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

Introduction

CH1 Background (1/2 to 1 page) - give context - background is not what is the problem, but what is the context of the problem. Why does it matter. How many people suffer from lack of clean water for eg or die from car accidents.

- problem statement (1 paragraph). Most important of entire report. Exactly what you have done. Nothing more or less. sentence or two. - objectives "fine-grain" SMART, numbers. If i meet these objectives, i have solved the problem statement in part. 1-5 objectives or sub-objectives or even requirements. Measured against objectives - metrics. Summary of work (not contribution for skripsie). what you have achieved. meeting objectives. - Scope: What will be done and not be done. Mainly what will not be done. Did not take into account organic material or eg did not consider mobility. Did think about other things - still important (e.g. future work). write it in such a way that it is positive / what u did do. did not consider cost size market manufacturability etc. - Structure of Report (Not exactly the same as the Table of Contents): Roadmap, in chapter 2 I did the following etc

ECSA chapter 1: I took a vaguely defined problem (prob statement), understood what i must do to solve (objectives), and then solved it (summary of work). Then,

1.1. Introduction

- **Background:**
  - To maintain state safety, Unmanned Aerial Vehicles (UAVs) are used in military missions for surveillance, reconnaissance, and intelligence gathering. However, they rely on GPS for navigation, which can be jammed, spoofed, or otherwise lost in adversarial environments. This severely limits the scope of where these missions can be conducted, as well as the safety of the UAVs.
- **Problem Statement:**
  - Traditional UAV navigation systems are GPS-based, which is susceptible to new GPS-denial technology. Additionally, existing solutions mainly involve emission technology (e.g., LIDAR) which can be detected by adversaries.

- **Motivation:!!!**
  - There is a need for a navigation redundancy system that avoids detection in GPS-denied environments. Intertial reference units drift over time FIX LATER. weighted based on confidence level flip.

1.2. Aims and Objectives

- **Aim:**
  - Develop a robust and accurate image-based GPS location estimator for UAV navigation post-GPS signal loss.
- **Objectives:**
  - Design, develop and test algorithms for efficient, robust and accurate feature extraction. The system shall have a good feature count of at least 1000 per image.
  - Design, develop and test algorithms for efficient, robust and accurate feature matching. The system requires at least 500 good matches to be sufficiently confident.
  - Use the matching algorithms to infer the changes in translation and rotation between the current and prior images when GPS signal is lost. This ultimately allows for the estimation of the UAV's new GPS location and heading. The system shall have a normalized x-y error under 100m and a heading error under 10 degrees for all estimations.
  - Improve the above method by utilizing stereo vision for depth awareness and increased accuracy. The system shall have a less than half the error of the monocular system.
  - Implement the system on a single-board computer (SBC) for real-time operation. The system shall have a processing time of under 10s per frame in the forward flight, and under 3s per frame in the backward flight.

1.3. Scope

- **Assumptions and Restrictions:**
  - Real-life footage from a UAV test-flight will be used. This camera shall be aimed perfectly downwards for all time, sufficiently detailed and distortion-free (e.g., warping, blurring, clouds, etc.).

  - The UAV will be flying at a constant altitude.
  - There will therefore be limited depth warping. That is, the accuracy of the system using 2D estimation will be sufficiently accurate'' but there will still be some room for improvement in 3D.
  - The UAV and its cameras will always be level with the ground. That is, no pitch and roll will occur.
  - The system output is an estimated GPS location and heading estimate, not a control system. This is sufficient for the navigators to manually control the UAV.

1.4. Methods and Approach

- **Data Collection:**
  - Capture sequential images from UAV flights at different altitudes.
  - The UAV should follow a predefined path with known GPS coordinates, heading, camera parameters, and altitude. Thereafter, the UAV should fly very close to the same path backward.
- **Design and Development:**
  - Research and test the accuracy and count of features in 5-10 different feature extraction methods.
  - Research and test the accuracy and count of matches in 5-10 different feature matching methods.
  - Implement and evaluate different homographic estimation methods.
  - Implement and evaluate stereo vision methods.
  - Implement and evaluate the system on a SBC.
- **Analysis:**
  - Use sequential image analysis to estimate new coordinates.
  - Implement and evaluate pathfinding algorithms to calculate a path backward using past captured features.

1.5. Summary of Work

- The project aims to develop a robust and accurate image-based GPS location estimator for UAV navigation post-GPS signal loss. The system will be designed,

developed, and tested for feature extraction, matching, homographic estimation, stereo vision, and computational requirements. The system will be implemented on a SBC for real-time operation.

- The project will have a significant impact on UAV navigation accuracy and reliability in GPS-denied environments, with potential applications in military and civilian contexts.

1.6. Structure

- **Chapter 2: Literature Review**
  - Discusses the current state of UAV navigation and GPS dependency, alternative navigation methods, feature detection and matching, homographic estimation, stereo vision, and computational requirements.
- **Chapter 3: Methodology**
  - Describes the data collection, design and development, and analysis methods used in the project.
- **Chapter 4: Results**
  - Presents the results of the project, including the accuracy and efficiency of the developed system.
- **Chapter 5: Discussion**
  - Discusses the implications of the results and the potential impact of the project.
- **Chapter 6: Conclusion**
  - Summarizes the project and its outcomes.

1.7. Potential Impact

- **Societal Impact:**
  - Enhance the reliability of UAV operations in civilian applications such as disaster response and border patrol.
- **Knowledge Impact:**
  - Contribute to advancements in UAV navigation technology and computer vision applications.

- **Economic Impact:**
  - Increase the practicality and usage of UAV systems in military and civilian contexts, potentially enhancing border control, enhancing international electronic engineering collaboration.
- **Safety Impact:**
  - Increase the ability for national surveillance missions to secure critical knowledge in a military context.

1.8. Conclusion

- The project aims to improve UAV navigation accuracy and reliability in GPS-denied environments through camera-based image matching. This may have profound impacts in a South African military context as well as broader impacts on civil safety and economic well-being.

References

1. Sathyamoorthy, D., et al. (2020). Evaluation of the Vulnerabilities of Unmanned Aerial Vehicles (UAVs) to Global Positioning System (GPS) Jamming and Spoofing. *Defense Science & Technology Journal*, 33, 334-343.

2. Rusnák, M. & Vásárhelyi, J. (2023). A review of using visual odometry methods in autonomous UAV navigation in GPS-denied environment. *Acta Universitatis Sapientiae Electrical and Mechanical Engineering*, 15, 14-32. https://doi.org/10.2478/auseme-2023-0002

3. Karab, E., Prasad, S., & Shahato, M. (2017). Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. *Acta Polytechnica Hungarica*, 14(6), 183-206.

4. Luglio-Miguel, F., Iores, G., & Salazar, S. & Lozanc, R. (2014). Dubins path generation for a fixed-wing UAV. *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, 339-345.

