keypoints increase, the accuracy of both methods converges. Divergence occurs at values below 3000 keypoints, and this should be considered if using FLANN. Notably, FLANN sometimes outperforms BFMatcher (indicated by negative values) due to its approximate matching. For instance, FLANN may find a less similar secondary match for a valid primary match, allowing it to be retained through Lowe's ratio test, where BFMatcher would find a more similar secondary match and discard the valid primary match.

Figure 4.8 shows the runtime comparison for BFMatcher and FLANN across different datasets. FLANN consistently outperforms BFMatcher in terms of speed, with significantly lower execution times across all datasets. Specifically, in the CITY datasets, where the

number of keypoints found were significantly higher (no keypoint maximum at this stage), BFMatcher's runtime was significantly higher than FLANN's. This is attributed to FLANN's approximate matching, which scales better with the number of keypoints.

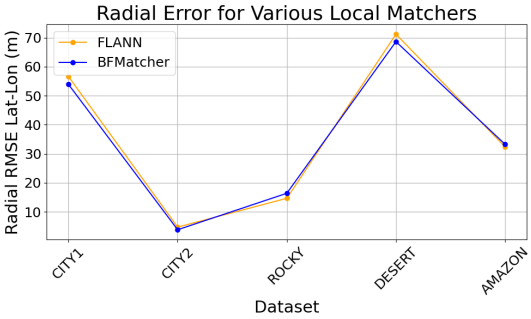


Figure 4.7: Radial Error for BFMatcher and FLANN.

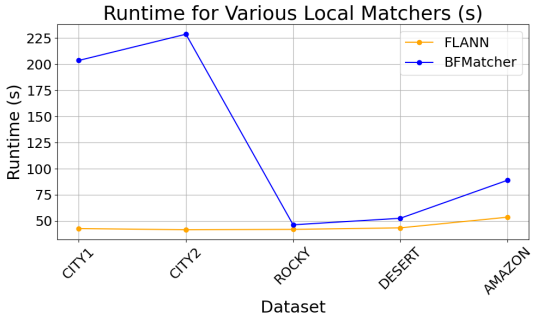


Figure 4.8: Runtime Comparison for BFMatcher and FLANN.

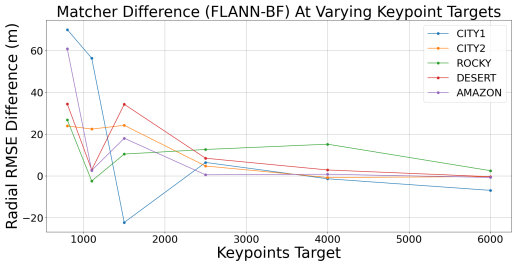


Figure 4.9: Convergence in RMSE Lat-Lon Error Between FLANN and BFMatcher Across Keypoint Targets.

Robustness Testing

BFMatcher did not have any tunable parameters, thereby scoring it a 5 for parameter robustness. FLANN had a few parameters that could be tuned, including the number of trees, the number of checks, and the search algorithm. However, FLANN maintained equivalent performance across these ranges; the default parameters were used.

Final Selection of Local Feature Matcher

Based on the comprehensive evaluation of accuracy, runtime, and robustness, FLANN emerges as the optimal choice for the UAV navigation system. FLANN offers significantly faster runtimes and better scalability while maintaining comparable accuracy to BFMatcher. However, at least 3000 keypoints should be targeted to ensure consistent performance across datasets.

4.4. Planar Transform Estimators

This section evaluates the performance of four planar transformation estimation methods: Partial Affine 2D (Rigid Transform plus minor scaling), Affine 2D, Homography, and

Rigid Transform via Singular Value Decomposition (SVD). As noted in Section 3.5, both rotational and translational stages are combined into a single transform evaluation since the conclusions were similar for both stages. For these tests, RANSAC was used on all methods to filter outliers, except for the Rigid Transform via SVD, which does not have a built-in outlier rejection mechanism.

Accuracy and Runtime Evaluation

Figures 4.10 and 4.11 summarize the RMSE and runtime values across datasets for each planar transformation estimation method. The results indicate that while each method exhibits strengths in different datasets, the Rigid Transform via SVD and Partial Affine 2D consistently emerge as the most accurate methods for rotational and translational estimations. The Rigid Transform via SVD slightly outperforms Partial Affine 2D in most datasets. This slight advantage is attributed to its strict modeling of rigid transformations, which aligns closely with the characteristics of the datasets.

The runtime results, shown in Figure 4.11, are influenced by the degrees of freedom in each method. Methods with fewer degrees of freedom, such as rigid transforms, demonstrate better performance across datasets. The Rigid Transform via SVD proves to be the fastest due to its lack of iterative optimization.

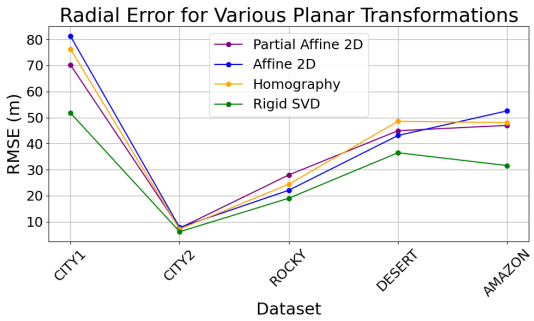


Figure 4.10: RMSE Comparison Across Datasets for Planar Transform Estimators.

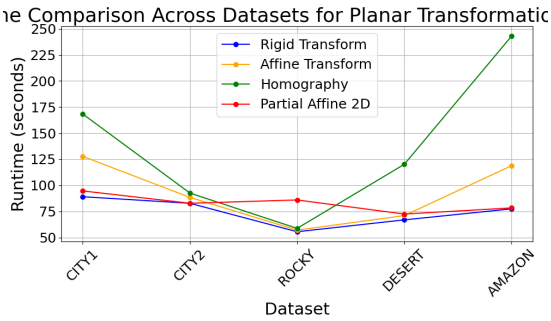


Figure 4.11: Runtime Comparison Across Datasets for Planar Transform Estimators.

Robustness Testing

The OpenCV methods (Partial Affine 2D, Affine 2D, Homography) utilize RANSAC, or less commonly LMedS or other outlier rejection methods, for robust outlier rejection. The thresholds for these methods significantly affect accuracy and require extensive testing to determine optimal values. For more complex, three-dimensional transformations, adjusting these thresholds may be more beneficial. However, in this application, none of these methods outperformed the accuracy of the Rigid Transform via SVD under any threshold. Thus, the OpenCV methods achieve a robustness score of 3, while the Rigid Transform via SVD achieves a score of 5 due to its lack of tunable parameters.

Final Selection of Rotational Estimator

Based on the comprehensive evaluation of accuracy, runtime, and robustness, the Rigid Transform via SVD emerged as the most suitable estimator for the UAV navigation system. It demonstrated the lowest combined radial RMSE in latitude and longitude across all datasets and the fastest runtime, largely due to its alignment with the application-specific transformations. Furthermore, it required no parameter tuning, making it highly suitable for real-time UAV applications.

4.5. Image Similarity Estimators

Accurate image similarity estimation, or global matching, is essential for UAV navigation systems to select appropriate reference images for comparison. Effective similarity estimators should provide accuracy and efficiency while maintaining robustness against small rotational offsets. The proximity radius for initial search space reduction was crudely set to include the five closest images, but this parameter is dependent on external factors such as the UAV's speed and image capture rate and is not the focus of this evaluation. This section specifically evaluates the global matching techniques used to estimate the similarity between images to identify the closest match for subsequent heading and position estimation.

