UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

# A Critical Analysis of Design Flaws in the Death Star

Luke Skywalker
99652154

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

Supervisor: Dr O. W. Kenobi

October 2099

# Acknowledgements

I would like to thank my dog, Muffin. I also would like to thank the inventor of the incubator; without him/her, I would not be here. Finally, I would like to thank Dr Herman Kamper for this amazing report template.

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 / *Student number* | Handtekening / *Signature* |
| Voorletters en van / *Initials and surname* | Datum / *Date* |

# Abstract


### English

The English abstract.

### Afrikaans

Die Afrikaanse uittreksel.

# Contents

# List of Figures

# List of Tables

# Nomenclature

### Variables and functions

| | |
|---|---|
| $p(x)$ | Probability density function with respect to variable $x$. |
| $P(A)$ | Probability of event $A$ occurring. |
| $\varepsilon$ | The Bayes error. |
| $\varepsilon_u$ | The Bhattacharyya bound. |
| $B$ | The Bhattacharyya distance. |
| $s$ | An HMM state. A subscript is used to refer to a particular state, e.g. $s_i$ refers to the $i^{\text{th}}$ state of an HMM. |
| $\mathbf{S}$ | A set of HMM states. |
| $\mathbf{F}$ | A set of frames. |
| $\mathbf{o}_f$ | Observation (feature) vector associated with frame $f$. |
| $\gamma_s(\mathbf{o}_f)$ | A posteriori probability of the observation vector $\mathbf{o}_f$ being generated by HMM state $s$. |
| $\mu$ | Statistical mean vector. |
| $\Sigma$ | Statistical covariance matrix. |
| $L(\mathbf{S})$ | Log likelihood of the set of HMM states $\mathbf{S}$ generating the training set observation vectors assigned to the states in that set. |
| $\mathcal{N}(\mathbf{x}\|\mu, \Sigma)$ | Multivariate Gaussian PDF with mean $\mu$ and covariance matrix $\Sigma$. |
| $a_{ij}$ | The probability of a transition from HMM state $s_i$ to state $s_j$. |
| $N$ | Total number of frames or number of tokens, depending on the context. |
| $D$ | Number of deletion errors. |
| $I$ | Number of insertion errors. |
| $S$ | Number of substitution errors. |

### Acronyms and abbreviations

| | |
|---|---|
| AE | Afrikaans English |
| AID | accent identification |
| ASR | automatic speech recognition |
| AST | African Speech Technology |
| CE | Cape Flats English |
| DCD | dialect-context-dependent |
| DNN | deep neural network |
| G2P | grapheme-to-phoneme |
| GMM | Gaussian mixture model |
| HMM | hidden Markov model |
| HTK | Hidden Markov Model Toolkit |
| IE | Indian South African English |
| IPA | International Phonetic Alphabet |
| LM | language model |
| LMS | language model scaling factor |
| MFCC | Mel-frequency cepstral coefficient |
| MLLR | maximum likelihood linear regression |
| OOV | out-of-vocabulary |
| PD | pronunciation dictionary |
| PDF | probability density function |
| SAE | South African English |
| SAMPA | Speech Assessment Methods Phonetic Alphabet |

## Feature Extractors

### What is a feature Extractor

A feature extractor is a computer vision algorithm or deep-learning model that identifies and extracts key features in an image. A feature is a pixel or weight of multiple pixels that represents a highly unique and distinctive part of an image. For instance, this could be an edge between a wall and the sky. Feature extractors are characterized by a spatial coordinate and a descriptor. A descriptor is a vector which contains information about the helps it be uniquely identified again in a new image. This can include information about the size, shape, and intensity of the feature. One can also infer the scale and orientation of the feature from the descriptor. A feature extractor also commonly includes a confidence score which indicates a score of how unique and well-defined the feature is. Ultimately, a feature extractors goal is to find unique and well-defined features that can be used to develop a complex understanding of the nuances and details of an image.

### Usage in this task

In this task, feature extractors are used to extract keypoints in images. These are used for intra-image rotational and translational estimates, and matching similar images.

feature extractors are nb bc they r iinvariant to scale rotation and transformation. As opposed to simply correlating entire image. also less computationally expensive. https://baotramduong.medium.com/feature-extraction-in-computer-vision-using-python-358d7c9863cb

common feature extraction techniques include edge detection, color histograms, and texture analysis

# 0.1. Choosing a Feature Extractor

When selecting a feature extractor, several critical factors must be considered:

- **Accuracy:** The extractor needs to be accurate to ensure GPS inference is accurate. That is, the extractor should provide a sufficient quality and quantity of keypoints. This is quantified as subtending over 500 good matches from similar image pairs.
- **Speed:** Be capable of real-time processing to ensure timely navigation and decision-making in dynamic environments. Specifically, initial tests must extract features in less than 1 second on a CPU. For the neural network-based models, the time taken to extract features must be less than 3 second on a CPU since they may be implemented with a GPU and CUDA libraries to improve performance.

- **Robustness:** The feature extractor should exhibit invariance to changes in scale, rotation, illumination, perspective and noise to ensure accurate and repeatable performance across different flight datasets and conditions.

Feature extractors have different parameters trading-off accuracy and speed. For example, there are detection thresholds, descriptor sizes, and keypoint densities that can be adjusted to optimize performance.

## 0.2. Types of Feature Extractors

### 0.2.1. Traditional Feature Extractors

#### SIFT (Scale-Invariant Feature Transform)

SIFT detects and describes local features in images. It is robust to changes in scale, rotation, and illumination, making it a reliable choice for many applications. However, its computational intensity can be a drawback in real-time scenarios.

- **Advantages:** High accuracy and robustness due to its multi-scale approach and precise keypoint localization. SIFT's descriptors are highly distinctive, enabling reliable matching across different views and conditions.
- **Disadvantages:** High computational cost and slower processing speed due to the extensive keypoint detection and descriptor computation steps, making it less suitable for real-time applications.

#### SURF (Speeded-Up Robust Features)

SURF is a faster alternative to SIFT, utilizing integral images for rapid computation of image convolutions. It offers good accuracy and robustness while being computationally more efficient than SIFT.

