VILNIUS UNIVERSITY

Rokas
JUREVIČIUS

Investigation of Particle Filter Based Visual Localization for Unmanned Aerial Vehicle Flights at Low-Altitude

DOCTORAL DISSERTATION

Technological sciences,
Informatics engineering T 007

VILNIUS 2019

The dissertation work was carried out at Vilnius University from 2014 to 2018.

Academic supervisor:
Dr. Virginijus Marcinkevičius (Vilnius University, Technological Sciences, Informatics Engineering – T 007).

VILNIAUS UNIVERSITETAS

Rokas
JUREVIČIUS

Bepiločio orlaivio vizualinės lokalizacijos grįstos dalelių filtru tyrimas skrydžiams mažame aukštyje

DAKTARO DISERTACIJA

Technologijos mokslai,
Informatikos inžinerija T 007

VILNIUS 2019

Disertacija rengta 2014 - 2018 metais Vilniaus universitete.

Mokslinis vadovas:
dr. Virginijus Marcinkevičius (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007).

Acknowledgements

I would like to express my very great appreciation to my scientific supervisor Dr. Virginijus Marcinkevičius for the guidance throughout the research and writing of this thesis. His encouragement has kept my progress on schedule and the academic knowledge shared with me has become the foundation for my future career.

I would also like to extend my thanks to Mindaugas Eglinskas for sharing ideas and discussions which lead to this research.

I wish to acknowledge the technical assistance provided by Justinas Šeibokas.

I would like to thank Lithuanian Space Science and Technology Institute for joint effort in collecting aerial data that was used in the research.

Finally, I wish to thank my family for their support and encouragement throughout my study.

Table of Contents

Acknowledgements **v**

Notation **xiii**

Abbreviations **xiv**

INTRODUCTION **1**

1. Research Context 1
2. Statement Of The Problem 2
3. Research Object 4
4. Research Aim And Objectives 4
5. Research Methods 5
6. Scientific Contributions Of The Research 6
7. Practical Value Of The Research 6
8. Defensive Claims 8
9. Approbation Of The Results 8
10. Outline Of The Thesis 10

1 VISUAL LOCALIZATION ON-BOARD UAV **11**

1.1 Unmanned Air Vehicle Platforms 11
1.2 Mobile Robot Localization 13
1.2.1 Bayesian Filter 15
1.2.2 Kalman Filter 17
1.2.3 Particle Filter 18
1.3 Monocular Visual Localization 19
1.3.1 Optical Flow 20
1.3.2 Visual Odometry 21
1.3.3 Bayesian Nonlinear Filtering 25
1.3.4 Visual SLAM 25
1.3.5 Other GPS-Denied Localization Methods 29

1.4 Parallel Computing Platforms for Size and Energy Constraint Environments 30
1.5 Datasets for Benchmark of Visual Localization Algorithm 34
1.5.1 Most Notable Publicly Available Datasets 35
1.5.2 Datasets For Benchmark Of Map-Based Localization Algorithms 36
1.6 Conclusions of Chapter 2 37

2 RESEARCH METHODOLOGY 39
2.1 Energy Efficiency Of Computing Platforms 39
2.1.1 Parallel implementation of the Sobel filter algorithm 39
2.1.2 Measuring The Power Consumption 41
2.2 Datasets of Aerial Imagery for Benchmark of Visual Localization Algorithms 43
2.2.1 Data From Test Flight 44
2.2.2 Simulated Dataset 46
2.3 Particle Filter Localization For Low-Altitude UAV Flights 51
2.3.1 Particles 56
2.3.2 Particle sampling 57
2.3.3 Motion model 58
2.3.4 Particle propagation 60
2.3.5 True Pose Estimation 60
2.3.6 Similarity Between Two Images 61
2.3.7 Image Similarity To Particle Probability Conversion Functions 62
2.3.8 DPC-PFL: Discriminatory Pearson Correlation based Particle Filter Localization Algorithm 64
2.4 Conclusions Of Chapter 3 68

3 EXPERIMENTS AND RESULTS 69
3.1 Comparison Of Computing Platforms 70
3.1.1 Execution Speed on Different Platforms 70
3.1.2 Energy Efficiency of Computing Platforms 72
3.2 Comparison of Image Similarity Metrics 75

3.3 Evaluation of UAV Heading Error Impact on the NCC Similarity Metric 80
3.4 Comparison Of Particle Sampling Techniques 81
3.5 Image Similarity To Probability Conversion Functions . . 84
3.5.1 Accuracy 85
3.5.2 Execution Time 88
3.5.3 Robustness 88
3.6 Comparison of DPC-PFL Against Visual Odometry . . . 90
3.7 Comparison Against Visual SLAM 91
3.8 Conclusions Of Chapter 4 92

GENERAL CONCLUSIONS **97**

Bibliography **98**

List of Figures

1.1 Different autonomous UAV platforms. 13
1.2 General scheme of a mobile robot localization process. . . 14
1.3 An example of a Dense optical flow of sample images. Green lines represent the direction and magnitude of the approximated camera movement. 20
1.5 Visual odometry calculates motion by aligning 3D world points p_j in image sequence and calculate transformation $T_{k,k-1}$ that causes the features to be reprojected from image patches u_n to u'_n. 23
1.4 SVO algorithm processing pipeline. 24
1.6 An overview of feature based and direct vSLAM methods. The key difference is that direct methods does not abstract image data using feature extraction and uses photometric error minimization for movement tracking. . . . 28
1.7 Platforms used in the experimental comparison. 31
1.8 SLAM Sensor Unit with integrated Xilinx Zynq processor with FPGA. 32

2.1 Input and output images of the *Sobel* filter. 40
2.2 Setup of power measurement sensor. 43
2.3 Spartan UAV used in test flights. 44
2.4 Sample images from the dataset captured during the test flight. 45
2.5 Sample patches of the same location in urban map created at different dates. Each figure is annotated with a creation date. 48
2.6 Flight plans for different scenarios. 49
2.7 Different flight trajectories recovered using the SVO algorithm and compared against ground truth (GT). 53
2.8 Sample image sequences from the dataset. 54

2.9 Particle filtering algorithm including VO 56
2.10 A preview of the particle sampling according to it's weight. This example contains 5 particles with given weight values. If a generated random uniform number is within the patch assigned to a specific particle, that particle will be sampled, in case of generated value = 0.68, particle $P_t^{(1)}$ would be drawn. 58
2.11 Simplified planar motion model for UAV 59
2.12 Conversion functions outputs. 66

3.1 Time taken to process a single frame using OpenCL and eSDK on the *Parallella* platform. 71
3.2 Execution time of *Sobel* filter algorithm on images of different resolutions. Note that CPU data does not contain read and write times, since these operations are required on accelerator devices. 72
3.3 Consumed energy histograms of different platforms. . . . 73
3.4 Image similarity metric values on simulated 500 UAV pose hypotheses. Image a) shows the similarity coefficient values calculated using NCC metric (eq. 2.15), most values are around 0 with few clear peak values of over 0.1. The image b) contains CC metric (eq. 2.14) value, which does not contain any clear peak values. The image c) contains SSD similarity (eq. 2.13) values, the values are mostly 0, but there is also a lot of peak values. 76
3.5 Correlation coefficient values when matching map with aerial imagery on all heading angles. 80
3.6 Hypothesized UAV locations on the known map. 82
3.7 Comparison of localization accuracy with different maps, map years depicts the date when map was created. 90
3.8 Flight trajectory reconstruction using Visual odometry (red), proposed DPC-PFL (green) and conventional GPS sensor (blue). 92

3.9 Absolute positioning error using Visual odometry (red) and proposed DPC-PFL (green). 93
3.10 Linear flight trajectories recovered using SVO, ORB-SLAM2, and proposed DPC-PFL. 94

List of Tables

2.1 Different trajectories, maps and altitudes used to perform 12 flight scenarios in the dataset 50
2.2 Number of images and distances of each flight. 50
2.3 Metadata fields available for each image in the dataset. . . 52

3.1 Measured average energy consumption per computed frame. 75
3.2 Shapiro-Wilk test results 75
3.3 Student's t-test results across measurements. 75
3.4 Similarity metric comparison. 78
3.5 Heading change impact on similarity coefficient 81
3.6 Comparison of sampling algorithms 84
3.7 Localization accuracy in meters using parametric logistic conversion function 86
3.8 Localization accuracy in meters using parametric rectifying conversion function 86
3.9 Conversion function ranking according to accuracy 87
3.10 Localization comparison against Softmax conversion function 87
3.11 Localization comparison against Softmax conversion function 89
3.12 The deviations of localization accuracy achieved using different conversion functions over different maps. 91
3.13 Localization accuracy comparison of ORB-SLAM (ORB), SVO Visual odometry, and the proposed algorithm – DPC-PFL. 93

Notation

S_t - Particle set on time t
$P^{(i)}$ - ith particle of the particle set
T - image from camera, represented by a sequence of pixel color intensity values
I - image patch from a map, represented by a sequence of pixel color intensity values
$s_t^{(i)}$ - probability value of ith particle on time t
R_t - image similarity value on time t
$b_t^{(i)}$ - ith particle weight on time t
$F(x)$ - image similarity to probability conversion function
$\hat{\delta}_{tran}$ - movement distance measured by odometry algorithm, with additional noise
$\hat{\delta}_{rot}$ - rotation angle measured by magnetometer sensor, with additional noise
$\langle x, y, \theta_{yaw} \rangle$ - a set describing UAV pose in 2D map space, x, y coordinates and heading angle θ
$\langle x', y', \theta'_{yaw} \rangle$ - posterior UAV pose
$bel(z_t)$ - belief over position hypothesis z_t
z_t - UAV (or robot's) position hypothesis on time t
m_t - sensor measurement data on time t

Abbreviations

BA - Bundle Adjustment
BRIEF - Binary Robust Independent Elementary Features
CC - cross-correlation
CENSURE - Center-Surround detector
CNN - Convolutional Neural Network
CPU - Central Processing Unit
CSV - Comma-Separated Values
DPC-PFL: Discriminatory Pearson Correlation based Particle Filter Localization
DSMAC - Digitized Scene-Mapping Area Correlator
DoF - Degree(s) of Freedom
EKF - Extended Kalman Filter
FAST - Features from Accelerated Segment Test
FPGA - Field-Programmable Gate Array
FPS - Frames Per Second
GPS - Global Positioning System
GPU - Graphics Processing Unit
GT - Ground Truth
I2C - Inter-Integrated Circuit
KF - Kalman Filter
KLD - Kullback–Leibler Divergence
KLV - Key-Length-Value
MCU - Microcontroller Unit
MISB - Motion Imagery Standards Board
MPEG - Moving Picture Experts Group
MPEG2-TS - MPEG2 Transport Stream
NAIP - National Agriculture Imagery Program
NCC - normalized cross-correlation
ORB - Oriented FAST and rotated BRIEF
RANSAC - Random Sample Consensus
RSD - Relative Standard Deviation

SD - Standard Deviation
SIFT - Scale-Invariant Feature Transform
SITL - Software In The Loop
SLAM - Synchronous Localization And Mapping
SSD - Sum of Squared Differences
SURF - Speeded up robust features
SVO - Semi-dense Visual Odometry
SfM - Structure from Motion
SoC - System on Chip
TERCOM - Terrain Contour Matching
UAV - Unmanned Air Vehicle
USGS - United States Geological Survey
VO - Visual Odometry
vSLAM - Visual SLAM

INTRODUCTION

1. Research Context

During the recent years, the development of powerful embedded computing platforms enables the use of computer vision and artificial intelligence onboard robotic platforms, such as Unmanned Air Vehicles (abbr. UAV) (e.g., drones), ground vehicles, submarines, and others. These capabilities take a step forward towards intelligent, autonomous, secure, and most importantly safe robots. In the case of UAVs, flight safety and security are vital issues, since UAVs are highly dependent on GPS signal which can be jammed or spoofed [1]. The UAVs play an essential role in the military, law enforcement, and rescue services by providing critical intelligence from above. GPS signal is vulnerable to jamming and spoofing. Encoded military standard GPS signals can be used to avoid spoofing, although they are still vulnerable to jamming, this type of GPS signals are not available for use in commercial aircraft. If the GPS signal is lost or the UAV is navigating in a GPS-denied environment, conventional autopilot systems fail to navigate safely when there is no available alternative localization method. UAV should estimate it is own position without the need of GPS signal or other radio signal based localization techniques. Therefore, visual localization, the process of pose estimation relatively to a known environment using optical sensors, may solve the problem of navigation in GPS-Denied scenarios.

Visual odometry calculated on imagery from a downward looking camera on a UAV can solve the GPS-Denied localization [2, 3]. Such a solution may deal with this problem to certain limitations: the visible area of the camera must contain enough visual features for tracking throughout the flight. Visual odometry has a fundamental issue of positioning drift and errors add up to infinity, within infinite flight time. A similar problem of mobile robot localization was solved using Monte Carlo localization [4]. Particle filter mixed with wheel odometry approach was used in [5], where a mobile robot used ceiling mosaic and an upward facing camera to localize its position using Particle filter

localization. [6] demonstrates the usage of Particle filter with a panoramic vision for robot localization. Stereo vision systems have been successfully applied to low/medium size UAVs due to its low weight and versatility. The problem of two cameras is the rigid distance between them, which limits the useful altitude range [7]. Computer vision techniques were demonstrated to be able to solve "kidnapped robot problem" (or global localization problem) using Visual odometry and Extended Kalman-filter based Synchronous Localization and Mapping (abbr. SLAM) in [7]. This solution relies on natural landmark seen in the which are used to calculate homography, recovering the flight. Particle filters have solved localization problem for autonomous robots [8] using laser scanners and panoramic vision [6]. A Particle filter based map matching solution was proposed in [9], it has shown potential results, but the evaluation of the algorithm is limited to a tiny dataset. Military navigation systems—TERMAC and DSMAC—successfully used Particle filtering techniques for missile guidance [10]. This thesis uses basic principles of Particle filter and previous works on UAV navigation to propose a new algorithm that improves localization accuracy, execution speed, and robustness to inaccuracies in maps of the environment.

The research questions are the following:

1. What are the means of UAV localization when GPS signal is unavailable?
2. What is the accuracy that can be achieved using visual localization and how can it be improved?
3. Which computing platform should be used to perform visual localization onboard UAV?
4. Which computer vision algorithms are most suitable for UAV localization during long distance, low altitude flights in GPS-Denied environment?

2. Statement Of The Problem

Autonomous flight of UAV is highly dependent on precise pose estimation of the aircraft body. Autonomous aircraft rely on a precise GPS

signal to perform autonomous flight missions. Consequently, flights are impossible to perform or fail if the GPS signal is lost or jammed. It is an especially worrying problem during military UAV missions since the GPS signal is entirely unavailable in such scenarios. UAVs can also be used for search and rescue missions in extreme situations, where GPS signal might not be available. External radio signal based navigation systems were developed as a feasible solution for this challenge, but all radio signals are vulnerable to jamming.

The field of computer vision is tackling the problem by proposing visual localization algorithms, that process video stream from onboard cameras and estimates aircraft pose in space. Visual odometry, SLAM, optical flow, and image registration algorithms were developed, which can provide backup localization for UAVs. Although, these techniques show stunning results in indoors or outdoor environments at near-ground (< 100 meters) altitude, not many researchers apply these algorithms in low altitude flights (between 100-3000 meters). Most researchers focus on micro UAVs, that are only capable of flying at near-ground altitudes or indoor environments since they are cheaper and more widely available. Problems faced at near-ground altitude is quite different than during low altitude, long distance flights that are far more expensive to perform.

Visual SLAM algorithms have significantly advanced to provide drift-free positioning systems, but they have focused for ground vehicles or UAVs flying in indoors or a near-ground altitude (usually 5-15 meters) outdoors flights. The military field developed TERCOM navigation systems for missile GPS-Denied navigation a few decades ago. TERCOM systems are still in use today, but they are dependent on accurate height maps of the environment to provide precise localization. An improved version of TERCOM is the DSMAC system, which used ortho-photo maps and optical camera sensor for navigation. Both systems require very recent elevation or ortho-photo map of the environment, which is usually captured using satellites moments before the flight.

The development of a backup navigation system for GPS is being performed by researches around the globe, to solve the GPS-Denied nav-

igation problem and contribute to autonomous flight safety. This thesis analyses map relative localization techniques using an optical sensor to provide precise and drift-free positioning for low altitude UAV flights in GPS-Denied environments. Improvements to current techniques are proposed to speed-up execution, reduce localization drift, improve precision and improve robustness to inaccuracies in maps so that localization could be feasible in out-of-date map.

For the UAV to be completely autonomous and independent of external sensors to perform navigation, sensors and computing hardware must be mounted on board the UAV. It means that hardware must be powerful enough to process images from camera at high frequency, but also is constrained in size and energy consumption. The thesis aims to develop and improve a map-based visual localization algorithm but also proposes a method for evaluating the energy efficiency of embedded parallel computing platforms. It is necessary to benchmark state-of-the-art hardware computing platforms and evaluating their ability to perform common vision task with the least energy consumption.

3. Research Object

The research object is visual localization of a UAV using computer vision, image processing, and Particle filtering algorithms, that suitable for execution onboard UAV with embedded computer hardware.

4. Research Aim And Objectives

The research aim is to develop new visual localization algorithm, that is capable of UAV pose estimation in GPS-Denied environments during low altitude and long-distance flights, running on embedded flight computer that outperforms existing methods in localization accuracy, execution speed, and robustness to errors in the aerial imagery.

To accomplish the research aim, followed tasks were performed:

1. Compare image similarity metrics to identify the most suitable metric for matching terrain images captured during a low altitude flight for UAV localization.
2. Compare particle sampling techniques used in Particle filters to identify which sampling technique requires the least computations for Particle filter localization algorithm.
3. Propose and compare image similarity to particle probability conversion function to improve the accuracy of a Particle filter localization algorithm.
4. Propose a visual Particle filter localization algorithm based on the results of previous tasks and evaluate the ability of the algorithm to reduce the accumulating error of Visual odometry.
5. Compare the proposed Particle filter localization algorithm against state-of-the-art SLAM algorithm.
6. Propose a new method for evaluation of embedded computing platforms to identify the most energy efficient platform currently available that suitable for computer vision tasks onboard UAV.

5. Research Methods

Research performed in this thesis is based on these scientific methods:

1. Literature review is performed on the latest scientific papers to identify, select and evaluate state-of-the-art algorithms solving the stated problem.
2. Quantitative and qualitative information gathering was performed to create datasets used for experiments and experimental data describing the performance of the proposed solution, or its components.
3. Statistical methods, e.g., *Student's t-test*, *Shapiro-Wilk test*, are used to perform confirmatory data analysis, ensuring the reliability of data and experimental setup.
4. Constructive research was used to propose improvements on the real-world of the problem and propose new methods to improve the theory.

5. Software development methods were used in the experimental part of this thesis, implementing localization algorithms and proposing new methods to improve existing algorithms.

6. Scientific Contributions Of The Research

The thesis contributes to the development of vision-based map relative localization systems for low altitude UAV flights. The main contributions of the thesis can be outlined as follows:

1. A modification of visual Particle filter localization algorithm is proposed. The thesis introduces a new component for Particle filter algorithm — image similarity to probability conversion function. The effects of the function on the accuracy, execution speed and robustness were measured in the experimental section of this thesis. The ability to achieve trade-off of accuracy for robustness by using function parametrization was presented.
2. Proposed Particle filter localization algorithm was able to outperform state-of-the-art algorithms — SVO and ORB-SLAM in the means of localization accuracy. The proposed localization algorithm drastically improves both - accuracy and execution speed over classical implementation. It allowed improving the accuracy results of state-of-the-art odometry and SLAM algorithms.
3. Embedded computing platform benchmark method is proposed to identify the most energy efficient parallel computing platform capable of common computer vision tasks. Three platforms were compared, and a platform based on NVIDIA TX1 GPU was identified as the fastest and most energy efficient platform for image processing onboard UAV among other compared platforms.

7. Practical Value Of The Research

The research is focused on low altitude UAV localization to provide GPS backup system during flight. The research provides valuable information for computer vision engineer researching on robotics, localization,

and image registration on embedded hardware. The most important practical contributions are the following:

1. New localization algorithm based on Particle filter was developed, and its source code [1] is publicly available for other researchers to benchmark against and use in the development of UAV backup localization systems.
2. The experimental results have shown that by using image similarity to probability density conversion function in Particle filter, it is possible to control the accuracy and robustness of the localization. By using *logistic* conversion function and changing conversion curve using a parameter, it is possible to achieve robustness and accuracy trade-off, e.g., increasing accuracy if up-to-date maps are available or increasing robustness to inaccuracies in an out-dated map with a certain loss of accuracy.
3. A new dataset of aerial imagery was created and published with open access. The dataset contains images from a simulated alongside with sensor data that can be used to develop and evaluate *optical flow*, *visual odometry*, *localization*, *SLAM*, and other vision algorithms. To the best of our knowledge, it is by far largest dataset of low altitude aerial imagery.
4. Different particle sampling techniques were evaluated and *KLD sampling* technique has shown to significantly improve localization algorithm execution runtime with similar accuracy compared to other techniques.
5. The new method of embedded computing platform comparison was used to identify the most energy efficient computing platform from currently most advanced available platforms. Nvidia Tegra X1 has shown the best energy efficiency results and price/performance rates.

[1] Source code is available online: `https://github.com/jureviciusr/particle-match`

8. Defensive Claims

The following claims are defended in this thesis:

1. Pearson Correlation Coefficient (or Normalized Correlation Coefficient) is the most suitable image similarity metric for visual localization of a UAV compared against the sum of squared differences and cross-correlation metrics.
2. The use of KLD sampling algorithm speeds-up the execution of Particle filter localization by reducing the number of particles used for localization without affecting positioning accuracy.
3. The use of logistic conversion function allows the proposed localization algorithm to achieve higher accuracy and improve robustness against inaccuracies in maps of the environment.
4. Proposed modification of Particle filter localization algorithm is more accurate then Visual odometry algorithm SVO and has reduced the accumulating error.
5. Proposed modification of Particle filter localization algorithm is more accurate than state-of-the-art algorithm ORB-SLAM.
6. The proposed method for energy efficiency measurement allows the comparison of embedded computing platforms and identifies the most energy efficient platform.

