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.
ε	The Bayes error.
ε_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^{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.
μ	Statistical mean vector.
Σ	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 μ and covariance matrix Σ.
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

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

Comparison of Methods

2.1. 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.

2.1.1. Datasets

Five distinct datasets were selected to comprehensively test the methods' ability to generalize and perform in different environments:

- **CITY1 and CITY2** (Cape Town): Both datasets originate from Cape Town. CITY1 incorporates both rotational and translational changes between frames, while CITY2 includes only translational changes. This distinction allows for the isolation of performance under rotational loads.
- **ROCKY**: Represents rugged terrain with varied features, testing the methods' ability to handle complex topographies.
- **DESERT and AMAZON**: Characterized by extreme sparsity and repetitive patterns, these datasets present significant challenges even for human observers to distinguish differences.

2.1.2. Testing Structure

Each method, including feature extraction, local feature matching, rotational estimation, image similarity computation, translational estimation, and optimization techniques, is subjected to rigorous testing based on the following criteria:

- **Accuracy**: Evaluated using the Root Mean Square Error (RMSE) of GPS estimations, accuracy assessments focus on the end error. This is justified because

outputs—whether images or degrees—propagate through the pipeline, meaning any errors or poor choices are ultimately reflected in the final GPS error.

- **Accuracy**: Evaluated using the Root Mean Square Error (RMSE) of GPS estimations, accuracy assessments focus on the end error. This approach is preferable because intermediate outputs do not always provide clear indicators of quality. For instance, the number of keypoints can be misleading; without context, it's difficult to determine if they represent good features or excessive noise. Outputs propagate through the pipeline, meaning any errors or poor choices are ultimately reflected in the final GPS error, making it more effective to measure accuracy at the end.

- **Runtime**: The entire system runtime per dataset is assessed to evaluate the computational efficiency of each method. Efficient runtimes are crucial for real-time UAV applications, as delays—combined with pilot and UAV response times—can result in consistently missing the target and failing to follow the intended path. As before, runtime may propagate through the system, and therefore the runtime per line is not necessarily indicative of better performance; Runtime per line is not tested.

- **Robustness**: Tested to verify each method's stability under parameter variations and challenging conditions. Robust methods maintain consistent performance despite environmental or parameter changes, ensuring reliable UAV navigation across different scenarios.

This structured testing approach ensures that each component of the UAV navigation system is thoroughly evaluated, facilitating the selection of methods that deliver optimal performance across all critical metrics.

2.1.3. Parameters

Parameters were held constant across tests and chosen to ensure optimal performance across methods. The focus was not on selecting the absolute best parameter set, as this was not relevant for inter-method testing; rather, the goal was to minimize bias while allowing each method to perform effectively. For instance, multiple methods were tested within a single runtime to enhance testing speed without compromising the integrity of the inter-method comparisons. The takeaway here is to avoid interpreting results objectively, as they do not reflect the realistic and overall performance of the system.

2.2. Feature Detectors

This section presents the evaluation results of three feature detectors: ORB, AKAZE, and SuperPoint with LightGlue. The primary objective of this evaluation is to identify the

most suitable detector for a UAV navigation system tasked with estimating rotational and translational transformations. Key performance metrics, including RMSE (Root Mean Square Error), runtime, and robustness across multiple datasets, were considered to assess each detector's effectiveness and suitability.

2.2.1. Testing Overview

The feature detectors were assessed based on their ability to accurately estimate transformations under various conditions. Consistent parameters were maintained across all datasets to ensure a fair comparison. Notably, AKAZE lacked a built-in keypoint target parameter, resulting in highly variable runtimes across datasets due to the necessity of selecting a parameter that identified sufficient keypoints for each dataset. To address this variability, dynamic thresholding was implemented for AKAZE, enabling keypoint target thresholds comparable to those of the other detectors.

The evaluation focused on three main criteria:

- **Accuracy (RMSE in GPS)**: Measures the precision of transformation estimates.
- **Runtime**: Assesses the computational efficiency of each detector.
- **Robustness**: Evaluates consistency and reliability across different datasets.

The results from these tests informed the selection of the most appropriate detector and threshold parameters for the four defined stages utilizing the detected keypoints: rotational alignment for global matching, global similarity computation, precise translation estimation, and precise rotational estimation.

2.2.2. Translation Estimation Performance

Table 2.1 presents the RMSE values in meters for each feature detector applied to the precise translation estimation stage. AKAZE, utilizing dynamic keypoint targeting, demonstrated the highest accuracy across all datasets while maintaining reasonable runtime. ORB exhibited respectable accuracy but was consistently outperformed by AKAZE. SuperPoint recorded the highest RMSE values, particularly in challenging datasets such as ROCKY and AMAZON, indicating its limited generalizability across diverse environments.