Accuracy and Runtime Evaluation

The effectiveness of the global matching methods is reflected in the Lat-Lon estimation error, as more similar matches yield a higher number of good matches and subsequently lower estimation errors.

Figures 4.12 and 4.13 summarize the RMSE values (in meters) and runtime comparisons for the Local Retrofit, Cross-Correlation, Histogram, and SSIM methods. The results indicate that the Histogram and Cross-Correlation techniques perform nearly equivalently in terms of accuracy, with SSIM slightly behind. The Local Retrofit method recorded the highest RMSE values, especially in the DESERT dataset. Its crude detection and matching allowed many false positives, leading to random choices of the best reference match. Consequently, it was excluded from further analysis.

The Histogram technique demonstrated the most consistent runtimes, followed closely by Cross-Correlation. SSIM exhibited longer runtimes due to its more complex structural similarity calculations.

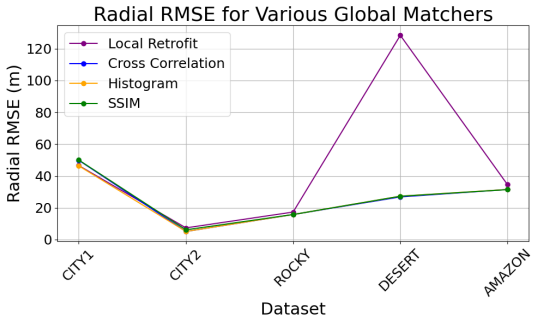


Figure 4.12: RMSE Comparison Across Datasets for Global Matching Techniques.

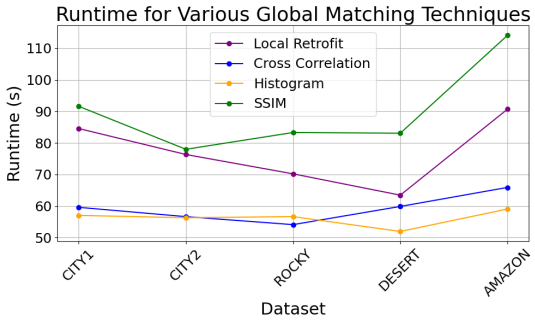


Figure 4.13: Runtime Comparison Across Datasets for Global Matching Techniques.

Robustness Testing

The global matching methods did not have tunable parameters, earning them a robustness score of 5. In contrast, the Local Retrofit model had various parameters that required precise tuning, such as detector type, detector threshold, grid size, and match threshold. Balancing runtime and accuracy was challenging with this method, and it required extremely precise tuning to achieve usable performance. As such, the Local Retrofit model achieved a robustness score of 2.

Considerations for Alignment Prior to Global Matching

This section details the considerations when selecting the level of precision for the rotational alignment techniques employed to align image pairs for unbiased similarity comparisons. To assess the impact of rotational misalignment, the estimated internal alignment angle was intentionally skewed, and the response from the global matcher was observed in terms of the resulting Lat-Lon error.

Initially, a 5-degree skew was introduced, and no significant changes in position estimation were observed. Figure 4.14 shows the percentage change in Lat-Lon error from the no-skew estimate when a 10-degree skew was applied. Cross-Correlation maintained the most consistent accuracy, closely followed by Histogram and then SSIM. All methods demonstrated high robustness to rotational misalignments up to 5 degrees but showed degradation beyond this threshold. Therefore, when employing a rotational alignment technique prior to global matching, a 5-degree misalignment is tolerable, allowing the choice of detector and matcher to be guided by efficiency rather than precision.

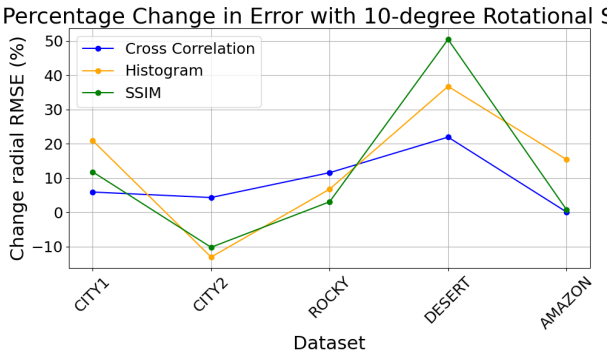


Figure 4.14: Percentage Change in Lat-Lon Error with 10-degree Rotational Offset

Final Selection of Global Matching Technique

Based on the comprehensive evaluation of accuracy, runtime, and robustness, the Histogram technique was identified as the most suitable global matching method for the system. Histogram consistently provided superior performance in terms of both RMSE and runtime while maintaining sufficient robustness to rotational error.

4.6. Optimization Techniques

Several methods were employed to enhance the performance of the UAV navigation system, focusing on filtering image matches to balance noise and maintain stability within the point sets. These optimization techniques are simpler to implement and were not assessed with the same depth as the methods in previous sections. Since precise parameter optimization was not the primary goal of this pipeline, parameters were tuned approximately, with functional ranges documented.

Planar Transform Outlier Rejection Methods

Two outlier rejection methods, LMedS (Least Median of Squares) and RANSAC (Random Sample Consensus), were evaluated for match filtration. Both methods performed nearly equivalently, with LMedS displaying a slightly lower radial error. LMedS was also significantly faster than RANSAC, making it the preferred choice for planar transform outlier rejection. The results are summarized in Figures 4.15 and 4.16. Both methods require precise tuning of thresholds to achieve optimal performance across datasets, earning them a robustness score of 3. However, these techniques were not included in the final optimal pipeline due to the superior performance of the Rigid Transform via SVD without outlier rejection.

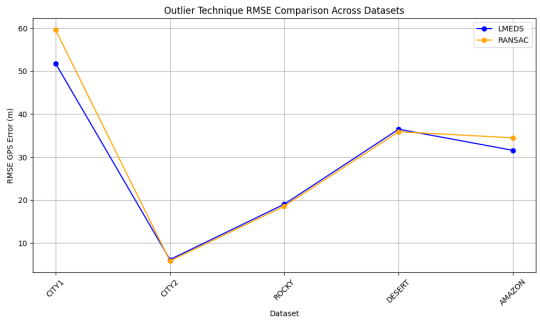


Figure 4.15: Radial Lat-Lon RMSE Comparison Across Datasets for LMEDS and RANSAC.

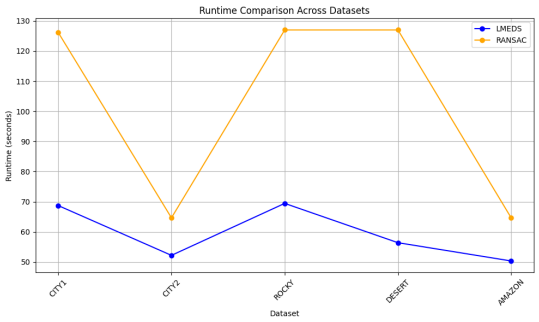


Figure 4.16: Runtime Comparison Across Datasets for LMEDS and RANSAC.

Lowe's Ratio Test

Lowe's Ratio Test was utilized to filter keypoint matches by comparing the distance of the best match to that of the second-best match. Initially, a static threshold was applied to determine match quality. However, this static approach was insufficient for handling variability across diverse datasets, resulting in inconsistent accuracy. To improve robustness, a dynamic thresholding strategy was adopted. This approach involves setting an initial threshold and incrementally increasing it, thereby allowing greater leniency until a predefined number of matches or a percentage of the keypoints is found.