- **Advantages:** Faster than SIFT due to its use of Haar wavelets and integral images, providing good balance between speed and accuracy. It maintains robustness to scale and rotation changes.
- **Disadvantages:** Still relatively computationally expensive compared to simpler methods like ORB, and can be less accurate than SIFT in certain complex scenarios.

#### ORB (Oriented FAST and Rotated BRIEF)

ORB combines the FAST keypoint detector and the BRIEF descriptor, providing a highly efficient feature extraction method suitable for real-time applications. It is designed to be both fast and invariant to rotation and scale.

- **Advantages:** High speed and efficiency, making it suitable for real-time applications. ORB's binary descriptors are computationally less intensive while providing sufficient discriminative power for many tasks.
- **Disadvantages:** Lower accuracy compared to SIFT and SURF, especially in complex scenes with significant variations in lighting and scale. The binary nature of BRIEF descriptors can sometimes lead to higher false match rates.

### 0.2.2. AKAZE (Accelerated-Keypoint-Affine-Zernike)

AKAZE is a feature extraction method that combines speed and accuracy by using nonlinear scale space and feature detection. It is designed to be robust to various transformations and lighting conditions.

- **Advantages:** High speed and efficiency due to its nonlinear scale space and feature detection approach. AKAZE is robust to scale, rotation, and illumination changes, making it suitable for diverse environments.
- **Disadvantages:** May not be as accurate as SIFT or SURF in certain scenarios, particularly those with complex textures or repetitive patterns. The trade-off between speed and accuracy may vary depending on the application.

### 0.2.3. Deep Learning-Based Feature Extractors

Deep learning-based feature extractors leverage neural networks to learn feature representations directly from data. These models need to be trained on large datasets to capture complex and hierarchical features effectively. They offer high accuracy and the ability to adapt to specific tasks through transfer learning.

- **Advantages:** High accuracy and the ability to learn complex and hierarchical features directly from data, enabling robust performance across diverse tasks and conditions.
- **Disadvantages:** Requires substantial computational resources for training and inference. The training process is data-intensive, requiring large labeled datasets to achieve optimal performance.

#### Pre-trained Models (SuperPoint)

Utilizing pre-trained models, like Superpoint, that are trained on large datasets are often highly accurate relative to their efficiency. They are trained on a variety of datasets and aim to generalize well to a variety of tasks.

- **Advantages:** High accuracy due to extensive pre-training on large and diverse datasets.
- **Disadvantages:** May not generalize well to specific datasets which are not representative of the application dataset if xxx.

## 0.3. Evaluation for UAV-Based Navigation

### 0.3.1. Requirements

For the Skripsie project, the feature extractor must meet the following initial requirements with default parameters:

-
-
- Must be free to use and not require any pre-training or live-training. The former excludes SURF, SIFT, and the latter, deep learning-based models.

### 0.3.2. Recommended Feature Extractors

The feature extractor should be applicable to application specific requirements. In the case of this project, it should be free to use and not require any pre-training or live-training due to scope. As such, the following feature extractors are recommended:

- ORB: A fast and efficient feature extractor that provides a good balance between speed and accuracy. It is suitable for real-time applications and can handle scale and rotation changes effectively.
- AKAZE: A feature extraction method that combines speed and accuracy by using nonlinear scale space and feature detection. It is robust to various transformations and lighting conditions.
- SuperPoint: A pre-trained model that offers high accuracy and efficiency for real-time applications. It learns feature representations directly from data, making it adaptable to specific tasks.

# Chapter 1

# Results

TESTS: amt of keypoints for a specific runtime that are accepted as good matches overall accuracy for heading and GPS change estimation

All parameter choices add a noise test to final results.

test 1: accuracy on a single dataset for rotational estimation (global matcher), global matching technique, and local matcher. gonna need to split those. say something about the matches are good no matter what - not a hard task. So test both methods (Plus neural local) on both stages (2x3 results) on a single dataset - see which is more accurate only. - total time and acc. test 2: Take the top 3 combinations and test them on all the datasets (or as many as are close together in accuracy) - optimize for datasets - see which is more accurate only. - total time and acc. test 3: mess up the parameters and see how it affects the stability of different methods as well as the overall accuracy - stability, acc and time. test 4: test whether one performs significantly better than normally when used as the global matcher technique

XXX - note that we dont look at time per extraction as the time of latter stages is affected by the amount and quality of keypoints. So it might extract rubbish faster, but then take longer to match. So we look at total time. Test 1 and 2 show accuracy. Test 2 and 3 show robustness.

XXX - need to make a note on why mean heading error is not compared much - it subtends GPS error - and its estimated in this project as we dont have an accurate heading.

XXX - say we are going to use histograms, its significantly faster, ensures no effects from the global match, could potentially test that after

However, when trying different data sets etc - and perhaps when trying it with the local matcher, robustness might affect these end results.

Table 3 shows the results for confidence thresholding applied to AKAZE and ORB detectors.

| Threshold | AKAZE | | ORB | |
|---|---|---|---|---|
| | **RMSE GPS (error)** | **Runtime (s)** | **RMSE GPS (error)** | **Runtime (s)** |
| **No Filter** | 61.64 | 42.43 | 46.76 | 35.73 |
| **1000** | 49.52 | 49.17 | 47.19 | 53.79 |
| **500** | 51.48 | 46.82 | 48.12 | 38.79 |
| **300** | 51.48 | 50.13 | 48.47 | 39.01 |

**Table 1.1:** RMSE GPS Error and Runtime for AKAZE and ORB with Different Confidence Match Limits

**Observations**: AKAZE performed optimally with a threshold of 1000 matches, while ORB performed best with a threshold of 300 matches. AKAZE achieving optimal performance at higher thresholds suggests that its keypoints are more distinctive and less ambiguous than ORB's. This aids the case that AKAZE requires less optimization after extraction than ORB.

NOTE ORB simply does not work on the desert dataset

Please note that these tests are run without optimal settings - i.e. for debug purposes certain parts run which normally would not.