Table 2.1: RMSE for Various Local Detectors (in meters)

Method	CITY1	CITY2	ROCKY	DESERT	AMAZON
ORB	70.89	10.56	43.32	80.66	46.39
Dynamic AKAZE (3000 keypoints)	66.80	6.91	22.48	33.07	39.27
SuperPoint	72.66	15.80	114.93	31.60	329.19

Runtime and Efficiency

Table 2.2 illustrates the runtime (in seconds) for each feature detector across different datasets. ORB proved to be the most efficient detector, making it suitable for applications requiring fast processing. Dynamic AKAZE, while exhibiting longer runtimes, achieved a balance between efficiency and accuracy. SuperPoint demonstrated the longest runtimes across all datasets, highlighting its limited applicability for time-sensitive applications unless optimized with GPU acceleration.

Table 2.2: Runtime for Various Local Detectors (in seconds)

Method	CITY1	CITY2	ROCKY	DESERT	AMAZON
ORB	75.25	70.77	111.64	58.29	75.34
Dynamic AKAZE (3000 keypoints)	112.18	104.16	127.86	108.86	119.76
SuperPoint	338.21	292.11	307.93	277.73	291.01

Robustness

In terms of robustness, AKAZE exhibited the highest consistency in accuracy across different datasets. ORB also performed well but showed greater variability in accuracy across datasets compared to AKAZE. SuperPoint, however, demonstrated significant performance variability, indicating lower robustness across diverse environments.

2.2.3. Rotation Estimation Performance

Table 2.3 presents the RMSE values in meters for ORB and AKAZE applied to the precise rotational estimation stage. Both detectors were optimized for this stage to ensure high accuracy while maintaining reasonable runtime.

Table 2.3: RMSE for ORB (8000 keypoints) and AKAZE (6000 keypoints) (in meters)

Method	CITY1	CITY2	ROCKY	DESERT	AMAZON
ORB (8000 keypoints)	69.42	8.17	22.38	32.55	37.93
Dynamic AKAZE (6000 keypoints)	68.14	8.34	21.72	35.17	39.25

Runtime and Efficiency

Table 2.4 presents the runtime comparisons for ORB and AKAZE in the rotational estimation stage. ORB outperformed AKAZE in terms of speed, especially in denser datasets, while maintaining similar levels of accuracy. This suggests that ORB is better suited for time-sensitive applications within this stage.

Table 2.4: Runtime for ORB (8000 keypoints) and AKAZE (6000 keypoints) (in seconds)

Method	CITY1	CITY2	ROCKY	DESERT	AMAZON
ORB (8000 keypoints)	95.80	106.63	104.79	99.56	84.70
Dynamic AKAZE (6000 keypoints)	108.44	131.10	120.77	111.25	98.70

Robustness

Regarding robustness, AKAZE demonstrated more consistent accuracy across various datasets compared to ORB, which exhibited greater variability in performance. Despite this, ORB's consistently lower runtimes make it a favorable option for applications where speed is critical.

2.2.4. Final Selection of Feature Detectors

Based on the comprehensive evaluation, the following detectors were selected for the respective stages of the UAV navigation system:

- **Translation Estimation Stage:** AKAZE, with dynamic keypoint targeting (3000 keypoints), exhibited the highest accuracy and robustness in translation estimation while maintaining reasonable runtime. This balance makes Dynamic AKAZE the optimal choice for precise translation inference.
- **Rotational Alignment for Translation Estimation Stage:** ORB, utilizing 8000 keypoints, offers an excellent balance between accuracy and efficiency. Although AKAZE demonstrated slightly better performance in rotational estimation, ORB's significantly lower runtime makes it the preferred option for this stage.
- **Rotational Alignment for Global Matching Stage:** ORB, with 6000 keypoints, is the optimal choice due to its superior runtime efficiency. This stage can accommodate relatively larger rotational estimation errors, making ORB ideal for global matching tasks.
- **Global Similarity Refinement:** ORB, employing 1500 keypoints, is selected for global image similarity comparison using local matching. The lower accuracy requirement and the necessity for high runtime efficiency make ORB the best fit for grid matching.

These selections ensure that each stage of the UAV navigation system leverages the most appropriate feature detector, optimizing both accuracy and computational performance.

2.3. Local Feature Matchers

This section evaluates two prominent local matchers, BFMatcher and FLANN, within the context of a UAV navigation system. The primary objective of this evaluation is to determine the most efficient and robust matcher in terms of accuracy, runtime, and consistency across diverse datasets. These matchers play a crucial role in the system's ability to accurately estimate rotational and translational transformations by effectively pairing detected feature points.