A lower initial threshold and smaller increment value increase the likelihood of approaching the desired match count but impact runtime. An initial threshold of 0.7 and an increment of 0.05 were selected to balance efficiency and accuracy. Working ranges for reliable performance were found to be between 0.5–0.75 for the initial threshold and 0.025–0.1 for the increment.

Through testing, a target of 500 matches was selected for its ability to consistently maintain stability across diverse datasets. The working range for reliable performance was found to be 300 to 700 matches. Additionally, to prevent low-quality matches from entering the system when fewer keypoints were detected, a maximum match-to-keypoint ratio of 75% was implemented to ensure sufficient keypoints were found.

N-Match or Absolute Thresholding

Absolute Thresholding, or N-Match Thresholding, involves filtering keypoint matches based on a fixed number of matches or a specific descriptor distance threshold. Filtering based on a specific descriptor distance threshold proved to be more effective in generalizing across datasets and was subsequently chosen.

The method was applied prior to Lowe's Ratio Test filtering, with a lenient threshold, to aid in computation for subsequent stages without removing potential matches. This step was especially beneficial in reducing the number of matches in instances when the

detector found an abnormally high number of keypoints in specific images.

During the testing phase, various match threshold values between 700 and 2500 keypoints were found to perform relatively equivalently. Thresholds below this range were too restrictive, while those above had negligible effect. A threshold of 1000 matches was chosen due to significant gains in runtime relative to the upper limit, with negligible impact on accuracy.

4.7. Summary

The following methods were selected for the optimal pipeline, used in the results section, based on the comprehensive evaluation of accuracy, runtime, and robustness across diverse datasets:

The chosen feature detector was ORB with 3000 keypoints for the coarse detection layer and AKAZE with 3000 keypoints for the dense detection layer. FLANN was selected as the local matcher, and the Rigid Transform via SVD was chosen for planar transformation estimation. Image similarity was evaluated using histograms. The optimization techniques applied were initial N-Match Thresholding with 1000 matches followed by Lowe's Filtering with 500 matches.

Chapter 5

Results

5.1. Performance Analysis

This section presents the results achieved by the optimal navigation system pipeline across various challenging datasets. The analysis evaluates the system's accuracy, efficiency, and generalizability, providing insights into its applicability in real-world UAV navigation scenarios. Details about the testing setup are provided in Chapter 4.

5.1.1. Key Metrics

The system's performance across the five diverse datasets—CITY1, CITY2, ROCKY, DESERT, and AMAZON—is evaluated based on two primary metrics: **Accuracy** and **Runtime**.

Accuracy is assessed in two primary ways:

1. *Absolute Radial Error (RMSE)*: This metric measures the root mean square error of the estimated positions compared to the ground truth positions, calculated as the average radial distance (in meters) between the estimated and true positions over the dataset. It provides a tangible understanding of the error magnitude. Additionally, errors may be represented per image, in pixels or in axial components, offering an intuition about the error size relative to the fixed image resolution of 1920×972 pixels.

2. *Relative Radial Error*: This metric expresses the per-image absolute radial error as a percentage of the ground truth radial displacement, averaged over the dataset. The ground truth radial displacement is the magnitude of the translation vector from the reference image's center to the current image's center, measured in meters. This normalized metric allows for a clearer understanding of the system's accuracy relative to the movement size. Errors may also be represented per image or in axial components.

Runtime is evaluated in three scenarios:

1. *Mean Add Time (With GNSS)*: The average time required to add one image to the pipeline during GNSS availability. This includes streaming the image and extracting its features. This time determines the system's processing rate while GNSS signals are available after parameter inference has concluded.

2. *Mean Parameter Inference Time (With GNSS)*: The average time needed per image to infer the pixel-to-meter conversion factor. This includes processing the entire

pipeline—finding the best match, estimating the transformation to estimate the UAV's position—and performing linear regression using the ground truth (once at the end of this mode). Adding this to the Mean Add Time provides the total time to process an image when GNSS is available and parameter inference is active. The inference mode is active for the first five images; further processing provided negligible improvements in accuracy.

3. *Mean Location Inference Time (Without GNSS)*: The average time taken per image to infer the UAV's location when GNSS signals are lost. This time encompasses processing the entire pipeline to estimate the UAV's position. This metric is critical as it determines how quickly a pilot can correct any path deviations to prevent further drift.

5.1.2. Requirements Compliance

This section provides a concise summary of the system's ability to meet the outlined requirements, as detailed in Section 1.5. Table 5.1 shows the system's compliance with the requirements; detailed performance evaluations are presented in the subsequent sections. The system meets both the accuracy and runtime requirements across all datasets, indicating its adaptability

Table 5.1: Compliance with Requirements for Radial Error and Processing Time

Requirement	CITY1	CITY2	ROCKY	DESERT	AMAZON
Max Radial Error $\leq$ 10%	0.6356%	0.2715%	6.2123%	0.3516%	0.7447%
Max post-GNSS Inference Time $\leq$ 2s	1.3609s	1.6271s	1.4374s	1.8392s	1.7091s
Max with-GNSS Time $\leq$ 5s	2.0377s	2.1342s	2.2412s	2.2418s	2.3975s

5.1.3. Detailed Results

This section presents the results, while Section 5.1.4 discusses the sources of error in the system.

In terms of accuracy, as shown in Figures 5.1 and 5.2, all datasets achieved a mean radial error below 2% of the ground truth displacement, a mean pixel error below 1.05 pixels, and a mean positional error below 3.2 meters. Evaluating the percentage error, the datasets' performance from best to worst is as follows: CITY2, DESERT, AMAZON, CITY1, and finally, with a significant difference in performance, ROCKY.

Regarding maximum single-image errors, the highest error occurred in the ROCKY dataset, with a maximum radial error of 6.21% of the ground truth displacement, corresponding to 27.54 meters or 4.93 pixels.

For runtime performance, as shown in Figure 5.3, the mean add (feature extraction) time across datasets was below 0.6 seconds, the mean parameter inference time was below 1.4 seconds, and the mean location inference time was below 1.3 seconds. This indicates that the system maintains real-time performance even during parameter inference, with the sum of the mean parameter inference and add times being below 2 seconds (within the 5-second requirement), and the location inference time below its requirement of 2 seconds.

Regarding maximum single-image runtimes, the system remained under the crucial 2-second threshold following GNSS signal loss. During the with-GNSS phase, the system produced runtimes slightly above 2 seconds while inference mode was active (combining the add time and parameter inference time). Once inference mode was off, this runtime stayed below 2 seconds. Over the entire flight, these occasional longer runtimes would be offset by shorter processing times when inference mode was off, resulting in an overall mean runtime below 2 seconds, well within the 5-second requirement.

Finally, these results are met across all datasets. The system's performance is consistent across diverse terrains, demonstrating its robustness and adaptability to varying environmental conditions without any dataset-specific tuning.

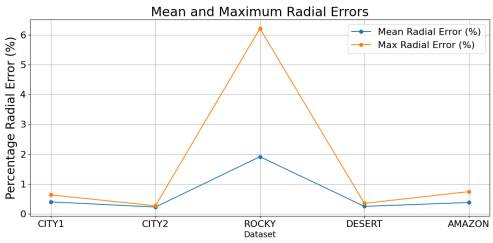


Figure 5.1: Mean Percentage Error Across Datasets (Movement Size Normalized).