9. Approbation Of The Results

Results obtained in this thesis were published in five papers: two papers in periodic scientific journals and three papers in reviewed scientific conference proceedings. The results were presented in three international scientific conferences. The following list presents the publications and presentations in conferences:

Papers in periodic scientific journals:

[A1] Jurevičius, R., Marcinkevičius, V., and Šeibokas J. (2019). Robust GNSS Denied Localization for UAV Using Particle Filter and

Visual Odometry. Machine Vision and Applications, Springer. ISSN: 1432-1769 [IF: 1.788]

[A2] Jurevičius, R., and Marcinkevičius, V. (2016). Energy Efficient Platform for Sobel Filter in Energy and Size Constrained Systems. Baltic Journal of Modern Computing, 4(1), 2015, p. 79-88. ISSN: 2255-8950

Papers in peer-reviewed scientific conference proceedings:

[A3] Jurevičius, R., and Marcinkevičius, V. (2017). Application Of Vision-Based Particle Filter and Visual Odometry for UAV Localization. WSCG '2017: short communications proceedings: The 25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2016 in co-operation with EUROGRAPHICS: University of West Bohemia, Plzen, Czech RepublicMay 29 – June 2, 2017, p. 67–71. ISBN: 978-80-86943-45-9

[A4] Jurevičius, R., Marcinkevičius, V., and Taujanskas, V. (2016). Comparison of Image Similarity Functions and Sampling Algorithms in Vision-Based Particle Filter for UAV Localization. Proceedings of CSIST 2016, p. 109–114. ISBN: 978-985-566-369-1

[A5] Jurevičius, R., and Marcinkevičius, V. (2015). Energy Efficient Platform for Sobel Filter Implementation in Energy and Size Constrained Systems. In Information, Electronic and Electrical Engineering (AIEEE), 2015 IEEE 3rd Workshop on Advances in (pp. 1–5). IEEE. ISBN: 978-1-5090-1201-5

Presentations in international scientific conferences:

1. Jurevičius R. Energy Efficient Platform for Sobel Filter Implementation in Energy and Size Constrained Systems. The 3rd Workshop on AIEEE'15, Riga, Latvia. November 13–14, 2015.
2. Jurevičius R. Comparison of Image Similarity Functions and Sampling Algorithms in Vision-Based Particle Filter for UAV Localization. International Congress on Informatics CSIST'2016, Minsk, Belarus. October 24–27, 2016.

3. Jurevičius R. Application of Vision-Based Particle Filter and Visual Odometry for UAV Localization. The 25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision WSCG '2017, Plzen, Czech Republic. May 29–June 2, 2017.

10. Outline Of The Thesis

The thesis is organized as follow: Chapter 1 provides an introduction and overview of the thesis; Chapter 2 describes methods of visual localization which can be performed onboard UAV and datasets used for algorithms development and evaluation; Chapter 3 presents the research methodology and presents the new methods proposed by in thesis; Chapter 4 presents experimental results of evaluation of the modification of the Particle filter localization algorithm and the comparison of different embedded computing platforms; General conclusions are presented afterward; Bibliographic references are included at the end of the thesis. The dissertation consists of 110 pages, 30 figures, and 16 tables.

Chapter 1

VISUAL LOCALIZATION ON-BOARD UAV

This chapter reviews current state-of-the-art research in the field of GPS-Denied UAV localization. The types of UAVs are presented to define the constraints of the objective. Algorithms of localization are presented to narrow down which techniques are the most suitable for the localization. Parallel computing platforms suitable for computer vision processing onboard the UAV are presented. The literature review and experimental ideas are based on research published in papers [A1], [A2], [A3], [A4], and [A5].

1.1 Unmanned Air Vehicle Platforms

The recent development in hardware sensors and embedded processors has brought many consumer-level UAVs capable of autonomous operations. The UAVs can be categorized according to their weight into these categories:

- Micro Aerial Vehicle (abbr. MAV): < 1 kg
- Small Unmanned Aerial System (abbr. sUAS): < 25 kg
- Unmanned Aerial System (abbr. UAS): > 200 kg

The MAVs usually are small quadrotor helicopters, since they are the easiest to carry, operate, and can be relatively very small. Military and law enforcement fields have been using UAVs for a few decades, most of them are sUAS or large UAS. These UAVs are usually fixed-wing UAVs, which can be very expensive, but can carry more payload, fly in higher altitudes and flight distances might be of tens of kilometers. Figure

1.1 presents examples of common UAV platforms used in research and development, agricultural, and other commercial activities. AscTec Pelican UAV is shown in figure 1.1a, a UAV that was specifically designed for computer vision algorithm development. The UAV has an onboard high-performance computing platform and various sensor - optical camera, laser scanner, IMU, GPS, Barometer, and other. It is a very popular platform among the computer vision research community. Figure 1.1b shows a more classical helicopter model UAV, which was modified with autonomous capabilities for localization research by G. Conte et al. [11]. Figure 1.1c shows the Sentera Pheonix fixed wing UAV, which is designed for aerial mapping and survey in agriculture. Figure 1.1d show a larger fixed wing UAV Baam Tech Futura. The fixed-wing UAVs are the most common platform in military and law enforcement fields, due to the ability of long-distance flights. The altitudes in which the UAVs operate can be categorized into these categories:

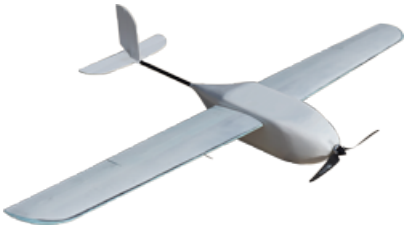
- Near-ground altitude: $<$ 100 meters
- Low altitude: between 100 and 3000 meters
- High altitude: $>$ 3000 meters

The small quadrotor MAVs, such as the AscTec Pelican, usually fly in near-ground altitude ($<$ 100 meters) or in indoor environments, so the use of vSLAM algorithms is quite common among these types of aircraft. The larger helicopters and fixed-wing UAVs flying in higher altitudes have not received that much of research activity, the most popular localization method for them is VO. The Stereo Vision system, consisting of two calibrated cameras, providing depth information is used for UAVs flying in low altitude. The depth information combined with optical data can improve the positioning accuracy of vision-based localization algorithms. On the other hand, flying at higher altitudes stereo camera systems are unable to provide depth information since the disparity between images is negligible. Low altitude ($>$100 meters) usually uses monocular localization techniques.

(a) AscTec Pelican UAV [1]

(b) Modified Autonomous RMAX Helicopter UAV [11] [2]

(c) Sentera Pheonix Fixed Wing UAV [3]



(d) Baam Tech Futura Fixed Wing UAV [4]

Figure 1.1: Different autonomous UAV platforms.

1.2 Mobile Robot Localization

Localization is one of the key challenges for a robotic platform to achieve complete autonomous operation. Localization is the process of estimating the robot's pose relative to its environment according to sensor data

[1] Ascending Technologies Home Page: `http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-pelican/`

[2] Research By G. Conte et al.: `https://www.researchgate.net/figure/The-Rmax-helicopter_fig1_242515144`

[3] Sentera Pheonix UAV Home Page: `https://sentera.com/phx-drone/`

[4] Baam Tech Futura UAV Home Page: `https://baam.tech/in-depth`

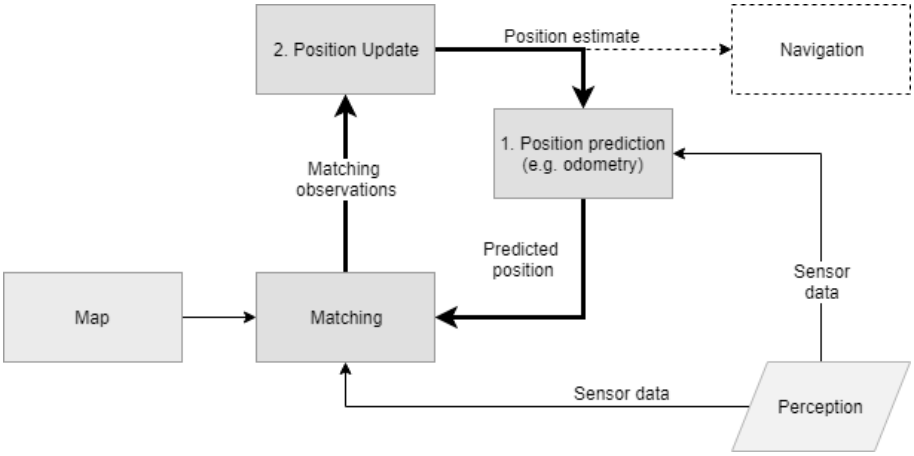


Figure 1.2: General scheme of a mobile robot localization process.

[4]. The pose is usually estimated relatively to an *a priori* known map or is building the map during operation. Estimating the precise robot's pose is required to perform autonomous navigation. This section presents general approaches to mobile robot localization using computer vision techniques since UAV is a mobile robot; the algorithms are generally suitable to all moving robotic platforms. A general representation of robotic localization is presented in figure 1.2. The general localization of a mobile robot is performed in two main steps [12]:

1. Position prediction;
2. Position update.

Position prediction step takes the current position and applies the measured motion from sensor data, predicting the new position. The current observations from sensors are used to match the predicted position to what the robot currently "sees" and update the prediction accordingly. The estimated robot position from position update is passed to the navigation module, that uses this information to control the robot's motors and plan actions to achieve the robot's goal.

Current state-of-the-art localization techniques are mostly probabilistic techniques, which are variants of the Bayes filter or at least employ

some Bayesian filter based techniques. These techniques include [13]:

- Kalman filters
- Particle filters
- Multihypothesis tracking
- Grid-based approaches
- Topological approaches

In probabilistic techniques, the robot's position is denoted as belief over position variable z_t by $bel(z_t)$ and is described in the following form:

$$bel(z_t) = P(z_t|z_{0:t-1}, m_{0:t-1}), \tag{1.1}$$

where:

- z_t - position variable on time t, it is a set of coordinates describing robot's position in a coordinate system (it can be a planar 2-dimensional environment or a complete 3-dimensional pose described using 6-degrees of freedom)
- m_t - sensor data (IMU, barometer, wind speed and other data used for dead-reckoning) on time t.

The estimated position can be seen as a conditional probability over all past robot positions $z_{0:t-1}$ and all past sensor measurements $m_{0:t-1}$. On each iteration of the localization process, the previous position z_{t-1} is transformed into current position z_t by applying new movement data to the previous location and correcting the location with current sensor data.

1.2.1 Bayesian Filter

Bayesian filtering can be used to estimate the position of a mobile robot platform. One of the first authors to explore iterative Bayesian estimation and describe Bayesian filtering was Y. Ho et al. [14], the authors show that the approach can solve nonlinear, non-Gaussian estimation problems [15]. In general, the Bayes filters estimate a dynamic system's

state from noisy sensor measurements [13]. Bayes filters are iterative algorithms implementing the Bayes rule:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}, \tag{1.2}$$

where:

- A and B are events.
- $p(A|B)$ is a conditional probability of event A occurring given B has occurred.
- $p(B|A)$ is a conditional probability of event B occurring given A has occurred.
- $p(A)$ and $p(B)$ are the probabilities of occurring events A and B independently.

This probabilistic approach is necessary since data from any sensor contains some amounts of noise which is addressed by filtering using the Bayes rule. In the context of mobile robot localization, the discrete Bayes rule is used more often [13]:

$$p(A|B) = \frac{p(B|A)p(A)}{\sum_{A'} p(B|A')p(A')}, \tag{1.3}$$

where A' is a single discrete element from the set A and $\sum_{A'} P(A') = 1$. Algorithm 1 shows the basic Bayes filter algorithm to calculate robot pose estimate shown in equation 1.1 using the discrete Bayes rule [16]. Line 3 of the discrete Bayes filter is the prediction step of the algorithm; it calculates belief over state z_t and measurements m_t. If the algorithm performs only localization without knowing the movement, this step usually uses odometry information to predict the state, or in other cases, control information sent to robot motors can be used to predict the state. In line 4 the algorithm multiplies the belief $\bar{bel}(z_t)$ with the probability of measurements m_t being observed in the current state. The resulting value is normalized using normalization constant η to obtain the final belief $bel(z_t)$.

Algorithm 1 Discrete Bayes Filter

Inputs: the previous robot position belief distribution $bel(z_{t-1})$, latest sensor data m_t

Outputs: current robot position belief distribution $bel(z_t)$

1: **function** DISCRETEBAYESFILTER($bel(z_{t-1})$, m_t)
2: **for** all z_{t-1} **do**
3: $\bar{bel}(z_t) = \sum_{z_{t-1}} p(z_t|m_t, z_{t-1})bel(z_{t-1})$ ▷ Prediction
4: $bel(z_t) = \eta p(m_t|z_t)\bar{bel}(z_t)$ ▷ Current state belief, normalized
return $bel(z_t)$

1.2.2 Kalman Filter

Kalman filter is the most widely used implementation of Bayes filter. Kalman filter was developed by R. E. Kalman in the 1960s, and its first application was to estimate spacecraft trajectories in the Apollo mission [17]. Since then the Kalman filter was used in a vast array of application, including robot pose estimation, aircraft trajectory estimation, various tracking application, such as GPS / GNSS and radar tracking. Despite the wide usage, the Kalman filter has its limits, such that it is implemented for continuous states and is not applicable for discrete spaces. Kalman filter approximates beliefs by their first and second moment, which is identical to unimodal Gaussian representation [13]:

$$bel(z_t) = N(z_t; \mu_t, \Sigma_t) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_t|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_t-\mu_t)^T\Sigma_t^{-1}(x_t-\mu_t)}, \tag{1.4}$$

where:

- μ_t is the distribution's mean (first moment) on time t,
- Σ_t is $d \times d$ covariance matrix (second moment) on time t,
- d is the dimension of the state,
- $N(z_t; \mu_t, \Sigma_t)$ denotes the belief of state z_t given a Gaussian distribution with mean μ_t and covariance Σ_t.

Kalman filters are optimal estimators, assuming the initial uncertainty is Gaussian, and the system is linear. The Kalman filter can

also be applied to some nonlinear systems which can be linearized using first-order Taylor series expansion, this type of Kalman filter is called the extended Kalman filter[13]. The Kalman filter works only on linear systems (or linearized nonlinear systems), so it usually is used for tracking problems when the state can be observed with a continuous Gaussian distribution and little noise [16].

1.2.3 Particle Filter

Particle filters are a nonparametric implementation of the Bayes filter. Particle filters approximate the systems state space posterior by a finite number of parameters called particles. The Bayes theorem is used in Particle filters to update a particles probability value when more data is available. The particles in the Particle filter is denoted as a finite set:

$$S_t := P_t^{(1)}, P_t^{(2)}, ..., P_t^{(n)} \tag{1.5}$$

Particle $P_t^{(i)}$ (where $1 \leq i \leq n$) is an instance of state at time t. In other words, a particle is a hypothesis of a true state space at time t, and in the case of localization, it represents the robot's position in space. Here n represents a number of particles, which can be a fixed number, e.g., $n = 500$, or in specific cases, the number of particles can be dynamic. Particle filters that use variable particle count to represent a distribution are called Adaptive Particle filters.

Algorithm 2 shows a basic Particle filter algorithm based on importance sampling [16]. In line 4 of the algorithm performs prediction step by sampling a particle from a previous set, while in line 6 the particle is propagated forward using odometry data or control signal value instead and inserted into the set $\bar{S}_t$ which will contain all the propagated values. The lines 8 and 9 take the propagated set $\bar{S}_t$ and draw new particles from it, a new set of particles is created for the next iteration. In the Particle filter algorithm, the set S_t represents the belief distribution $bel(z_t)$ in Bayes filter algorithm. The set $\bar{S}_t$ is the representation of $\bar{bel}(z_t)$.

Different sampling techniques can be used in Particle filters to achieve different goals, in some cases, a fixed amount of particles are

Algorithm 2 Importance sampling based Particle filter algorithm.

Inputs: the previous particles S_{t-1}, latest sensor data m_t
Outputs: newly drawn particles S_t

function PARTICLEFILTER(S_{t-1}, m_t)
  $\bar{S}_t = S_t = \emptyset$
  for i = 1 to n **do**
    sample $P_t^{(i)} \sim p(z_t^{(i)}|m_t, P_{t-1}^{(i)})$ ▷ $z_t^{(i)}$ denotes the particle's position
    $b_t^{(i)} = p(m_t|P_t^{(i)})$
    $\bar{S}_t = \bar{S}_t \cup < P_t^{(i)}, b_t^{(i)} >$
  for i = 1 to n **do**
    draw P_t with probability $\propto b_t^{(i)}$
    $S_t = S_t \cup P_t$
  return S_t

necessary to achieve a fixed iteration run time, while embedded platforms use sampling techniques that can adapt particle counts to reduce computations.

1.3 Monocular Visual Localization

Monocular visual SLAM algorithms have been developed over the last few years and achieved localization GPS-Denied environment. Visual SLAM algorithm developed in [18] was shown to achieve centimeter-level precision in an in-doors environment. ORB-SLAM2 and LSD-SLAM achieved very accurate results on TUM-RGBD and KITTI datasets [19] [20], however, these experiments were performed on forward facing camera images only, which is not the case for low altitude (> 100 meters) UAV flights. Additionally, SLAM techniques suffer from increasing localization error due to the lack of a global localization. The problem of GPS-Denied navigation is solved by Visual odometry algorithms, although they achieved very high performance and precision, they also suffer from accumulating error (position drift) over long distance flights

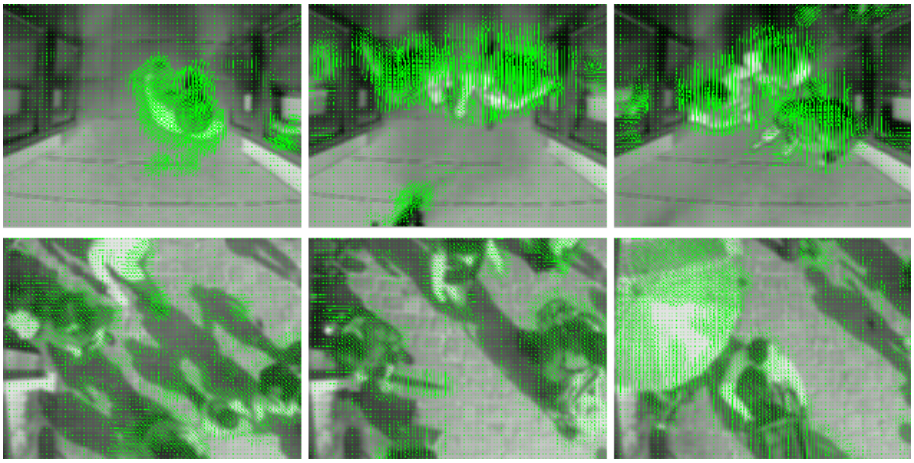


Figure 1.3: An example of a Dense optical flow of sample images. Green lines represent the direction and magnitude of the approximated camera movement. [21]

(> 1km distance). The thesis focuses on middle-sized sUAS UAVs flying in low latitude flights, so the localization techniques covered in this section only includes monocular systems, since, as mentioned in section 1.1, stereo vision systems do not provide any gains at altitudes above 100 meters.

This section describes algorithms and systems developed to solve pose estimation in GPS-denied environments and discusses the applicability of the algorithms for low altitude UAV flights.

1.3.1 Optical Flow

Optical flow is a method of calculating the movement vectors of pixels given two images of the same scene. The optical flow techniques are divided into two categories: global and local methods. Global methods, such as Horn-Schunck [22], estimate motion vectors on the whole image, even if it contains homogeneous zones. Visualization of a global method used to estimate dense optical flow is presented in figure 1.3. Local methods, such as Lucas-Kanade [23], calculate optical flow only for specific

pixels of interest and their neighborhoods, which allows faster execution and is more robust to image noise compared to global methods. The Horn-Schunck and Lucas-Kanade methods are considered classical optical flow methods. Applications of optical flow inspired by bionic insect vision were used to estimate UAV self-motion in several pieces of research [24–26]. More recently, optical flow combined with IMU data was implemented to perform fixed hovering and landing on moving platform for UAV [27]. Additionally, optical flow can be used to detect moving objects in the scene from UAV [28]. Although the optical flow is suitable for some of the maneuvers of UAV, such as hovering and landing, it is unsuitable for long distance flights, since the optical flow methods, even combined with IMU data, cannot acquire suitable positioning accuracy [29].

1.3.2 Visual Odometry

Visual odometry (abbr. VO) is the process of calculating aircraft (or robot) egomotion from the camera image stream. VO methods usually fall into three categories[30]:

1. feature-based methods,
2. appearance-based (also known as direct) methods,
3. hybrid methods.

Feature-based methods extract visual features and track them. Depending on the focus of the method, one of the following features can be used:

- Haris [31],
- Shi-Tomasi [32],
- SIFT [33],
- SURF [34],
- CENSURE [35],
- BRIEF [36],
- FAST [37, 38],
- ORB [39].

SIFT and SURF features are considered to be very computationally demanding, novel algorithms tend to use ORB or FAST features to achieve real-time execution. Appearance-based VO approaches (also known as direct methods) does not rely on engineered visual features but uses pixel intensity directly to reconstruct motion. These methods are also very computationally demanding, so they use GPU resources for real-time execution. Also, over the last few years, hybrid approaches were developed, that computes odometry directly on pixel intensity values of selected regions. VO can be calculated using two hardware approaches — using monocular or stereo imagery. This research focuses on low altitude (>100 meters) flights, where stereo cameras do not provide any advantage, the following review of algorithms will be focused on monocular VO only.

Nistér et al. in 2004 [40] proposed first monocular VO algorithm. In the mentioned paper, five-point RANSAC [41, 42] was introduced to calculate motion hypotheses, and later this method became popular among other VO algorithms. A method of combining IMU sensor data with the optical flow in an EKF framework was proposed in [43]. The first implementation of the proposed algorithm was a part of a SLAM algorithm, but later the method was published as a standalone visual navigation method [44]. Some VO algorithms were proposed for UAVs that were dependent on PTAM (a feature-based SLAM algorithm, the next section will present it in details) [45–47]. A semi-direct VO (abbr. SVO) was proposed to calculate motion in real-time onboard UAV [2, 48]. The algorithm uses a sparse approach to select features for matching between frames. The sparse approach uses pixels at corners detected using FAST corner detector or along intensity gradient edges. This approach uses fewer features, but the selected features are very strong, and the algorithm achieves precise image alignment using less computational resources. The sparse approach used in this algorithm allows high-speed execution — 55 frames per second on embedded computer hardware[2]. Figure 1.5 shows the transformation calculated from image sequences. Due to the new era of deep learning and CNN, Visual odometry is also

one of the fields that are explored by deep learning approach, CNN is proposed to be used for Visual odometry calculations [49, 50]. The approach is to learn the motion from pixel values over two frames. The deep learning based methods are still heavily developed, and accuracy results are only available on the datasets, that was used to train the network, so the general use case for them is still unsure.

SVO with a downward facing camera is used in this research to calculate aircraft motion. It provides more accurate positioning compared to other algorithms and real-time execution on embedded platforms. The algorithm processing pipeline is shown in figure 1.4. The pipeline can be described in these steps:

1. Feature tracking — features are extracted from the input image and matched against features from the previous image.
2. The matched features are aligned with the constructed 3D map from previous iterations.
3. Mapping thread uses the processed image to detect 3D displacement of the camera. Then the features are mapped into 3D space and the constructed map is used during feature alignment.