Chapter 2

Feature Extractors

2. What information did u draw from or acquire to solve problem. What did I learn not from EE engineering. Additional to undergrad.

e.g. water quality: UV spectrum, optical sensing, color spectrum moving signals. Capture additional technologies and information to do design. Anything beyond scope of EE, lecturer might not know. I needed to know this before addressing problem.

eg i have these 3 options at my disposal or many.

2.1. Intropara

This chapter focuses on the additional prerequisite knowledge required to solve the problem above what the reader is expected to know. The image-based navigation system involves many functional parts which need to be understood and integrated to solve the problem. This includes the technique of feature extraction, feature matching, and homographic estimation based on the change in rotation and translation of the landscape.

Feature Extractors

In this report, we conduct an in-depth evaluation of various feature extractors to determine the most suitable one for real-time, robust and accurate UAV-based navigation.

- what is a feature extractor - why is it important - what are the factors that affect the performance of a feature extractor - what are the types of feature extractors (How many are there) - Parameters of feature extractors - which ones solve the problem. e.g. built for identifying people is not good for landscapes. - which are the best ones for our problem - what are the parameters associated with feature extractor optimization - techniques to optimize feature extractors (plots, optimization techniques etc) - how to evaluate feature extractors - what are the limitations of feature extractors

2.1.1. Feature Matchers

- What is a feature matcher - Why is it important - What are the types of feature matchers (How many are there) - Which ones solve the problem. e.g. good at matching people

is not good for landscapes. - What are the factors that affect the performance of a feature matcher - Which are the best ones for our problem - What are the parameters associated with feature matcher optimization - Techniques to optimize feature matchers (plots, optimization techniques etc) - How to evaluate feature matchers - What are the limitations of feature matchers

2.1.2. Homographic Estimation

- What is homographic estimation (rot translation warping) - Why is it important - What are the types of homographic estimation (How many are there) - what are the alternatives to homographic estimation (e.g. optical flow, affine transformation, and perspective transformation) - What is the most accurate way to estimate homography - What are the factors that affect the performance of homographic estimation - How should we optimize homographic estimation - How do we evaluate the performance of homographic estimation - What are the limitations of homographic estimation

Accuracy considerations

- Pixel to GPS coordinate conversion factor accuracy - Warping in the data (fisheye lens, distortion) - Image resolution and quality - Depth of objects causing relatively velocity differences - Parameters of extraction and matching - Preprocessing of images (e.g., blurring) - Outlier removal techniques

Stereo Vision

- What is stereo vision - Why is it important - What are the techniques of stereo vision (How many are there) - What are the factors that affect the performance of stereo vision - What is the basic data requirements for accurately estimating depth - How should we optimize stereo vision - How can we evaluate the performance of stereo vision - What are the limitations of stereo vision - What are the performance considerations of stereo vision

Feature Extractors

In this report, we conduct an in-depth evaluation of various feature extractors to determine the most suitable one for real-time, robust and accurate UAV-based navigation.

- what is a feature extractor - Usage in this task - what are the types of feature extractors (How many are there) - which ones solve the problem. e.g. built for identifying people is not good for landscapes. - what are the factors that affect the performance of a feature extractor - which are the best ones for our problem - what are the parameters associated with feature extractor optimization - techniques to optimize feature extractors

(plots, optimization techniques etc) - how to evaluate feature extractors - what are the limitations of 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 match images, estimate motion or depth, classify objects and perform other computer vision tasks.

Usage in this task

In this task, feature extractors are used to identify and extract key features in images captured by a UAV. We then use the extracted featured to match images, and ultimately estimate the new GPS coordinates and heading of the UAV. Accurate features are required to find the image that has the most correlation with the current image. Then, many features are required to accurately estimate the new GPS coordinates and heading of the UAV. In both cases, there is a requirement on the extractor to propose as many accurate keypoints as possible to the next stage.

What factors affect the performance of a feature extractor

The performance of a feature extractor is influenced by several factors, including: - Accuracy: The ability of the feature extractor to detect and describe features accurately is crucial for reliable image matching and estimation tasks. That is, it should be able to identify unique features that have very distinct and strong descriptors. - Robustness: The feature extractor should be invariant to changes in scale, rotation, illumination, and noise to ensure consistent performance across different environments and conditions. - Speed: Real-time processing requires fast computation and feature extraction to minimize latency, which is essential for timely decision-making in dynamic environments. - Repeatability: The feature extractor should be able to detect the same feature in different images, even if the feature is viewed from different angles or under different lighting conditions. -

Scalability: The algorithm should efficiently handle large datasets or high-resolution images, maintaining performance as the scale of data increases.

How many feature extractors exist and what are the main ones?

There exists a wide variety of feature extractors, each with its unique strengths and weaknesses. Some of the most commonly used feature extractors include: - SIFT (Scale-Invariant Feature Transform): SIFT is a classic feature extractor known for its robustness to scale, rotation, and illumination changes. It is widely used in various computer vision applications. - SURF (Speeded-Up Robust Features): SURF is a faster alternative to SIFT, offering similar robustness and accuracy while being computationally more efficient. - ORB (Oriented FAST and Rotated BRIEF): ORB is a fast and efficient feature extractor that combines the FAST keypoint detector with the BRIEF descriptor. It is suitable for real-time applications. - SuperPoint: SuperPoint is a recently developed, deep learning-based feature extractor that offers high accuracy and efficiency for real-time applications. It learns feature representations directly from data, making it adaptable to specific tasks. -

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

HOG, LBP (texture), Color Histogram, SIFT, SURF (comp efficient surf), ORB Gabor (texture and edges) Zernike (shape representation) Hu Moments (shape recognition) Haralick Texture Features

keras applications https://keras.io/api/applications/ https://keras.io/api/applications/HASHTAGusage-examples-for-image-classification-models

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

2.2. Choosing a Feature Extractor

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

- **Accuracy:** Accurate feature detection and description are crucial for navigation tasks to ensure precise localization and mapping. Inaccurate features can lead to errors in navigation and obstacle avoidance.
- **Speed:** Real-time processing necessitates fast computation and feature extraction to minimize latency, which is essential for timely decision-making in dynamic environments.

- **Robustness:** The feature extractor should exhibit invariance to changes in scale, rotation, illumination, and noise to ensure consistent performance across different flight conditions and environments.
- **Complexity:** Computational complexity directly impacts processing time and resource consumption, which is crucial for resource-constrained UAV systems where power efficiency is a consideration.
- **Scalability:** The algorithm should efficiently handle large datasets or high-resolution images, maintaining performance as the scale of data increases.

2.3. Parameters Associated with Feature Extractors

Feature extractors are characterized by several parameters that influence their performance:

2.3.1. Descriptor Size

The descriptor size, or the length of the feature vector, significantly impacts both the discriminative power and computational load of the feature extractor.