Another note, this data is run in debug mode, with many methods running sequentially instead of a single method. Further, this is ran in a high accuracy mode for both, with an aim to get good results for both while keeping the time between methods reasonably similar without moving far off optimal points.

The first is the rotational estimator, the latter the translational estimator

XXX make note abt dynamic adjustment for ORB - ambig match minimum

XXX note global local is actually rot and translation. Say how rot is used for global matching and translational estimation.

TEST 1: Ideally better points should subtend better estimations on all transformations.

NOTE that these methods are fully optimized within the constraint of having similar runtimes. However, orb performs optimally at higher speeds naturally - it loses acc with too many forced keypoints. lets note after these initial results we dont want to go down a rabbit hole of optimization as that will take too long. This is on datsetrot

datsetrot

| Rotational stage Detector | ORB | | AKAZE | |
|---|---|---|---|---|
| **Translational stage Detector** | **ORB** | **AKAZE** | **ORB** | **AKAZE** |
| **Mean Global Good Matches** | 661.24 | 661.24 | 736.86 | 736.86 |
| **Mean Local Good Matches** | 898.05 | 12557.3 | 882.3 | 12256.30 |
| **RMSE (GPS Error) (radial)** | 75.78 | 76.18 | 73.31 | 68.94 |
| **MAE (GPS Error)** | 53.02 | 53.53 | 51.30 | 48.31 |
| **Runtime (seconds)** | 31.62 | 73.14 | 47.67 | 68.18 |

**Table 1.2:** Comparison of Rotational and Translational Inference Methods with Good Matches, Error Metrics, and MAE

From whats above, we firsly see how AKAZE finds significantly more keypoints than ORB. note the stages are not exactly the same, other parameters have been tuned to find the highest accuracy for the specific task. In terms of accuracy, all methods perform well. Because of the difficulty in perfect optimization, and trying to keep the runtime similar, we see that the MAE and RMSE are similar. Therefore, we will test the accuracy on a different dataset. We will keep the parameter set optimized for dataset x and test on dataset y to see how it performs with unoptimized parameters.

XXX include MAE XXX test CUDA

ORB performs the best. But note this result is not perfect. Firstly, there are an extremely large number of interdependent parameters which can be optimized for every dataset or dynamically adjusted within datasets for optimal performance. Finding the perfect strategy for dynamic and static adjustment is out of the scope of this project. However, we have found well-optimized parameters wherever possible.

Test 3: General robustness. AKAZE worked better out of the box. When using default parameters and only changing the main threshold, AKAZE worked on the low feature desert dataset, while ORB required changing many parameters.

In order to test robustness, we will use the optimized parameter set for the initial dataset (CPT rotations) on the other 4 datasets, that is, amazon rain forest, desert, city without rotations, and minor rotation rocky environment.

xxx - we could test non planar, and low - coverage. both not essential as the outcome is relatively obvious. xxx - we need to do a dynamic thresholding similar to inference. its not as simple as more or less features. different datasets will perform optimally with different numbers of features. we need a balance between quality and quantity. This must be done through testing errors in forward path and adjusting based on that. not implemented. time intensive to iteratively update, can be done - weak to changes in landscape. increasing far abv minimum req kps is best option. but long-term in flight might be better to have a dynamic threshold. This is the single most sensitive stage. Param choices here are critical. Maybe a benefit to superpoint.

test across one dataset.

datset

| Metric | CityROT | CityTR | Rocky | Desert | Amazon |
|---|---|---|---|---|---|
| **Mean Global Good Matches** | 661.24 | 680.56 | 630.29 | 656.40 | 628.30 |
| **Mean Local Good Matches** | 615.35 | 639.75 | 683.90 | 669.55 | 569.15 |
| **RMSE - GPS error** | 75.78 | 18.12 | 28.10 | 357.19 | 76.30 |
| **MAE - GPS error** | 53.02 | 12.51 | 19.84 | 249.82 | 51.86 |
| **Runtime (seconds)** | 35.84 | 36.65 | 31.89 | 26.11 | 33.54 |

**Table 1.3:** Robustness testing of ORB (Rotational) and ORB (Translational) across datasets with Parameters Optimized for CityROT dataset

| Metric | CityROT | CityTR | Rocky | Desert | Amazon |
|---|---|---|---|---|---|
| **Mean Global Good Matches** | 661.24 | 680.56 | 630.29 | **FAIL** | **FAIL** |
| **Mean Local Good Matches** | 1000.0 | 1000.0 | 974.05 | **FAIL** | **FAIL** |
| **RMSE - GPS error** | 72.44 | 6.32 | 63.77 | **FAIL** | **FAIL** |
| **MAE - GPS error** | 50.94 | 4.32 | 43.01 | **FAIL** | **FAIL** |
| **Runtime (seconds)** | 64.37 | 157.77 | 60.39 | **FAIL** | **FAIL** |

**Table 1.4:** Robustness testing of ORB (Rotational) and AKAZE (Translational) across datasets with Parameters Optimized for CityROT dataset

Further tests with AKAZE as the rotational estimator fail in the same cases. Although certain parameter sets will work with AKAZE for all cases, the issue lies in the fact AKAZE takes extensive time already, and said parameter set will cause even more delays. This is because the AKAZE thresholds do not dynamically adjust based on the amount of keypoints in the scene. ORB, however, does. This is a significant advantage of ORB in terms of robustness.

| Metric | CityROT | CityTR | Rocky | Desert | Amazon |
|---|---|---|---|---|---|
| **Mean Global Good Matches** | 661.24 | 680.56 | 630.29 | 656.40 | 628.30 |
| **Mean Local Good Matches** | nan | nan | nan | nan | nan |
| **RMSE - GPS error** | 78.42 | 14.24 | 23.03 | 39.64 | 42.02 |
| **MAE - GPS error** | 55.43 | 9.79 | 16.07 | 27.20 | 28.10 |
| **Mean Heading Error** | 0.90 | 0.016 | 0.17 | 0.88 | 1.97 |
| **Runtime (seconds)** | 106.61 | 120.39 | 109.36 | 115.53 | 108.83 |