2.3.1. Testing Overview

The local matchers were assessed under various conditions to evaluate their performance concerning accuracy (RMSE in GPS), runtime, and robustness. The evaluation was meticulously designed to understand how each matcher handles noisy keypoints, limited post-match filtering, and varying detector thresholds. The goal was to identify which matcher offers the most suitable balance between match quality and computational efficiency, thereby ensuring real-time applicability in UAV navigation.

2.3.2. Accuracy Evaluation

Table 2.5 presents the Root Mean Squared Error (RMSE) in GPS 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.

Table 2.5: RMSE GPS Accuracy for BFMatcher and FLANN (in meters)

Matcher	CITY1	CITY2	ROCKY	DESERT	AMAZON
FLANN	56.59	4.70	14.63	71.20	32.35
BFMatcher	53.89	3.79	16.36	68.64	33.24

Observations:

- **BFMatcher Accuracy:** BFMatcher demonstrated slightly superior accuracy in CITY1 and CITY2 compared to FLANN. However, the improvement was not substantial enough to outweigh its increased computational demands.
- **FLANN Accuracy:** FLANN maintained competitive accuracy levels, trailing BFMatcher only marginally in most datasets. Notably, FLANN exhibited strong performance in the ROCKY dataset, underscoring its reliability in challenging environments.

2.3.3. Runtime Evaluation

Table 2.6 illustrates the runtime (in seconds) for BFMatcher and FLANN across different datasets. The results clearly show that FLANN consistently outperforms BFMatcher in terms of speed, with significantly lower execution times across all datasets.

Table 2.6: Runtime Comparison for BFMatcher and FLANN (in seconds)

Matcher	CITY1	CITY2	ROCKY	DESERT	AMAZON
FLANN	42.72	41.61	41.96	43.42	53.62
BFMatcher	203.49	228.59	46.33	52.59	88.95

Observations:

- **FLANN Efficiency:** FLANN exhibited significantly faster runtimes across all datasets, particularly excelling in CITY1 and CITY2 where it completed tasks in less than a quarter of the time required by BFMatcher.
- **BFMatcher Runtime:** The exhaustive matching approach of BFMatcher resulted in considerably higher runtimes, rendering it less suitable for real-time applications where speed is critical.

2.3.4. Robustness Under Varying Detector Thresholds

Figure 3.1 depicts the divergence in RMSE GPS error between FLANN and BFMatcher as the number of keypoints increases. Positive values indicate that BFMatcher outperforms FLANN, while negative values suggest FLANN performs better.