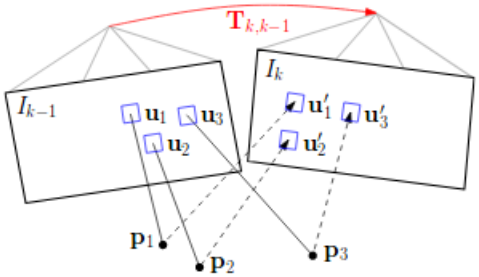


Figure 1.5: Visual odometry calculates motion by aligning 3D world points p_j in image sequence and calculate transformation $T_{k,k-1}$ that causes the features to be reprojected from image patches u_n to u'_n. [2]

The problem of any VO algorithm is accumulating drift over long distance flights. Since VO computes the camera path incrementally, the

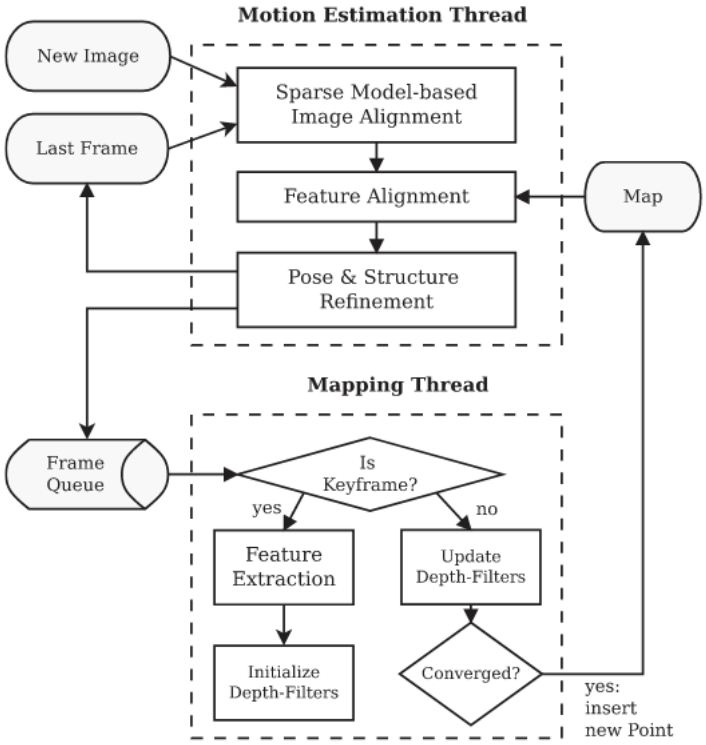


Figure 1.4: SVO algorithm processing pipeline. [2]

errors introduced by each frame are accumulated [30]. Bundle adjustment (abbr. BA) method was proposed to reduce the drift. It works by using the sliding window approach for image sequences and tries to adjust positions of features between more than two frames. A work from K. Konolige et al. [51] has shown that sliding window BA approach can reduce the position drift by a factor of 2 to 5 over a 10 kilometer VO experiment. However, since the algorithm uses a map built by its own, the drift is unavoidable, so VO is a suitable method for short distance flights, but long-distance flights require the ability of global localization to remove the drift altogether.

1.3.3 Bayesian Nonlinear Filtering

Another approach of UAV localization is the use of algorithms based on nonlinear filtering. Nonlinear filtering makes an inference of the state, by computing the posterior distribution for state vector given all the observations at the time. Traditional approaches use the Kalman Filter (abbr. KF) [52] which computes posterior distribution for linear Gaussian systems. Nonlinear, non-Gaussian state-space estimation is solved by using a general KF and adding additional noise to and second-order moments; this approach is named the extended Kalman Filter (abbr. EKF) [53]. These approaches are usually used for computationally limited applications, where the systems are mostly linear or mildly nonlinear. In the case of UAV localization, KF and its' variants are suitable for signal processing of the sensor data, but the localization in a map might be highly nonlinear, especially during the global search, when the position of aircraft is highly uncertain. The Particle filter development started in 1993 [54] when resampling stage was introduced. Resampling allowed the application to approximate the more nonlinear models and it was used in fastSLAM [55] algorithm to solve the SLAM problem. It was also incorporated in UAV localization techniques using height measurements. [56].

1.3.4 Visual SLAM

For the case of localization in a very dynamic environment with no available map, Visual odometry is extended to build a 3D structure of the environment, which is used for localization during algorithm runtime. Such systems are referred to as Simultaneous Localization and Mapping (SLAM) systems. The earliest research that proposed a solution to robotic mapping with localization problem was in the 1990s, in papers by R. Smith et al. [57, 58]. Another influential paper by [59] has stated the problem of mapping and localization as a single problem. The early researches were made using a range sensor, specifically ultrasonic distance sensors to map and navigate in a 2D environment. Since then, a vari-

ety of 2D SLAM systems were developed [60–63], which easily maps the unknown environment and can navigate in it. With the advancement of photogrammetry and computing power of small computers, optical camera based SLAM systems were developed. SLAM systems that use optical camera sensor data as input are also referred to as visual SLAM (vSLAM), the algorithms are widely used in computer vision, robotics, and augmented reality. vSLAM methods can be categorized the same way as the VO algorithms:

1. Feature-based methods,
2. Direct methods,
3. Hybrid methods.

An overview of feature-based and direct vSLAM methods key components is presented in figure 1.6. The basic components in a SLAM system are [64]:

1. Initialization,
2. Tracking,
3. Mapping.

During the initialization, a global coordinate system is defined, and the currently visible environment is reconstructed. Tracking and mapping steps are iteratively performed on the initial map. Mapping step extends the map of the environment, while tracking matches image sequences and performs localization inside the map. Tracking is performed by matching current landmarks with landmarks from previous images. Tracking calculates the relative motion between two consecutive images. Feature-based methods usually use visual feature descriptors as landmarks and match them using one of the RANSAC [42] family algorithms. The direct method does not always on landmark matching; some direct methods estimates camera motion by minimizing the photometric residual between two intensity images [65]. Structure from Motion (SfM) is the most common method of reconstructing the environment from the image sequence. SfM is a crucial component used for photogrammetry, robotic mapping, SLAM, and some VO algorithms. SfM recreates a

3D model of a scene from sequences of images. In the case of VO and SLAM, SfM is used to build a 3D model of the environment. Direct methods recreate a dense point-cloud of the environment while feature based methods might recreate the environment with only the features it is extracting. SfM is solving some of the critical challenges faced in robotic mapping applications [66]:

1. Measurement error is statistically dependent,
2. Data association problem,
3. Environment is not static; it changes over time.

These problems are also relevant from a localization perspective, just as the SLAM algorithms solve the mapping and localization as a single problem. One of the first SfM methods that reconstruct dense environments was proposed by C. Tomasi et al. [67]. Over the recent years and especially with interest for a robotic mapping application, improved methods were developed [68].

Additional modules are incorporated in the systems to ensure the long running of the algorithm:

- Relocalization,
- Global map optimization.

The relocalization is performed in case the tracking of position is lost and the current position has to be searched in the created map, or the mapping must be restarted. Global map optimization usually refines the already built map using BA if the system detects that the robot platform has returned to the point that was previously visited and a loop was formed.

First monocular vSLAM was developed in 2003 and was called MonoSLAM [69, 70]. To increase the computational efficiency of MonoSLAM - PTAM algorithm was introduced [3], it separated tracking and mapping into separate threads on CPU. The PTAM algorithm was very influential and many VO algorithms that were developed after it was proposed, used PTAM as its' internal framework. PTAM algorithm is composed of the following four main components [64]:

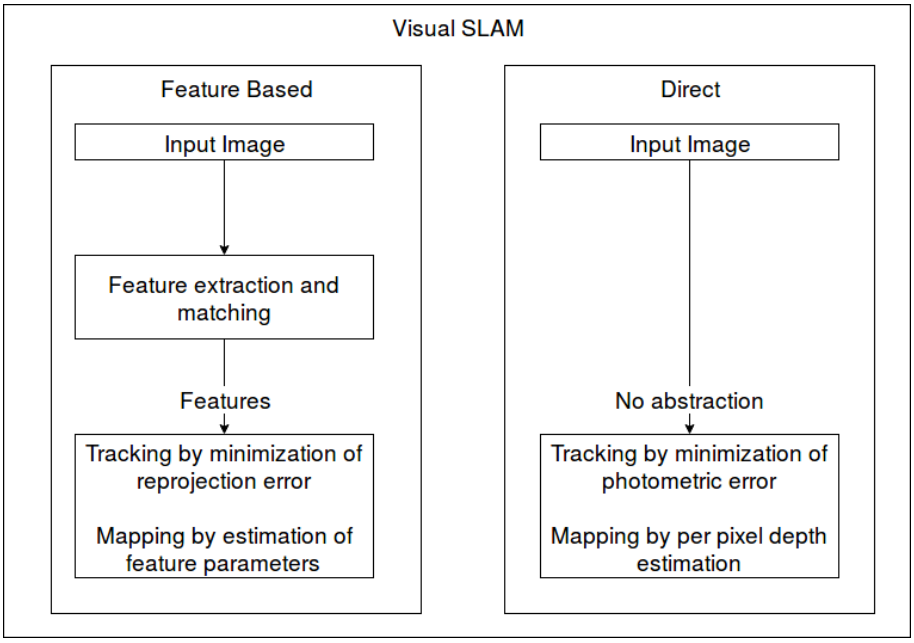


Figure 1.6: An overview of feature based and direct vSLAM methods. The key difference is that direct methods does not abstract image data using feature extraction and uses photometric error minimization for movement tracking.

1. Map initializing using five-point RANSAC algorithm [41].
2. Camera poses are estimated from matched feature points and the input image.
3. 3D position of features is estimated by triangulation.
4. The tracking process is recovered by randomized tree-Based search [71].

ORB-SLAM [19, 72] an extension of PTAM, which includes BA, vision-based closed-loop detection, and 7 DoF pose-graph optimization. ORB-SLAM is the most complete feature-based monocular vSLAM system [64]. ORB-SLAM achieves state-of-the-art results on TUM-RGBD and KITTI datasets. The algorithm is available for stereo and mono sequences. The concept of monocular ORB-SLAM2 algorithm is based

on ORB feature [39] extraction and matching. The ORB features are tracked in every frame of the image sequence; the camera is localized by matching the features to the local map and minimizing the reprojection error by applying motion only BA, and the camera is localized relative to the local map.

DTAM [73] is one of the first direct vSLAM method. The method uses all pixels in the imagery for reconstruction, so it is a dense method. It was optimized for execution on a GPU to achieve real-time execution of the algorithm. DTAM was also shown to be able to execute on mobile device [74] (smart-phone), so it is possible to run it on embedded hardware. One of the currently leading direct methods of SLAM is the LSD-SLAM [20, 75]. It reconstructs only areas containing intensity gradient, thus ignoring textureless areas. The semi-dense approach allows the execution of the algorithm on CPU in real-time.

1.3.5 Other GPS-Denied Localization Methods

This section covers other visual proposed localization methods that do not use optical flow, VO, or vSLAM. An algorithm of extracting features and matching them to GIS data is proposed in [76]. By extracting corners and lines, algorithms try to identify roads and buildings and match them to a GIS model, thus providing a location. A similar approach of detecting and matching roads in aerial imagery for localization is proposed in [77, 78]. Image registration technique is used to match aerial images with orthophoto maps in research by P. Jende et al. [79]. The proposed algorithm extracts corner features from UAV camera images and matches them against extracted features from an orthophoto map. The method was shown to work quite well in urban areas where the roads provide robust features for matching, but authors suggest that without the strong features, localization accuracy might drop. A map matching localization approach is presented in [9], where a UAV is localized relatively to imagery from Google maps. During a test flight, algorithms achieved root mean square error of ~6 meters, which is comparable to GPS root mean square error of ~3 meters. Unfortunately, the

authors do not provide absolute mean error, which could be used for an actual comparison against other methods.

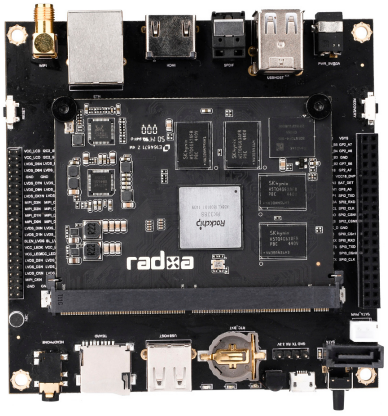
Terrain Contour Matching (abbr. TERCOM) system was developed for cruise missile navigation before the GPS was widely available [80]. TERCOM navigation systems are still in use, but they are dependent on the latest height maps, which are not universally available. Digital Scene Mapping and Area Correlation (abbr. DSMAC) was an improved system with the same idea of TERCOM, but instead of heights, it uses aerial imagery obtained using optical camera sensor, but it is also very dependent on up-to-date maps [10]. DSMAC systems are susceptible to shadows, so the maps must be made at the same time of the day as the navigation is performed. DSMAC systems were proven to be useful and used for lunar module landing in the Apollo missions.

1.4 Parallel Computing Platforms for Size and Energy Constraint Environments

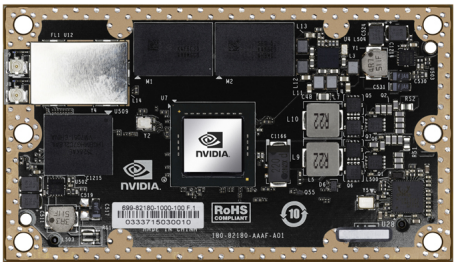
Sophisticated computer vision applications often used in robotics requires a lot of computing power, i.e., recent research demonstrates that vision-aided navigation is feasible during flight-time to operate in GPS-denied environments, but the challenge is in dealing with the power and weight restrictions onboard a UAV while providing necessary positioning accuracy [81]. Most modern computers have enough computing power to run complex computer vision algorithms, though such machines often consume tenths or hundreds of watts of electrical power and are large compared to small robotic systems or UAVs. The computing power requirements can be ignored until the system has to employ image processing and computer vision algorithms for obstacle avoidance, object recognition or vision-aided guidance using digital cameras [82].

Researchers developing a framework for computer vision algorithm benchmark mentioned that in energy and performance constrained context, such as a battery-powered robot, it is essential to achieve sufficient accuracy while maximizing battery life [84]. By using less power on

(a) The Parallella platform.



(b) Radxa Rock2 Square development board.



(c) NVIDIA Tegra X1 SoC.

Figure 1.7: Platforms used in the experimental comparison.

computations may allow the usage of more sensor which could increase aerial vehicle security and flight distances.

As described in [85], there are three leading high-performance platforms for image processing (specifically sliding-window algorithms): multi-core systems, GPU and FPGA. The research conducted in [85] states, that multi-core CPU systems may be very energy inefficient compared with FPGA and GPU implementations.

FPGA based computing platforms suffer from complex development environments; it requires a lot of development effort to implement such

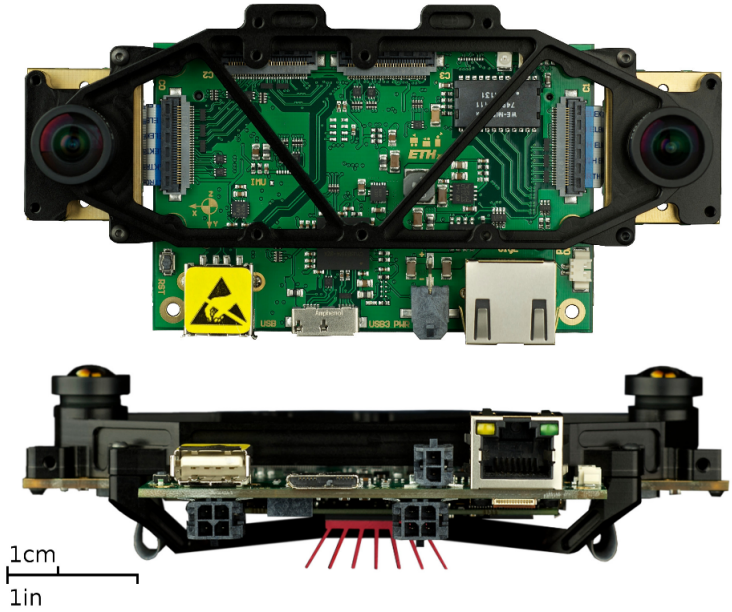


Figure 1.8: SLAM Sensor Unit with integrated Xilinx Zynq processor with FPGA. [83]

complex algorithms as vSLAM. A platform developed in work by J. Nikolic et al. [83], designed manufactured an integrated sensor platform based on Xilinx Zynq-7020 processor containing ARM two core CPU and FPGA on a single chip (see figure 1.8). Since the development of SLAM system on the FPGA requires much effort, the FPGA is responsible for sensor data acquisition, pre-processing and sensor data synchronization and the SLAM algorithm used to perform positioning was executed off-line.

The new Parallella (see figure 1.7a) platform may be of interest since it implements low power RISC architecture. The platform contains Xilinx Zynq-7010 processor with integrated FPGA programmable logic and a 16-core Epiphany co-processor. The platform uses FPGA logic only for data line interconnections between Xilinx CPU and Epiphany co-processor. The Epiphany co-processor was designed to execute parallel

applications with high energy efficiency. The 64-core Epiphany processor variant has shown to have a high potential in energy efficient computing by achieving 2400 cycles per second per Watt (c/s/W) compared to 79 c/s/W for the Intel i7-4770K CPU [86]. So it should be possible to achieve the same computational power using 30 times less power. These numbers are very rough since a single cycle in Intel's instruction set may do such amounts of works which could require more than several cycles in Epiphany's RISC architecture. Still, research may be conducted to compare this platform with other efficient platforms - FPGA or an embedded GPU. I. Grasso has researched embedded Mali GPU performance and energy for high-performance computing (HPC), the potential is that Mali 604 GPU has an 8.4x computing speedup compared with Cortex A-15 CPU core while using 32% of the energy [87]. Previous research conducted by J. Knezovic [88] implementing a blow-fish password hacking algorithm has shown to be very efficient from an energy perspective. The efficiency of the algorithm implementation was measured in password cracks per second per Watt of power. The Epiphany 16 core co-processor was able to crack as many passwords as Intel's T7200 processor, but epiphany processor requires 17 times less energy to do the same calculations [88]. Three hardware platforms were chosen for the comparison. All of the platforms were chosen to be small, maximum power usage under 10 watts and small enough to fit most UAVs. One platform used in this research will be Parallella [89], which uses a 16 core RISC co-processor designed for parallel computing.

Another platform will be Radxa Rock2 SoM (see figure 1.7b), which employs Rockchip RK3288 quad-core ARM CPU and Mali T764 GPU. The Mali GPU is one of the few widely available embedded GPU cores with OpenCL 1.1 [90] support for general purpose parallel computing. The ARM Mali family GPU are the most common co-processor in the smartphone industry.

In 2015 NVIDIA introduced a new embedded SoC - Tegra X1. It was specially designed for mobile platforms and robotics, including UAVs in mind. It was designed to be small and energy efficient while provid-

ing cutting-edge performance. Airvision Core X1 platform, utilizing NVIDIA Tegra X1 SoC, delivers real-time computer vision and navigation for Unmanned Aerial Vehicles (UAV) (see figure 1.7c). This hardware platform uses ARM quad-core CPU (up to 2GHz clock speed) and Maxwell architecture GPU with 256 cores (up to 1GHz clock speed) and is programmable using CUDA toolkit [91]. In 2017 NVIDIA released an improved SoC - Tegra X2, which is an improved version of Tegra X1.

1.5 Datasets for Benchmark of Visual Localization Algorithm

Several methods have been proposed to solve the GPS-denied localization problem using imaging sensors and computer vision algorithms. A survey by Yuncheng Lu et al. [29] suggests that current methods can be categorized into map-based, map-building, and map-less systems. Map-less systems include Visual odometry and optical flow algorithms. Map-building systems are usually recognized as SLAM algorithms, that build maps of the environment during runtime and localizes relatively to the created map. Both map-less and map-building systems have several public datasets that are widely used by the computer vision community to benchmark these systems. Map-based systems do not have established public datasets, and researchers create their datasets to measure their accuracy. This problem was already mentioned in research papers by A. Nassar et al. [92] and M. Shan et al. [9]. The self-made datasets are usually small since the creation of a dataset is very laborious. Due to the absence of publicly available dataset, the comparison between algorithms becomes ambiguous. Real-life test flights, of course, are mandatory for any system to be proven reliable, but an initial off-line benchmarking can provide substantial information about the algorithm performance within common pitfalls and can be used to compare against each other. This section presents a new dataset that was explicitly created for map-based system benchmark providing different trajectories, environments, and altitudes. Despite a map-based system focus, it can also be used for map-

less and map-building systems to measure the accuracy for high altitude (> 200 meters) UAV localization. The dataset can be downloaded online from `https://zenodo.org/record/1211730#.W3HJ_XV95hE`.

This section reviews popular datasets used for aerial localization systems benchmarks. Since there is no public dataset suitable for map-based system benchmark, recent works of such systems are also reviewed, to observe the volume of data that is used in the state-of-the-art map-based localization system development.

1.5.1 Most Notable Publicly Available Datasets

SLAM and VO research areas have a few datasets that have been established as a baseline for benchmarking. The most notable are KITTI [93], TUM-RGBD [94], ICL-NUIM [95], EUROC [96] and a very recent Zurich Urban Micro Aerial Vehicle Dataset [97]. The following list provides short content overviews of these datasets:

1. TUM-RGBD dataset was created using a handheld Kinect sensor in an indoor environment. The dataset provides ground truth locations of the camera, that were precisely captured, using external sensors mounted in the environment.
2. ICL-NUIM dataset provides handheld RGB-D data sequences from a simulated environment of small indoor spaces, such as living room and office. Ground truth contains camera locations, the dataset is very similar to TUM-RGBD, except it was created in a simulated environment.
3. EUROC dataset is created in various indoor environments with a stereo camera mounted on a micro UAV. The dataset contains precise ground truth trajectory of the aircraft from VICON motion capture system and 3D scans of the environments.
4. KITTI dataset was created using a stereo camera and other sensors mounted on a car, and the imagery was captured while driving in an outdoor urban environment. The dataset contains a lot of material (130 gigabytes of video footage) and can be used as a

benchmark for localization algorithms that use the front-facing camera images for UAVs flying at low altitude.

5. The Zurich dataset was specifically created to benchmark localization systems of UAVs flying in low altitude in outdoor urban environments. The Zurich dataset contains monocular front-facing camera images captured from a UAV flying at the altitude of 5-15 meters. The dataset contains images from 2 kilometers of flight distance in Zurich city.