**Table 1.5:** Robustness testing of ORB (Rotational) and SUPERPOINT-LightGlue (Translational) across datasets with Parameters Optimized for CityROT dataset

| Translational Detector | CityROT | CityTR | Rocky | Desert | Amazon |
|---|---|---|---|---|---|
| **ORB** | 75.78 | 18.12 | 28.10 | 357.19 | 76.30 |
| **AKAZE** | 72.44 | 6.32 | 63.77 | **FAIL** | **FAIL** |
| **SUPERPOINT-LightGlue** | 78.42 | 14.24 | 23.03 | 39.64 | 42.02 |

**Table 1.6:** RMSE (GPS error) for different Translational Detectors (ORB as Rotational Detector) across datasets

| Translational Detector | CityROT | CityTR | Rocky | Desert | Amazon |
|---|---|---|---|---|---|
| **ORB** | 35.84 | 36.65 | 31.89 | 26.11 | 33.54 |
| **AKAZE** | 64.37 | 157.77 | 60.39 | **FAIL** | **FAIL** |
| **SUPERPOINT-LightGlue** | 106.61 | 120.39 | 109.36 | 115.53 | 108.83 |

**Table 1.7:** Runtime (seconds) for different Translational Detectors (ORB as Rotational Detector) across datasets

# Chapter 2

# Local Feature Matchers

## 2.1. Background

Feature matching is a fundamental component of image-based navigation and pose estimation systems. It involves identifying correspondences between features detected in different images, enabling the estimation of geometric transformations such as translation and rotation. These correspondences form the basis for understanding relative movement, orientation, and position between images.

This study focuses on local feature matchers, which identify correspondences between individual keypoints within images. In contrast, global matchers evaluate the entire image holistically, treating it as a single entity. Local feature matching is essential for applications where precise geometric transformations are required.

## 2.2. Context and Processing Stages

This chapter evaluates several state-of-the-art local feature matching techniques, including **BFMatcher**, **FLANN**, and **LightGlue**. These matchers represent widely researched methods with real-world applicability. LightGlue is evaluated in a separate chapter alongside the neural network-based extractor **SuperPoint**, as it is optimized for deep-learning-based feature extraction.

### Pipeline Overview

- **Feature Extraction (Prior Step):** Features are extracted from images using methods such as ORB, AKAZE, or SuperPoint. The quality of extracted keypoints significantly affects subsequent matching performance. These features are normalized in the **global heading space** (True North) to ensure that transformations can be interpreted in global coordinates.
- **Matching and Filtering:** Keypoints are matched between images using different feature matchers (e.g., BFMatcher, FLANN, LightGlue). Each matcher outputs a

list of potential matches, along with their similarity scores, represented by a distance metric.

- **Search Technique:** The search technique determines which potential matches are retained. Techniques such as radius matching, vanilla matching, and KNN-based matching are explored. This stage allows the matcher to control the number and quality of matches passed for further filtering.
- **Optimization:** After initial matching, additional filtering techniques are applied to improve match quality. The goal is to maximize correspondences while minimizing false positives. These optimization methods include Lowe's ratio test, cross-checking, and RANSAC filtering.
- **Pose Estimation (Post Step):** The normalized matches are used to estimate transformations (e.g., translation) or to measure global image similarity, depending on the state of the pipeline.

## 2.3. Methodology

### Feature Matchers

- **Brute-Force Matcher (BFMatcher):** The BFMatcher compares each feature in one image with every feature in the second image. While it guarantees the best match, it is computationally expensive.
- **Fast Library for Approximate Nearest Neighbors (FLANN):** FLANN provides fast nearest neighbor search algorithms, optimized for large datasets and high-dimensional features. It offers a trade-off between accuracy and computational efficiency, making it suitable for real-time applications.
- **LightGlue:** LightGlue is a machine-learning-based matcher that improves matching accuracy through deep learning. Although highly effective, it is computationally expensive. It is tested alongside SuperPoint in the feature extraction chapter, as its true potential is only realized when paired with neural network-based feature extractors.

### Search Techniques

- **Radius Search:** Retains matches within a specific distance (radius) around each keypoint. However, it does not guarantee a fixed number of matches per keypoint, potentially leading to either insufficient or excessive matches. Due to these limitations, it was not used in further experiments.

- **Vanilla Matching:** Returns the single best match for each keypoint. Empirical testing showed that this approach provided poor accuracy due to its inability to remove ambiguity with further processing, and it was not further investigated.

- **K-Nearest Neighbors (KNN):** Retains the top $K$ matches for each keypoint, allowing the use of post-filtering techniques like Lowe's ratio test to remove ambiguous matches. Matches with only one neighbor were discarded to avoid compromising accuracy when Lowe's ratio could not be applied. Empirical tests showed that values above $K=2$ introduced impractical computational overheads. This method was used for further experiments with $K=2$.

### Optimization Techniques

- **Lowe's Ratio Test:** Filters matches based on the ratio between the best match's distance and the second-best match's distance. If the ratio is below a given threshold, in other words, the matches are sufficiently different, the match is retained. While effective at removing ambiguity, Lowe's ratio is moderately sensitive to datasets and parameters. A dynamic threshold was implemented, adjusting iteratively until a sufficient number of matches were retained while keeping iterations to a minimum.

- **Cross-Check:** Filters matches by retaining only those that are mutual in both matching directions. This ensures that matches are one-to-one, as is often not the case due to imperfections in the dataset, reducing false positives.

- **RANSAC:** Filters outliers by estimating transformations between images using the Random Sample Consensus (RANSAC) algorithm. It is more closely tied to the rotational and translational estimation stages, where it is tested further.

- **Confidence Thresholding or Weighting:** Matches were filtered based on their confidence scores, determined by similarity in descriptor space. However, confidence-based filtering proved extremely sensitive to changes in parameters and datasets, limiting its applicability. This method was not further pursued.