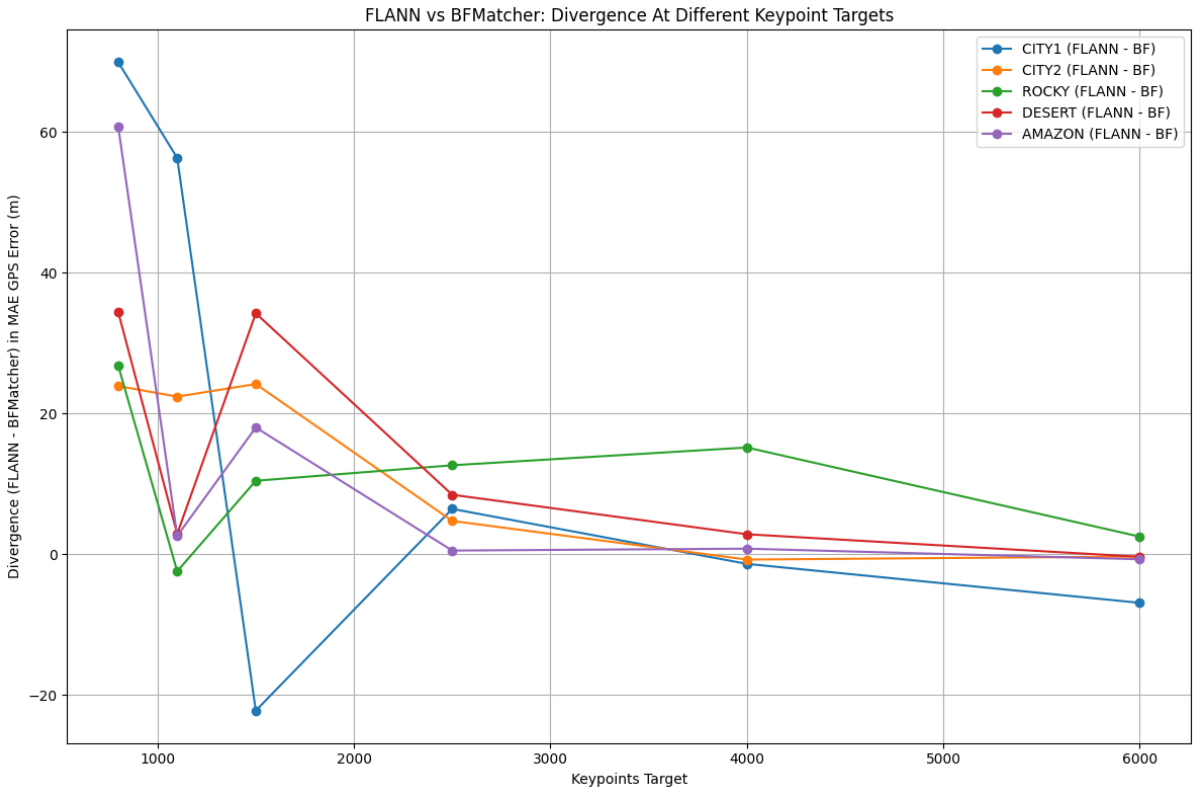

Figure 2.1: Divergence in RMSE GPS Error Between FLANN and BFMatcher Across Keypoint Targets

Observations:

- **Convergence:** Both matchers demonstrated similar accuracy levels as the number of keypoints increased, with performance becoming nearly identical beyond 2500 keypoints. Given that feature detectors are typically configured with a keypoint target of at least 3000, both matchers are expected to perform comparably in standard operational conditions.
- **Outliers:** Although BFMatcher generally outperforms FLANN, there are instances where FLANN achieves better accuracy. This can occur due to FLANN's approximate matching, which may preserve valid matches that BFMatcher might discard using Lowe's ratio thresholding.
- **Scalability:** FLANN's runtime scalability with increasing keypoint counts was superior to that of BFMatcher, making FLANN a more scalable solution for larger datasets.

2.3.5. 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. This makes FLANN highly suitable for real-time applications where computational efficiency is paramount.

- **Overall Choice: FLANN** is selected as the primary local feature matcher for the UAV navigation system, balancing high performance and efficiency across diverse operational conditions.

These selections ensure that the UAV navigation system leverages the most appropriate local matcher, optimizing both accuracy and computational performance to achieve reliable and efficient navigation.

2.4. Rotational Estimators

This section evaluates the performance of three rotational estimation methods: Partial Affine 2D, Affine 2D, and Homography. The objective is to identify the most suitable method for UAV navigation by comparing accuracy, runtime, and robustness across various datasets. These estimators are critical for accurately determining the rotational transformations required for precise navigation and alignment of UAV imagery.

2.4.1. Testing Overview

The rotational estimators were assessed based on their ability to accurately estimate rotations under diverse conditions. Consistent parameters were maintained across all datasets to ensure a fair comparison. The evaluation focused on three primary criteria:

- **Accuracy (RMSE in GPS)**: Measures the precision of rotational estimates.
- **Runtime**: Assesses the computational efficiency of each estimator.
- **Robustness**: Evaluates the consistency and reliability of each method across different datasets.

Additionally, an improvement technique involving the application of a rigid transform using Singular Value Decomposition (SVD) was explored. Although initially less effective as a standalone method, the rigid transform consistently enhanced overall accuracy when applied after other estimation methods.

2.4.2. Accuracy Evaluation

Root Mean Square Error (RMSE) was utilized to evaluate the estimation error of each method across five datasets. Table 2.7 summarizes the RMSE values (in meters) for Partial Affine 2D, Affine 2D, and Homography.

Table 2.7: RMSE Comparison Across Datasets for Partial Affine 2D, Affine 2D, and Homography

Dataset	Partial Affine 2D	Affine 2D	Homography
CITY1	70.18	81.22	76.25
CITY2	7.54	7.80	7.16
ROCKY	27.98	22.10	24.40
DESERT	44.94	43.11	48.56
AMAZON	46.98	52.56	48.04

Observations:

- **Partial Affine 2D Accuracy:** Partial Affine 2D exhibited the lowest combined normalized RMSE across all datasets, indicating superior overall accuracy.
- **Affine 2D Accuracy:** Affine 2D demonstrated competitive accuracy, particularly excelling in the ROCKY dataset.
- **Homography Accuracy:** Homography maintained respectable accuracy but was outperformed by Partial Affine 2D in most datasets.
- **Dataset-Specific Performance:** The most accurate estimator varied per dataset, highlighting the importance of method selection based on specific environmental conditions.

2.4.3. Runtime Evaluation

Table 2.8 presents the runtime (in seconds) for each rotational estimator across the five datasets. The results reflect the computational efficiency of each method relative to their transformation complexity.