1.5.2 Datasets For Benchmark Of Map-Based Localization Algorithms

The dataset reviewed in the previous section provides a variety of challenging environments — indoor and industrial spaces, urban city streets with live traffic. However, no dataset could be used to measure localization accuracy of a low altitude UAV flight with a downward facing camera. Due to the lack of a dataset, the following list of researches relies on self-made simulated datasets or real-life test flights:

1. Navigation system using VO, inertial navigation, and image registration proposed by G. Conte et al. [11, 98] uses images from an offline dataset which contains imagery from a 1-kilometer distance flight of a rectangular trajectory. The proposed system was additionally evaluated during a test flight.
2. Geo-referenced localization approach published by F. Lindsten et al. [99] was evaluated using a self-made dataset captured from a flight covering a distance of 400 meters.
3. Map matching approach presented by M. Shan et al. [9] was evaluated using images from a 3-minute flight (around 360 meters of distance) at an altitude of 80 meters. This dataset is available publicly and can be used for benchmarking, although it is rather small and contains only a single flight.
4. A research by A. Nassar et al. [92] proposed deep Convolutional Neural Network (abbr. CNN) for image registration from a downward facing camera a UAV. Due to the lack of datasets, authors

created two datasets of their own, (1) using images from 1.2-kilometer distance flight at 300 meters altitude over the city of Potsdam, Germany and (2) from a flight of 0.5-kilometer distance over Famagusta, Cyprus.

5. Navigation system based on detection and matching of road intersections proposed by S. Dumble et al. [78] evaluates the system on a test flight of around 7 kilometers distance.
6. Image feature based localization approach is proposed by T. Wang et al. [100], and the system is evaluated using self-made image-in-loop simulation. The volume of the dataset is undisclosed.

Unfortunately, datasets from most of these researches are not publicly available, or the volume of the datasets are relatively small and does not cover more than one aspect of map-based GPS-denied localization. Due to the lack of publicly available datasets for map-based localization systems benchmark, a new dataset was created and is presented in section 2.2.2.

1.6 Conclusions of Chapter 2

This section reviews the development of visual localization methods used for UAV localization.

- State-of-the-art in visual localization for near-ground altitude and in-doors environments is currently achieved by vSLAM algorithms, which show incredible performance and accuracy. The ORB-SLAM algorithm is currently the state-of-the-art algorithm among vSLAM methods.
- VO is among the most popular methods for position estimation during low altitude flights, although VO can provide localization, all VO algorithms suffer from position drift over a long period of flights.
- Particle filter techniques can be used to solve localization drift for long-distance flights; the technique is known to solve nonlinear, non-Gaussian state-space estimation problems.

- For the system to be able to work onboard UAV in real-time, it is necessary to compare computing platforms and choose the most energy efficient platform capable of computer vision tasks.
- It is required to propose a method for the benchmark of embedded computing platforms that are capable of computer vision tasks onboard UAV. Currently, there is no unified benchmark of energy efficiency for embedded computing platforms.
- Since real-life test flights require much effort and there are no publicly available datasets for map-based localization algorithm benchmark, a new dataset is required to perform algorithm development and comparison against other algorithms.

Chapter 2

RESEARCH METHODOLOGY

This chapter presents the proposed algorithms and functions to achieve precise visual localization based on the literature review performed in chapter 1. New datasets are presented in this section which was created for evaluation of the localization algorithm. The algorithms are implemented and evaluated in chapter 3 using the new datasets. The methodology is based on research published in papers [A1], [A2], [A3], [A4], and [A5].

2.1 Energy Efficiency Of Computing Platforms

This section describes the method that is used to measure the energy efficiency of computing platforms. A typical computer vision algorithm, Sobel filter, is performed on the target platform while execution time and the energy consumption is measured in high frequency (1 kHz) to measure the exact amount of energy used to perform the calculations.

2.1.1 Parallel implementation of the Sobel filter algorithm

Parallel programming will be used to employ all of the 16 cores of the *Parallella* platform, a more detailed description of the *Parallella* and other computing platforms can be found in section 1.4. The software can be developed eSDK (Epiphany SDK, developed by Adapteva[101]) or OpenCL implementation for the Epiphany processor by the Brown Deer Technology (OpenCL, part of COPRTHR framework[102]). Due to

(a) 480p Input (b) 480p Vertically filtered (c) 480p Horizontally filtered

Figure 2.1: Input and output images of the *Sobel* filter.

better performance, the algorithm was implemented using eSDK framework. The software for Mali GPU was developed using OpenCL framework since it is the only framework supported by the Mali GPU for general purpose computing. A data parallelization technique was chosen for the implementation which is described in OpenCL introduction [103] - the input data is divided into even sub-arrays, the computational core count selects the sub-array count in the accelerating hardware.

An evaluation of the two frameworks and platforms was carried out by implementing *Sobel filter* [104] algorithm. Two convolutions of the input image was calculated using frame

$$I_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * T$$

for vertical image convolution and

$$I_h = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * T$$

for horizontal image convolution [104]. In the convolution frame equations '$*$' symbol notes convolution operation with 2D pixel array T. T denotes a input image as a set of pixel intensity values in a two dimensional matrix (a grayscale image). Input image example is available in fig. 2.1a, also output horizontal and vertical images are available in fig.

2.1 b and c respectfully. The convolution will be calculated for 100 iterations using the same image and the execution time (in seconds) for each frame will be measured. For a more real-life evaluation and in-depth execution analysis, the single iteration will be divided into three stages:

Write. Time taken to write the image to a shared memory buffer.
Exec. Time taken for the parallel hardware to complete calculations.
Read. Time taken to read the results from shared memory.

Measuring the execution time will allow calculating possible execution framerate (FPS) for an image stream. Measuring energy consumption in millijoules (mJ) will allow comparing the energy efficiency of each device. Few different image sizes were selected to see what framerates are feasible with each platform (see fig.2.1). The performance is compared against a single core of a CPU, which would be a typical implementation of the *Sobel* convolution filter without any additional hardware acceleration. Algorithm 3 shows an overview of the steps that are performed while measuring a single calculation of Sobel filter on a single image. The implemented algorithm of the Sobel filter was intended to be able to process an image of any given size. Keeping that in mind and the fact that Epiphany co-processor's single core has only 32 kB of available local memory, the use of external shared DDR memory block was implemented. The program writes all image into a shared memory block, then executes each of the parallel computing cores.

2.1.2 Measuring The Power Consumption

The energy consumption of computing platforms will be analyzed during the execution of the implemented *Sobel* filter algorithm. Current / voltage sensor INA219 [1] is used to measure power at ~1 kHz sampling rate. The sensor setup (see fig. 2.2) was made to avoid power measurement influence possible by the additional electronics. The data is captured using MCU, which transmits measurements to a laptop where they

[1] Texas Instruments INA219 Current sensor, data sheet available online: `http://www.ti.com/lit/ds/sbos448f/sbos448f.pdf`

Algorithm 3 Procedure for measurement of energy consumption

Inputs: Image T
Outputs: Vertically filtered image I_v, horizontally filtered image I_h, write time t_w, execution time t_e, read time t_r, consumed energy E
function MEASURESOBELFILTER(T)
$startPowerMeasurementCapture()$ ▷ Capture energy consumption in a separate thread
$t_w^{start} = timeNow()$ ▷ Record current time
$upload(T)$ ▷ Upload image to the accelerator memory
$t_w = timeNow() - t_w^{start}$ ▷ Measure write time
$t_e^{start} = timeNow()$
$I_v, I_h = performSobel(T)$ ▷ Calculate Sobel filter on the accelerator
$t_e = timeNow() - t_e^{start}$ ▷ Measure execution time
$t_r^{start} = timeNow()$ ▷ Record current time
$download(T)$ ▷ Download image from the accelerator memory
$t_r = timeNow() - t_r^{start}$ ▷ Measure read time
$endPowerMeasurementCapture()$
$E = retrieveEnergy()$ ▷ Integrate the power measurements
return $< I_v, I_h, t_w, t_e, t_r, E >$

are recorded. A laptop uses a wired LAN connection to receive messages from the platform performing calculations to capture the beginning and end of the computation process. Energy equation $E = \int_0^t P(t)\mathrm{d}t$ derived from power equation in [105] is used to calculate the amount of energy used to process each frame. $P(t)$ is the measured power in Watts on time t. To check the reliability of the measured data Shapiro–Wilk test [106] will be used on calculated energy values. The statistical p-value threshold of 0.05 will be used to check the *null* hypotheses that measured data is of normal distribution. To provide additional confidence on measured data Student's t-test will be performed to prove that mean values from both experiments has significant differences. Figure 2.2 shows the setup of current / voltage sensor for the experiment. The platform

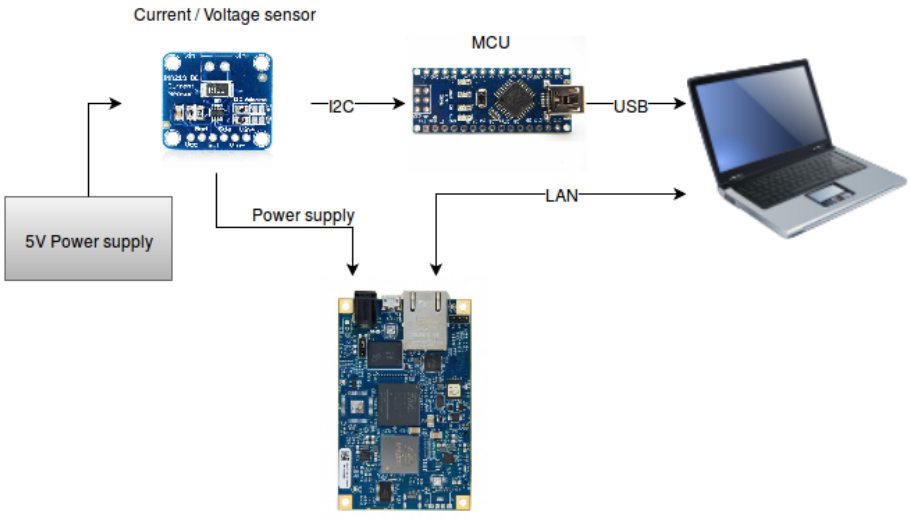


Figure 2.2: Setup of power measurement sensor.

being measured is connected to the power supply over the sensor. MCU reads sensor measurements over the I2C interface and is connected to a computer that is recording the launch of the algorithm and energy consumption measurements.

2.2 Datasets of Aerial Imagery for Benchmark of Visual Localization Algorithms

Datasets of aerial imagery from low altitude with location information are not common. As described in section 1.5, researchers tend to create their datasets to develop the system and measure the execution performance. Two datasets were created:

1. Test flight dataset using an actual UAV,
2. Dataset from a simulated environment.

The datasets will be used to perform experiments on the proposed localization method. A test flight using fixed-wing UAV was performed

Figure 2.3: Spartan UAV used in test flights.

to collect real-life data. Performing a test flight is an expensive and time-consuming process, to improve the experiment credibility, additional data was collected using a simulation environment. The simulated dataset takes into account different aspects of visual localization so that it could be used for the benchmark of other localization systems. The simulated dataset is made public, and to the best of knowledge, it is by far the largest publicly available dataset of aerial imagery for visual localization benchmark.

2.2.1 Data From Test Flight

A fixed-wing Spartan UAV (see figure 2.3) was used to collect aerial imagery and sensor data during ~1 km flight. The Spartan UAV manufactured in Lithuania by the Space Science and Technology institute [2]. Basler acA640-120uc industrial camera with global shutter was used to collect aerial imagery alongside with other sensor data provided from the

[2]UAVs of Space Science and Technology Institute: `http://space-lt.eu/en/technologies/unmanned-aircraft-vehiclesuav/`

(a) (b) (c)

(d) (e) (f)

Figure 2.4: Sample images from the dataset captured during the test flight.

UAV flight controller. Images were recorded in 640x480 resolution at 90 FPS. Data was recorded using MPEG2-TS video format, and sensor data were recorded as metadata alongside the video stream in KLV format according to MISB 0601 metadata standard [3]. The video playback allows data to be read with the same timing as it was recorded on UAV. Sample images from the video file is provided in figure 2.4. Metadata includes GPS coordinates in WGS84 format, UAV attitude quaternion as measured by flight controller IMU, camera parameters (horizontal and verticals fields of view and focal length), location of current frame corners and image center in GPS coordinates.

[3]MISB ST0601 Metadata Standard: `http://www.gwg.nga.mil/misb/docs/standards/ST0601.6.pdf`

2.2.2 Simulated Dataset

Section 1.5 presents currently available datasets, and the problem of missing dataset researchers are facing when developing map-based localization systems. To define requirements for the new dataset an overview of common map-based localization system components and their pitfalls is needed.

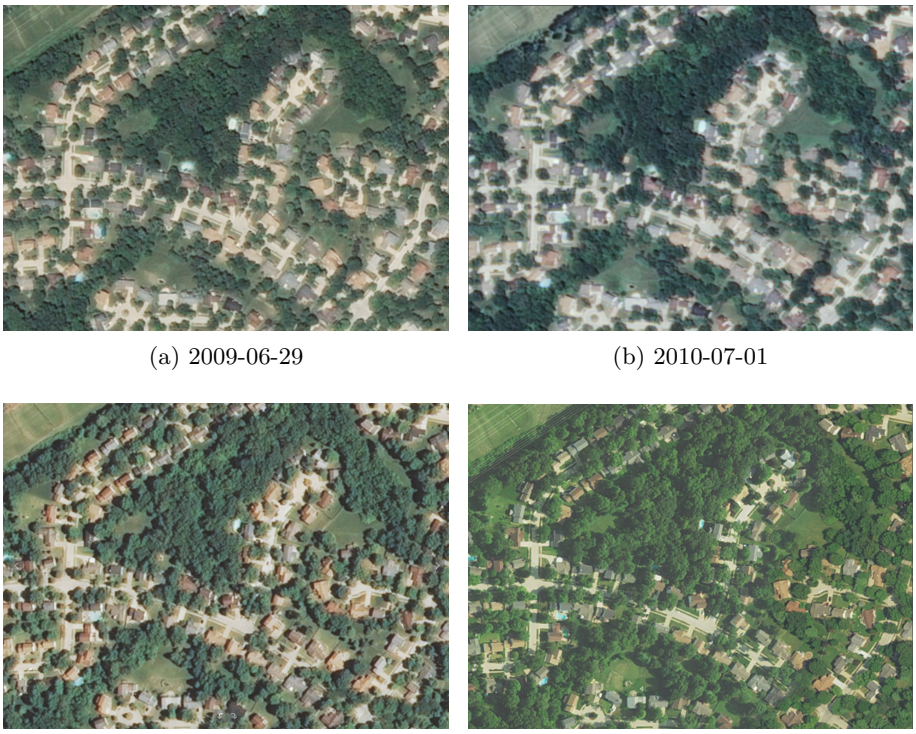
Map-based localization systems use image registration algorithms to match image from an onboard camera with a map. These algorithms are prone to errors when the maps are not up-to-date. Another common component is Visual odometry, which provides motion information between the images. Usually, Visual odometry relies on a high framerate video feed to accurately calculate vehicle motion in real-time. For the dataset to be suitable for Visual odometry, authors of popular Visual odometry algorithms suggest, that the image sequences should be of at least 50 frames per second [2, 3]. The rapid motion may be captured more precisely using higher framerates, but advanced real-time systems rarely exceed 60 FPS [107], so recording more than 60 images per second is not practical. From the researches reviewed in section 1.5, it is clear that researchers tend to use imagery from urban environments. Since urban environments contain buildings, roads, and other artifacts, such imagery provides a rich texture that computer vision algorithms can successfully process and extract information. An additional environment, such as forests, would provide useful information about system accuracy when the imagery is not rich with texture and features. Different trajectories at different altitudes should also provide additional validation of system behavior in different flight scenarios. The creation of a dataset is very laborious, so researchers might be using small datasets to evaluate localization accuracy, which might not expose the long-term issues of localization drift, so the distance should also be considered as a critical requirement for the dataset. Advanced localization systems are using loop-closure algorithms to improve localization accuracy and reduce drift over time. These algorithms detect a point in space that was already passed during the localization and optimizes the flight tra-

jectory to match the passed point. It would be beneficial for a system to test this feature using the new dataset.

One of the key challenge defined in robotic mapping problem [66] that is also challenging in UAV localization problem: environments change over time, and maps currently available from satellite imagery might be at least few years behind. Some changes may be slow, such as new buildings or roads, but some changes are visible very quickly - moving cars, effects of seasons on trees, etc. A previous work of road feature mapping [77] includes tests of the matching algorithm using maps of different dates. Different maps provided information on the robustness of the algorithm, and it is also important to be able to test algorithm robustness to changes in the environment. After the careful review of localization systems, the key requirements for the dataset were defined as follows:

1. flights should be performed in multiple environments: urban and forest,
2. image capture framerate should be at least 50 frames per second,
3. maps of different dates should be available to test algorithm robustness to changes in the imagery,
4. different trajectories should be available,
5. flight distances should be 1 kilometer or longer.

A number of different flight scenarios are set up to satisfy the defined requirements. A single flight scenario is a combination of a map, trajectory, and altitude. Table 2.1 shows different trajectories, maps, and attitudes that are used while performing simulated flights. Three trajectories were chosen — straight line, rectangular, and circular. Previews of the trajectories are given in figure 2.6. Rectangular and circular trajectories have a returning point, so these scenarios can be used to test loop-closure algorithms. Each flight scenario is given an abbreviation, e.g., FL-200, where the first character stands for map (F - forest, U - urban), the second character stands for trajectory (L - straight line, R - rectangular, C - circular), and the number stands for the altitude in meters. A total of 12 flight scenarios is performed.



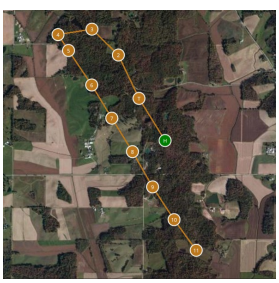
(a) 2009-06-29 (b) 2010-07-01

(c) 2012-06-22 (d) 2014-07-20

Figure 2.5: Sample patches of the same location in urban map created at different dates. Each figure is annotated with a creation date.

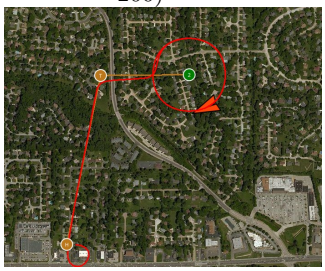
The new dataset was created using orthophoto maps from the United States Geological Survey's (abbr. USGS) National Agriculture Imagery Program (abbr. NAIP) database [108] and Gazebo Robotics simulation environment[4]. Maps were used as a texture for the simulator ground plane. Flights over the urban environment used the map created at the year 2009 and flights over the forest environment used the map created in 2008. Sample patches of maps of the same location are given in figure 2.5. PX4 flight controller software in software in the loop (abbr. SITL) mode was used to control the aircraft model in

[4]Gazebo Simulator home page: `http://gazebosim.org/`



(a) Straight line trajectory (FL-200)

(b) Rectangular trajectory (FR-200)



(c) Circular trajectory (FC-200)

Figure 2.6: Flight plans for different scenarios.

the simulated environment. A software service was developed which captured images from the simulator as soon as it reached target altitude and location. Image metadata was collected using the CSV file format. Flight trajectories for each scenario were planned manually using QGroundControl[5] software. The flight plans were also saved alongside the images. Previews of planned trajectories from the ground control software are shown in figure 2.6. The flight plans include plane takeoff, but this part is not included in the dataset, and image sequences begin when the plane reaches its starting point and altitude.

[5]Qgroundcontrol: Ground control station for small air land water autonomous unmanned systems, website: `http://qgroundcontrol.com/`

Table 2.1: Different trajectories, maps and altitudes used to perform 12 flight scenarios in the dataset

Trajectory	**Map**	**Altitude, meters**
Line (L)	Forest (F)	200
Rectangular (R)	Urban (U)	300
Circular (C)		

Table 2.2: Number of images and distances of each flight.

Scenario	**Image count**	**Flight distance, meters**
FL-200	11837	2936
FL-300	9020	2736
FR-200	16056	4748
FR-300	15307	4541
FC-200	3601	1065
FC-300	3450	1028
UL-200	8735	2592
UL-300	8416	2496
UR-200	14997	4418
UR-300	15361	4542
UC-200	3422	1016
UC-300	3272	970
Total	113474	33088

Dataset consists of 12 archives for each of the planned scenarios. Each archive contains *jpeg* encoded images, flight plans from QGroundControl software and a CSV file containing image metadata. Table 2.3 presents a detailed list and explanation of fields available in the CSV file. The dataset contains a total of 113474 images, alongside with geographical coordinates and UAV pose at the moment of image capture. The total distance covered by flight is over 33 kilometers; the statistics of each flight is available in table 2.2. All images are captured at

640x480 resolution. Images are rectified and do not contain camera distortion, so camera calibration is not required. The captured images contain identical pixel intensity values to the map, so to avoid perfect matches between the map and an image, Gaussian noise (mean: 0.0, standard deviation: 0.02) is added to each image. Sample images from the dataset are shown in figure 2.8.

The metadata fields are given in the table 2.3 contains information on the aircraft attitude during the capture of an image. Fields $< ImuX, ImuY, ImuZ, ImuW >$ and $< OrientationX, OrientationY, OrientationZ, OrientationW >$ represents an orientation in space using a quaternion in the following notations:

$$q = W + X \cdot i + Y \cdot j + Z \cdot k, \tag{2.1}$$

where:

- i, j, k are the imaginary numbers used in the quaternion,
- X, Y, Z, W are the scalar values of the rotation quaternion.

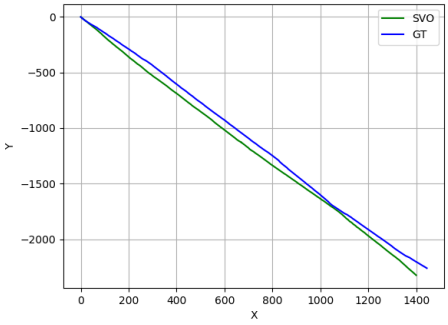
The images were processed using SVO visual odometry algorithm to provide a baseline for comparison, and the resulting coordinates are available in the metadata file. Visual odometry and SVO algorithm are presented in details in section 1.3.2. The recovered trajectories using SVO algorithm is shown in figure 2.7. From the recovered trajectories we can see that while SVO algorithm works pretty accurate on straight line trajectory, it fails to recover rectangular and circular trajectories. The problem is that SVO fails to accurately calculate trajectory while the plane is performing turn maneuver, which causes pure rotational movement which is known to cause problems for the SVO algorithm.

2.3 Particle Filter Localization For Low-Altitude UAV Flights

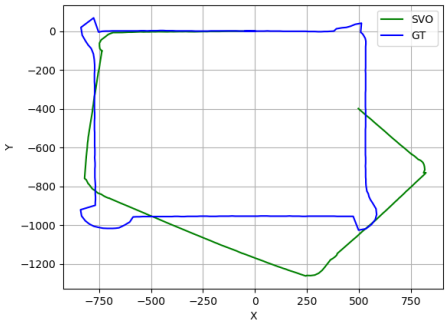
Particle filter localization (or Monte Carlo localization) is a recursive Bayes filter that estimates the posterior distribution of robot poses con-

Table 2.3: Metadata fields available for each image in the dataset.