- **Implementation:** Common descriptors include SIFT (128 dimensions), SURF (64 or 128 dimensions), and ORB (32 or 256 dimensions). Larger descriptors like those used in SIFT capture more detailed information about each keypoint.
- **Pros:** Larger descriptors are more discriminative and better at distinguishing between different keypoints, improving matching accuracy across varied conditions.
- **Cons:** Increased descriptor size results in higher computational costs for storage and matching, which can be a bottleneck in real-time applications. For instance, matching thousands of 128-dimensional descriptors can be significantly slower than matching 32-dimensional descriptors.
- **Impact:** The choice of descriptor size must balance the need for detail and accuracy against the available computational resources and real-time constraints.

Mathematically, if d is the descriptor size and N is the number of keypoints, the storage requirement is $O(N \times d)$. The matching complexity, assuming a brute-force approach, is $O(N^2 \times d)$. Hence, optimizing d is crucial for real-time performance.

2.3.2. Keypoint Detection Threshold

The keypoint detection threshold determines the sensitivity of the feature detector to identifying keypoints in the image.

- **Implementation:** In algorithms like SIFT and SURF, the threshold is used to filter out weak keypoints. ORB uses the FAST detector with a threshold to select keypoints.
- **Pros:** A lower threshold increases the number of detected keypoints, which enhances robustness and ensures that important features are not missed, especially in complex or textured scenes.
- **Cons:** Increasing the number of keypoints also increases computational demands, both in terms of memory usage and processing time for descriptor computation and matching.
- **Impact:** Selecting an appropriate threshold is crucial. Too high a threshold might miss important features, reducing robustness, while too low a threshold might result in an unmanageable number of keypoints, slowing down processing.

Mathematically, if t is the threshold, then the number of keypoints N can be modeled as a function $N = f(t)$. Lowering t increases N, thus increasing the computational load $O(f(t) \times d)$.

2.3.3. Octave Layers

Octave layers refer to the levels in the scale-space pyramid used for multi-scale feature detection, enhancing robustness to scale variations.

- **Implementation:** In SIFT, the image is repeatedly blurred and subsampled to create a scale-space pyramid with multiple octaves, each containing several layers.
- **Pros:** More octave layers improve the feature extractor's ability to detect features at various scales, making it robust to changes in object size and distance.
- **Cons:** More layers increase computational complexity and processing time. Each additional octave layer involves further blurring, subsampling, and keypoint detection steps.
- **Impact:** The number of octave layers must be chosen to balance scale invariance against computational efficiency. Insufficient layers might miss features at certain scales, while too many layers could slow down processing.

Mathematically, if O is the number of octaves and L is the number of layers per octave, the computational complexity for constructing the pyramid is $O(O \times L \times N)$, where N is the number of pixels.

2.3.4. Grid Size

The grid size affects the granularity of feature extraction, impacting the detail and computational efficiency.

- **Implementation:** Grid size refers to the spatial subdivision of the image into regions for feature extraction. Smaller grid sizes provide finer granularity.
- **Pros:** Finer grids capture more detailed features and spatial relationships, improving the robustness and accuracy of the feature descriptors, particularly in textured regions.
- **Cons:** Smaller grid sizes require more computational resources for both extraction and matching, as more regions need to be processed and compared.
- **Impact:** The grid size must be carefully selected based on the complexity of the scene and the available computational resources. Finer grids are beneficial for detailed scenes but may not be practical for real-time processing on resource-limited platforms.

If G is the number of grid cells, then the computational complexity increases with G, as each cell needs to be processed independently, leading to a complexity of $O(G \times f(t) \times d)$.

2.3.5. Orientation Assignment

Orientation assignment ensures rotation invariance by assigning an orientation to each keypoint, which is crucial for consistent feature matching under different rotational perspectives.

- **Implementation:** SIFT assigns an orientation based on the gradient direction around the keypoint. ORB uses intensity centroid methods for orientation assignment.
- **Pros:** Assigning orientations to keypoints makes the descriptors invariant to image rotations, improving matching accuracy when the image or object is viewed from different angles.
- **Cons:** The orientation assignment step adds to the computational complexity and processing time. Incorrect orientation assignment can degrade matching performance.
- **Impact:** Ensuring accurate and efficient orientation assignment is vital for applications involving rotational movements. It enhances the robustness of the feature extractor but needs to be balanced against the additional computational overhead.

If θ is the orientation angle, the computation involves determining θ for each keypoint, adding a complexity of $O(N)$ where N is the number of keypoints.

2.4. Types of Feature Extractors

2.4.1. 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.

Mathematically, SIFT's computational complexity is $O(O \times L \times N \times d)$ for the scale-space construction and $O(N \times d)$ for descriptor computation, where O is the number of octaves, L is the number of layers, N is the number of keypoints, and d is the descriptor size.

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.

Mathematically, SURF's computational complexity is reduced to $O(O \times N \times \log N)$ for the integral image computation and $O(N \times d)$ for descriptor computation.

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.

Mathematically, ORB's complexity is $O(N \times \log N)$ for FAST keypoint detection and $O(N \times d)$ for BRIEF descriptor computation, where d is typically smaller than in SIFT or SURF.

2.4.2. Deep Learning-Based Feature Extractors

CNN-Based Extractors

Convolutional Neural Networks (CNNs) have revolutionized feature extraction by learning feature representations directly from data. Models such as VGG, ResNet, and Inception have demonstrated high accuracy and robustness in various image processing tasks.

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

Mathematically, CNNs involve multiple convolutional layers with a complexity of $O(N \times K^2 \times C)$ per layer, where N is the number of pixels, K is the kernel size, and C is the number of channels. The total complexity depends on the depth and architecture of the network.

Pre-trained Models

Utilizing pre-trained models on large datasets like ImageNet can significantly expedite the development process and improve accuracy. These models can be fine-tuned for specific tasks, leveraging their learned representations.

- **Advantages:** High accuracy due to extensive pre-training on large and diverse datasets. Reduced training time and resource requirements during fine-tuning for specific applications.

- **Disadvantages:** May not generalize well to specific tasks without adequate fine-tuning. Potential for overfitting if the fine-tuning dataset is not representative of the target application.

Mathematically, the fine-tuning process involves updating a subset of weights, reducing the complexity to $O(M \times K^2 \times C)$, where M is the number of modified weights, typically much smaller than the total number of weights.

SuperPoint

SuperPoint is a self-supervised framework for interest point detection and description, employing a fully convolutional neural network to detect points and describe them in a unified process. It is designed for real-time applications and offers a balance between accuracy and efficiency.

- **Advantages:** High accuracy due to its end-to-end learning of keypoints and descriptors. It is robust to various transformations and efficient for real-time applications, leveraging deep learning techniques for feature extraction.
- **Disadvantages:** Requires substantial computational resources for training. While inference is efficient, domain-specific fine-tuning may still be necessary to achieve optimal performance in specialized tasks.