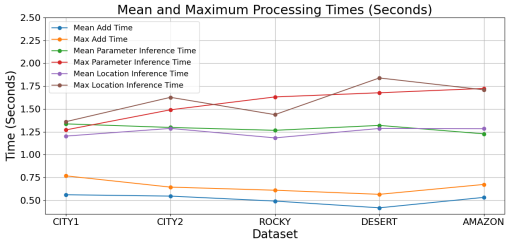


Figure 5.2: Mean Pixel and Metre Error Across Datasets.

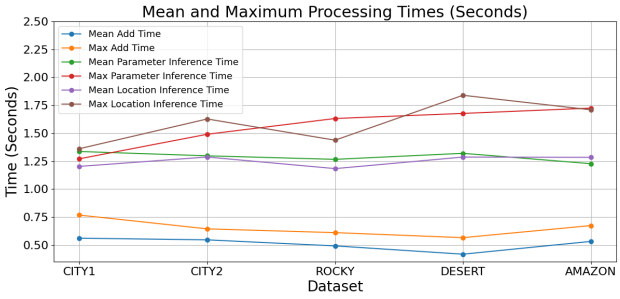


Figure 5.3: Mean and Max Runtime Performance Across Datasets.

5.1.4. Visual Results

Figures 5.4 and 5.5 illustrate the actual versus estimated flight paths of the UAV in the ROCKY and DESERT datasets, respectively. Even in the worst-performing dataset, ROCKY, the system maintains a highly accurate flight path on a broader scale, demonstrating that the errors are negligible for practical purposes.

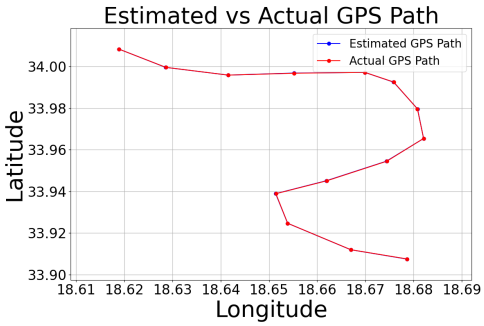


Figure 5.4: Flight Path of UAV in Rocky Dataset (Worst)

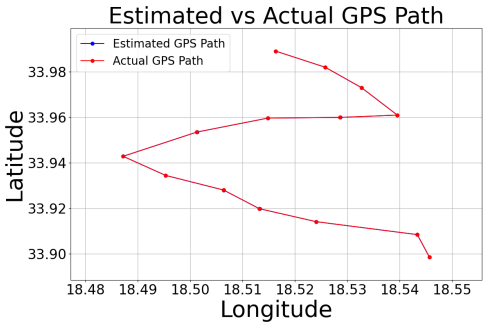


Figure 5.5: Flight Path of UAV in Desert Dataset (Best).

Figure 5.6 presents examples of the bottom 50 matches, after filtering, in the CITY2 dataset. This illustrates the reliability of the feature matching on a broader scale.

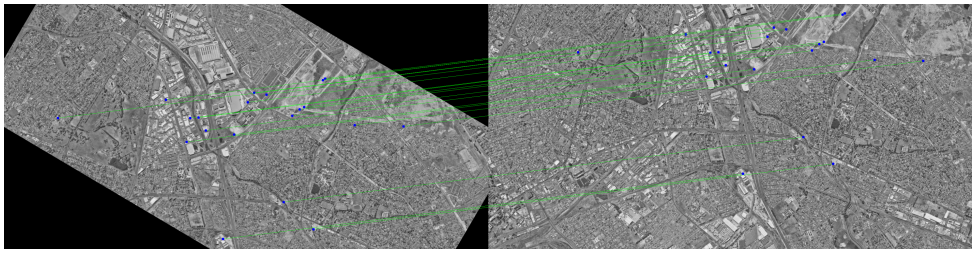


Figure 5.6: Matches Between Two Images in CITY2 Dataset.

Figure 5.7 shows the overlay of two images from the CITY2 dataset after applying the estimated rotation and translation. This visual representation demonstrates the system's ability to accurately compute the transformation between images. The visible borders result from the rotation and translation causing parts of the images to extend beyond the canvas boundaries. The near-perfect alignment of the images indicates the effectiveness of the transformation estimation. Notably, slight blurring near the outer edges of the overlaid section suggests minor discrepancies. Specifically, the outer pixels move at different rates compared to those at the center because of the varying angles at which light enters the camera lens. While these are minor sources of error in this study, cameras with significant fisheye distortion might experience more pronounced errors in these regions.

Figure 5.7: Overlay of Two Images in CITY2 Dataset.

Figure 5.8 illustrates the distribution of pixel deviations in the X and Y directions across the datasets. To improve clarity, some axial values are grouped, and the deviations

are quantized into bins; thus, the values represent ranges rather than exact pixel offsets. This visualization provides an intuitive understanding of the pixel error distribution and any potential axial bias. As shown, the errors are distributed relatively evenly across both axes, indicating no significant axial bias in the system. Moreover, the majority of errors are within 2 pixels, demonstrating the system's robustness and accuracy in localization estimates. The largest radial error, as noted in Figure 5.8, is 4.92 pixels. While this error is relatively small, the sources of these errors are discussed in the following section.

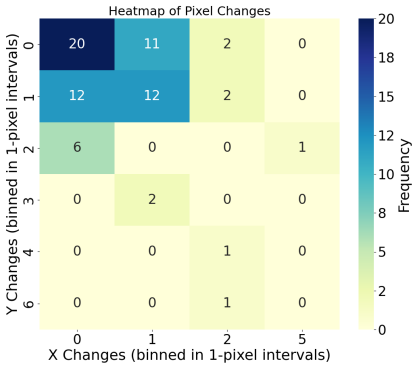


Figure 5.8: Heatmap of Pixel Deviations in X and Y Directions

Analysis of Identified Sources of Error

The system's maximum error was observed to be 6.2% of the UAV displacement on the ROCKY dataset. Additionally, notable outliers were detected in the CITY1 and AMAZON datasets, with maximum errors of 0.64% and 0.74%, respectively. Below, the sources of these errors are analyzed, highlighting their prevalence across all datasets and their amplified effects in specific cases.

Differences in Overlap: Although the mean performance across all datasets is relatively good, a significant outlier was observed in one image in the ROCKY dataset. This is attributed to the step size, or displacement, between that image and its best reference match. The radial translation between these images is approximately 873 pixels—the largest step size by a considerable margin across all datasets. This large displacement reduced the mutual overlap to 59.4%, the lowest among all datasets, leading to a higher error of 6.2% of the UAV displacement. Therefore, maximizing the path overlap is crucial for maintaining a high level of accuracy, as insufficient overlap was the primary reason for the poorer performance in the ROCKY dataset.

Quantization Effects: When images are captured from slightly different positions, sub-pixel shifts can cause feature edges to distort. At a fixed resolution, details smaller than a pixel cannot be accurately represented, leading to potential inaccuracies due to averaging. For large features, the impact of this distortion is minimal relative to the feature size. However, smaller features experience significant boundary distortions, causing high levels

of descriptor and keypoint inaccuracies. This effect is exacerbated at higher altitudes where features are predominantly small, impacting descriptor fidelity and keypoint precision. The CITY datasets, with their dense, small-scale features, are especially affected, resulting in increased sub-pixel errors. While CITY2 experiences sub-pixel shifts as well, its simpler frame-to-frame translations (without significant rotation) help reduce the overall error, making the effect less pronounced.