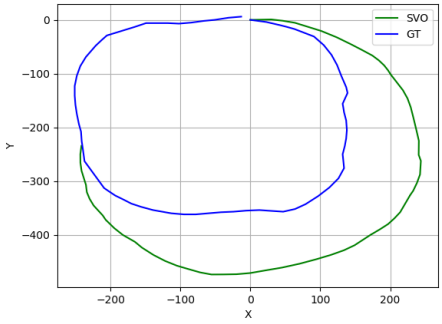
Field	**Description**	**Unit**
Filename	Name of the image file	-
Latitude	Latitude of the aircraft position	degrees
Longitude	Longitude of the aircraft position	degrees
RelativeAltitude	Altitude from the earth's surface	meters
ImuX	IMU quaternion, X component	-
ImuY	IMU quaternion, Y component	-
ImuZ	IMU quaternion, Z component	-
ImuW	IMU quaternion, W component	-
PoseX	GT position relative to flight start location, X component	meters
PoseY	GT position, Y component	meters
PoseZ	GT position, Z component	meters
OrientationX	GT orientation, relative to world coordinate frame, X component	-
OrientationY	GT orientation, Y component	-
OrientationZ	GT orientation, Z component	-
OrientationW	GT orientation, W component	-
MapX	Image center location on map, X axis	pixels
MapY	Image center location on map, Y axis	pixels
SvoX	Pose estimate, calculated using SVO algorithm in Euclidean space, X component	meters
SvoY	Pose estimate, Y component	meters
SvoZ	Pose estimate, Z component	meters



(a) Scenario FL-200.



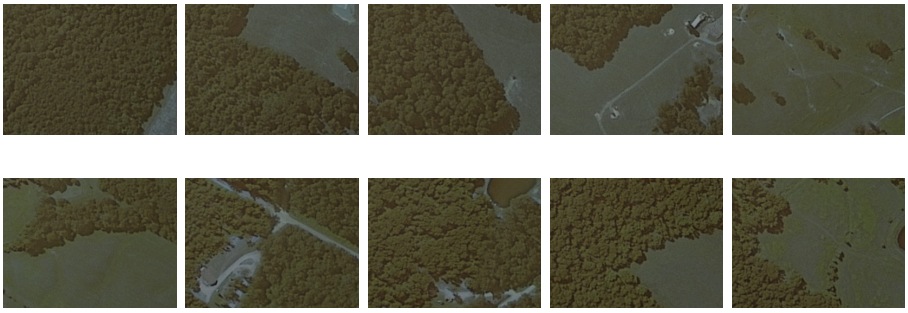
(b) Scenario UR-200



(c) Scenario UC-300

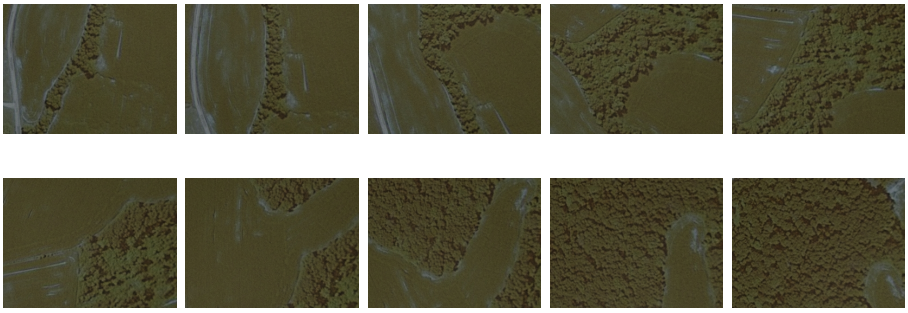
Figure 2.7: Different flight trajectories recovered using the SVO algorithm and compared against ground truth (GT).

ditioned on sensor data [4]. The particle filter is known to provide better performance compared to other nonlinear approaches (e.g., the EKF) since it can provide optimal estimation in nonlinear non-Gaussian state-space models [109]. The problem is that robot pose is changing over time and the aircraft pose probability distribution must be evaluated iteratively. It is possible that aerial imagery might be obstructed or the images might not contain any texture, in case of flying over forests or lakes. In that case, only dead-reckoning may be used to estimate the pose until enough data is collected to provide a precise location of the aircraft. The Particle filter is known to be robust to significant noise in



(a) Forest map, 200 meters altitude, Straight line trajectory (FL-200)

(b) Urban map, 200 meters altitude, Rectangular trajectory (FR-200)



(c) Forest map, 200 meters altitude, Circular trajectory (FC-200)

Figure 2.8: Sample image sequences from the dataset.

the sensor measurements and may recover position even if there is not enough texture in the imagery for the VO to reconstruct motion.

Particle probabilities are obtained by calculating image similarity R with formulas presented in section 2.3.6 and mapping them to an interval of $[0; 1]$. To map the image similarity values to particle probability, a conversion function from section 2.3.7 is used:

$$s = F(R) \tag{2.2}$$

Then we can extend position belief presented in equation 1.1 for the UAV position belief in a Particle filter as

$$bel(z_t) = p(z_t | z_{0:t-1}, m_{1:t-1}, s_{1:t-1}) \tag{2.3}$$

Figure 2.9 shows the diagram of the Particle filter localization algorithm. The algorithm is divided into seven steps:

1. **Particles.** During the initialization, this step initializes the algorithm by generating some initial particles. On other iterations, this step passes the current particles to the next step.
2. **Particle sampling.** This step takes current particles and samples a particle according to its probability and passes to particle propagation.
3. **Odometry.** This step takes raw image data from the camera sensor and calculates the movement of the UAV from the imagery.
4. **Motion model.** This step takes motion measured by the odometry and adds additional noise for each of the drawn particles from the sampling step.
5. **Particle propagation.** This step predicts the next position of a particle by adding motion data using the motion model to the drawn particles from the sampling step.
6. **Particle map matching.** This step takes a patch of the map according to a predicted particle's location and calculate image similarity value using the patch and latest camera image.
7. **True pose estimation.** This step takes all the latest particles and estimates the current position of the aircraft.

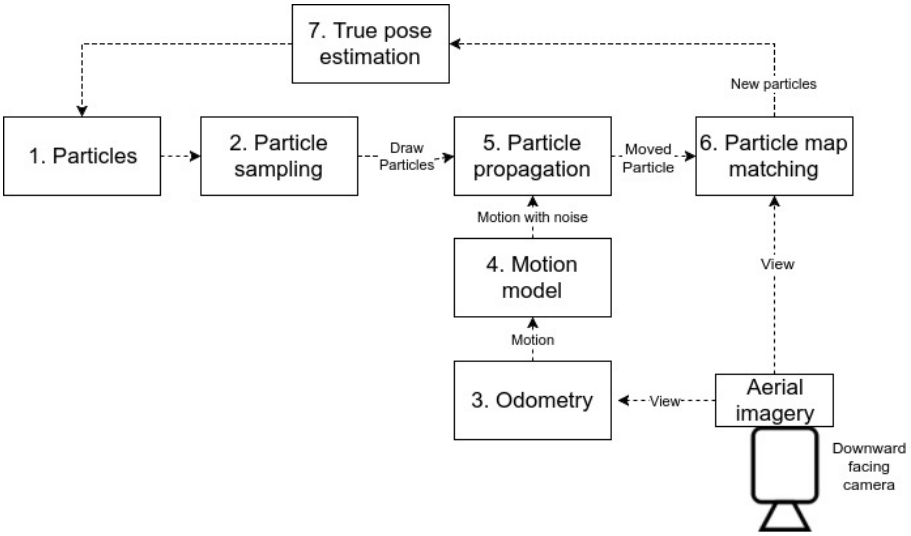


Figure 2.9: Particle filtering algorithm including VO

2.3.1 Particles

Particle is a hypothesis for the aircraft's true position relatively to a given map. Several particles are maintained in the algorithm to evaluate more than one possible location of the aircraft and propagate the possibilities over time. Each particle is assigned a probability value s_t on time t using a selected conversion function $F(x)$

$$s_t = F(R_t), \tag{2.4}$$

it is calculated using UAV image similarity value R_t with the map image on the particle location. Initial particle probability value is assigned $s_0 = 1$. Particles are also assigned weight value which is used during sampling. The particle weight b_i is calculated by normalizing all probabilities

$$b_{i,t} = \frac{s_{i,t}}{\sum_{j=0}^{n} s_{t,j}}, \tag{2.5}$$

where

- n is the particle count,

- i is single particle index,
- t is time of current iteration.

2.3.2 Particle sampling

Sampling is the stage of the Particle filter when particles are resampled according to their probability. Each iteration resamples particles to find the most plausible UAV location over time. Few sampling techniques that are commonly used for Particle filtering applications:

- Rejection Sampling [110],
- Importance Sampling [111],
- KLD-Sampling [112].

Rejection Sampling is a practical method that is easy to implement and to deploy. The principle of rejection sampling is to evaluate the survival of a randomly selected particle to survive with a probability equal to its belief value. Particles with higher belief are more likely to be resampled, though bad particles are also able to survive with less probability. Importance sampling was introduced to deal with higher uncertainty in the measurements. This sampling technique uses weighted probabilities to resample the particles. By generating a random uniform number in the interval [0; 1], we select a particle which weights the nearest patch. Patch for a particle g_i is calculated from the particle weight

$$g_t^{(j)} = 1 - \sum_{j=i}^{n} b_t^{(j)}. \tag{2.6}$$

The higher particle belief is, the larger weight it gets and a wider patch it gets in the interval of [0; 1] and value is more likely to be resampled. Figure 2.10 presents an example of sampling with 5 particles.

This way an important value may be resampled more times than in rejection sampling, due to the usage of the particle weight. Importance sampling is the most common method used in Monte Carlo localization. KLD-sampling is an extension of the importance of sampling. It uses

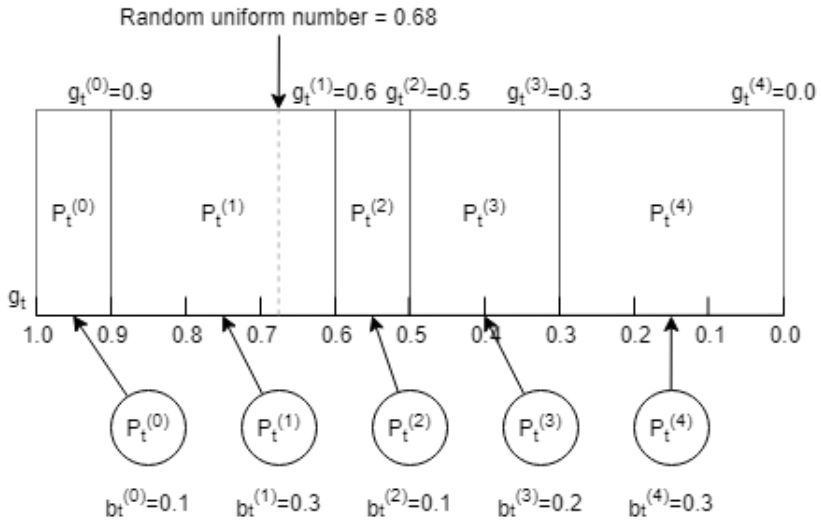


Figure 2.10: A preview of the particle sampling according to it's weight. This example contains 5 particles with given weight values. If a generated random uniform number is within the patch assigned to a specific particle, that particle will be sampled, in case of generated value = 0.68, particle $P_t^{(1)}$ would be drawn.

Kullback-Leibler divergence [113] to dynamically calculate a required number of particles to calculate the accurate position. The Particle filter with variable particle count is referred to as the Adaptive Particle filter.

2.3.3 Motion model

Motion model is used for dead-reckoning of the UAV pose from odometry data (visual and movement speed sensors). The UAV pose may be described using six parameters $\langle x, y, z, \theta_{roll}, \theta_{pitch}, \theta_{yaw} \rangle$ in space relative to the known environment. Parameter z is equivalent to altitude, which can be measured using sensors (barometer, laser) with relatively high precision and the altitude is only required for image scaling so that it will be ignored in the model. Roll and pitch angles are required for the

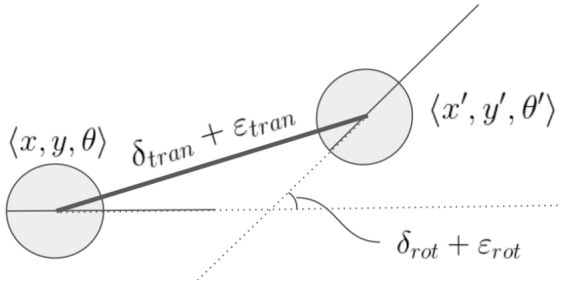


Figure 2.11: Simplified planar motion model for UAV

calculation of camera relative elevation angle and image center on the map. Those parameters can be ignored by using camera gimbal hardware in the case if the gimbal system is configured to look downward. The search space thus is narrowed down to pseudo-planar movement using only three parameters (see figure 2.11), where

- aircraft pose $z_t = \langle x, y, \theta_{yaw} \rangle$, where θ_{yaw} is an angle, equivalent to UAV heading angle and
- z_t is the UAV pose on time t in orthophoto map plane.

The pose update can be calculated after receiving new sensor data using these equations [16]:

$$x' = x + \alpha_1 \hat{\delta}_{tran} cos(\theta_{yaw} + \alpha_3 \hat{\delta}_{rot}), \tag{2.7}$$

$$y' = y + \alpha_2 \hat{\delta}_{tran} sin(\theta_{yaw} + \alpha_3 \hat{\delta}_{rot}), \tag{2.8}$$

$$\theta'_{yaw} = \theta_{yaw} + \alpha_3 \hat{\delta}_{rot}, \tag{2.9}$$

where:

- x', y' and θ'_{yaw} are posterior UAV location relative to the orthophoto map
  α_n - measurement noise coefficient
- $\hat{\delta}_{tran}$ - transitional (movement speed) measurement with measurement noise ϵ_{tran}, obtained:

$$\hat{\delta}_{tran} = \delta_{tran} + sample\_normal(\epsilon_{tran}) \tag{2.10}$$

- $\hat{\delta}_{rot}$ - rotational (heading angle) measurement with measurement noise ϵ_{rot}, obtained:

$$\hat{\delta}_{rot} = \delta_{rot} + sample\_normal(\epsilon_{rot}) \tag{2.11}$$

  - *sample_normal* is an algorithm that samples normal distribution according to known measurement error. Algorithm implementations can be found in [16].

2.3.4 Particle propagation

This step of the Particle filter uses sensor data, VO and motion model to propagate the particles after re-sampling. Propagation moves the re-sampled particles into their current locations by adding relative movement that was measured since last Particle filter iteration with additional noise from the motion model discussed in section 2.3.3. An additional technique of sensor fusion can be used to fuse data from IMU and VO to improve VO accuracy and in case of VO failure, using IMU data to propagate can provide dead-reckoning of the UAV pose.

2.3.5 True Pose Estimation

True location of the UAV is calculated by recursively estimating particle probability values as described in [16]. The particle with highest probability can be considered to be the true UAV pose P_t^{final} for current time t. Weighted sum of all particle locations can also be used as final pose of the UAV. Particle weight is calculated using equation 2.5 and is used to sum all particle locations:

$$P_t^{final} =< \sum_{i=0}^{n} (b_t^{(i)} * x_t^{(i)}), \sum_{i=0}^{n} (b_t^{(i)} * y_t^{(i)}) >, \tag{2.12}$$

where $x_t^{(i)}$ and $y_t^{(i)}$ are the x and y coordinate values of ith particle on time t.

2.3.6 Similarity Between Two Images

Similarity between two images, which are sequences of pixel intensity values, is a measure that quantifies the dependency between the images. A measure R is considered a similarity measure if it increases in value if the dependency of two sequences increases. Image similarity is calculated to evaluate a hypothesized UAV location probability to be the true location. A correlation value can be calculated between a map patch of an according UAV location and the imagery retrieved from camera sensor. Normalized Sum of Squared Differences (SSD), also known as normalized L_2 norm, is a very popular measure to calculate image similarity:

$$R = 1 - \frac{\sum_{x=0}^{w}\sum_{y=0}^{h}(T(x,y) - I(x,y))^2}{\sqrt{\sum_{x=0}^{w}\sum_{y=0}^{h}T(x,y)^2 \cdot \sum_{x=0}^{w}\sum_{y=0}^{h}I(x,y)^2}}, \tag{2.13}$$

note that subtraction from 1 is added, to satisfy similarity condition No. 1, otherwise it would be considered dissimilarity measure as value would decrease with increasing dependency. Another popular image similarity measure is cross-correlation (CC):

$$R = \frac{\sum_{x=0}^{w}\sum_{y=0}^{h}(T(x,y) \cdot I(x,y))}{\sqrt{\sum_{x=0}^{w}\sum_{y=0}^{h}T(x,y)^2 \cdot \sum_{x=0}^{w}\sum_{y=0}^{h}I(x,y)^2}} \tag{2.14}$$

An extension to the CC similarity measure is normalized correlation coefficient, also known as Pearson Correlation [114] or normalized cross-correlation (NCC) [115], which subtract mean of the image, thus making the measure robust to changes in contrast and exposure:

$$R = \frac{\sum_{x=0}^{w}\sum_{y=0}^{h}(T'(x,y) \cdot I'(x,y))}{\sqrt{\sum_{x=0}^{w}\sum_{y=0}^{h}T'(x,y)^2 \cdot \sum_{x=0}^{w}\sum_{y=0}^{h}I'(x,y)^2}}, \tag{2.15}$$

where

- $T'(x,y) = T(x,y) - \frac{\sum_{x'=0}^{w}\sum_{y'=0}^{h}T(x',y')}{w \cdot h}$
- $I'(x,y) = I(x,y) - \frac{\sum_{x'=0}^{w}\sum_{y'=0}^{h}I(x',y')}{w \cdot h}$

- w, h are the image dimensions (width and height).

A two stage approach by A. Goshtasby et al. [116] was proposed to increase computation speed of the correlation coefficient. The book *Image Registration* by Goshtasby [117] provides an evaluation of these and other measures. In the evaluation, normalized SSD was the fastest measure to calculate (7.25 milliseconds for an image pair) compared to any other measure in the evaluation. On the other, Pearson Correlation, which is slower (12.13 milliseconds for an image pair) was one the best measures that was able to match images when the pair contained images of different scene lighting and exposure.

2.3.7 Image Similarity To Particle Probability Conversion Functions

This section describes $F(x)$ used for image similarity to particle probability conversion. Image similarity is calculated using Pearson Correlation Coefficient [114] between map and an aerial image obtained from the UAV camera. Pearson correlation coefficient is chosen since it was the best image similarity measure according to the comparison carried out in section 3.2. Pearson Correlation Coefficient provides an arbitrary similarity value that is proportional to image similarity in range of $[-1; 1]$, where one means that images are proportional and -1 means that one image is entirely anti-correlated. The similarity cannot be used to sample the Particle set, which is the basic idea of the Particle filtering algorithm. The similarity value must be recalculated to a probability distribution, which can be sampled.

The easiest approach to calculate particle probability is to convert negative image similarity values to positive. The range of Pearson Correlation Coefficient is fixed in range of[-1; 1], the following formula can be used:

$$F(x) = \frac{x + 1}{2} \tag{2.16}$$

Equation 2.16 provides high probabilities at poor similarity values, e.g. if $x = -0.1$, then $F(x) = 0.45$. Fig. 2.12a shows the visual

representation of such conversion. This means that particles with poor image-map similarity survives with high probability.

Softmax is a popular function used to convert arbitrary output values to categorical probability distributions in neural networks and machine learning [118]. Softmax conversion function (see figure 2.12b) can be implemented by the following equation :

$$F(x) = e^x \tag{2.17}$$

Due to its wide use amongst machine learning applications, it is going to be used as a baseline in the experimental sections of this paper.

The main idea for the conversion function is to achieve robust localization with as little particles as possible. It should assign low probabilities (non-zero) for the negative similarities and boost positive similarities to improve their survivability during sampling. Two functions are proposed to deal with the conversion in the described fashion. The first function uses linear rectification with a single parameter d describing what probability value is assigned at $x = 0$ using positive parameter value and where the function starts rising on the X-axis using negative parameter values (see figure 2.12c):

$$F(x,d) = \begin{cases} 0, & \text{if } d < 0 \text{ and } x \leq |d| \\ (1+|d|)(x-|d|) + d^2, & \text{if } d < 0 \text{ and } x > |d| \\ d(1+x), & \text{if } d \geq 0 \text{ and } x \leq 0 \\ x(1-d) + d, & \text{if } d \geq 0 \text{ and } x > 0 \end{cases} \tag{2.18}$$

Previous work by A. Nakhmani et al. [119] has used a rectified Pearson correlation coefficient for a Particle filter based visual tracking algorithm, this conversion function would be equivalent to proposed rectified function with parameter value $d = 0$. The second function was developed from generalized logistic function [120]:

$$l(x) = \frac{A}{(1 + \delta e^{-kx})^{\frac{1}{v}}} \tag{2.19}$$

and by applying fixed coefficients: $A = 1$, $\delta = 1$, $k = 5$, the logistic function becomes:

$$L(x, v) = \frac{1}{(1 + e^{-5x})^{\frac{1}{v}}}. \tag{2.20}$$

Parameter v is going to be used as a hyper-parameter to control the shape of the curve. To achieve a curve in range of [0; 1] at input range of [-1; 1], a division is added, so the final conversion equation is:

$$F(x, v) = \frac{L(x, v)}{L(1, v)}. \tag{2.21}$$

See figure 2.12d for the curves with different v values used in the research. This function has a more practical form without different if-cases. It also contains a single hyper-parameter, that is used to control the function curvature. Each of these conversion functions is implemented, and their impact using different hyper-parameters on localization accuracy, speed, and robustness is analyzed.

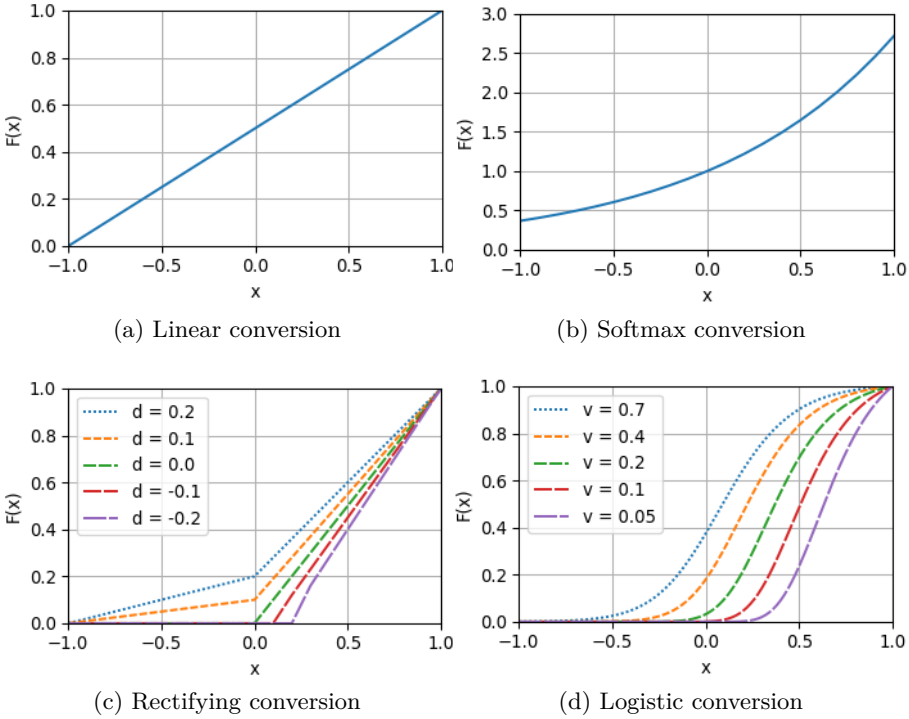
This section describes the methodology and experimental setups used for performance measurements of Particle filter localization.