Mathematically, SuperPoint's complexity is similar to CNNs, with the additional step of keypoint extraction and description being integrated into a unified process, leading to an overall complexity of $O(N \times K^2 \times C)$ for inference.

2.5. Evaluation for UAV-Based Navigation

2.5.1. Requirements

For the Skripsie project, the feature extractor must:

- Be capable of real-time processing to ensure timely navigation and decision-making in dynamic environments.
- Utilize the available high processing power of UAV systems efficiently, balancing accuracy and speed to maintain performance without excessive computational burden.
- Provide high accuracy to ensure reliable navigation, obstacle avoidance, and mapping, which are critical for UAV operations.

2.5.2. Recommended Feature Extractor

Given the requirements of real-time processing, high accuracy, and efficient utilization of processing power, a deep learning-based feature extractor is recommended. Particularly, a pre-trained CNN model such as ResNet or VGG, or a specialized model like SuperPoint, offers the best balance of these attributes. These models leverage the high processing power of modern UAV hardware, providing robust and accurate feature extraction essential for reliable navigation.

2.6. Conclusion

In conclusion, the choice of a feature extractor depends on the specific needs of the application. For UAV-based navigation in the Skripsie project, deep learning-based extractors, especially pre-trained models and SuperPoint, offer superior performance in terms of accuracy and real-time processing capabilities. These models provide a robust solution, leveraging advanced computational resources to meet the stringent demands of UAV navigation tasks.

Chapter 3

Results

this table shows the st. dev. related to the estimation vs ground truth coordinates ratio. A more stable analysis implies a lower variation in the factor relating estimation (pixel change) to that of the true GPS coordinate change. A gaussian blur was applied with varying, square kernel sizes and the st. dev. was analysed. code as of 4aug (sift etc)

Kernel Size	Standard Deviation X	Standard Deviation Y	Norm
1	0.15985	0.03265	0.16317
3	0.16372	0.03547	0.16753
5	0.16166	0.03307	0.16401
11	0.17329	0.02227	0.17472
13	0.17310	0.03345	0.17630
15	0.17972	0.04036	0.18417
17	0.18370	0.05149	0.19177
19	0.15727	0.03997	0.16225
21	0.18859	0.03368	0.19159
23	0.15842	0.03414	0.16207
25	0.17395	0.03374	0.17720
27	0.15788	0.02341	0.15960
29	0.15515	0.03875	0.16095
41	0.15982	0.03484	0.16359
61	0.42036	0.08671	0.42918
91	1.22744	0.03476	1.22793
121	0.23488	0.12851	0.26695
181	1.22640	2.65690	2.93890

Table 3.1: Standard Deviation and Norm of the Ratio of Actual to Estimated GPS Location for Various Kernel Sizes

A kernel size of 27 resulted in the lowest normalized standard deviation. It was subsequently made the primary use. A kernel size of 11 resulted in the lowest Y deviation, while a kernel size of one resulted in a fairly low deviation and ensures no important keypoints are removed. For this reason, those three kernel sizes will be tested with in

future.

Comparison of SIFT with BFMatcher and SuperPoint with LightGlue for UAV Navigation

In this analysis, we compare two feature extraction and matching techniques used for estimating GPS coordinates in UAV navigation. The first method utilizes the SIFT algorithm with BFMatcher, while the second employs SuperPoint with LightGlue. Both methods were applied to a dataset of aerial images, and the deviations between actual and estimated GPS coordinates were calculated. It's worth noting that the parameters for both methods were not optimized, which might affect their performance. The table below summarizes the results, showing the deviations in meters and the Euclidean norm for each method.

Image	**Deviation X (m)**	**Deviation Y (m)**	**Norm (m)**
SIFT + BFMatcher			
10	53.62	16.81	56.15
9	1287.48	5.02	1287.49
8	24.02	4.42	24.42
7	2.33	0.62	2.42
6	452.92	27.80	453.77
5	58.87	28.16	65.29
4	28.74	86.32	90.82
3	7.51	14.52	16.31
2	24.96	16.10	29.64
SuperPoint + LightGlue			
10	45.38	2.83	45.47
9	1281.47	41.44	1282.15
8	23.98	6.96	24.95
7	3.48	0.80	3.57
6	452.04	15.92	452.34
5	58.04	27.70	64.24
4	15.15	63.60	65.37
3	2.67	18.44	18.64
2	21.39	20.56	29.59

Table 3.2: Comparison of Deviation and Norm for SIFT + BFMatcher and SuperPoint + LightGlue

Analysis

In this comparison, we observe that the SuperPoint + LightGlue combination provides lower deviation norms in 6 out of 9 cases for the X deviations, suggesting better performance in those instances. The unoptimized parameters may have affected the overall performance, but the results indicate that SuperPoint + LightGlue could offer more accurate estimates under the same conditions.

Percentile Outliers removal

Lower Cutoff (%)	Normalized Error
0	59.57
1	58.61
2	57.71
3	57.01
4	55.33
6	55.80
10	54.96
12	54.06
14	54.00
16	54.11
18	53.98
19	53.45
20	53.82
21	54.07
22	54.44
24	54.06
26	54.00
28	54.44
30	53.66
32	54.39
34	54.14
36	54.01
38	53.49
40	53.20
41	53.46
42	53.68
43	53.82
44	53.30
45	52.87
46	53.25
47	52.48
48	52.73
49	53.74

Table 3.3: Normalized Error vs. Lower Percentile Cutoff

15 seconds fat run w.out global matcher.

All parameter choices

1. feature extractor 2. Feature matcher 3. color or grayscale 4. Blur kernel size 5. percentile cutoff 6. outlier removal method 7. Amt of features to detect 8. Extractor threshold 9. How many matches to detect 10. Match threshold 11. Image size and resolution 12. Scaling factor for GPS to pixels 13. Speed / runtime 14. Accuracy

[45.69197059475034] Time taken to execute the code: 94.5152 seconds - ORB

[47.40683503538506] Time taken to execute the code: 77.3410 seconds - AKAZE

[45.79188281755529] - Time taken to execute the code: 78.6795 seconds - Still akaze average 900 flights a day spoofed check source https://www.youtube.com/watch?v=bFM9HHB9JXI&ab_channel=DrBenMiles remove the backslashes in the above link

The next critical feature in an image-based navigation system is the feature matcher. The feature matcher serves two main purposes. The first is to find the most similar image to that being analyzed. The second is to estimate the relative pose (translation and rotation) between the two images. Thus, we break up this section into two components, global matching and local matching. xxx check scale if or not if To compute matches, The descriptor first transforms each feature individually to a normalized space. This process aligns the feature's orientation, adjusts scale, and normalizes other factors like intensity, ensuring that each descriptor is in a consistent format for direct comparison during matching. The actual matching procedure involves comparing the descriptor of a keypoint to that of keypoints in the other image until the most similar keypoint is found in terms of their descriptors. There are, however, basic thresholds of accuracy which need to be met and further techniques to filter out bad matches.