Table 2.8: Runtime Comparison Across Datasets for Partial Affine 2D, Affine 2D, and Homography

Dataset	Partial Affine 2D	Affine 2D	Homography
CITY1	56.76	65.27	68.79
CITY2	52.20	54.55	65.11
ROCKY	69.44	76.16	78.92
DESERT	56.40	71.52	69.97
AMAZON	50.34	58.16	58.27

Observations:

- **Runtime Efficiency:** Partial Affine 2D consistently demonstrated the fastest runtimes across all datasets, followed by Affine 2D and Homography.

- **Complexity Correlation:** The runtime generally correlated with the transformation model's complexity, with more complex models requiring longer computation times.
- **Dataset-Specific Efficiency:** Partial Affine 2D maintained superior efficiency even in more demanding datasets such as ROCKY and DESERT.

2.4.4. Robustness Testing

Robustness testing assessed each method's sensitivity to parameter variations, including Lowe's ratio, RANSAC thresholds, and keypoint quantity. This evaluation was essential to determine the performance consistency of each method under different conditions. For the sake of brevity, the results are omitted; The observations are summarized below.

Observations:

- **Parameter Sensitivity:** All methods exhibited high robustness against parameter variations, maintaining consistent performance across different settings.
- **Homography Variability:** Homography showed greater variability in accuracy when fewer matches were available, attributable to the complexity of its transformation model.
- **Consistent Performance:** Partial Affine 2D maintained stable performance across all datasets, reinforcing its suitability for diverse environmental conditions given its simplified transformation model.

2.4.5. Improvement Technique

An additional method, the rigid transform using Singular Value Decomposition (SVD), was initially evaluated as a primary rotational estimator. However, it performed significantly worse than the other methods and was subsequently excluded from standalone consideration. Notably, when the rigid transform was applied after other estimation methods, it consistently enhanced overall accuracy. This improvement is likely due to the rigid transform's ability to make minor, precise corrections once the point cloud is largely aligned by other methods, despite its initial inefficiency in handling outliers in the case of large misalignments.

2.4.6. Final Selection of Rotational Estimator

Based on the comprehensive evaluation of accuracy, runtime, and robustness, Partial Affine 2D emerged as the most suitable primary rotational estimator for the UAV navigation system. It demonstrated the lowest combined normalized RMSE across all datasets, the

fastest runtime, and high robustness. Additionally, the application of the rigid transform using SVD after Partial Affine 2D further enhanced overall accuracy, ensuring precise rotational alignment.

2.5. Image Similarity Estimators

Accurate image similarity estimation, or global matching, is essential for UAV navigation systems to choose reasonable images to compare to. They should ensure accuracy and efficiency while maintaining robustness against small rotational offsets. This section evaluates different global matching techniques across multiple datasets to identify the most suitable method for UAV navigation based on accuracy, runtime, and robustness.

2.5.1. Accuracy Evaluation

Naturally, the appropriateness of the choice of best match is implicitly passed through to subsequent stages and realized as an error in GPS estimations. Root Mean Square Error (RMSE) was utilized to assess the GPS estimation accuracy of each global matching technique across the five datasets. Table 2.9 summarizes the RMSE values (in meters) for Local Retrofit, Cross Correlation, Histogram, and SSIM.

Table 2.9: RMSE Comparison Across Datasets for Various Global Matching Techniques (in meters)

Global Matching Technique	CITY1	CITY2	ROCKY	DESERT	AMAZON
Local Retrofit	46.56	7.37	17.32	128.43	34.54
Cross Correlation	49.88	5.12	15.77	26.84	31.50
Histogram	46.35	5.12	15.77	27.28	31.50
SSIM	50.03	6.16	15.77	27.28	31.44

Observations:

- **Histogram Accuracy:** The Histogram technique consistently achieved the lowest RMSE across most datasets, particularly excelling in CITY1, CITY2, and ROCKY.
- **Cross Correlation Accuracy:** Cross Correlation closely followed Histogram in accuracy, demonstrating strong performance in CITY2 and DESERT.
- **SSIM Accuracy:** SSIM maintained comparable accuracy to Histogram and Cross Correlation but exhibited slightly higher errors in CITY1 and DESERT.

- **Local Retrofit Accuracy:** The Local Retrofit method recorded the highest RMSE values, especially in the DESERT dataset, indicating poor generalizability and higher complexity. Consequently, it was excluded from further analysis.

2.5.2. Runtime Evaluation

Table 2.10 compares the computational efficiency of each global matching technique across the five datasets.