## 2.4. Experimental Setup and Results

The experiments evaluated BFMatcher and FLANN for time efficiency, accuracy, and robustness across diverse, real-world datasets. Other stages of the pipeline, were optimized to ensure fair comparisons between the matchers. Metrics included Mean Absolute Error (MAE) of the GPS estimation and runtime in seconds. Mean heading error was excluded, as it was inferred through GPS error, and ground truth heading data was unavailable. Three testing stages were conducted: optimization to identify the best parameters, robustness to

assess generalizability, and performance evaluation to compare the matchers under similar optimized conditions.

## Optimization Testing

The goal of optimization testing was to identify parameters that maximize performance without compromising generalizability. FLANN and BFMatcher were tested with different configurations, specifically Lowe's ratio and cross-checking.

Confidence thresholding, as noted earlier, was not pursued due to its sensitivity to parameter changes. RANSAC was tested separately within the pose estimation stages.

### Test 1: Cross-Checking

Both matchers were tested with cross-checking to improve accuracy by reducing false positives. Cross-checking ensures that matches are **mutual** in both directions between two images. However, due to **noise** and imperfections in real-world data, matching between images is inherently **asymmetrical**, meaning the top matches in one direction often differ significantly from those in the reverse direction.

Empirical testing showed that only **FLANN** had potential for practical use with cross-checking, as BFMatcher's built-in cross-checking was too slow for real-time application. Despite attempts to optimize FLANN with cross-checking, the combined impact of noise, asymmetry, and computational costs limited its effectiveness. The results are summarized below.

**Initial Approach** Initially, cross-checking was applied **after Lowe's ratio filtering**. However, this resulted in poor accuracy across **4 out of 5 datasets**. Applying Lowe's ratio first left too few matches for cross-checking, as the asymmetry between matching directions caused many mutual matches to be lost. As a result, the approach became unstable and unreliable. Table 2.1 summarizes the comparison.

| Matcher | Metric Type | CITY1 | CITY2 | ROCKY | DESERT | AMAZON |
|---|---|---|---|---|---|---|
| FLANN (No Cross-Check) | MAE GPS (m) | 56.59 | 4.70 | 14.63 | 71.20 | 32.35 |
| | Runtime (s) | 42.72 | 41.61 | 41.96 | 43.42 | 53.62 |
| FLANN (With Cross-Check) | MAE GPS (m) | 70.71 | 7.96 | 23.64 | 41.37 | 44.39 |
| | Runtime (s) | 71.60 | 72.28 | 88.99 | 70.98 | 62.28 |

**Table 2.1:** Comparison of FLANN with and without Post-Lowe's Cross-Check across Datasets (MAE GPS and Runtime)

**Experimenting with Pre- and Post-Lowe's Filtering** However, in the desert dataset, there were significant improvements. To explore this further, three strategies were explored to attempt to help the cross-checking process generalize better across datasets. These strategies tested variations of a relaxed Lowe's threshold prior to cross-checking, to reduce computational load without removing too many matches. After cross-checking, a set stricter Lowe's threshold was applied to remove false positives.

- **Low or No Pre-Filtering:** Minimal filtering allowed sufficient matches to pass to cross-checking, but resulted in unacceptable runtimes.
- **High Pre-Filtering:** Aggressively filtering matches reduced runtime but led to instability and poor results across most datasets, as too few matches remained. There were no static parameters that could balance this trade-off.
- **Dynamic Pre-Filtering:** Lowe's ratio was incremented dynamically until **n matches** were found. $n$ was tuned to achieve maximum acceptable runtime. Although this method ensured stability, it did not improve accuracy over no cross-checking in **4 out of 5 datasets**.

### Testing Conclusion

Cross-checking was tested thoroughly but failed to consistently improve performance with any combination of pre- and post-filtering. Its effectiveness was limited by asymmetry in the matching process and the characteristics of the datasets used. As a result, a generalizable increase in accuracy could not be achieved in real-time. Therefore, cross-checking was not adopted.

#### Test 2: Lowe's Ratio Test

Lowe's ratio was employed to reduce ambiguity by comparing the best and second-best matches for each keypoint. A match was retained if the ratio between these distances fell below a defined threshold. This method effectively filtered false positives, ensuring only distinct matches were kept. However, determining an optimal static threshold was challenging, as performance varied across different datasets and feature extraction methods.

During testing, the optimal static threshold was found to be 0.8 for AKAZE and 0.7 for ORB, with slight variations depending on other parameters. These values improved accuracy but lacked generalizability across datasets that differed in keypoint density and quality. Consequently, a dynamic approach was implemented, adjusting the threshold iteratively until a minimum number of matches were obtained to ensure stability. The initial thresholds and increment steps were tuned to have minimal influence in most cases, aiming to preserve the generalizability of the method. This dynamic method better balanced quality and efficiency than fixed thresholds.

While dynamic tuning enhanced performance, it compromised generalaizbility. Future iterations could benefit from real-time threshold adjustments to adapt to changing conditions in-flight or during GPS loss, maintaining both generalizability and accuracy.

### 2.4.1. Robustness Evaluation

The robustness tests assess the matchers' ability to generalize, in terms of accuracy and performance, across diverse datasets and non-ideal conditions, ensuring reliability in real-world applications. These tests explore the impact of noisy keypoints through various detector thresholds and limited post-match filtering, providing insights into the matchers' ability to ensure accurate matches under challenging conditions.