Matches have specific confidence levels. When calculating transformations we can include confidence weightings to improve the accuracy of the estimations.

Plan

These methods offer complex tradeoffs within different accuracy and speed metrics. As such, tests will be conducted on individual methods as well as hybrid approaches. All of the above methods will be included in these tests except the deep-learning approaches which are too computationally intensive for the task of finding the most correlated image. The best method is that which results in the combination of image pairs (matches) which subtends the lowest mean squared deviation in actual and estimated GPS locations and heading.

To summarize: the invariant methods are: Hashing, BOVW; variant methods are

Intermediary results:

These results indicate the time to complete the set of best correlated images. The score has not been used as the highly variant parameter set often, with enough computation,

can achieve the correct score. So instead of holding computational time constant, which is not clear, we increase the parameters until we see it achieve the correct score. Then the time is recorded and a higher accuracy is dually implied in a lower time. Where excessive time is required and the correct combination has not been met, there will be an indication of failure to meet score. Excessive time is defined relatively as 5 images above 5 seconds.

Search Space (Images)	**Runtime (Seconds)**
12	5.04
11	5.14
10	4.90
9	4.30
8	3.74
7	3.56
6	3.12
5	2.59
4	1.99
3	1.51
2	0.98
1	0.49

Table 3.4: KNN Matching: Search Space vs. Runtime

For BF_matching with KNN search, the run-time per image is roughly half a second, and the time complexity is linear. Flann_matcher does not get the correct result even with excessive time

Matcher	**Mean Normalized Error**	**Total Runtime (seconds)**
Graph Matcher	45.79188281755529	73.0386
88.3136 seconds **Flann Matcher**	47.32259915429281	124.3599
BF Matcher	45.90913	85.0412

Table 3.5: Mean Normalized Error and Total Runtime for GraphMatcher, FlannMatcher, and BFMatcher

This is for 13 images, given all other parameters constant for each test. As visible FLANN_matcher and BF_matcher perform the best, with FLANN slightly edging out BF. However, BF_matcher is significantly faster than FLANN_matcher. Graph_matcher is slower and more inaccurate than both other methods. These results are largely unexpected. The first point to note is that BF_matcher, an exhaustive search, takes less time than the approximate matchers.

However, without any best image finder, the code runs in 20s.

ORB detects less accurate and more noisy keypoints. AKAZE detects more accurate and robust keypoints. This means the homography model will be more accurate with AKAZE. As such, when using homography to determine the relative pose between images, one cannot employ a single outlier threshold for both. AKAZE requires a lower threshold due to its higher accuracy. However, ORB requires a higher threshold to remove the noise. eg 1.0 for AKAZE and 35.0 for ORB.

0 is bf, 1 is flann, 2 is graph

global+local matching with akaze (all with global graph 2 which is best) [45.909132551707636] FOR 0 which is BF 85.0412 seconds [45.86020977142852] FOR 1 which is flann 105.3263 seconds [47.24097141956378] FOR 2 (RUN-FIRST) WHICH IS graph 115.4037 seconds

global+local matching with ORB

[45.69197059475034] 81.0717 seconds Detector: ORB, Global Matcher: GraphMatcher, Local Matcher: BFMatcher

46.09923686293978 202.3633 seconds Detector: ORB, Global Matcher: GraphMatcher, Local Matcher: FlannMatcher

[46.04487996277007] Time taken to execute the code: 77.1281 seconds Detector: ORB, Global Matcher: GraphMatcher, Local Matcher: GraphMatcher

Global matchers via local matcher retrofit grid

Detector	**Local Matcher**	**Error Score**	**Time (seconds)**
AKAZE	BFMatcher	45.9091	85.0412
	FlannMatcher	45.8602	105.3263
	GraphMatcher	47.2409	115.4037
ORB	BFMatcher	45.6920	81.0717
	FlannMatcher	46.0992	202.3633
	GraphMatcher	46.0449	77.1281

Table 3.6: Error Scores and Execution Times for AKAZE and ORB Detectors with Different Local Matchers (Global Matcher: GraphMatcher)

global ssim etc

score 1 gets out of 13: 12 [51.1096468381182]

GLOBAL MATCHERS ACCuracy TEST:

All of these are done with BF

Global Matcher	**Error Score**	**Time (seconds)**	**Detector**
Histogram Comparison (1)	51.1003587945317	71.0535	AKAZE
SSIM (2)	49.8271	69.9308	AKAZE
Hash Comparison (3)	140.4049	85.8228	AKAZE

Table 3.7: Global Matchers with AKAZE Detector and FlannMatcher as Local Matcher

note that the relativity hereof is crucial. A single image mistmatch can be the reason for either small or dramatic changes.

Baseline time with global matcher off: 28.6975

ORB FAILS dramatically in the global matching methods.

[32.21786664268455] Time taken to execute the code: 335.3992 seconds Local Detector: Superpoint, Local Matcher: AKAZE, Global Detector: ORB, GLOBAL Matcher: BFMatcher

[124.24910955944776] Time taken to execute the code: 391.6816 seconds Local Detector: Superpoint, Local Matcher: AKAZE, Global Detector: ORB, GLOBAL Matcher: graphmatcher

Detector: AKAZE, Global Matcher: GraphMatcher, Local Matcher: BFMatcher Time taken to execute the code: 41.3234 seconds total match analysis time: 26.0431 seconds [51.1003587945317]

[40.32015300294837] Time taken to execute the code: 70.8431 seconds total match analysis time: 0.0000 seconds Detector: AKAZE, Global Matcher: BFMatcher, Local Matcher: BFMatcher

kernel 1 subtends to much noise for acc matching. image 5 mathches 2.

LIGHTGLUE: light glue, with Superpoint, even with highly optimized parameters runs extremely slow on a CPU. As such, on a CPU, it is not feasible to use SuperGlue over Light Glue for near real-time applications. Further, any global matcher used with LightGlue should be highly efficient to compensate for the slow performance of the formers. As such, local matching grid adaption will not be tested and instead, SSIM and histogram image comparison will be tested. Future implementations can involve a GPU for SuperGlue and LightGlue, as well as local matcher adaption for effective global matching.

[28.164948628902334]. params are very simple for this: 256 extractions, low matching thresholds. 0.0x for the two. and 0.5155 for threshold. Time taken to execute the code: 150.6657 seconds Detector: ORB, Global Matcher: SSIM, Local Matcher: LightGlue.