Table 2.10: Runtime Comparison Across Global Matching Techniques (in seconds)

Global Matching Technique	CITY1	CITY2	ROCKY	DESERT	AMAZON
Local Retrofit	84.56	76.34	70.22	63.47	90.69
Cross Correlation	59.62	56.67	54.14	59.90	65.92
Histogram	57.06	56.33	56.68	51.98	59.09
SSIM	91.60	77.99	83.31	83.08	114.07

Observations:

- **Histogram Efficiency:** The Histogram technique was the most efficient in terms of runtime, outperforming all other methods consistently across all datasets.
- **Cross Correlation Efficiency:** Cross Correlation followed closely behind Histogram, offering slightly higher runtimes but still maintaining high computational efficiency.
- **SSIM Efficiency:** SSIM exhibited the longest runtimes, particularly in the AMAZON and DESERT datasets, rendering it less suitable for real-time applications.
- **Local Retrofit Efficiency:** The Local Retrofit method had the highest runtimes and was deemed non-viable due to its excessive computational cost and poor accuracy.

2.5.3. Robustness to Rotational Offsets

Robustness testing evaluated each global matching technique's sensitivity to a 10-degree rotational offset. All matchers that made it this far saw no change in choice combination below a 5-degree offset. The impact on RMSE GPS error and the percentage change in error were recorded to assess each method's stability under rotational misalignments.

Table 2.11: RMSE (GPS Error) and Percentage Change with 10-degree Rotational Offset

Method	Metric	CITY1	CITY2	ROCKY	DESERT	AMAZON
Cross Correlation	RMSE	54.81	5.34	17.55	32.51	31.52
	% Change	5.89%	4.29%	11.54%	21.92%	0.06%
Histogram	RMSE	56.17	4.45	16.83	37.20	36.35
	% Change	20.93%	-13.09%	6.73%	36.73%	15.38%
SSIM	RMSE	55.93	5.53	16.25	41.01	31.67
	% Change	11.79%	-10.23%	3.04%	50.43%	0.73%

Observations:

- **Cross Correlation Robustness:** Cross Correlation exhibited the lowest percentage change in GPS error under a 10-degree rotational offset, indicating strong robustness.
- **Histogram Robustness:** While Histogram demonstrated competitive RMSE values, it showed significant deviations in the DESERT and AMAZON datasets when subjected to rotational offsets.
- **SSIM Robustness:** SSIM displayed higher error rates and greater variability, particularly in the DESERT dataset, making it less robust to significant rotational misalignments.

2.5.4. Improvement Technique

An additional method, the rigid transform using Singular Value Decomposition (SVD), was initially evaluated as a primary global matching technique. However, it performed significantly worse than the other methods and was subsequently excluded from standalone consideration. However, when the rigid transform was applied after other estimation methods, it consistently enhanced overall accuracy. This improvement is likely due to the rigid transform's ability to make minor, precise corrections once the point cloud is largely aligned by other methods, despite its initial inefficiency in handling large misalignments.

2.5.5. Final Selection of Global Matching Technique

Based on the comprehensive evaluation of accuracy, runtime, and robustness, the **Histogram** technique is identified and chosen as the most suitable global matching method for the system. Histogram consistently provided superior performance in terms of both RMSE and runtime, particularly under large rotational offsets. Although Cross Correlation also demonstrated strong performance, Histogram's marginally better accuracy and runtime efficiency make it the preferred choice.

2.6. Translational Estimators

This section evaluates the performance of various translational estimation methods for UAV navigation. The methods assessed include Phase Correlation, Rigid Transform, Affine Transform with RANSAC, Homography Transform, and Partial Affine 2D. Each method is evaluated based on accuracy, runtime, and robustness across multiple datasets to identify the most effective option for UAV navigation.

2.6.1. Accuracy Evaluation

Root Mean Square Error (RMSE) was utilized to assess the GPS estimation accuracy of each translational estimation method across five datasets. Table 2.12 summarizes the RMSE values (in meters) for each method.

Table 2.12: RMSE GPS Error Comparison Across Datasets for Different Translation Methods

Method	CITY1	CITY2	ROCKY	DESERT	AMAZON
Phase Corr	1437.85	1349.16	629.82	1263.15	1121.81
Rigid Transform	64.31	7.42	21.82	29.01	38.52
Affine Transform	127.77	88.41	57.01	70.96	118.77
Homography	168.35	92.54	58.69	120.15	242.97
Partial Affine 2D	64.11	6.34	19.07	29.94	40.08

Observations:

- **Partial Affine 2D** achieved the lowest RMSE, establishing it as the most accurate estimator across all datasets. Its superior performance compared to the direct algebraic **Rigid Transform** is attributed to the utilization of RANSAC for effective outlier rejection.
- Both **Homography** and **Affine Transform** methods exhibited significantly higher RMSE values than Partial Affine 2D and Rigid Transform. This increase is primarily due to their greater degrees of freedom and inherent complexity.
- **Phase Correlation** recorded the highest RMSE values, indicating lower accuracy relative to the other methods. This reduced performance is likely due to its heightened sensitivity to noise, as it processes the entire image context.