Depth and Perspective Variations: Google Earth's 3D terrain model introduces changes in perspective and ground height as images are captured from different positions. These variations are intensified by significant altitude changes in the landscape. The ROCKY dataset, characterized by pronounced elevation differences in mountainous areas, demonstrates the effects of these perspective distortions and scale discrepancies.

Homogeneous and Repetitive Terrain: The effectiveness of keypoint detection relies on the presence of unique, high-contrast features. Similarly, the accuracy of matching depends on whether the local environment surrounding a feature is sufficiently distinct. In homogeneous and repetitive environments, such as the AMAZON dataset, feature detectors struggle to identify a diverse set of keypoints, and matchers face challenges in selecting correct correspondences. Although modern computer vision techniques mitigate this issue to a large extent, in highly repetitive terrains, even these advanced methods encounter difficulties.

Variability in Terrain and Environmental Conditions: Each dataset's terrain has unique attributes, such as the density of good features and the uniqueness of keypoints. Optimal feature counts vary with terrain type; however, this study applied a uniform target number of keypoints and matches across all datasets. This static approach does not fully account for the variability in feature distribution and environmental conditions, leading to generalized performance rather than terrain-specific optimization.

Optical Distortion Effects: Camera lenses cause distortion, particularly at the edges of the frame. This occurs because light from the edges enters the lens at steeper angles compared to the center, resulting in different perceived motion. This change in angle affects how features move across frames, making the outer pixels appear to move differently than those at the center. This non-uniform movement can lead to discrepancies in pixel displacement, impacting keypoint detection and matching accuracy.

Algorithmic Limitations: The image processing and matching algorithms employed have intrinsic constraints, particularly under challenging conditions or when handling repetitive features. Optimized for computational efficiency, these algorithms may not capture every image feature perfectly, contributing to potential errors in feature detection and matching.

5.2. Adverse Conditions Evaluation

To ensure the system's robustness and practical viability, it is essential to evaluate its performance under various challenging conditions. This section presents tests conducted under reduced resolution, which is particularly relevant for UAVs with limited computational resources. Although dynamically varying lighting conditions and low reference image overlap are also important, they are not included in this evaluation due to current limitations in test data and scope.

5.2.1. Low Resolution Testing

This section evaluates the system's performance under varying resolution factors, relative to the original 1920×972 resolution. The analysis focuses on the trade-off between navigational accuracy and computational efficiency as the resolution decreases. The goal is to determine whether the system can maintain performance in scenarios where the UAV has limited power and computational resources and can only operate at lower resolutions.

Results

The results, as shown in Figures 5.9 and 5.10, indicate that the system maintains stable performance with minimal changes in accuracy until the resolution is reduced to an absolute resolution of 336×221 pixels. At this point, there is a sharp increase in error in the AMAZON dataset, indicating that the system's accuracy becomes significantly compromised. Interestingly, the accuracy remains relatively constant down to this critical resolution, suggesting algorithmic robustness to lower levels of detail.

In terms of runtime, the location inference time decreases as the resolution decreases. A reduction of the resolution by half, to 960×486 pixels, results in slightly less than a 50% decrease in runtime compared to the original resolution. This decrease is due to the reduced number of features and matches possible, leading to a lower computational load. Importantly, the system's accuracy is not compromised at this resolution.

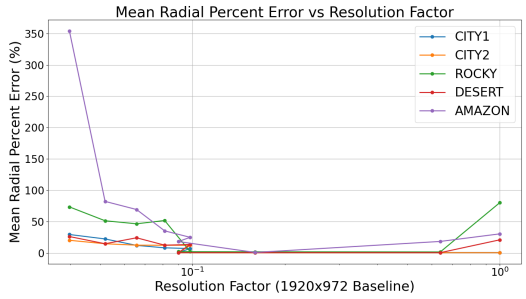


Figure 5.9: Radial (%) Error vs Resolution Factor

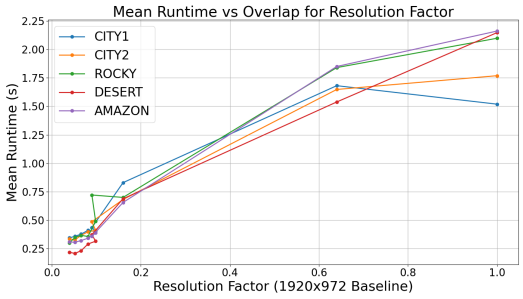


Figure 5.10: Location Inference Time (s) vs Resolution Factor

Conclusion

The evaluation shows that the system can maintain stable performance with minimal changes in accuracy at resolutions slightly above 336×221 pixels. However, below this resolution, the system's accuracy becomes significantly compromised, leading to a sharp increase in error. The system's runtime decreases linearly as the resolution decreases, with an almost 50% reduction in runtime observed at a resolution of 960×486 pixels while maintaining accuracy. This indicates that the system can operate effectively at lower resolutions, providing a viable solution for UAVs with limited computational resources.

5.2.2. Overlap Testing

This section evaluates the system's performance as the overlap between images decreases. Overlap is defined as the percentage of pixels shared between images relative to the image size. By analyzing the system's accuracy and runtime as overlap decreases, the minimum overlap required for reliable navigation and real-time performance my be identified.

Methodology

To simulate a decrease in overlap, images from the CITY2 dataset are progressively cropped. Since the CITY2 dataset involves only translational changes, calculating the overlapping pixels simplifies to considering the difference between the image size and the translation vector. Cropping is performed incrementally by 50 pixels per iteration, scaled in the x-direction according to the aspect ratio, alternating between x and y crops. Because translation vectors vary between images, the overlap is presented as that of the bottleneck image (the one with the lowest overlap) for each dataset to maintain clarity.

Results

The accuracy plot in Figure 5.11 shows that the error remains nearly constant until the overlap drops below approximately 30%, after which the error starts to increase significantly. It is important to note that below the lowest tested value, the overlap of the bottleneck image was below 0%, so further error measurements are not shown. An interesting observation is that the error decreases again after initially increasing at very low overlap values. However, this reduction is unreliable and results from coincidental high densities of good matches despite the low number of overlapping pixels. Therefore, extremely low levels of overlap cannot be relied upon for consistent accuracy. Maintaining an overlap above 30% ensures stable and accurate navigation.

The runtime plot in Figure 5.12 indicates a linear relationship between runtime and overlap. As expected, runtime decreases as overlap decreases because fewer features and matches are processed, reducing computational load. The rate of decrease is slightly

above 0.1 seconds per 10% reduction in overlap. While runtime decreases moderately with reduced overlap, the potential for decreased accuracy and instability may outweigh the benefits. Minor crops can be used once the path is established to slightly reduce computational load, but periodically referencing the full image is essential to maintain accurate positioning.

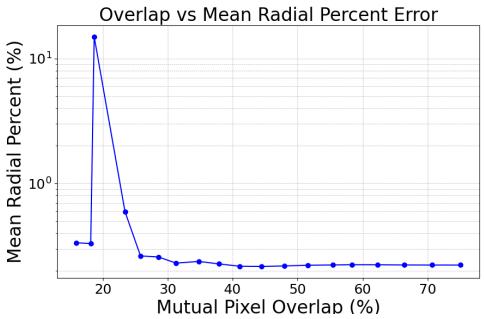


Figure 5.11: Overlap Percentage vs. Mean Error Percentage.