#### Test 1: Robustness Evaluation Under Limited Post-Filtering

This robustness test evaluates the performance of BFMatcher and FLANN without using Lowe's ratio or other ambiguity-filtering techniques. However, RANSAC and other downstream processes were applied to maintain accuracy in the estimation stages. The same keypoints were passed into the matching stage to ensure a fair comparison. The primary goal was to observe the raw quality and number of matches produced by each matcher. The results are summarized in Table 2.2.

| Matcher | Metric Type | CITY1 | CITY2 | ROCKY | DESERT | AMAZON |
|---|---|---|---|---|---|---|
| BFMatcher | MAE GPS (m) | 777.64 | 507.75 | 465.81 | 534.85 | 319.65 |
| | Runtime (s) | 140.56 | 145.91 | 115.26 | 108.71 | 192.19 |
| | Mean Matches | 9036.15 | 9062.00 | 10416.30 | 6460.95 | 10451.65 |
| | Mean Keypoints | 9059.20 | 9041.27 | 10429.07 | 6480.47 | 10479.13 |
| FLANN | MAE GPS (m) | 826.73 | 526.42 | 471.96 | 539.96 | 340.21 |
| | Runtime (s) | 50.88 | 46.10 | 50.39 | 50.69 | 64.91 |
| | Mean Matches | 9045.45 | 9063.75 | 10434.75 | 6507.80 | 10479.50 |
| | Mean Keypoints | 9059.20 | 9041.27 | 10429.07 | 6480.47 | 10479.13 |

**Table 2.2:** Performance Comparison of BFMatcher and FLANN Under Limited Post-Filtering

**Observations**

- **Performance Comparison:** BFMatcher achieved slightly better accuracy across most datasets. However, the improvements were marginal compared to the significant variation in runtime, with BFMatcher taking two to three times longer than FLANN across all datasets.
- **Matches Found:** Both matchers detected a similar number of matches, indicating that the primary difference lies in match quality. BFMatcher's exhaustive approach yields higher-quality matches but at the cost of substantial runtime.
- **Implications:** In most cases, the minor accuracy improvements offered by BF-Matcher do not justify its high computational cost. FLANN remains the better option for real-world applications, where runtime efficiency is critical.

### Test 2: Robustness Evaluation Under Different Detector Thresholds

This test evaluates the performance of BFMatcher and FLANN under varying keypoint thresholds to assess the differences in accuracy, stability, and scalability. The goal is to determine how the two matchers perform across a range of detection thresholds—from low

to high keypoints—and to observe whether the methods converge or diverge at specific higher levels. This test is also essential to identify the stability of the methods at low keypoints. The results are summarized in Figure 2.1.

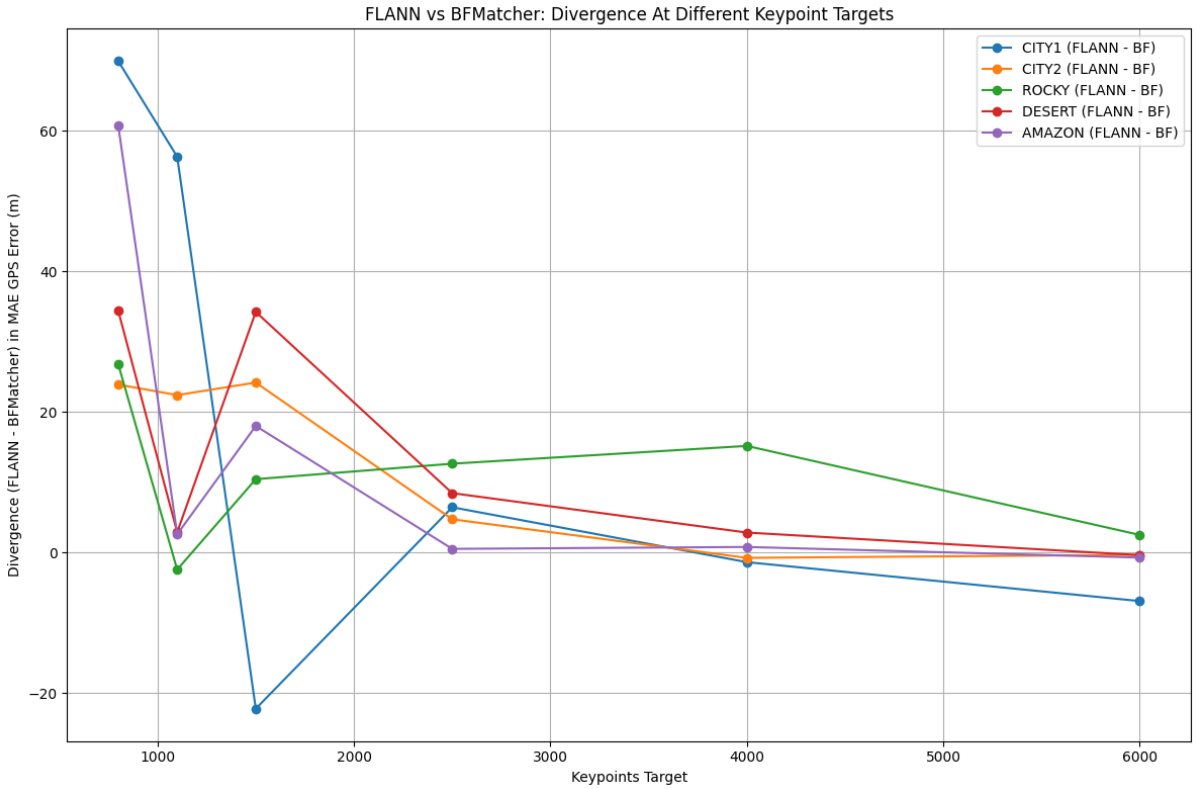


**Figure 2.1:** Divergence in MAE GPS Error Between FLANN and BFMatcher Across Keypoint Targets.

### Observations

- **Divergence and Convergence:** As the number of keypoints increases, the difference in MAE GPS error between BFMatcher and FLANN decreases, showing that both matchers converge to similar levels of accuracy. At higher keypoint counts, the performance gap becomes negligible, with both methods providing comparable accuracy. Beyond 5000 keypoints, the error difference is minimal, well within acceptable limits for real-time applications.

- **Match Quantity vs Quality:** Both matchers identified a similar number of matches in most tests, but the main distinction was in match quality. BFMatcher's exhaustive approach produced slightly higher-quality matches but required more processing time. FLANN, however, provided competitive accuracy with significantly faster runtimes, maintaining acceptable match quality.

- **Lowe's Ratio and Abnormalities:** FLANN outperformed BFMatcher in several cases, primarily due to the effects of Lowe's filtering. Specifically:

  - When Lowe's ratio was too lenient, incorrect matches could pass, but FLANN's approach often filtered these out by not identifying a suitable second match.
  - When Lowe's ratio was too strict, valid matches could be discarded if the second-best match was too similar, but FLANN occasionally allowed these through by selecting slightly lower-quality second matches.