2.6.2. Runtime Evaluation

Table 2.13 presents the runtime (in seconds) for each translational estimation method across the five datasets.

Table 2.13: Runtime Comparison Across Datasets for Different Translation Methods (in seconds)

Method	CITY1	CITY2	ROCKY	DESERT	AMAZON
Phase Corr	1437.85	1349.16	629.82	1263.15	1121.81
Rigid Transform	89.05	99.74	103.46	104.84	87.43
Affine Transform	127.77	88.41	57.01	70.96	118.77
Homography	168.35	92.54	58.69	120.15	242.97
Partial Affine 2D	94.62	82.77	85.94	72.43	78.33

Observations:

- **Partial Affine 2D** demonstrated the fastest runtime among the translational methods, making it the most computationally efficient. It was followed by the **Rigid Transform**.
- **Phase Correlation** exhibited significantly longer runtimes, making it less viable for real-time UAV applications.

2.6.3. Robustness Testing

Robustness was assessed by evaluating each method's sensitivity to parameter variations, such as keypoint quantity and threshold changes. Table 2.14 highlights the standard deviation of RMSE across datasets to indicate consistency.

Table 2.14: RMSE Variance Across Datasets for Different Translation Methods (Robustness Test)

Method	CITY1	CITY2	ROCKY	DESERT	AMAZON
Phase Corr	111.35	76.54	55.69	110.15	101.97
Rigid Transform	64.61	7.53	21.14	31.31	37.96
Affine Transform	127.77	88.41	57.01	70.96	118.77
Homography	168.35	92.54	58.69	120.15	242.97
Partial Affine 2D	64.11	6.34	19.07	29.94	40.08

Observations:

- **Partial Affine 2D** and **Rigid Transform** exhibited the lowest variance, indicating strong robustness to parameter changes.
- **Homography** showed the most variability and performed inconsistently under different conditions.

2.6.4. Final Choice of Translational Estimator

Based on the evaluation of accuracy, runtime, and robustness, **Partial Affine 2D** was selected as the most suitable translational estimator for the UAV navigation system. It demonstrated the highest accuracy, fastest runtime, and strong robustness across all datasets.

- **Primary Translational Estimator: Partial Affine 2D** is chosen for its optimal performance in minimizing GPS error, computational efficiency, and consistent robustness across diverse operational conditions.

Optimization Techniques

compare RANSAC vs other outlier rejection methods

Chapter 3

Local Feature Matchers

3.1. 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 3.1 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 3.1: 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.

3.1.1. 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 3.2.

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 3.2: 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 3.1.