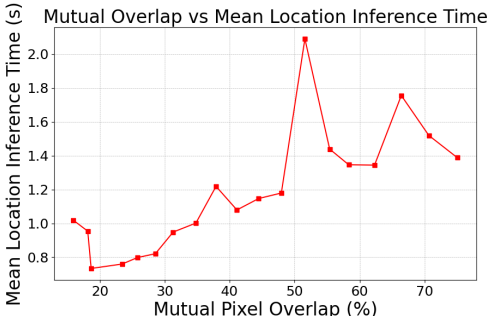


Figure 5.12: Overlap Percentage vs. Mean Localization Time.

Conclusion

The results demonstrate that the UAV can maintain reasonable accuracy until the overlap drops below 30%. While runtime decreases with reduced overlap, the marginal runtime benefits may not justify the increased risk of navigation errors. Therefore, maintaining an overlap a margin above 30% is recommended for reliable navigation, with above 60% being optimal as per 5.1.4, with occasional full-image references used to verify the integrity of the estimates.

5.2.3. Low-Light Testing

In UAV navigation, long missions may require the UAV to return along the outbound path at a significantly different time from the outbound journey, potentially under different lighting conditions. In this test, the lighting conditions of the reference images are altered relative to those of the current images, simulating evening and nighttime conditions. This adds an extra layer of difficulty beyond simply changing all images to low-light conditions; however, the ability to operate under low-light conditions is also implicitly tested here.

Since darker images have less uniqueness and contrast, the number of features found may become too low if the dynamic detector threshold is not adjusted. Therefore, the detector threshold is adjusted for the simulated nighttime conditions, acknowledging that this more lenient threshold leads to more keypoints and subsequently higher runtime. In practice, the detector threshold should be recalculated periodically to ensure a consistent number of keypoints are found across images taken at different times.

Methodology

This section evaluates the robustness of the navigation method under simulated evening and nighttime conditions across five datasets: CITY1, CITY2, ROCKY, DESERT, and AMAZON. The parameters used to simulate the lighting effects are shown in Table 5.2, with no effects applied for daytime conditions. Nighttime conditions are simulated through darkening and adding noise, rather than actual nighttime conditions which would include dynamic lighting changes. The alpha parameter controls contrast, while beta reduces the brightness of the image. The noise sigma adds Gaussian noise to mimic low-light imperfections. The weight balances the processed image and Gaussian noise effect for realism in night simulations, with higher values implying that most of the image remains unchanged. To ensure that all values are visible on the plot, the percentage increase from daytime error is used, alongside a logarithmic scale on the y-axis.

Table 5.2: Parameter Settings for Different Lighting Effects

Effect	Alpha	Beta	Sigma	Weight (main)
Early Eve	0.7	-15	10	0.97
Late Eve	0.6	-30	15	0.95
Late Night	0.5	-50	20	0.92

Representative examples from the CITY1 dataset illustrate these conditions:

Figure 5.13: Daytime.

Figure 5.14: Early Evening.

Figure 5.15: Late Evening.

Figure 5.16: Night-Time.

Simulated Low-Light Conditions in Cape Town reproduced from [11].

Results

Figure 5.17 illustrates the percentage increase in error under each lighting condition across datasets, relative to the daytime performance.

The CITY1 dataset shows the smallest increase in error across all conditions, maintaining an error increase under 10%. CITY2 follows, with slightly higher increases in error as lighting conditions become more challenging. This could be attributed to CITY2's minimal rotations, which do not implicitly reduce non-mutual information and may allow more false positives under difficult conditions compared to CITY1. Further, the lower altitude of CITY2 compared to CITY1, as per 5.3 implies a lower number of distinct features and subsequently lower contrast range in the dataset, leading to poorer identification

in low-light scenarios. The strong performance of the CITY datasets can be attributed to their dense feature environments with numerous high-contrast elements. However, in real-world scenarios, city lights—unaccounted for due to dataset limitations—could introduce dynamic changes impacting accuracy, especially if they occupy a significant portion of the view.

The ROCKY dataset ranks third in performance, with the DESERT and AMAZON datasets experiencing the largest increases in error. The substantial error rise in these datasets is due to their already sparse environments becoming even more feature-poor under low-light conditions, significantly increasing navigation error. The late evening and nighttime settings are particularly challenging, with errors exceeding 100 times the baseline daytime error in these sparse environments.

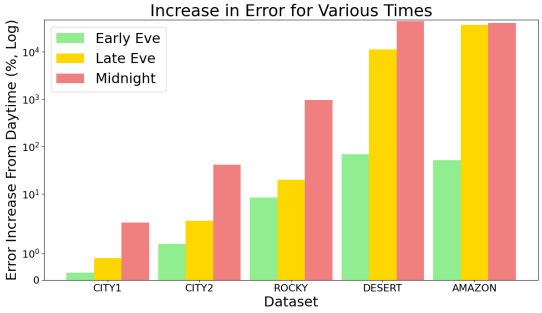


Figure 5.17: Mean percentage increase in error under nighttime and evening conditions.

Conclusion

Despite the high relative error percentages, the actual error is only really understood when considered relative to the ground truth distances rather than to the baseline, daytime percentage error. In all datasets, during early evening conditions, this error remains below 5% of the mean ground truth distance between compared images. In denser environments like the CITY datasets, errors remain under 2% across all lighting conditions. For sparser environments, it is important to mitigate illumination changes during image capture, although the pipeline demonstrates robustness to moderate changes in illumination.

5.2.4. Static Camera Tilt Testing

In practice, the camera may not be perfectly aligned facing directly downward. This test evaluates the system's performance under a static forward tilt (pitch) of 7 degrees, exceeding the typical offset a camera may experience due to improper mounting. Radial error is measured in meters to provide a physical understanding of the system's performance with and without the tilt.

As shown in Table 5.3, the system maintains a mean error below 26 meters across all datasets during tilt, with the effect varying depending on the mean ground distance and

Table 5.3: Accuracy Results of Tilted Camera Test

Metric	CITY1	CITY2	ROCKY	DESERT	AMAZON
With Tilt Error (m)	11.13	12.04	17.20	25.11	14.53
No Tilt Error (m)	4.80	2.76	5.88	2.36	2.53
Ground Height (km)	6.78	5.21	5.14	5.00	5.70

thus the perspective change. The system's robustness to camera tilts is evident, with the error comparable to that without the tilt. This indicates that the system can effectively handle static tilts.

5.3. Conclusion

The results presented in this chapter confirm that the system meets the accuracy and time requirements established in the objectives. The system demonstrated consistent and strong accuracy and runtime performance across various datasets, including sparse-feature environments, showcasing its ability to generalize effectively. Additionally, in practical scenarios, once the path is established, there will be significantly less translation between the current and reference images, leading to lower errors *ceteris paribus*.

Furthermore, the applied stress tests showed the pipeline's robustness to low resolution, reduced overlap, static camera tilts, and low-light conditions, with the system maintaining stable performance under these adverse conditions. These results underscore the system's versatility and effectiveness in real-world UAV navigation scenarios, providing a reliable and efficient solution for autonomous navigation.

However, it is noted that sparse datasets do not perform optimally under large variations in lighting, and maintaining an overlap above 30% is critical to keeping errors under 5% of the UAV's displacement.