  This demonstrates the limitations of static thresholds in Lowe's ratio and highlights the potential issues with overly rigid filtering.

- **Accuracy and Scalability:** Both matchers showed decreasing error rates as the number of keypoints increased, improving overall stability. However, BFMatcher's runtime increased much faster than FLANN's as the keypoint count grew, making FLANN a better option for handling larger datasets efficiently.

- **Implications:** As keypoint counts rise, both matchers achieve comparable accuracy, but FLANN's superior efficiency and scalability make it the more practical choice for large-scale, real-time applications, balancing accuracy with computational cost.

## 4.3 Accuracy and Runtime Evaluation

This evaluation compares the accuracy and runtime of BFMatcher and FLANN using optimized parameters for each dataset and matching method. The goal is to assess their trade-offs between precision and computational efficiency. Results are summarized in Table 2.3.

| Matcher | Metric Type | CITY1 | CITY2 | ROCKY | DESERT | AMAZON |
|---|---|---|---|---|---|---|
| FLANN | MAE GPS (m) | 56.59 | 4.70 | 14.63 | 71.20 | 32.35 |
| | Runtime (s) | 42.72 | 41.61 | 41.96 | 43.42 | 53.62 |
| BF | MAE GPS (m) | 53.89 | 3.79 | 16.36 | 68.64 | 33.24 |
| | Runtime (s) | 203.49 | 228.59 | 46.33 | 52.59 | 88.95 |

**Table 2.3:** Performance comparison of optimized BFMatcher and FLANN Across Datasets

**Observations**

- **BFMatcher** achieved slightly better MAE in some cases, but this precision came at the cost of significantly longer runtimes, with execution times often more than double those of FLANN.

- **FLANN** maintained faster runtimes across all datasets, demonstrating better computational efficiency, though with marginally higher MAE values in some scenarios.
- **Test conclusion:** FLANN is more suited for real-time applications, offering a practical balance between speed and accuracy, with scalability to handle larger datasets efficiently.

## Conclusion

This study evaluated BFMatcher and FLANN for local feature matching. BFMatcher provided slightly higher-quality matches but was computationally expensive and more stable when fewer keypoints were available. FLANN, however, delivered comparable accuracy with significantly better runtime and scalability, making it ideal for real-time applications when sufficient keypoints are available.

In practice, with modern computational power generating high keypoint counts, the differences between the two methods become negligible. While BFMatcher shows more robustness under low-keypoint conditions, its lack of scalability makes it less practical compared to FLANN, which provides a more efficient solution without compromising accuracy.

## Future Work

Future research should focus on:

- **Adaptive Filtering:** Develop dynamic filtering techniques and inference models for determining optimal Lowe's threshold to address its limitations.
- **Extreme Testing:** Explore the matchers' performance under more severe environmental conditions to improve robustness.
- **Neural Network Integration:** Investigate ways to efficiently integrate learning-based matchers like LightGlue into real-time systems.

These improvements will enhance the robustness, efficiency, and applicability of feature matching in future systems.

XXX - flat earth buildings assumption XXX - try 512 x 512 XXX - different datasets

## 2.5. Rotational Estimators

### 2.5.1. Introduction

Rotational estimators are employed to align UAV-captured images when GPS data is unreliable. It is crucial that this step is accurate, as subsequent steps, the estimation of translational shifts and image similarity, are highly sensitive to small errors in rotation. Rotational estimation is used in this task to estimate angles between images, and subsequently the heading of the UAV relative to a known reference point. Since the primary aim of this task is to use reference images, with known headings, adjusting for heading changes due to the Earth's curvature is not necessary. These steps are crucial to ensuring reliable navigation in GPS-denied environments.

TEST 1: Best estimation technique for transformation TEST 2: Best feature Extractor for Rotational Estimation TEST 3: Best feature matcher for Rotational Estimation TEST 4:

Tests: Across detectors: akaze orb superpoint

Four methods were selected for rotational estimation based on their applicability, computational efficiency, and accuracy. Other methods were considered but ultimately excluded due to their efficiency or accuracy.

### 2.5.2. Methods

- **Homography Estimation**: A method supporting 8 degrees of freedom (DoF), accounting for rotation, scaling, translation, and perspective correction. It is suitable for complex transformations but may introduce unnecessary errors when perspective correction is not needed.
- **Affine Estimation**: A method with 6 degrees of freedom that handles rotation, scaling, and translation without perspective correction. It is computationally efficient and offers sufficient transformation handling for most tasks.
- **2x2 Rotation Matrix**: A simplified method that accounts only for rotational and translational freedom. It lacks support for scaling and perspective, making it less effective for complex transformations.
- **Vector-based Estimation**: A direct estimation of rotation using vectors derived from image points. This method lacks robust outlier removal and complexity, limiting its effectiveness for real-world applications.

**Conclusion:** These methods were chosen to provide a range of complexity and computational efficiency, with homography and affine estimation being the primary focus due to their broader applicability in image transformations.

### 2.5.3. Improvement Techniques

To enhance the accuracy and robustness of these methods, the following improvement techniques were applied:

- **RANSAC (Random Sample Consensus)**: A robust outlier removal technique that iteratively refines the inlier set to accurately estimate transformations.

### 2.5.4. Initial Accuracy Testing

Initial tests were conducted using optimal, yet generalizable, parameters, and the performance of each method was evaluated based on its Mean Absolute Error (MAE) in heading relative to the ground truth. Since the ground truth heading data was unavailable for changes in heading, a dataset with only fixed North headings was used. The results are summarized below in table 2.4.

| Method | MAE (heading) | Result |
|---|---|---|
| Homography | 0.1207 | Accurate |
| Affine | 0.1129 | Best-performing |
| 2x2 Rotation Matrix | 1.0277 | Poor performance |
| Vector-based | 7.3571 | Poor performance |

**Table 2.4:** Initial Testing Results (MAE - Mean Absolute GPS Error)

**Conclusion:** The affine method achieved the lowest MAE, making it the most accurate for aligning images. Homography performed slightly worse due to its additional degrees of freedom, which introduced unnecessary errors. The 2x2 rotation matrix and vector-based methods were unsuitable for the task due to their lack of transformation handling complexity. Based on these results, affine and homography were selected for further testing.