2.3.8 DPC-PFL: Discriminatory Pearson Correlation based Particle Filter Localization Algorithm

This section proposes a variant of adaptive Particle filter localization algorithm based on KLD sampling and Pearson correlation coefficient similarity measure to solve GPS-Denied localization on an aircraft flying in low altitude. The main novelty of this algorithm is the use of image similarity to particle probability conversion function, which is used to convert Pearson correlation values to particle probability with lower-probability values for low correlation and higher-probability value for increasing correlation values. Since the conversion function is used to reduce the survivability of particles with low correlation values and increase chances of survival of particle with better correlation values, the function is named particle discrimination function and the algorithm is

Algorithm 4 Proposed Particle Filter Localization Algorithm - DPC-PFL

Inputs: Set of particles S_{t-1} obtained from previous iteration, Latest camera image T, Conversion function parameter v (if applicable)
Outputs: Proposed location $< X, Y >$ of the UAV in map coordinates
function FILTERPARTICLES(S_{t-1}, T, v)
$H = 0, k = 1, S_t = \emptyset, i = 0$
do
$P^{(i)} \sim S_{t-1}$ ▷ Sample a particle from the particle set
$PropagateParticle(P^{(i)})$ ▷ Propagate particle using motion model equations 2.7, 2.8, and 2.9
$I = ExtractMapImage(P^{(i)})$ ▷ Extract map image corresponding to particle location
$r_t^{(i)} = CalcSimilarity(T, I)$ ▷ Calculate image similarity using NCC measure
$s_t^{(i)} = F(r_t^{(i)}, v)$ ▷ Convert image similarity value using logistic conversions function
$H = H + s_t^{(i)}$
$S_t = S_t \cup P^{(i)}$ ▷ Insert particle into the particle set
if $P^{(i)}$ falls into empty bin **then**
$bin =$ non-empty ▷ Mark bin as non-empty
$k = k + 1$ ▷ Increase marked bin counter
$i = i + 1$ ▷ Increase number of particles
while $i < \frac{1}{2\epsilon} Z^2_{k-1,1-\delta}$ ▷ Until K-L bound is reached
$n = i$ ▷ Save the number of particles
$x' = 0, y' = 0, \theta'_{yaw} = 0$
for i = 1 .. n **do**
$b_t^{(i)} = s_t^{(i)} / H$ ▷ Calculate particle weight
$x' = x' + (b_t^{(i)} * x_t^{(i)})$ ▷ Calculate weighted sum of particle coordinates
$y' = y' + (b_t^{(i)} * y_t^{(i)})$ ▷ $x_t^{(i)}, y_t^{(i)}$ are ith particle's coordinates in map
$\theta'_{yaw} = \theta'_{yaw} + (b_t^{(i)} * \theta_t^{(i)})$ ▷ Final pose heading angle θ_{yaw}
return $< x', y', \theta'_{yaw} >$



(a) Linear conversion

(b) Softmax conversion

(c) Rectifying conversion

(d) Logistic conversion

Figure 2.12: Conversion functions outputs.

named Discriminatory Pearson Correlation based Particle Filer Localization (abbr. DPC-PFL).

Localization is achieved by sampling a set of particles using their probability distribution. Particles are hypothesized UAV locations on the map and are assigned a probability value that is proportional to the likeliness of being the true location. Particles are iteratively resampled using sampling technique which adapts sampled particle count n depending on Kullback-Leibler divergence (abbr. KLD sampling):

$$n = \frac{1}{2\epsilon} Z^2_{k-1,1-\delta} = \frac{k-1}{2\epsilon} \left\{ 1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta} \right\}^3, \quad (2.22)$$

where :

- $z_{1-\delta}$ is the upper $1-\delta$ quantile of the standard normal $N(0,1)$ distribution;
- ϵ is the upper bound of the approximated discrete distribution by the KL-divergence [121];
- k is the number of bins that are marked taken.

The technique has shown good results against other sampling techniques on simulated flight data — it provides the same localization accuracy, but dynamic particle count allows to decrease computational costs up to 1.7 times (see section 3.4 for details of KLD sampling algorithm evaluation against other algorithms). The particles can be sampled only if the probability distribution is known. The problem is how probability distribution could be calculated from image similarity. Particle filter uses a finite amount of particles; thus a probability mass function is going to be used as a discrete probability distribution. Algorithm 4 shows the detailed implementation of the Particle filter localization. The initial particle set S_0 is generated around the starting point within 300 meters radius, an initial probability is set to 1. During the first iteration, all particles have an equal likelihood to be sampled. KLD algorithm parameters ϵ and δ were chosen empirically and the values used were $\epsilon = 0.05$ and $\delta = 0.9$. The bins used in the algorithm is implemented by dividing map coordinates into a 2-D grid of 5 meters. If a particle falls into a bin (a grid square), it is marked as taken. The number of bins is used in the Kullback-Leibler divergence calculation to adapt the number of particles according to their distribution on the map. Image similarity value is calculated using NCC similarity measure, the measure was chosen since it provides highest similarity values to near ground-truth positions compared to SSD and CC similarity measures (see section 3.2 for details). The image similarity value R is converted to particle probability $b^{(i)}$ using one of the conversion functions $F(x)$ described in section 2.3.7. The recommended conversion function is the logistic function with parameter value 0.2 since this function achieved highest scoring results in experiments performed in section 3.5. In the case, if the given map is very recent, lower values of 0.1 or 0.05 can be used to increase algorithm ac-

curacy but reducing robustness to changes between aerial imagery and the map. The sampling algorithm can sample the calculated particle probability in the resampling stage. Sampled particles must be propagated given their motion since the last iteration. Visual odometry algorithm is used to calculate relative UAV movement over time and is added using a planar motion model presented in section 2.3.3.

2.4 Conclusions Of Chapter 3

The conclusions of chapter 3 can be outlined as follows:

- Two image similarity to particle probability conversion functions are proposed to map image similarity to particle probability and allow the trade-off between accuracy and execution speed.
- Modification of Particle filter based localization algorithm named DPC-PFL is proposed to solve the drift of Visual odometry, the most common method for low-altitude UAV localization in GPS-denied environments.
- New method is proposed for measuring the energy efficiency of a parallel computing platform using a common computer vision task - computing Sobel filter. The method includes measurements of energy consumption by the platform to accurately compare the energy efficiency of each processing iteration by the hardware.

Chapter 3

EXPERIMENTS AND RESULTS

This chapter presents experiments performed during the research and the results that were obtained. The plan of experiments is as follows:

1. Comparison of computing platforms. The goal of this experiment is to measure energy consumption and execution time of Sobel filter calculation on selected computing platforms, to identify the most efficient computing platform.
2. Comparison of image similarity metrics. The best-identified metric will be used in the following experiments.
3. Evaluation of UAV Heading Error Impact on the NCC Similarity Metric. This experiment is performed to measure how the error of the magnetometer sensor (used to measure heading direction) affects the image similarity value. The similarity value is calculated between an actual image from UAV and according to map patch with different angular errors.
4. Comparison of particle sampling techniques. Few selected sampling techniques will be evaluated to choose the best particle sampling technique for the case of UAV visual localization using Particle filter.
5. Evaluation of image similarity to probability conversion functions. The experiment is performed by running localizations on the simulated dataset using different image similarity to probability conversion functions. Position estimation accuracy, execution speed, and robustness to the difference in map imagery are evaluated using different conversion functions. The best performing conversion function will be used in the next experiments.

6. Comparison against Visual odometry. Measure the accuracy of the proposed localization algorithm DPC-PFL and evaluate whether the algorithm can reduce accumulating drift of VO on the test-flight dataset.
7. Comparison against Visual SLAM. Compare the DPC-PFL against state-of-the-art vSLAM algorithm - OSB-SLAM.

The experiments will be performed in the planned order. Results from each item of the plan will be presented in a separate section of this chapter. The results of this chapter were based on works published in papers [A1], [A2], [A3], [A4], and [A5].

3.1 Comparison Of Computing Platforms

This section presents comparison of three computing platforms fit for execution computer vision algorithms on-board UAV. The platforms are evaluated using methodology presented in section 2.1. Three platforms are evaluated (see section 1.4 for details on the platforms):

- Parallella platform with Epiphany 16 co-processor,
- Radxa Rock 2 platform with ARM Mali-T764 GPU,
- Airvision Core X1 platform with NVIDIA X1 Embedded GPU, based on NVIDIA Maxwell architecture.

3.1.1 Execution Speed on Different Platforms

The *Sobel* filter was implemented on both OpenCL and eSDK frameworks for the Epiphany processor, and a brief comparison of results was made. The main difference between these implementations is that eSDK required the manual implementation of input/output image buffers, while OpenCL framework does manage buffers by itself. Figure 3.1 shows that eSDK framework has done the same calculations at least 3 times faster. The reason might be that the OpenCL is poorly implemented for the Epiphany processor and has too much overhead. Further experiments will be done using only eSDK since it shows better results

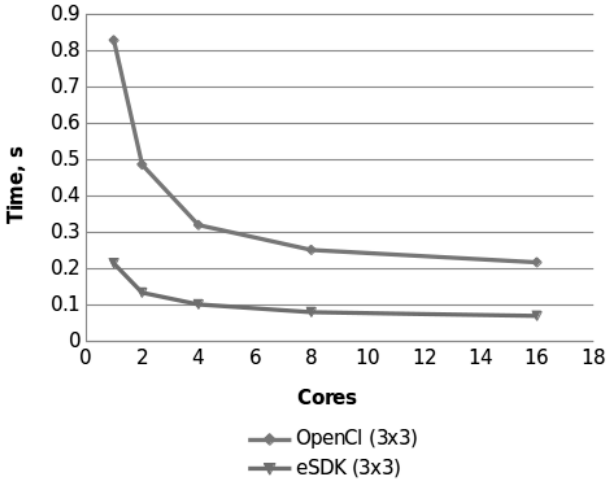


Figure 3.1: Time taken to process a single frame using OpenCL and eSDK on the *Parallella* platform.

without a doubt. The result values below are average values of 100 iterations processing the same image.

Figure 3.2 shows the execution time taken to process the images of different resolutions on ARM CPU, Epiphany Co-Processor, ARM MALI GPU and NVIDIA TX1 GPU. Images of following resolutions were processed:

- 640x480 (480p)
- 1280x720 (720p)
- 1920x1080 (1080p)

The results show that the execution time on Epiphany co-processor takes longer compared to CPU implementation, while Mali GPU operates approximately twice faster than the CPU. The NVIDIA Tegra X1 GPU performs 15–50 times faster compared to CPU implementation, the values are displayed on a different scale since they are barely visible on the scale on the left side.

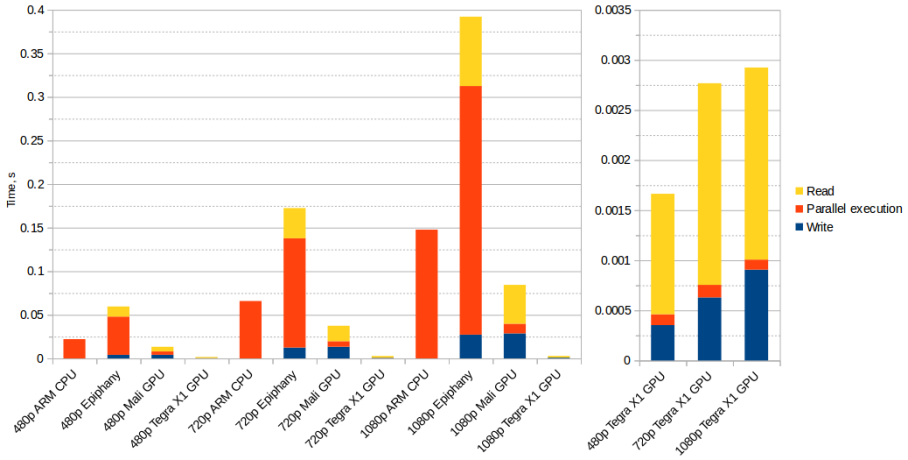
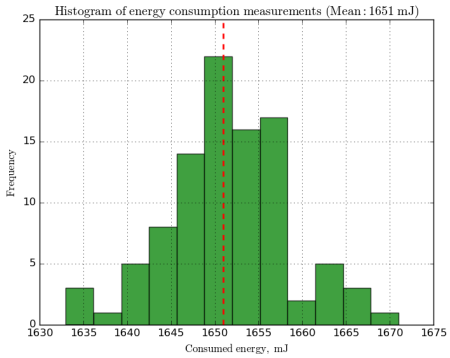


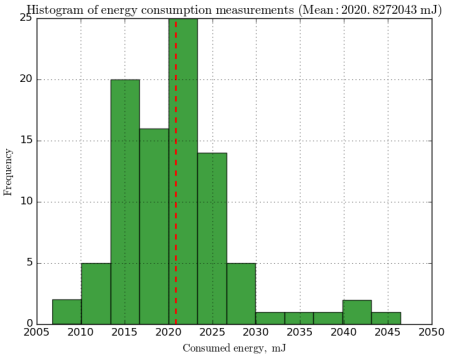
Figure 3.2: Execution time of *Sobel* filter algorithm on images of different resolutions. Note that CPU data does not contain read and write times, since these operations are required on accelerator devices.

3.1.2 Energy Efficiency of Computing Platforms

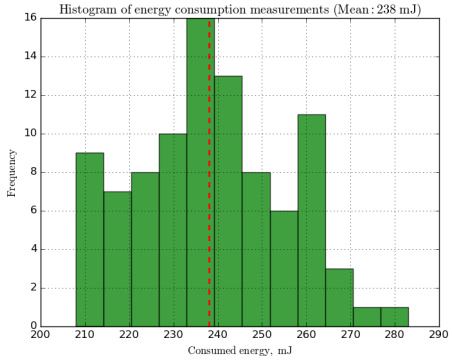
Histograms presented in figure 3.3 shows consumed energy values for each of the experiment iterations, and the red dashed line represents the average value of all measured values. The experiment is carried out by performing 100 iterations of *Sobel* filter on an input image of 1920x1080 resolution. Table 3.1 presents average measured energy consumption for each experiment. The results show that by using Mali GPU T764 (on *Radxa Rock2* platform) may reduce energy consumption 6.85 times comparing with 16 core Epiphany co-processor (on *Parallella* platform) and consumes 84.2 times less energy when using Nvidia Tegra X1 (*Airvision Core X1* platform). Table 3.2 shows the results of performing a Shapiro-Wilk test against measurements on each platform. The test shows that all measurement value p-values are above 0.05, so the null-hypothesis that values are from a normally distributed population cannot be rejected with a 95% probability. The results of Student's t-test across the measurements are provided in table 3.3, the t value represents the t-Statistics of the test and p is the p-value of a null-hypothesis stating that



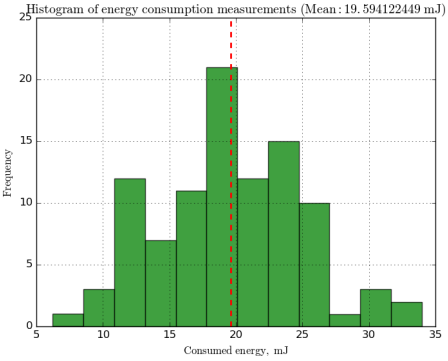
(a) Consumed energy histogram on *Parallella* with Epiphany-16 co-processor.



(b) Consumed energy histogram on *ARM Cortex A9 CPU*.



(c) Consumed energy histogram on *Radxa Rock2* with ARM Mali T764 GPU.



(d) Consumed energy histogram on *Airvision Core X1* with NVIDIA Tegra X1 GPU.

Figure 3.3: Consumed energy histograms of different platforms.

measurements are measured dependently. The results of the Student's t-test show that the experimental results were captured independently with a very high probability (p-value < 0.001).

The experiments shows a comparison of various heterogeneous platforms and though Parallella platform has a new innovative co-processor technology it may be inefficient with input data-intensive tasks, such as convolution filters (like benchmarked Sobel filter), with only 2.54 FPS

processing speed on high-resolution images. Due slow computations, the platform consumes 3x more energy than a single core ARM CPU.

Nvidia Tegra X1 GPU powered platform showed the best results compared with any other platform, which may be expected since it uses 256 cores for processing while the Parallella platform has only 16 cores. It uses ~12x less energy for the same amount of calculations and is ~29x times faster than a second-best platform – Radxa Rock2, this is not only because execution is a lot faster because of the number of cores, but memory transfer is ~26x faster.

Although Nvidia Tegra X1 platform was around six times more expensive than other platforms at the time of experiment execution (600$ versus 99$ and 120$ for Parallella and Radxa Rock2 respectively), to compare value-per-money metric we see that Parallella calculates 0,025 FPS/$ (minimum price per unit is 99$), Radxa Rock2 can calculate 0.10 FPS/$ (minimum price per unit is 120$), Nvidia Tegra X1 0.60 FPS/$ (minimum price per unit is 600$) so money-value is better with Nvidia Tegra X1 although Radxa Rock2 is a cheaper solution. Parallella platform is not able to surpass a single core application nor by processing speed (~3x slower) nor by energy consumption (~3x more energy). Radxa Rock2 is ~5x faster and 7x more energy efficient than a single core application, while Nvidia Tegra X1 is ~142x times faster and ~84x more energy efficient.

Processing results on the Parallella platform are somewhat disappointing; the acceleration of processing is only three times faster using 16 parallel cores versus single-core implementation (on both eSDK and OpenCL). The reason may be the required data buffering due to insufficient memory on the co-processor. It is possible to avoid usage of the external memory buffer by writing an image directly into the core‘s internal memory. It could improve performance, but constrains image size. Further experiments would be required to back up this statement. Also, the FPGA system reviewed in the related work has shown potential in performance and power efficiency compared with traditional platforms.

Table 3.1: Measured average energy consumption per computed frame.

Platform	Energy, mJ
ARM Single Core	587
Parallella	1651
Radxa Rock2	238
Airvision Core X1	19.6

Table 3.2: Shapiro-Wilk test results

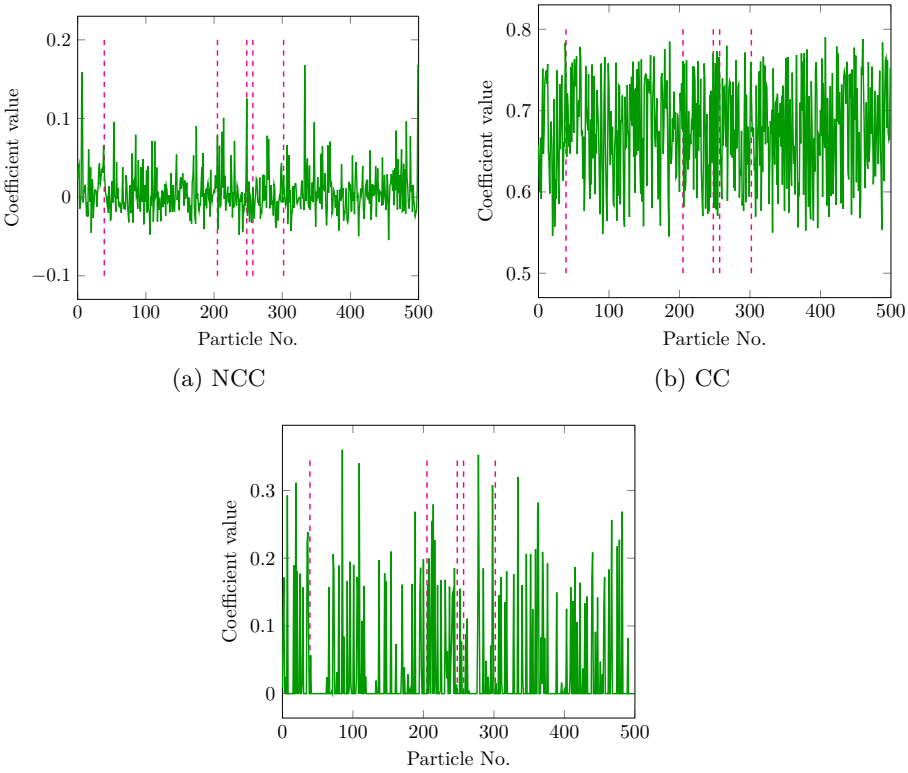
Data set name	W	p-value
ARM Single core consumed energy	0.979	0.144
Airvision Core X1 consumed energy	0.984	0.153
Parallella consumed energy	0.984	0.286
Radxa Rock2 consumed energy	0.980	0.165

3.2 Comparison of Image Similarity Metrics

Image similarity are evaluated using images from test flight dataset described in section 2.2.1. The similarity metrics described in section 2.3.6 will be evaluated by calculating the metrics between the test flight image and according to image patch from a map. The map is located in a suburb region of Lentvaris near Vilnius City, the capital of Lithuania. The

Table 3.3: Student's t-test results across measurements.

	ARM Single Core		Parallella		Radxa Rock2		Airvision Core X1	
Statistic	**t**	**p**	**t**	**p**	**t**	**p**	**t**	**p**
ARM Single Core	-	-	-1207	<0.01	187	<0.01	752	<0.01
Parallella	1207	<0.01	-	-	742	<0.01	1801	<0.01
Radxa Rock2	-187	<0.01	-742	<0.01	-	-	119	<0.01
Airvision Core X1	-752	<0.01	-1801	<0.01	-119	<0.01	-	-



(a) NCC

(b) CC

(c) SSD

Figure 3.4: Image similarity metric values on simulated 500 UAV pose hypotheses. Image a) shows the similarity coefficient values calculated using NCC metric (eq. 2.15), most values are around 0 with few clear peak values of over 0.1. The image b) contains CC metric (eq. 2.14) value, which does not contain any clear peak values. The image c) contains SSD similarity (eq. 2.13) values, the values are mostly 0, but there is also a lot of peak values.

area was selected to contain forests, grass plains and civil buildings with streets. The map contains both, texture-rich imagery like streets with many houses and grass plains which does not contain a lot of distinct visual features.