FULL NEURAL TEST Local_Alg_Detector_Choices = [ORB, AKAZE] Local_Alg_Matcher_Choices = [BFMatcher, FLANN, GraphMatcher] Global_Matcher_Choices = [Same as Local Matcher, Histogram, SSIM]

Local_Alg_Detector_Choices = [Superpoint] Local_Alg_Matcher_Choices = [Lightglue]

Total combinations = 2 * 3 * 3 * 1 * 1 = 18. Exhaustively, we could try each algorithmic combination in the prepoc and global matcher choice. Then there would be 5 options for Global matcher choice. We could also add neural approaches here but have decided against this for time cost. Further, we could add an extra step in local matching, which is currently implicit but there, to have different preproc and final local matching. That is, a different rotational and translational inference. Then we could use ORB, AKAZE, Superpoint for the local alg detector and we could use BF matcher, FLANN, graph matcher, LightGlue for the local alg matcher. This would take the 1x1 to a 3x4. This would take the total

combinations to 2 * 3 * 5 * 3 * 4 = 360. This is too much. Instead we try make a few educated guesses.

This applies to both models.

Firstly, as discussed before, we remove neural network options from search space reduction because of computational cost. Secondly, we make the assumption that each algorithmic-based local detector and matcher, considered jointly, is either better at both translation and rotational inference, or worse, but not better at one and worse at another than another algorithmic local matcher. This is not strictly true, but it is a good assumption that is crucial to limit the scope. This means that for all 4 possible stage choices: Rotational normalization in Global matching, Global Matching, Rotational inference in Local matching, and translational inference in local matching; If they are using an algorithmic combination of detectors and extractors, that combination is wholly better or worse for all the other stages if it is better or worse for one stage. Thirdly, we make the assumption that

SELFCODE: PREPROCDET; PREPROCMAT; GLOBALMAT (LOCAL is the same: super, light) 1 is ORB, 2 is AKAZE - rotational normalization detector for global matcher 0 is BF, 1 is FLANN, 2 is GRAPH - rotational normalization matcher for global matcher 0 is pass thru, 3 is histogram, 4 is SSIM - global matcher therefore the code ranges are [1-2], [0-2], [0-4]

[32.502594573455845] Preprocessing Local Algorithmic Detector: ORB, Preprocessing Local Algorithmic Matcher: BF, Global Matcher: Same as Preprocessing, Local Detector: Superpoint, Local Matcher: Lightglue Time taken to execute the code: 202.7905 seconds code: 1, 0, 0

[32.712598500355305] Preprocessing Local Algorithmic Detector: ORB, Preprocessing Local Algorithmic Matcher: FLANN, Global Matcher: Same as Preprocessing, Local Detector: Superpoint, Local Matcher: Lightglue Time taken to execute the code: 383.5403 seconds code: 1, 1, 0

[35.96082195165782] Preprocessing Local Algorithmic Detector: ORB, Preprocessing Local Algorithmic Matcher: GRAPH, Global Matcher: Same as Preprocessing, Local Detector: Superpoint, Local Matcher: Lightglue Time taken to execute the code: 215.8990 seconds code: 1, 2, 0

[37.623884922918975] Preprocessing Local Algorithmic Detector: AKAZE, Preprocessing Local Algorithmic Matcher: BF, Global Matcher: Same as Preprocessing, Local Detector: Superpoint, Local Matcher: Lightglue Time taken to execute the code: 170.9718 seconds code: 2, 0, 0

[37.48360017845881] Time taken to execute the code: 275.1002 seconds Preprocessing Local Algorithmic Detector: AKAZE, Preprocessing Local Algorithmic Matcher: FLANN, Global Matcher: Same as Preprocessing , Local Detector: Superpoint, Local Matcher: Lightglue code 2, 1, 0

[41.34034888283645] Preprocessing Local Algorithmic Detector: AKAZE, Preprocessing Local Algorithmic Matcher: GRAPH, Global Matcher: Same as Preprocessing, Local Detector: Superpoint, Local Matcher: Lightglue Time taken to execute the code: 246.5933 seconds code: 2, 2, 0

Preprocessing & Global Detector	**Preprocessing & Global Matcher**	**Error Score**	**Time (seconds)**
ORB	BF	32.5026	202.7905
ORB	FLANN	32.7126	383.5403
ORB	GRAPH	35.9608	215.8990
AKAZE	BF	37.6239	170.9718
AKAZE	FLANN	37.4836	275.1002
AKAZE	GRAPH	41.3403	246.5933

Local Detector: Superpoint
Local Matcher: Lightglue
Global Matcher: Same as Preprocessing, no isoglob matchers

assume from prior table, and initial tests that the global matchers perform worse. good conclusion.

In this software, we have two distinct stages: global (preprocessing - rotational normalization, global matching) and local (rotational and subsequent translational inference) matching. Each stage operates independently. This modular structure allows us to evaluate the performance of each stage based on endpoint metrics, such as time and accuracy, and select the best method for each, without needing to test every possible combination of methods across stages.

Within each of the two stages there are sub-stages which are interdependent. This is because the keypoints, and nuanced rotational patterns follow on from initial stages into latter ones. Global matchers lose a large portion of coupling due to recomputation of descriptors. However, between global and local matching, there may exist nuanced coupling in that global matchers may tend to have different levels of preferences to different patterns or feature distributions to that of the local matcher. However, in this scope we state that each potential match is either similar enough for mismatches to not have a significant effect on the estimation, or the potential matches are far enough apart that the minor preference difference in the local and global matchers becomes negligible; As such, the single combination of global matchers that performs the best in one local matcher will perform the best, or very close to that, in any other local matcher. If we wanted an extremely precise estimate, we would need to test all combinations. However, to maintain scope and still achieve a highly accurate estimate, we can make this assumption.

ALG GLOBAL ISOTEST Algorithmic global testing

global_detector_choice Set 1 for ORB, 2 for AKAZE. global_matcher_choice = 1 Set 0 for BFMatcher, 1 for FlannMatcher, 2 for GraphMatcher global_matcher_technique = 0

code range is [1-2], [0-2], [0,3-4]

[36.37745610974454] should give same results as above.