Overall, the system has proven to be a versatile and effective solution for UAV navigation, showing remarkable potential to sustain operation under a wide range of operational challenges and environmental conditions.

Chapter 6

Conclusion

Global Navigation Satellite System (GNSS) vulnerabilities, such as jamming and spoofing, pose significant risks to Unmanned Aerial Vehicle (UAV) navigation, potentially leading to loss of control and mission failure. Recognizing the limitations of existing alternatives—which are often prohibitively costly or unreliable—this project addressed the urgent need for a robust, GNSS-independent navigation system for UAVs operating in GNSS-denied environments. The primary objective was to develop an accurate, adaptable, and generalizable image-based navigation system to serve as a redundancy to GNSS.

To achieve this objective, effective methods for feature extraction, matching, similarity computation, and planar transformation estimation were carefully selected and integrated to optimize the overall system pipeline. The implemented system was thoroughly tested across diverse datasets representing various terrains and environmental conditions.

The results demonstrated that the system successfully met all outlined objectives. The maximum radial error remained below 10%, with 4 out of 5 datasets exhibiting errors below 1% of the radial image displacement. The system maintained real-time performance during both the GNSS-available phase—where it captured features along the outbound path—with response times below 5 seconds, and during GNSS-denied scenarios—with response times consistently under 2 seconds across all datasets.

Despite challenges such as terrain variability, quantization errors, depth changes, and optical distortions from camera lenses, the system maintained robust accuracy across diverse datasets. Its resilience was further evidenced under varying lighting conditions, static camera tilts, reduced reference image overlap, and changes in resolution, showcasing its adaptability to real-world operational demands. However, it is important to acknowledge that the system's performance was notably impacted under extreme low-light conditions and image overlaps below 30%, suggesting areas for future improvement.

In conclusion, this project successfully developed a versatile and effective image-based GNSS-redundant navigation system for UAVs, demonstrating consistent and reliable performance across a wide range of terrains and operational conditions. The findings validate the feasibility of image-based navigation systems as a viable redundancy measure to address the vulnerabilities associated with GNSS dependency, thereby enhancing flight safety and mission reliability. With future enhancements, this system holds significant potential for strengthening UAV navigation reliability, supporting critical operations in national security, disaster response, and beyond.

6.1. Recommendations for Future Work

This study established a foundational framework for image-based UAV navigation, presenting several avenues for future enhancement. Addressing distortions from roll, pitch, and altitude changes is a pivotal initial step to ensure real-world applicability of the system. Incorporating a multi-image weighted inference system could improve location accuracy by leveraging information from multiple reference images, thereby mitigating the impact of individual image distortions or outliers. Implementing non-planar stereo matching techniques would enhance depth perception and altitude estimation in complex terrains, allowing the system to account for three-dimensional variations in the environment. Integrating real-time mapping data with reference images could eliminate the need for prior image capture and fixed return paths, enabling UAVs to navigate freely within mapped regions without positional drift. Deploying the system on higher-performance single-board computers (SBCs) and utilizing higher-resolution imaging sensors could support more sophisticated computations and detailed feature extraction, improving overall system performance. By pursuing these enhancements, the navigation system could become more robust, adaptable, and scalable, making it suitable for a broader range of UAV applications and operational scenarios.

Bibliography

[1] M. Weiss, "Personal communication," RF Engineer, June 2024.

[2] Geotab Team, "What is gps?" 2024, accessed: October 17, 2024. [Online]. Available: https://www.geotab.com/blog/what-is-gps/

[3] J. Khalil, "Gnss spoofing threatens airline safety, alarming pilots and aviation officials," *GPS World*, 2024. [Online]. Available: https://www.gpsworld.com/gnss-spoofing-threatens-airline-safety-alarming-pilots-and-aviation-officials/

[4] Y. Zhuang, X. Sun, Y. Li, J. Huai, L. Hua, X. Yang, X. Cao, P. Zhang, Y. Cao, L. Qi, J. Yang, N. El-Bendary, N. El-Sheimy, J. Thompson, and R. Chen, "Multi-sensor integrated navigation/positioning systems using data fusion: From analytics-based to learning-based approaches," *Information Fusion*, vol. 95, pp. 62–90, 2023.

[5] M. J. Wright, L. Anastassiou, C. Mishra, J. M. Davies, A. M. Phillips, S. Maskell, and J. F. Ralph, "Cold atom inertial sensors for navigation applications," *Frontiers in Physics*, vol. 10, p. 994459, 2022.

[6] J. Brewer, "Line of sight requirements for reliable low-power rf communication," White Paper, ATEK Access Technologies, 2024, senior Electrical Design Engineer.

[7] Scout Aerial Australia, "An introduction to lidar technology and its applications," 2024, accessed: October 17, 2024. [Online]. Available: https://www.scoutaerial.com.au/article-lidar/

[8] M. Y. Arafat, M. M. Alam, and S. Moh, "Vision-based navigation techniques for unmanned aerial vehicles: Review and challenges," *Drones*, vol. 7, no. 2, p. 89, 2023, accessed: 2024-10-29. [Online]. Available: https://doi.org/10.3390/drones7020089

[9] D.-G. Sim, R.-H. Park, R.-C. Kim, S. U. Lee, and I.-C. Kim, "Integrated position estimation using aerial image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 1–18, Jan 2002. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=982881

[10] X. Zhang, F. Xiao, M. Zheng, and Z. Xie, "Uav image matching from handcrafted to deep local features," *European Journal of Remote Sensing*, vol. 57, no. 1, 2024. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e074c475d93d5e79ded55b5554ea57cb5fb71207

[11] Google LLC, "Google earth," https://earth.google.com/, 2024, accessed: 2024-10-29.

[12] P. Barnum, B. Hu, and C. Brown, "Exploring the practical limits of optical flow," no. Technical Report 806, August 2003.

[13] M. Trajkovic and M. Hedley, "Fast corner detection," *Image and Vision Computing*, vol. 16, pp. 75–87, Feb 1998, accessed: 2024-10-29. [Online]. Available: https://www.researchgate.net/publication/223831601_Fast_Corner_Detection/citation/download

[14] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," vol. 6314, pp. 778–792, 2010, accessed: 2024-10-29. [Online]. Available: https://researchgate.net/publication/221304115_BRIEF_Binary_Robust_Independent_Elementary_Features/citation/download

[15] S. A. K. Tareen and Z. Saleem, "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk," in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2018, pp. 1–10, accessed: 2024-10-29. [Online]. Available: https://ieeexplore.ieee.org/document/8346440

[16] G. Reddy, C. Vani, and E. Krishna, "Implementation of mldb as high performance machine learning database for high performance applications," *Materials Today: Proceedings*, Mar 2021, accessed: 2024-10-29. [Online]. Available: https://www.researchgate.net/publication/349871153_Implementation_of_MLDB_as_high_performance_machine_learning_database_for_high_performance_applications/citation/download

[17] OpenCV, *cv::AKAZE Class Reference*, 2018. [Online]. Available: https://docs.opencv.org/3.4/d8/d30/classcv_1_1AKAZE.html

[18] Rpaultrat, "Efficient neural feature detector and descriptor," GitHub, 2023, accessed: 2024-10-24. [Online]. Available: https://github.com/rpaultrat/SuperPoint

[19] Cvg, "Lightglue: Local feature matching at light speed," GitHub, 2023, accessed: 2024-10-24. [Online]. Available: https://github.com/cvg/LightGlue