Images of a hypothesized location are cropped out of the map, re-scaled and noise are added using Gaussian and Salt and pepper algorithms. Salt and pepper algorithm is changing 5% image pixels into completely white pixels and 5% pixels completely black. Gaussian noise is added using 25 mean value and 40 sigma value of standard deviation. Images are compared in gray-scale color space, a conversion from colored to gray-scale color space is performed before calculating similarity value.

To compare the image matching coefficients, a set of 500 search hypothesis is generated and image similarity is calculated using a single ground truth location. Figure 3.4 shows the similarity values as calculated by formulas given in section 2.3.6. CC metric (see figures 3.4b) does not distinguish any poses from the common; there are many peak values. This issue will cause problems while localizing because many particles will survive during localization and it may take considerable time to filter out the final pose. SSD function (see figure3.4c) contains a lot of zero values, which means that these values will undoubtedly be omitted on the next iteration of Particle filtering. It causes a very early pose convergence which is likely to be incorrect. Particle filter requires some iterations to collect enough evidence over time to select a proper pose. Meanwhile, NCC (see figure 3.4a) has few high probability particles, and others are in a low probability zone. Thus, it allows the survival of very probable locations, but less probable locations can survive for consideration with lower probability. This property allows filtering the values over time with some chance of survival for images with lower similarities, which in some cases, over time, maybe the ground truth pose.

The dashed vertical lines mark five particles nearest to ground truth, most of them are one of the peak probability values, from figure 3.4a we can see that even particles with highest correlation coefficient values

Table 3.4: Similarity metric comparison.

Value	**NCC**	**CC**	**SSD**
Top 5 Min	**0.0122**	0.7185	0
Top 5 Average	**0.0550**	0.7471	0.0000
Top 5 Max	**0.1253**	0.7703	0.0000
Top 5 particles above Min (N^+)	**5**	5	5
Top 5 particles above Average (N^+)	**2**	2	5
Top 5 particles above Max (N^+)	**1**	1	5
Particles Above Top 5 Min (N^-)	**166**	151	141
Particles Above Top 5 Average (N^-)	**31**	74	141
Particles Above Top 5 Max (N^-)	**3**	15	141
Top 5 Min Accuracy	**2.92%**	3.21%	3.42%
Top 5 Average Accuracy	**6.45%**	2.70%	3.42%
Top 5 Max Accuracy	**25.00%**	6.25%	3.42%

may not be the closest particle to the ground truth.

Table 3.4 shows a statistical comparison of the similarity values; each row of the table is explained in the following list:

1. Top 5 Min - the minimum similarity value of 5 particles nearest to the ground truth.
2. Top 5 Average - the average similarity value of 5 particles nearest to the ground truth.
3. Top 5 Max - the highest similarity value of 5 particles nearest to the ground truth.
4. Top 5 particles above Min - number of particles from the top 5 that received higher similarity value than top 5 minimum.
5. Top 5 particles above Average - number of particles from the top 5 that received higher similarity value than top 5 average.
6. Top 5 particles above Max - number of particles from the top 5 that received higher similarity value than top 5 maximum.
7. Particles Above Top 5 Min - Number of particles that received a higher similarity value than the top 5 minimum.

8. Particles Above Top 5 Average - Number of particles that received a higher similarity value than the top 5 average.
9. Particles Above Top 5 Max - Number of particles that received a higher similarity value than the top 5 maximum value.
10. Top 5 Min Accuracy - shows the accuracy of the metric with a bottom-line of top 5 minimum.
11. Top 5 Average Accuracy - shows the accuracy of the metric with a bottom-line of top 5 average.
12. Top 5 Max Accuracy - shows the accuracy of the metric with a bottom-line of top 5 maximum.

The accuracy in the table 3.4 is calculated by expressing the number of the positive particle (any of the top 5) in the range of particle above the top 5 minimum, average, or maximum. The accuracy can be expressed as:

$$\alpha_{accuracy} = \frac{N^+}{N^- + N^+}, \tag{3.1}$$

where:

- $\alpha_{accuracy}$ is the accuracy value,
- N^+ is the number of particles from the top 5 particle set that received a similarity value higher than one the minimum, average, or maximum,
- N^- is the number of particles from non-top 5 particle set (the rest of the particles) that received a similarity value higher than one of the minimum, average, or maximum.

From the table 3.4 we can see that NCC metric calculates higher similarity values for the nearest particles compared to other metrics. The NCC metric has only three particles with higher similarity then top 5 particles maximum similarity compared to 15 particles of CC and 141 particles of SSD. Despite that, the NCC has 10% lower top 5 minimum accuracy than the CC, but the average and maximum accuracies are better (2.4 times better on average and four times better on maximum). The NCC metric provides far better chances of survival for the top

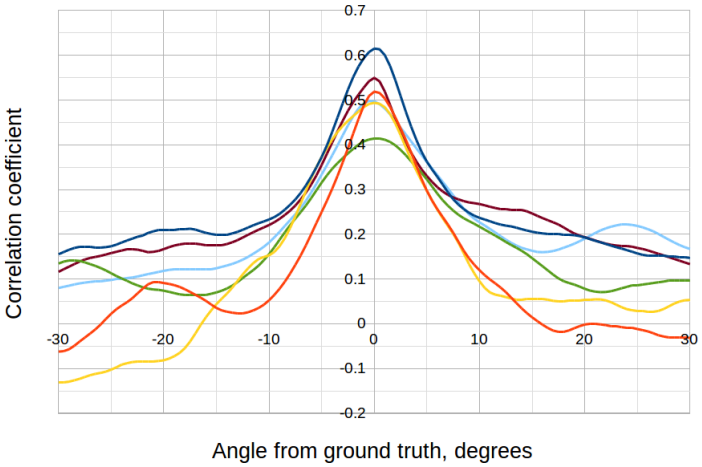


Figure 3.5: Correlation coefficient values when matching map with aerial imagery on all heading angles.

particles compared to other metrics. Therefore, (NCC) will be used for Particle filter localization in the rest of the experiments.

3.3 Evaluation of UAV Heading Error Impact on the NCC Similarity Metric

This section investigates the magnetometer error impact on NCC image similarity metric. The NCC metric is rotation variant, so the error in similarity value with inaccurate rotational data is measured. To evaluate how the angular errors affect NCC similarity values 500 aerial images were selected from test flight dataset with according map patches and correlation values between the pairs were calculated while rotating the aerial image around its center point and keeping the map patch unmodified. The change in correlation value is observed relative to the correlation value of perfectly aligned images. Figure 3.5 presents NCC similarity values for 6 sample images captured during real flight and matched with according orthophoto map patches. Table 3.5 contains average similarity change values versus heading change. Data from table

Table 3.5: Heading change impact on similarity coefficient

Heading change, °	Average similarity change, %
+10	67.78
+5	34.59
+2	9.38
-2	10.15
-5	35.49
-10	62.08

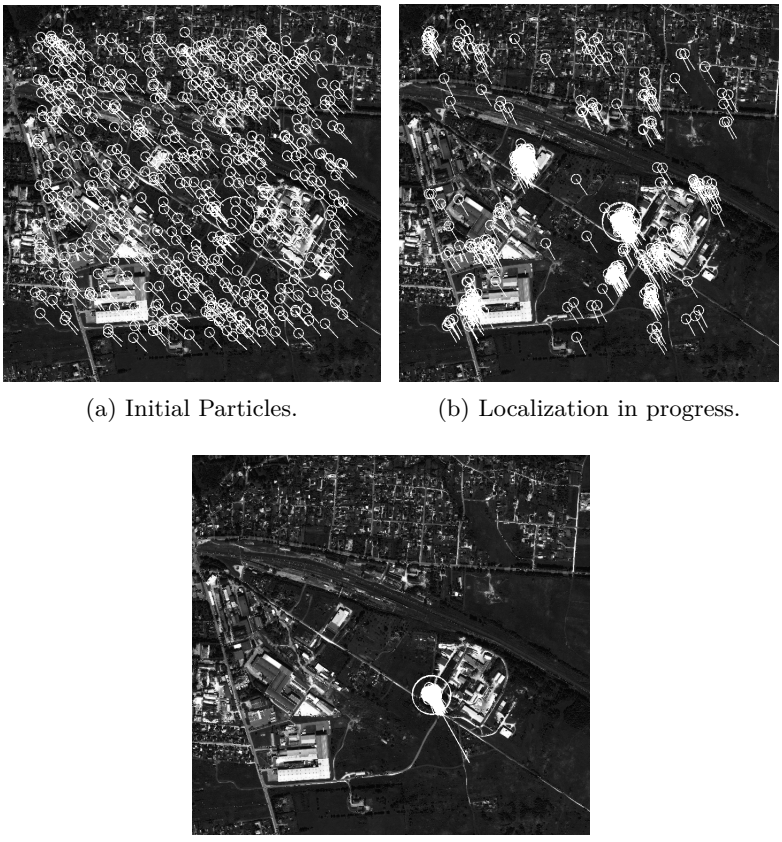
3.5 suggests that +/- 2 degrees of heading angle error can be ignored because it affects the correlation coefficient only up to 10% on average. The NCC metric can be used with heading measurements from magnetometers which measures the heading angle with 2-degree accuracy.

3.4 Comparison Of Particle Sampling Techniques

This section presents an experiment performed to evaluate three particle sampling algorithms presented in section 2.3.2:

1. Rejection Sampling
2. Importance Sampling
3. KLD-Sampling

This experiment aims to identify the sampling algorithm which can localize the UAV faster and with comparable accuracy. The experiment is set up by imitating the loss of GPS signal during UAV flight. The localization is started in a region of 312-meter radius (1040 in pixels) of last known position (worst case GPS accuracy is 156 meters, so doubled value makes the experiment more thorough). The effectiveness of the



(a) Initial Particles.

(b) Localization in progress.

(c) Final location converged.

Figure 3.6: Hypothesized UAV locations on the known map.

algorithm is measured by the percentage of successful predictions of the ground truth pose after 50 iterations (around 1 kilometer of flight distance). The result is considered successful if the particle with the highest probability is no further from ground truth then 15 meters. An example search of UAV location is presented in figure 3.6, where 500 particles are scattered randomly over an area of a map to localize the true pose of UAV. In the initial state (figure 3.6a) the particles are scattered uniformly among the map, the ground truth location is marked with a circle

marker that is five times larger than the hypothesized locations. During the localization (figure 3.6b, the weak hypotheses were eliminated, and the surviving particles are concentrated in few places which contain similar imagery to the one visible from the UAV. At the end of localization (figure 3.6c, all the surviving particles are near the ground truth location. The pose is marked with a large circle in the image center. To measure the accuracy of the localization 100 start locations are randomly selected, and each sampling algorithm is evaluated using the same starting conditions. The experiment allows 50 iterations for a single localization to converge. Then the best-matched hypothesis is compared to a ground truth value. Cropped out of the map image frame has the dimensions of 400x400 pixels. Such image size would be visible while using the 90-degree camera at around 100-meter altitude. This small size of the image was chosen because usage of a global shutter camera would be recommended to avoid shutter distortions. Most global shutter cameras are usually industrial low-resolution cameras. Higher resolutions may require additional resources and are more expensive. Quality image is more important than high resolution. Higher resolution may not vary significantly in matching quality. The camera should be calibrated correctly to be used in real-world scenarios.

Table 3.6 shows the experimental results of the algorithm comparison. The final particle count is the average particle count at the last iteration of the algorithm. Rejection and importance sampling algorithms have fixed particle counts, and the KLD-sampling particle count is variable over time. The average time is calculated by adding up the execution time of each 100 flight starting point tests. Experiments were concluded on an Intel i5-4200M processor with 3.1 GHz operating frequency.

Table 3.6: Comparison of sampling algorithms

Algorithm	End particle count	Average duration, s	Successful localization
Rejection	500	215	90 %
Importance	500	219	99 %
KLD	150	81.8	94 %

3.5 Image Similarity To Probability Conversion Functions

This thesis proposes a new component for visual Particle filter localization - image similarity to particle probability conversion function (also referred to as the discriminatory conversion function). Two parametric conversion functions were proposed in section 2.3.7. Accuracy, execution speed, and robustness to inaccurate maps are measured using different conversion functions and different parameter values. The simulated dataset is used in the experiments; 10 flights are performed for each of the dataset scenarios and conversion functions with different parameter values. A total of 1440 simulated flights were performed on the simulated dataset (presented in section 2.2.2) and average values are presented. To measure the improvement, two baseline functions were used: normalization to a range [0; 1] (see equation 2.16) and softmax conversion (see equation 2.17). For details of conversion function, see section 2.3.7. The final results are divided into three subsections: accuracy, execution time, and robustness.

A single combination of conversion function, a flight scenario, and a map used for matching is run for ten times, and average results are provided to account for randomness in the Particle filter algorithm. Maps of different dates are used for matching to test whether the algorithm can cope with changes in the environment. Four maps created on two-year intervals in the forest and urban environment are used for localization. During each experimental flight iteration, a measure of

accuracy in meters and duration in particle evaluations are recorded. Rectifying function is used with parameter values: 0.2, 0.1, 0.0, -0.1, -0.2. Logistic function is used with parameter values: 0.7, 0.4, 0.2, 0.1, and 0.05. Different metrics are used to evaluate accuracy, speed, and robustness:

- Accuracy is measured by calculating Euclidean distance between the ground truth location and algorithm's output location in the map plane. A ranking method is used to select which conversion function gives the most accurate results. Average accuracies from each conversion function are ranked amongst each of the flight scenarios, given the best - 1 point for the most accurate result, 2 points for the second most accurate and so on, finally, the function with least points will be chosen as the most accurate conversion function.
- Speed is measured by the number of average evaluated particles during the single experimental flight. Since the number of iterations in each simulated flight is the same, average particle evaluations per iteration is proportionate to time of execution, eliminating stochastic changes introduced while measuring execution time and it is independent of system resources.
- Robustness is evaluated by measuring the standard deviation of average accuracy using maps that were created on different dates. USGS provides imagery of the same regions every two years starting from 2008 (for the regions chosen for this research). This way we can measure how well the algorithm copes with changes introducing in the environment that is caused by time.

Visualization of the results is available online [1] .

3.5.1 Accuracy

Tables 3.7 and 3.8 show average accuracy results on each flight scenario with logistic and rectifying conversion functions using different para-

[1] Presentation video online in YouTube platform: `https://youtu.be/tcz_gFbivqA`

Table 3.7: Localization accuracy in meters using parametric logistic conversion function

Scenario	**v = 0.7**	**v = 0.4**	**v = 0.2**	**v = 0.1**	**v = 0.05**
FL-200	39.27	37.41	**55.91**	225.41	238.70
FL-300	26.74	26.09	**27.66**	25.63	160.18
FR-200	62.71	52.73	**74.06**	62.36	58.20
FR-300	64.34	59.25	**31.83**	32.54	58.48
FC-200	57.32	68.53	**92.39**	102.86	49.36
FC-300	95.63	112.06	**126.93**	156.09	140.20
UL-200	27.17	27.64	**26.56**	25.46	28.27
UL-300	27.27	25.88	**25.95**	26.60	27.57
UR-200	54.69	42.43	**37.09**	66.53	52.23
UR-300	50.00	43.55	**41.27**	45.07	55.10
UC-200	70.42	56.57	**49.09**	48.38	65.56
UC-300	73.71	64.74	**54.29**	63.16	67.49

Table 3.8: Localization accuracy in meters using parametric rectifying conversion function

Scenario	**d = 0.2**	**d = 0.1**	**d = 0.0**	**d = -0.1**	**d = -0.2**
FL-200	39.97	**37.66**	50.79	145.09	233.22
FL-300	27.01	**25.74**	27.14	56.05	184.69
FR-200	59.64	**67.15**	72.57	72.03	79.35
FR-300	59.50	**51.41**	40.72	29.18	35.53
FC-200	58.26	**73.73**	78.47	67.71	26.03
FC-300	92.44	**116.12**	125.66	142.34	112.88
UL-200	29.81	**27.40**	26.67	27.03	26.61
UL-300	26.66	**26.34**	25.01	25.60	25.27
UR-200	58.50	**44.85**	36.39	42.53	49.48
UR-300	53.54	**42.82**	39.55	39.92	42.68
UC-200	78.07	**72.78**	49.43	50.33	48.05
UC-300	76.03	**67.38**	57.79	56.38	61.53

Table 3.9: Conversion function ranking according to accuracy

Function	**Parameter value**	**Score**	**Rank**
Rectifying	0.2	75	8
Rectifying	0.1	54	2
Rectifying	0	56	3
Rectifying	-0.1	69	7
Rectifying	-0.2	88	10
Logistic	0.7	82	9
Logistic	0.4	68	6
Logistic	0.2	52	1
Logistic	0.1	58	4-5
Logistic	0.05	58	4-5

Table 3.10: Localization comparison against Softmax conversion function

	Accuracy improvement		
Scenario	Softmax / Linear	Logistic / Linear	Logistic / Softmax
FL200	7.79%	15.59%	7.24%
FL300	6.47%	32.40%	24.35%
FR200	14.77%	10.18%	-4.00%
FR300	6.37%	85.07%	73.99%
FC200	2.48%	-40.14%	-41.59%
FC300	-2.58%	-23.61%	-21.58%
UL200	12.10%	65.12%	47.30%
UL300	12.37%	48.47%	32.12%
UR200	22.90%	224.89%	164.34%
UR300	6.49%	98.27%	86.19%
UC200	0.81%	95.25%	93.67%
UC300	2.79%	63.48%	59.04%
Average	7.73%	56.25%	43.42%

meter values. Columns in bold show the most accurate localization with a given parameter value of a function from ranking results in table 3.9. Table 3.9 shows the experimental results ranked by accuracy for each of the experimental results, by ranking from 1 (best) to worst (10), the sum of the ranks is used to determine the most accurate (lowest value) conversion function and its parameter value.

Table 3.10 shows the comparison of selected best conversion function with a parameter value (logistic with v = 0.2) to the baseline Softmax conversion function, the comparison results shows, that the best logistic function is 43% more accurate and also requires 3 times less iterations (see table 3.11) on average to achieve the result.

3.5.2 Execution Time

Table 3.11 presents execution time speed-up measured of the best ranked logistic conversion function with 0.2 parameter value. The second column shows localization speed-up using Softmax conversion function against the most trivial linear conversion function. The Softmax function achieved 14% execution time speed-up in some cases and 7% speed-up on average. The best ranked logistic conversion function achieved 3.18x speed-up compared against the linear and 2.96x speed-up compared against Softmax conversion function. The execution time reduction achieved by using logistic conversion function is a very significant speed-up and even after the 3x speed-up already achieved by implementing KLD-sampling based localization.

3.5.3 Robustness

By comparing average accuracies on different maps, we can evaluate the algorithms ability to localize with changes in the imagery. Figure 3.7 shows accuracies using different maps for localization. The flight data was collected using the 2008 map, so with an increasing date of map creation, it incorporates more changes. Data shows that using a rectifying function with parameter in the range $0.0, \ldots, 0.2$ and logistic conver-

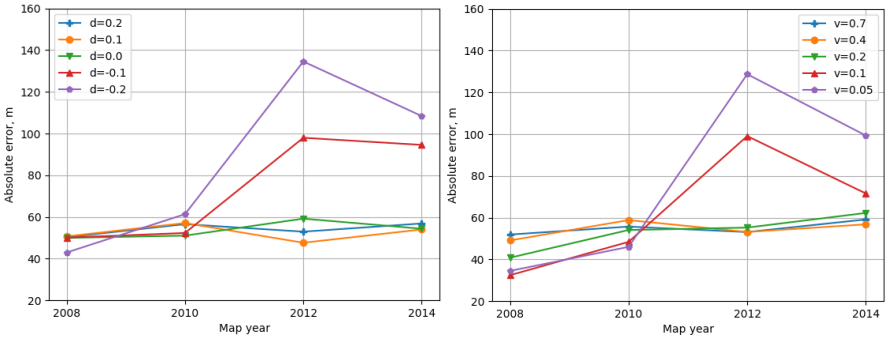
Table 3.11: Localization comparison against Softmax conversion function

	Speedup		
Scenario	Softmax / Linear	Logistic / Linear	Logistic / Softmax
FL200	1.06	2.41	2.26
FL300	1.07	2.32	2.17
FR200	1.11	3.23	2.92
FR300	1.14	3.36	2.95
FC200	1.00	2.66	2.65
FC300	0.99	2.51	2.53
UL200	1.14	3.76	3.31
UL300	1.12	3.40	3.03
UR200	1.14	3.66	3.21
UR300	1.10	3.37	3.07
UC200	1.01	3.62	3.58
UC300	1.02	3.89	3.83
Average	1.07	3.18	2.96

sion function with parameter in the range $0.2, \ldots 0.7$ provides similar accuracy with different maps, even with a map dated in 2014 which is six years apart from the imagery used in the flight. The accuracy also continues to increase with logistic function values 0.1 and 0.05 while using very recent maps, but the accuracy decreases drastically with maps that were created later, so using these values would provide benefits only if the map is very recent and does not have drastic changes. Table 3.12 shows the Standard Deviation (abbr. SD), Average, and Relative Standard Deviation (abbr. RSD) of localization accuracy using different probability conversion functions. The RSD value is calculated using the following formula:

$$RSD = \frac{100 \cdot SD}{Accuracy}, \tag{3.2}$$

where



(a) Accuracies using rectifying conversion function

(b) Accuracies using logistic conversion function

Figure 3.7: Comparison of localization accuracy with different maps, map years depicts the date when map was created.

- SD — Standard Deviation value,
- $\overline{Accuracy}$ — Average accuracy .

The rectifying function with parameter values of 0.2, 0.1, and 0 and logistic function with parameter value of 0.7 and 0.4 relative standard deviation values are under 10% which shows these function abilities to be robust using any maps from this experiment. Although, other functions are not that robust to older maps, but their accuracy is better using recent maps.

3.6 Comparison of DPC-PFL Against Visual Odometry

This experiment evaluates localization accuracy compared with conventional GPS positioning system and pure Visual odometry positioning. Accumulated odometry error correction is expected when using Particle filter in combination with odometry. Flight trajectory reconstruction using odometry and DPC-PFL is presented in figure 3.8. Trajectory errors in meters are presented in figure 3.9, the figure shows that odo-

Table 3.12: The deviations of localization accuracy achieved using different conversion functions over different maps.

Function	Parameter value	SD	$\overline{\text{Accuracy}}$	RSD
Rectifying	0.2	3.14	54.16	5.8%
Rectifying	0.1	4.09	52.36	7.8%
Rectifying	0	4.14	53.64	7.7%
Rectifying	-0.1	26.06	73.73	35.4%
Rectifying	-0.2	42.09	86.80	48.5%
Logistic	0.7	3.20	54.95	5.8%
Logistic	0.4	4.29	54.45	7.9%
Logistic	0.2	8.96	53.08	16.9%
Logistic	0.1	28.99	62.85	46.1%
Logistic	0.05	44.49	77.11	57.7%

metry suffers from cumulative errors as it was introduced in section 1.3.2. The dashed lines are the error trend-lines. The vertical line shows the breaking point of the trend-lines at 35 seconds of flight time. Since the breaking point of accuracies shows that the proposed algorithm adds a lot fewer errors during long time flights. Additional experiments are required to validate whether errors will not add up after long flights. Particle filter localization was able to keep error values in an around 50-meter range. After the 1 kilometer flight, the final error was reduced by a factor of 2 compared to localization from Visual odometry only. The DPC-PFL reduces error trend-line slope by a factor of 11 compared against SVO.