### 2.5.5. Rotational and Translational stage sensitivity

To assess the overall suitability of the rotational estimators, it is crucial to evaluate their impact on the system as a whole. Focusing solely on heading errors does not capture the full extent of how minor inaccuracies propagate through the system. By examining the Mean Normalized Error (MNE) in GPS coordinates, a more realistic understanding of the cumulative effect of small rotational errors and their influence on overall system performance is obtained. RMSE was used. Pair with local retrofit to global matcher.

| Method | Mean Normalized Error (GPS) |
|---|---|
| Homography | 48.90 |
| Affine | 41.28 |

**Table 2.5:** Accuracy Comparison for Global and Local Matching Techniques (MNE - Mean Normalized Error)

**Conclusion:** The affine method consistently outperforms homography in both global and local matching scenarios. The significant reductions in GPS error, even with relatively minor decreases in rotational error, highlight the sensitivity of both global matching and translational estimation stages to rotational inaccuracies. This underscores the critical importance of optimizing the accuracy of the rotational stage. It is clear that the extra degrees of freedom in homography, which were not present in the dataset, introduce unnecessary errors, making affine the preferred method for this task.

### 2.5.6. Time Constraints

Time efficiency was evaluated to assess the computational performance of each method. The table below summarizes the average, minimum, median, and maximum runtimes for each approach. Lower computation times enable higher frame rates, provide greater tolerance for minor errors, and allow for faster processing of larger search spaces, ultimately improving accuracy.

| Method | Mean Time (ms) | Max Time (ms) | Min Time (ms) | Median Time (ms) |
|---|---|---|---|---|
| Affine | 1.276 | 11.55 | 0.00 | 0.999 |
| Homography | 22.58 | 50.74 | 0.98 | 10.80 |

**Table 2.6:** Time Analysis for Affine and Homography Methods

**Conclusion:** Affine is significantly faster than homography, with a much lower mean and median runtime. The greater variability in homography's runtime (indicated by the difference between mean and median), and mean runtime was seen and is as a result of its greater degrees of freedom. Affine's speed makes it particularly well-suited for real-time UAV applications.

### 2.5.7. Robustness Testing

Robustness testing was performed to assess each method's sensitivity to different parameter settings, such as RANSAC thresholds, Lowe's ratio for match filtering, and keypoint confidence thresholds. These tests are essential to determine the methods' reliability under varying conditions and data quality.

### RANSAC Threshold Testing

We varied RANSAC thresholds to evaluate how sensitive each method is to outlier removal i.e. amt of false positives and false negatives in the dataset. A lower threshold implies more aggressive filtering, which reduces the number of keypoints but increases the reliability of those retained. A higher threshold retains more keypoints but may introduce more noise.

| Threshold | Homography (Mean Heading Error) | Affine (Mean Heading Error) |
|---|---|---|
| 0.2 | 0.1384 | 0.1089 |
| 0.5 (Default) | 0.1207 | 0.1129 |
| 5 | 0.1249 | 0.1003 |
| 25 | 0.1222 | 0.0997 |
| 50 | 0.1226 | 0.1112 |

**Table 2.7:** Effect of RANSAC Thresholds on Heading Error

**Conclusion:** Both methods are robust to changes in keypoint quality and number, but homography exhibits greater variability at lower thresholds, as it requires more datapoints to accurately estimate its increased degrees of freedom.

### Lowe's Ratio Robustness Testing

Evaluations were done for different Lowe's ratios. This tests the ability for the method to handle incorrect matches caused by ambiguity, as well as perform when too many matches are filtered out.

| Lowe's Ratio | Homography (Mean Heading Error) | Affine (Mean Heading Error) |
|---|---|---|
| 0.6 | 0.1289 | 0.1098 |
| 0.7 | 0.1255 | 0.1006 |
| 0.8 | 0.1207 | 0.0997 |
| 0.9 | 0.1139 | 0.1111 |
| 0.95 | 0.1211 | 0.1054 |

**Table 2.8:** Effect of Lowe's Ratio on Heading Error

**Conclusion:** Both methods are robust to changes in Lowe's ratio, handling variations in match quality and quantity effectively, as before.

### Keypoint Count Testing

xxx redo this with varied amt of kps. Keypoint confidence thresholds were varied to evaluate how each method performs with different levels of keypoint quality. Lower

thresholds admit more keypoints but decrease the reliability of individual keypoints.

| Keypoint Confidence Threshold | Homography (Mean Heading Error) | Affine (Mean Heading Error) |
|---|---|---|
| 0.001 | 0.1404 | 0.1019 |
| 0.0008 | 0.1207 | 0.0974 |
| 0.0005 | 0.1247 | 0.0997 |
| 0.0002 | 0.1247 | 0.1024 |

**Table 2.9:** Effect of Keypoint Confidence Thresholds on Heading Error

xxx test for akaze and orb.

**Conclusion:** Both methods are robust to changes in keypoint confidence thresholds, but affine remains more consistent at lower thresholds, maintaining lower mean errors with noisier keypoints.

#### Overall Robustness

In summary, affine outperforms homography in terms of robustness, particularly when a low number of keypoints is avaiable. Further, it outperforms it in terms of accuracy across all tests. This is due to its lower degrees of freedom, which limits the introduction of errors due to a more complex model which is necessary to account for more distortions, which are not present in the given application.

### 2.5.8. Conclusion

Affine estimation is selected as the primary method for future work due to its superior accuracy, faster computation, and greater robustness. Its 6 degrees of freedom provide sufficient understanding of image transformations without introducing unnecessary errors from perspective correction. These degrees are : xxx

Best Rotator: Affine, AKAZE AND BF

xxx test optimal RANSAC threshold

# Chapter 3

# Summary and Conclusion

# Appendix A

# Project Planning Schedule

This is an appendix.

# Appendix B

# Outcomes Compliance

This is another appendix.