[20] OpenCV, *cv::BFMatcher Class Reference*, 2022. [Online]. Available: https://docs.opencv.org/4.x/d3/da1/classcv_1_1BFMatcher.html

[21] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. X, 2014, accessed: 2024-10-29. [Online]. Available: https://www.cs.ubc.ca/research/flann/uploads/FLANN/flann_pami2014.pdf

[22] OpenCV, *Feature Matching in OpenCV*, 2018. [Online]. Available: https://docs.opencv.org/3.4/dc/dc3/tutorial_py_matcher.html

[23] Z. Rezvani, S. Shekarizeh, and M. Sabokrou, "Global-local processing in convolutional neural networks," 2023. [Online]. Available: https://doi.org/10.48550/arxiv.2306.08336

[24] V. Sharma, "Image-template matching using cross-correlation," *Medium*, 2022, accessed: 2024-10-26. [Online]. Available: https://vipin-sharma.medium.com/image-template-matching-using-cross-correlation-2f2b8e59f254

[25] A. Rosebrock, "How-to: 3 ways to compare histograms using opencv and python," *PyImageSearch*, 2014, accessed: 2024-10-26. [Online]. Available: https://pyimagesearch.com/2014/07/14/3-ways-compare-histograms-using-opencv-python/

[26] ——, "Image difference with opencv and python," *PyImageSearch*, 2017, accessed: 2024-10-26. [Online]. Available: https://pyimagesearch.com/2017/06/19/image-difference-with-opencv-and-python/

[27] OpenCV, "Warp affine transformation," 2024, accessed: 2024-10-24. [Online]. Available: https://docs.opencv.org/4.x/d4/d61/tutorial_warp_affine.html

[28] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using svd," 2017, january 16, 2017.

[29] OpenCV, "Homography estimation," 2024, accessed: 2024-10-24. [Online]. Available: https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html

[30] R. B. Fisher, "The ransac (random sample consensus) algorithm," 2002, accessed: 2024-10-28. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FISHER/RANSAC/

[31] D. S. Farin, "Automatic video segmentation employing object/camera modeling," Ph.D. dissertation, Technische Universiteit Eindhoven, 2005.

[32] J.-W. Bian, W.-Y. Lin, Y. Liu, L. Zhang, S.-K. Yeung, M.-M. Cheng, and I. Reid, "Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence," *International Journal of Computer Vision*, vol. 128, June 2020.

Appendix A

Project Planning Schedule

Week	Weekly Plan
Week 0 (Before 21/7)	Approached with and accepted the project topic: "An Image-Based Navigation System for UAVs."
Week 1 (21/7-28/7)	Review Literature on image-based navigation and GNSS vulnerabilities to assess current advancements and research gaps.
Week 2 (28/7-4/8)	Refine project scope and requirements based on the literature review. Define clear objectives and start report writing.
Week 3 (4/8-11/8)	Submit the supervisor-student agreement with the expected outcomes. Begin developing the system-level design aligned with the requirements.
Week 4 (11/8-18/8)	Research and choose suitable algorithms for feature extraction, matching, and transformation estimation.
Week 5 (18/8-25/8)	Develop a detailed software implementation plan, with the necessary modules, based on the system design.
Week 6 (25/8-1/9)	Start coding and get comfortable with the primary sections on feature detection and matching.
Week 7 (1/9-8/9)	Implement different planar transformation estimation techniques for rotation and translation.
Week 8 (8/9-15/9)	Optimize transformation estimations and integrate all approaches into a single main file.
Week 9 (15/9-22/9)	Transform relative to absolute location estimations.
Week 10 (22/9-29/9)	Develop image similarity estimation techniques, integrate components into a unified pipeline, and Start comparative tests.
Week 11 (29/9-6/10)	Optimize algorithms for better efficiency and accuracy. Prepare diverse datasets covering various terrains and conditions for testing.
Week 12 (6/10-13/10)	Conduct testing across datasets to evaluate system performance.
Week 13 (13/10-20/10)	Test the system's robustness to various operational challenges.
Week 14 (20/10-27/10)	Finalize all results and perform any additional necessary tests. Evaluate with regards to the objectives
Week 15 (27/10-3/11)	Complete the final report. Start preparation for presentations.

Table A.1: Project Timeline and Weekly Plan

Appendix B

Outcomes Compliance

Student's Graduate Attribute (GA) Achievement Plan

GA 1: Problem Solving

I have met this objective by taking a broadly defined problem: developing an image-based GNSS-independent navigation system for UAVs, and systematically solving this through research, synthesis of an optimal pipeline, experimentation and analysis. Using mathematical and statistical methods, I created a system that identifies, matches, and estimates UAV transformations in real time. Further, this involved investigating the sources of pipeline errors and solving them via improving understanding of fundamental global positioning concepts. Finally, this involved the numerous software engineering challenges that required constant problem-solving to ensure logical soundness, robustness and accuracy.

GA 2: Application of Scientific and Engineering Knowledge

My project utilized computer vision theories, image processing techniques, and pre-trained machine learning models. I applied algorithms to detect and analyze landscape changes, leveraging homographic transformations to account for relative UAV motion without GNSS signal. Statistical methods improved accuracy by testing different outlier rejection methods, ensuring reliable operation across multiple terrains.

GA 3: Engineering Design

I developed a high-level flow diagram for the image processing pipeline, integrating multiple resources and techniques with robust measures to handle inaccuracies without detailed pipelines available in literature. My engineering principles and design skills ensured development of a system that was built from the ground up to be adapt to various, dense and sparse, environments without manual intervention, maintaining stability and precision. This design also ensures modular design to facilitate scalability and future enhancements.

GA 4: Investigations, Experiments, and Data Analysis

I collected extensive data across various environments and conditions, systematically testing the navigation system to identify accuracy and robustness. Statistical analysis ensured accurate quantification of errors, such as knowing how to best quantify system performance. For instance, this included looking at maxima of error, quantifying them in different metrics for different perspectives, and analyzing potential sources of errors based on nuanced differences such as variance in ground image overlap. Ongoing experimentation with feature detection and matching algorithms, including neural network-based methods, fine-tuned the system's performance.

GA 5: Engineering Methods, Skills, and Tools (including IT)

I employed various Tools, including Python libraries like OpenCV, Matplotlib, Sklearn for data analysis and advanced frameworks for image processing. Rigorous research into current solutions and adherence to software engineering practices and engineering methods ensured systematic solving of parts of the problem, modularity, code optimization, and error handling. Different computational techniques allowed for precise text and visual feedback for evaluation and debugging, allowing continuous optimization.

GA 6: Professional and Technical Communication

This project involves both an oral presentation and written report; thereby demonstrating my ability to communicate effectively, both orally and in writing. Further, this project involved using diagrams and tables to communicate complex concepts concisely, facilitating understanding of the project's technical aspects.

GA 8: Individual Work

I took responsibility for all aspects of the development and implementation process. This included problem-solving, design, and optimization, as well as the testing and analysis of results. My role required me to work independently to research, develop, and refine solutions, utilizing various resources and overcoming challenges with weekly feedback from my supervisor.

GA 9: Independent Learning Ability

This project demanded independent learning to solve complex navigation challenges in GNSS-denied environments. This was done through researching different research materials to gain an understanding and relevant technical studies. I expanded my understanding

of Computer Vision independently, equipping myself with the skills necessary to achieve project goals without requiring extensive guidance.