[36.37745610974454] Preprocessing Global Detector: ORB, Preprocessing Global Matcher: BF, Global Matching Technique: Same as Global Matcher, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 71.1368 seconds code is 1, 0, 0

[FAIL] X2 Preprocessing Global Detector: ORB, Preprocessing Global Matcher: BF, Flann, Graph, Global Matching Techniques: Histograms and SSIM, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 71.1368 seconds Fails under conditions [1], [0,1,2], [3,4]

[36.497046238519495] Preprocessing Global Detector: ORB, Preprocessing Global Matcher: FLANN, Global Matching Technique: Same as Global Matcher, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 261.3577 seconds code is 1, 1, 0

[33.09399194594492] Preprocessing Global Detector: ORB, Preprocessing Global Matcher: GRAPH, Global Matching Technique: Same as Global Matcher, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 126.0975 seconds code is 1, 2, 0

[40.237965285920886] Preprocessing Global Detector: AKAZE, Preprocessing Global Matcher: BF, Global Matching Technique: Same as Global Matcher, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 68.8457 seconds code is 2, 0, 0

[45.85650551390119] Preprocessing Global Detector: AKAZE, Preprocessing Global Matcher: BF, Global Matching Technique: Histogram, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 87.7977 seconds code is 2, 0, 3

[45.85650551390119] Preprocessing Global Detector: AKAZE, Preprocessing Global Matcher: BF, Global Matching Technique: SSIM, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 52.2356 seconds code is 2, 0, 4

[40.10291200291962] Preprocessing Global Detector: AKAZE, Preprocessing Global Matcher: FLANN, Global Matching Technique: Same as Global Matcher, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 126.9023 seconds code is 2, 1, 0

[45.85650551390119] Preprocessing Global Detector: AKAZE, Preprocessing Global Matcher: FLANN, Global Matching Technique: Histogram, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 111.3775 seconds code is 2, 1, 3

[45.85650551390119] Preprocessing Global Detector: AKAZE, Preprocessing Global

Matcher: FLANN, Global Matching Technique: SSIM, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 150.3223 seconds code is 2, 1, 4

[45.13035674520387] Preprocessing Global Detector: AKAZE, Preprocessing Global Matcher: GRAPH, Global Matching Technique: Same as Global Matcher, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 98.8956 seconds

[45.85650551390119] Preprocessing Global Detector: AKAZE, Preprocessing Global Matcher: GRAPH, Global Matching Technique: Histogram, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 79.3778 seconds

[45.85650551390119] Preprocessing Global Detector: AKAZE, Preprocessing Global Matcher: GRAPH, Global Matching Technique: SSIM, Local Detector: AKAZE, Local Matcher: BF Time taken to execute the code: 74.2499 seconds

Global Detector	**Global Matcher**	**Global Matching Technique**	**Time [seconds]**	**Error Score**
ORB	GRAPH	Same as Global Matcher	126.0975	33.0940
ORB	BF	Same as Global Matcher	71.1368	36.3775
ORB	FLANN	Same as Global Matcher	261.3577	36.4970
AKAZE	BF	Same as Global Matcher	68.8457	40.2379
AKAZE	FLANN	Same as Global Matcher	126.9023	40.1029
AKAZE	BF	Histogram	87.7977	45.8565
AKAZE	BF	SSIM	52.2356	45.8565
AKAZE	FLANN	Histogram	111.3775	45.8565
AKAZE	FLANN	SSIM	150.3223	45.8565
AKAZE	GRAPH	Histogram	79.3778	45.8565
AKAZE	GRAPH	SSIM	74.2499	45.8565
AKAZE	GRAPH	Same as Global Matcher	98.8956	45.1304
ORB	BF, Flann, Graph	Histogram and SSIM (Fails)	FAIL	FAIL

Table 3.8: Error Scores, Execution Times, and Global Matching Techniques for Various Configurations

Different global matchers:

Fourier Mellin Transform with phase correlation - fail phase correlation with log polar transform - fail

Concept

The flow of the camera system shall be as follows:

1. The controller shall receive footage in real-time from the UAV and sync it with known telemetry data such as GPS coordinates, altitude, heading and speed.
2. The controller shall then process the video footage and telemetry data to extract and store the relevant features
3. When the GPS signal is lost, the UAV will turn around
4. The controller shall extract and match the features of the current image with that of the previous images
5. It will find the most correlated images and use that image to determine the change in feature locations to infer the UAV's position

Feature Extractors

these are the extractors orb, surf, sift, akaze, brisk deep learning extractors: superpoint, d2-net, r2d2

This study will introduce the most popular and state-of-the-art feature detectors. There will be rough research as to why these extractors were chosen. Where possible, we will compare methods and remove superceded methods. Thereafter, we will consider the methods which are most likely to work for the given application. The study will test these methods and compare them to determine the best method for the given application. The study will consider primarily the accuracy of the method, in terms of its error. It will also consider the computational cost in cases where it does not meet the semi-real-time requirements or where accuracy is similar.

SIFT (Scale-Invariant Feature Transform)

SIFT detects keypoints using the Difference of Gaussians (DoG), where Gaussian-blurred images at multiple scales are subtracted to highlight significant intensity changes. Keypoints are identified as maxima or minima in the DoG and are stored with their scale information. SIFT then assigns an orientation by analyzing gradients around each keypoint, ensuring rotation invariance. The final 128-dimensional descriptor encodes gradient information relative to the keypoint's orientation and scale, making it robust to changes in scale, rotation, and illumination. NOT USED NOT FREE

SURF (Speeded-Up Robust Features)

SURF detects keypoints using the Hessian matrix, which measures image curvature to find corners and edges. To speed up this process, SURF uses box filters and integral images, allowing quick computation. Keypoints are detected where the Hessian determinant is highest. SURF assigns an orientation using Haar wavelets, which measure intensity changes in the horizontal and vertical directions around the keypoint. The keypoint descriptor is built by summing wavelet responses within a square region, creating a compact 64-dimensional (or 128-dimensional) descriptor for feature matching. Good balance of speed and robustness, suitable for real-time mapping.

ORB (Oriented FAST and Rotated BRIEF)

ORB detects keypoints using the FAST algorithm, which identifies corners by analyzing a circle of pixels around each point. The keypoints are ranked using the Harris corner measure, and only the strongest are kept. ORB assigns an orientation to each keypoint by calculating the direction of the intensity gradient, ensuring rotation invariance. The descriptor is a modified version of BRIEF, which compares pixel intensities within the keypoint's neighborhood. The pattern is rotated according to the keypoint's orientation, resulting in a 256-bit binary descriptor that is fast and efficient for real-time applications. Fast and efficient, suitable if you need quick results with moderate accuracy.

AKAZE (Accelerated-KAZE)

AKAZE detects keypoints in a nonlinear scale space, created using nonlinear diffusion filtering, which preserves edges better than traditional Gaussian blurring. Keypoints are found by locating extrema in this scale space. Each keypoint is assigned an orientation based on the local gradient direction, ensuring rotation invariance. The descriptor, M-LDB, is a binary string formed by comparing intensity differences between pairs of points around the keypoint. AKAZE is robust to varying conditions and computationally efficient. Best balance of accuracy and robustness for complex landscapes.

BRISK (Binary Robust Invariant Scalable Keypoints)

BRISK detects keypoints using a scale-space pyramid, applying the FAST detector at multiple scales. Keypoints are refined through non-maximal suppression to retain only the strongest. Orientation is assigned by analyzing the intensity gradients around the keypoint. The descriptor is a binary string generated by comparing pixel intensities in a circular pattern, resulting in a 512-bit descriptor. BRISK is designed for speed and is effective in scenarios with scale and rotation variations. Fastest, but less robust, better for simpler or less variable environments.