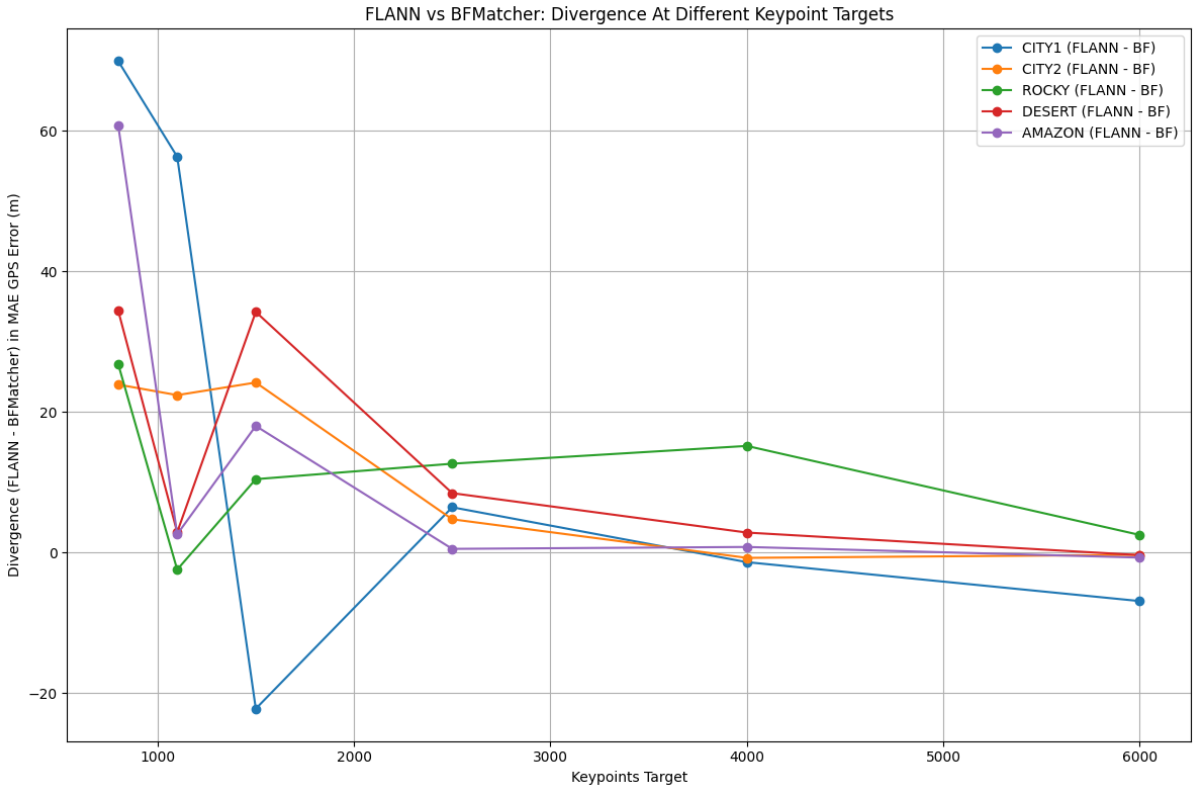

Figure 3.1: Divergence in MAE GPS Error Between FLANN and BFMatcher Across Keypoint Targets.

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 3.3.

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 3.3: 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.

Note: The runtime of individual methods should not be used for direct time comparisons, as it must be evaluated within the context of the entire pipeline. For example, a match filtration method might execute faster because it filters out fewer matches, but the resulting increase in matches could slow down subsequent steps. The goal is to assess the overall impact of the method on the entire pipeline, not just the method's isolated runtime.

THEORY To Be Completed.

3.2. Rotational Estimators

3.2.1. Initial Accuracy-Runtime Testing

Initial tests were conducted with optimal, yet generalizable, parameters to evaluate each method's Mean Absolute Error (MAE) in heading estimation. Since ground truth heading data was unavailable for heading changes, a dataset with fixed North headings was used. Note that methods without built-in RANSAC that aimed to estimate purely based on the point cloud proved ineffective in empirical testing. Results are summarized in Table 3.6.

Table 3.4: RMSE Comparison Across Datasets for Partial Affine 2D, Affine 2D, and Homography

Dataset	Partial Affine 2D	Affine 2D	Homography
CITY1	70.18	81.22	76.25
CITYT2	7.54	7.80	7.16
ROCKY	27.98	22.10	24.40
DESERT	44.94	43.11	48.56
AMAZON	46.98	52.56	48.04

Table 3.5: Runtime Comparison Across Datasets for Partial Affine 2D, Affine 2D, and Homography

Dataset	Partial Affine 2D	Affine 2D	Homography
CITY1	56.76	65.27	68.79
CITYT2	52.20	54.55	65.11
ROCKY	69.44	76.16	78.92
DESERT	56.40	71.52	69.97
AMAZON	50.34	58.16	58.27

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

Table 3.6: Initial Testing Results (MAE - Mean Absolute GPS Error)

Conclusion: The affine and homography methods performed sufficiently well to be considered for further testing. The 2x2 rotation matrix and vector-based methods were discarded due to poor performance. Their inability to iteratively solve and filter matches led to significant error points, making them unsuitable for the task at hand.

3.2.2. Detailed Time, Robustness & Accuracy Testing

Dataset	Homography		Affine	
	RMSE GPS	Runtime (s)	RMSE GPS	Runtime (s)
CITY1	73.7828	66.1684	65.5642	62.3034
CITY2	7.5038	68.6799	7.3890	64.9879
ROCKY	23.2824	88.6713	22.3195	80.8048
DESERT	49.0210	76.0751	38.6201	72.3996
AMAZON	48.6687	69.6631	44.5493	68.0990

Observations: Affine consistently outperformed homography in both accuracy (RMSE GPS) and runtime across all datasets. Homography's additional degrees of freedom (namely, perspective transformation) introduce unnecessary complexity and error when such transformations aren't significantly present.

3.2.3. Method Sensitivity

Sensitivity tests were conducted to assess each method's robustness to parameter variations (namely, Lowe's ratio, RANSAC thresholds, detector type, keypoint quantity). Both methods exhibited robustness to parameter changes, though homography showed marginally higher deviations when only having access to few matches, due to its more complex model. Both methods rated a 5/5 for robustness across varying datasets and conditions.

3.2.4. Conclusion

Affine estimation is selected for future work due to its superior accuracy, computational efficiency, and robustness. With its 6 degrees of freedom, it strikes a good balance for this

task. However, if the UAV flies at lower altitudes where perspective distortion becomes significant, the homography method may need to be reconsidered.

insert image showing generalized rot pattern which angles. ie how to rotate.

Chapter 4

Summary and Conclusion

Appendix A

Project Planning Schedule

This is an appendix.

Appendix B

Outcomes Compliance

This is another appendix.