Others

Feature detectors need to be paired with descriptor extractors to be useful. The above do both. There are also individual feature detectors and descriptor extractors which would be paired with either the above methods or ones below. Below, they are analysed. Pure feature detectors (keypoints no descriptors) STAR is used for blob-like detection although not very robust and usable for mapping. FAST, purely an extractor, not a descriptor. not very robust, but very fast. MSER, niche applications like text detection. Not very robust or fast. GFTT Excellent for motion tracking but lacks robustness in scale and rotation invariance needed for UAV mapping. Harris-corner detector: Effective in stable environments with well-defined corners, but limited by its lack of scale and rotation invariance Dense: More suited to dense sampling in texture or object recognition tasks, not specifically useful for detecting DISTINCTIVE features for mapping. SimpleBlobDetector: Best in environments with simple, blob-like features but not ideal for general UAV mapping tasks

Feature descriptors (no keypoints): BRIEF generates binary descriptors by comparing pixel intensities in a fixed pattern around each keypoint. It is fast but not scale or rotation invariant

FREAK: similar to brief in efficiency and a lack of scale invariance, however, it is rotation invariant. Once again, not suitable for UAV mapping.

Given the focus on efficiency of both of these methods, it is not necessary to pair them with any of the above methods as the accuracy, our main consideration is likely to be lower than the above methods.

Comparison of Feature Detectors

Feature Detector	**ORB**	**SURF**	**SIFT**
Time to Compute 300 Keypoints (ms)	15	115	120
Average Total Number of Keypoints per Image	7300	2300	1800
Matching Percentage Given Illumination Changes	96%	89%	88%
Matching Percentage Given Rotation Changes	100%	100%	92%

The given study indicates that ORB is the fastest, it ties for first with SURF in terms of rotation invariance, and it is the best at handling illumination changes. For our application, rotation and illumination invariance are critical. The speed of the detector is also important as the UAV will be processing images in real-time. ORB is the best choice for this application. Second to ORB is SURF, followed by SIFT, however more study will be conducted to determine the better between the two

Comparison of sift and surf

in practice sift better surf at scale inv, surf better at rot inv, warping, blur. both good for illumination

Table 3.9: Comparison of SIFT and SURF for different image transformations

Image Transformation	SIFT Matching Pairs	SURF Matching Pairs
Rotate 90°	215	309
Different Scale	153	32
Blur	29	217
Different Hue	153	169
Different Saturation/Value	49	220
Warp	266	565
RGB Noise	23	57

The above is a comparison of the invariance of SIFT and SURF. It tests multiple distortion types as well as execution time, and the number of keypoints detected. SIFT performs better in terms of scale invariance, while SURF performs better in terms of rotation invariance, warping, blur and marginally in illumination changes. This study was limited in terms of the number of images used and scope of testing. However, it provides a good indication of which feature detector is likely to work better for the given application. UAVs are likely to experience rotation changes and illumination changes on the reverse flight. The scope of this study is limited to relatively stable altitudes, and hence scale invariance is not as important. SURF is likely than SIFT to offer superior performance in this application. Due to scope constraints, the study will not test SIFT in practice.

Deep-Learning Based Feature Detectors

Deep learning-based feature detectors have gained popularity in recent years due to their ability to learn extremely discriminative features directly from data. This study makes use of pre-trained detectors to maintain scope. Some popular deep learning-based feature detectors include SuperPoint, R2D2, and D2-Net. These detectors offer high performance and robustness to a large number of transformations, making them suitable for precise and challenging applications. There are two limitations for these models. Firstly, they require a significant amount of matrix multiplication, which is computationally expensive and requires a GPU. Secondly, the pre-trained data is unlikely to have included images of landscapes akin to that of the UAV flight path. This means that the model may not be able to generalize well to the given application. In future work, it may be beneficial to train a model on the specific dataset to improve performance.

xxx - choose slam or homography later

Superpoint (Self-Supervised Interest Point Detection and Description - 2018)

SuperPoint is an end-to-end neural network designed for keypoint detection and description, trained on synthetic data to perform these tasks simultaneously. The network consists of two stages: a base CNN that processes the image to produce a dense feature map and a head that detects keypoints from this map. The keypoints are then assigned descriptors by sampling the surrounding feature map. SuperPoint is unique in that it learns both the keypoint locations and the descriptors in a fully supervised manner, optimizing them for tasks like visual SLAM and image matching. The keypoints detected by SuperPoint are highly repeatable, and the descriptors are robust, making the method effective for various computer vision applications. Despite being more computationally efficient than other deep learning methods, SuperPoint still requires considerable resources compared to traditional approaches, making it more suitable for scenarios where the computational cost is justified by the need for high accuracy and robustness.

D2-Net (2019) D2-Net is a deep learning-based approach that integrates keypoint detection and description into a single process using a Convolutional Neural Network (CNN). The network processes the input image to generate feature maps, which are used to identify keypoints by finding local maxima in these maps. This step ensures the keypoints are robust and scale-invariant, as they are derived from multiple scales within the network. For each detected keypoint, a high-dimensional descriptor is generated by sampling a patch of the feature map around the keypoint, capturing the local structure of the image. D2-Net is particularly strong in handling significant geometric transformations like rotation, scale changes, and perspective distortions, making it highly reliable for tasks requiring robust feature matching. However, due to the complexity of the network and the feature maps, D2-Net is computationally demanding, requiring significant processing power and memory.

R2D2 (2019) R2D2 (Repeatable and Reliable Detector and Descriptor) focuses on detecting keypoints that are both repeatable and reliable across different images and conditions. It uses a neural network to produce a dense map of keypoints, where each keypoint is scored based on its reliability—how distinct it is compared to others—and its repeatability—how consistently it can be detected across varying views. The network optimizes for these properties, ensuring that the detected keypoints are both stable and distinctive. The descriptors in R2D2 are generated by sampling the feature map around each keypoint, similar to D2-Net, but with a focus on maintaining robustness to noise and ensuring distinctiveness. This makes R2D2 highly effective in challenging environments with significant changes in viewpoint, lighting, or partial occlusion. The approach, while accurate, requires more computational resources compared to traditional methods due to the deep learning-based processing.

These are the deep-learning extractors. DELF and LF-NET are also popular, however,

slightly older and have been surpassed in most aspects by the following: (xxx - cite this)

D2-Net: Best for handling large transformations and providing dense, accurate keypoints and descriptors. R2D2: Excellent for repeatable and reliable keypoints, especially in challenging conditions. SuperPoint: Offers a good balance of accuracy and efficiency, with strong performance in real-time applications.

Chapter 4

Summary and Conclusion

Appendix A

Project Planning Schedule

This is an appendix.

Appendix B

Outcomes Compliance

This is another appendix.