3.7 Comparison Against Visual SLAM

This section presents the comparison results of the proposed DPC-PFL algorithm with state-of-the-art ORB-SLAM algorithm. Results of 4 flight scenarios are compared since ORB-SLAM was not able to reconstruct other flights and lost position tracking mid-flight; incomplete res-

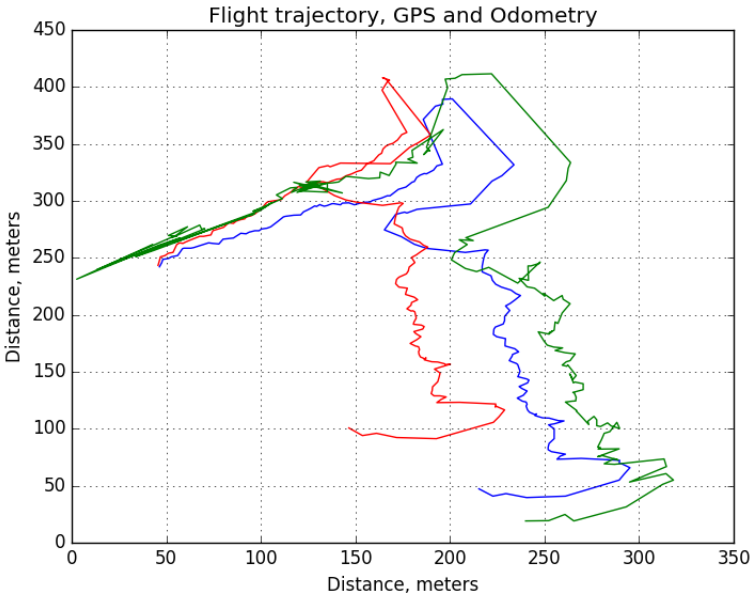


Figure 3.8: Flight trajectory reconstruction using Visual odometry (red), proposed DPC-PFL (green) and conventional GPS sensor (blue).

ults are not included. Figures 3.10a to 3.10d shows flight trajectories recovered by 3 algorithms - SVO (which is used internally by Particle filter), proposed DPC-PFL with logistic conversion function and parameter value of 0.2, and ORB-SLAM. According to results in table 3.13, ORB-SLAM provides similar results to SVO, while being a little bit more precise; meanwhile, the proposed DPC-PFL outperforms both SVO and ORB-SLAM with over ~2.6 times higher precision in average.

3.8 Conclusions Of Chapter 4

The conclusions of the experiments performed in this chapter can be outlined as follows:

- Normalized Correlation Coefficient is the most suitable metric for Particle filter localization compared against Sum of Squared Dif-

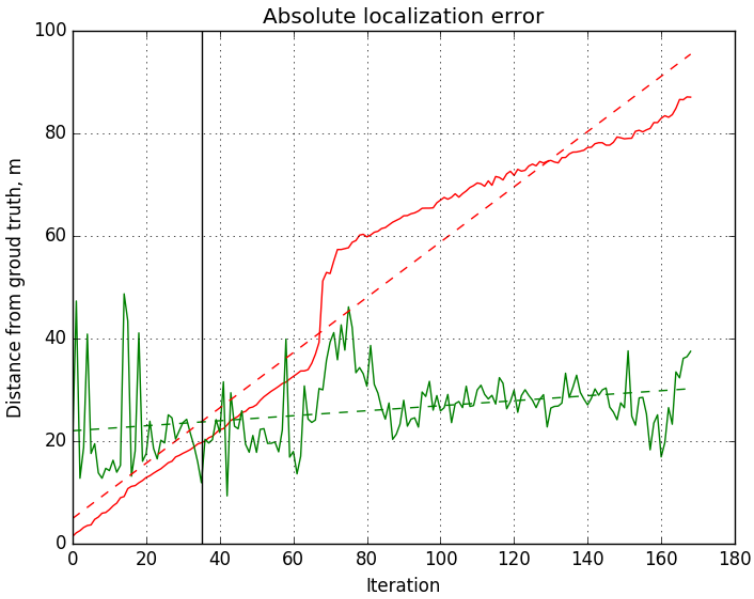
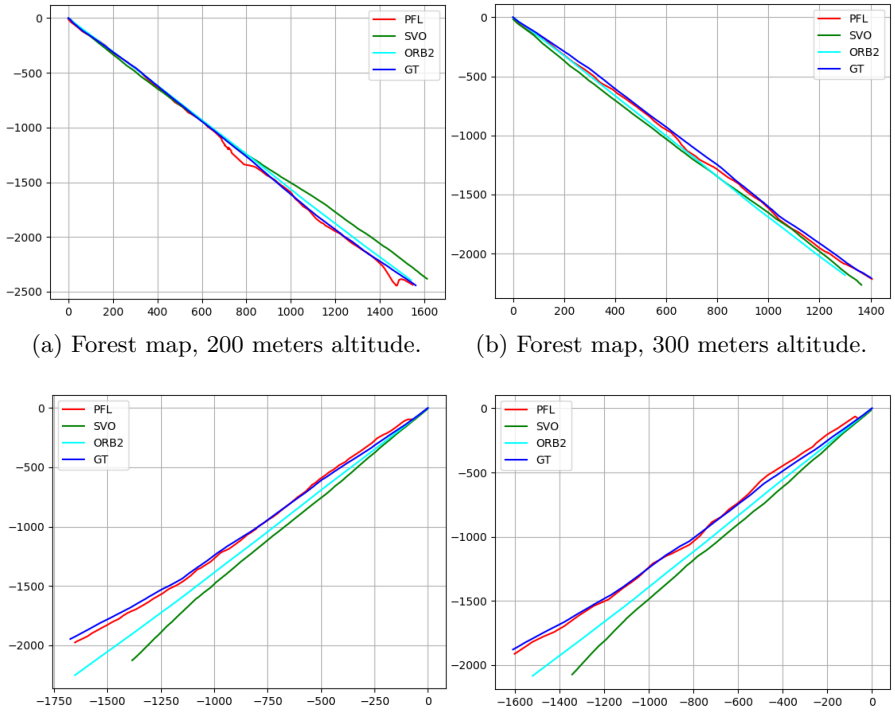


Figure 3.9: Absolute positioning error using Visual odometry (red) and proposed DPC-PFL (green).

Table 3.13: Localization accuracy comparison of ORB-SLAM (ORB), SVO Visual odometry, and the proposed algorithm – DPC-PFL.

	Accuracy, m			Relative accuracy		
Scenario	SVO	PFL	ORB	PFL/SVO	ORB/SVO	PFL/ORB
FL-200	48.36	27.60	49.56	75.26%	-2.43%	79.61%
FL-300	45.00	17.08	58.55	163.53%	-23.14%	242.88%
UL-200	130.95	21.93	118.48	497.03%	10.53%	440.17%
UL-300	129.88	21.06	86.05	516.84%	50.94%	308.67%
Average				313.16%	8.97%	267.83%



(a) Forest map, 200 meters altitude.

(b) Forest map, 300 meters altitude.

(c) Urban map, 200 meters altitude.

(d) Urban map, 200 meters altitude.

Figure 3.10: Linear flight trajectories recovered using SVO, ORB-SLAM2, and proposed DPC-PFL.

ferences and Cross-Correlation since it calculates the highest similarity values to particles that are closest to the ground truth. The metric can be used with magnetometers that measure heading angle with a $\pm 2^{\circ}$ error since such error reduces the similarity value only by 10%.

- KLD sampling technique was shown to be as accurate as importance and rejection sampling techniques but also adapts particle count to reduce computational load when it is not necessary. The experimental results suggest that KLD sampling is at least 2.8x faster than traditional importance sampling technique with similar

localization success rate.

- Localization performed using Softmax function 2.17 compared with linear conversion function 2.16 shows that localization using Softmax function reduces positioning error by 7% on average.
- Localization using the proposed logistic function with a parameter value of 0.2 was shown to provide the highest ranking results compared against linear and logistic conversion function with different parameter values.
- Particle filter localization using logistic conversion function with a parameter value of 0.2 achieves 43% higher accuracy using three times fewer computations than the baseline Softmax probability conversion function.
- Localization using the rectifying conversion function with a parameter values 0.2, 0.1, and 0.0 and the logistic conversion function with a parameter values 0.7, 0.4, and 0.2 has shown robust localization accuracy, the relative standard deviation is under 20% for all cases.
- Proposed localization algorithm DPC-PFL has shown the ability to reduce the accumulating drift which is a known drawback of Visual odometry algorithms. During the localization performed on real test flight data, absolute localization error was reduced two times, and the slope of accumulating error was reduced 11 times.
- DPC-PFL algorithm was compared against state-of-the-art ORB-SLAM algorithm. The results show that ORB-SLAM provides 9% higher accuracy than the Visual odometry algorithm SVO. Meanwhile, Particle filter using the proposed logistic function was able to achieve ~2.6 times better accuracy than both ORB-SLAM and SVO. However, the results were only provided for the straight line flight trajectory, since ORB-SLAM was unable to complete whole trajectories on the other flight scenarios, while the DPC-PFL completed all flights.
- The comparison of computing platforms concludes that Airvision Core X1 with NVIDIA TX1 GPU is the most energy efficient par-

allel computing platform for computer vision tasks onboard GPU since it is ~142x times faster and ~84x more energy efficient than single-core ARM CPU and ~29x times faster and ~12x more energy efficient then second best platform *Radxa Rock 2* with *ARM Mali T764* GPU.

GENERAL CONCLUSIONS

1. Pearson Correlation Coefficient is the best image similarity metric for localization in low altitude GPS-Denied flight compared to the Sum of Squared Differences and Cross-Correlation. It provides higher similarity values to hypotheses that are closer to the ground truth location compared to the other image similarity metrics. The usage of Pearson Correlation Coefficient allows increasing the detection probability of a particle that is close to the ground truth 4 times compared to Cross-Correlation.
2. Particle filter localization algorithm using KLD sampling technique performs 2.6 times faster compared to rejection and importance sampling based Particle filter localizations with a similar localization success rate.
3. Particle filter based localization algorithm using logistic conversion function with parameter value 0.2 achieved 43% more accurate localization and is 3 times faster compared to the same algorithm using baseline Softmax conversion function.
4. The Proposed particle filter localization algorithm reduces the localization error 2 times compared to state-of-the-art Visual odometry algorithm SVO. Additionally, the algorithm reduces the accumulating error slope of odometry 11 times.
5. Proposed localization algorithm based on Particle filter, KLD sampling, logistic conversion function with parameter value 0.2 and Normalized Cross Correlation achieves 2.6 times better localization accuracy compared against state-of-the-art Visual SLAM ORB-SLAM algorithm.
6. The proposed method for measuring the energy efficiency of computing platforms identified the *Airvision Core X1* as the most energy efficient platform for computing onboard UAV. It is ~142x times faster and ~84x more energy efficient than single-core ARM CPU and ~29x times faster and ~12x more energy efficient then second best platform *Radxa Rock 2* with *ARM Mali T764* GPU.

Bibliography

[1] Andrew J Kerns, Daniel P Shepard, Jahshan A Bhatti, and Todd E Humphreys. Unmanned aircraft capture and control via gps spoofing. *Journal of Field Robotics*, 31(4):617–636, 2014.

[2] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

[3] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[4] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1):99–141, 2001.

[5] Frank Dellaert, Sebastian Thrun, and Charles E Thorpe. *Mosaicing a large number of widely dispersed, noisy, and distorted images: A bayesian approach*. Carnegie Mellon University, The Robotics Institute, 1999.

[6] Henrik Andreasson, André Treptow, and Tom Duckett. Localization for mobile robots using panoramic vision, local features and particle filter. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3348–3353. IEEE, 2005.

[7] Fernando Caballero, Luis Merino, Joaquin Ferruz, and Aníbal Ollero. Vision-based odometry and slam for medium and high altitude flying uavs. *Journal of Intelligent and Robotic Systems*, 54(1-3):137–161, 2009.

[8] Sebastian Thrun. Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc., 2002.

[9] Mo Shan, Fei Wang, Feng Lin, Zhi Gao, Ya Z Tang, and Ben M Chen. Google map aided visual navigation for uavs in gps-denied environment. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 114–119. IEEE, 2015.

[10] Farid Kendoul. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29(2):315–378, 2012.

[11] Gianpaolo Conte and Patrick Doherty. An integrated uav navigation system based on aerial image matching. In *Aerospace Conference, 2008 IEEE*, pages 1–10. IEEE, 2008.

[12] Roland Siegwart, Illah Reza Nourbakhsh, Davide Scaramuzza, and Ronald C Arkin. *Introduction to autonomous mobile robots*. MIT press, 2011.

[13] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE pervasive computing*, (3):24–33, 2003.

[14] YC Ho and RCKA Lee. A bayesian approach to problems in stochastic estimation and control. *IEEE Transactions on Automatic Control*, 9(4):333–339, 1964.

[15] Zhe Chen et al. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182(1):1–69, 2003.

[16] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. 2005.

[17] Mohinder S Grewal and Angus P Andrews. Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems*, 30(3):69–78, 2010.

[18] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

[19] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[20] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[21] Jacky Baltes, Amirhossein Hosseinmemar, Joshua Jung, Soroush Sadeghnejad, and John Anderson. Practical real-time system for object counting based on optical flow. In *Robot Intelligence Technology and Applications 3*, pages 299–306. Springer, 2015.

[22] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[23] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[24] Mandyam V Srinivasan. How bees exploit optic flow: behavioural experiments and neural models. *Phil. Trans. R. Soc. Lond. B*, 337 (1281):253–259, 1992.

[25] Juergen Haag, Winfried Denk, and Alexander Borst. Fly motion vision is based on reichardt detectors regardless of the signal-to-noise ratio. *Proceedings of the National Academy of Sciences*, 101 (46):16333–16338, 2004.

[26] Franck Ruffier, Stéphane Viollet, S Amic, and N Franceschini. Bio-inspired optical flow circuits for the visual guidance of micro air vehicles. In *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*, volume 3, pages III–III. IEEE, 2003.

[27] Bruno Herissé, Tarek Hamel, Robert Mahony, and François-Xavier Russotto. Landing a vtol unmanned aerial vehicle on a moving platform using optical flow. *IEEE Transactions on robotics*, 28(1): 77–89, 2012.

[28] Josef Maier and Martin Humenberger. Movement detection based on dense optical flow for unmanned aerial vehicles. *International Journal of Advanced Robotic Systems*, 10(2):146, 2013.

[29] Yuncheng Lu, Zhucun Xue, Gui-Song Xia, and Liangpei Zhang. A survey on vision-based uav navigation. *Geo-spatial Information Science*, 21(1):21–32, 2018. doi: 10.1080/10095020.2017.1420509.

URL https://doi.org/10.1080/10095020.2017.1420509.

[30] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry, part i: The first 30 years and fundamentals [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[31] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[32] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, 1993.

[33] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[34] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[35] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer, 2008.

[36] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[37] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[38] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, 32(1): 105–119, 2010.

[39] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571.

IEEE, 2011.

[40] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. Ieee, 2004.

[41] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004.

[42] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[43] Stephan Weiss, Markus W Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 957–964. IEEE, 2012.

[44] Stephan Weiss, Roland Brockers, and Larry Matthies. 4dof drift free navigation using inertial cues and optical flow. 2013.

[45] Michael Blösch, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Vision based mav navigation in unknown and unstructured environments. In *Robotics and automation (ICRA), 2010 IEEE international conference on*, pages 21–28. IEEE, 2010.

[46] Stephan Weiss, Markus W Achtelik, Simon Lynen, Michael C Achtelik, Laurent Kneip, Margarita Chli, and Roland Siegwart. Monocular vision for long-term micro aerial vehicle state estimation: a compendium. *Journal of Field Robotics*, 30(5):803–831, 2013.

[47] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Accurate figure flying with a quadrocopter using onboard visual and inertial sensing. *Imu*, 320:240, 2012.

[48] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry

for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017.

[49] Peter Muller and Andreas Savakis. Flowdometry: An optical flow and deep learning based approach to visual odometry. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 624–631. IEEE, 2017.

[50] Cheng Zhao, Li Sun, Pulak Purkait, Tom Duckett, and Rustam Stolkin. Learning monocular visual odometry with dense 3d mapping from dense 3d flow. *arXiv preprint arXiv:1803.02286*, 2018.

[51] Kurt Konolige, Motilal Agrawal, and Joan Sola. Large-scale visual odometry for rough terrain. In *Robotics research*, pages 201–212. Springer, 2010.

[52] Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear estimation*. Number EPFL-BOOK-233814. Prentice Hall, 2000.

[53] Gerald L Smith, Stanley F Schmidt, and Leonard A McGee. *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. National Aeronautics and Space Administration, 1962.

[54] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

[55] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598, 2002.

[56] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010.

[57] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[58] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal*

of Robotics Research, 5(4):56–68, 1986.

[59] John J Leonard and Hugh F Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, pages 1442–1447. Ieee, 1991.

[60] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

[61] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 155–160. IEEE, 2011.

[62] Bruno Steux and Oussama El Hamzaoui. tinyslam: A slam algorithm in less than 200 lines c-language program. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pages 1975–1979. IEEE, 2010.

[63] Luca Carlone, Rosario Aragues, José A Castellanos, and Basilio Bona. A linear approximation for graph-based simultaneous localization and mapping. In *Robotics: Science and Systems*, volume 7, pages 41–48, 2012.

[64] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16, 2017.

[65] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

[66] Sebastian Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1(1-35):1, 2002.

[67] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[68] Frank Dellaert, Steven M Seitz, Charles E Thorpe, and Sebastian Thrun. Em, mcmc, and chain flipping for structure from motion with unknown correspondence. *Machine learning*, 50(1-2):45–71, 2003.

[69] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *null*, page 1403. IEEE, 2003.

[70] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007.

[71] Brian Williams, Georg Klein, and Ian Reid. Real-time slam relocalisation. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[72] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[73] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[74] Peter Ondrúška, Pushmeet Kohli, and Shahram Izadi. Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE transactions on visualization and computer graphics*, 21(11):1251–1258, 2015.

[75] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1935–1942. IEEE, 2015.

[76] Duo-Yu Gu, Cheng-Fei Zhu, Jiang Guo, Shu-Xiao Li, and Hong-Xing Chang. Vision-aided uav navigation using gis data. In *Vehicular Electronics and Safety (ICVES), 2010 IEEE International Conference on*, pages 78–82. IEEE, 2010.

[77] A Volkova and PW Gibbens. Satellite imagery assisted road-based

visual navigation system. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:209, 2016.

[78] Steven J Dumble and Peter W Gibbens. Airborne vision-aided navigation using road intersection features. *Journal of Intelligent & Robotic Systems*, 78(2):185–204, 2015.

[79] Phillipp Jende, Francesco Nex, Markus Gerke, and George Vosselman. A fully automatic approach to register mobile mapping and airborne imagery to support the correction of platform trajectories in gnss-denied urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 141:86–99, 2018.

[80] Joe P Golden. Terrain contour matching (tercom): a cruise missile guidance aid. In *Image processing for missile guidance*, volume 238, pages 10–19. International Society for Optics and Photonics, 1980.

[81] Girish Chowdhary, Eric N Johnson, Daniel Magree, Allen Wu, and Andy Shein. Gps-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft. *Journal of Field Robotics*, 30(3):415–438, 2013.

[82] Balemir Uragun. Energy efficiency for unmanned aerial vehicles. In *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, volume 2, pages 316–320. IEEE, 2011.

[83] Janosch Nikolic, Joern Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul T Furgale, and Roland Siegwart. A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 431–437. IEEE, 2014.

[84] Luigi Nardi, Bruno Bodin, M Zeeshan Zia, John Mawer, Andy Nisbet, Paul HJ Kelly, Andrew J Davison, Mikel Luján, Michael FP O'Boyle, Graham Riley, et al. Introducing slambench, a performance and accuracy benchmarking methodology for slam. *arXiv preprint arXiv:1410.2167*, 2014.

[85] Jeremy Fowers, Greg Brown, Patrick Cooke, and Greg Stitt. A

performance and energy comparison of fpgas, gpus, and multicores for sliding-window applications. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '12, pages 47–56, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1155-7. doi: 10.1145/2145694.2145704. URL `http://doi.acm.org/10.1145/2145694.2145704`.

[86] Andreas Olofsson, Tomas Nordström, and Zain Ul-Abdin. Kick-starting high-performance energy-efficient manycore architectures with epiphany. *arXiv preprint arXiv:1412.5538*, 2014.

[87] Ivan Grasso, Petar Radojkovic, Nikola Rajovic, Isaac Gelado, and Adrian Ramirez. Energy efficient hpc on embedded socs: Optimization techniques for mali gpu. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 123–132. IEEE, 2014.

[88] Josip Knezović. Are your passwords safe: Energy-efficient bcrypt cracking with low-cost parallel hardware. In *WOOT'14 8th Usenix Workshop on Offensive Technologies Proceedings 23rd USENIX Security Symposium*. Hrvatska znanstvena bibliografija i MZOS-Svibor, 2014.

[89] Adapteva Inc. Parallella - 1.x reference manual. `http://www.parallella.org/docs/parallella_manual.pdf`, 2014.

[90] ARM Limited. Arm mali opencl sdk. `http://malideveloper.arm.com/resources/sdks/mali-opencl-sdk/`.

[91] Nvidia Corporation. Nvidia cuda getting started guide for linux. `http://developer.download.nvidia.com/compute/cuda/7_0/Prod/doc/CUDA_Getting_Started_Linux.pdf`, 2016.

[92] Ahmed Nassar, Karim Amer, Reda ElHakim, and Mohamed ElHelw. A deep cnn-based framework for enhanced aerial imagery registration with applications to uav geolocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1513–1523, 2018.

[93] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International*

Journal of Robotics Research, 32(11):1231–1237, 2013.

[94] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012.

[95] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *Robotics and automation (ICRA), 2014 IEEE international conference on*, pages 1524–1531. IEEE, 2014.

[96] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[97] András L Majdik, Charles Till, and Davide Scaramuzza. The zurich urban micro aerial vehicle dataset. *The International Journal of Robotics Research*, 36(3):269–273, 2017.

[98] Gianpaolo Conte and Patrick Doherty. Vision-based unmanned aerial vehicle navigation using geo-referenced information. *EURASIP Journal on Advances in Signal Processing*, 2009:10, 2009.

[99] Fredrik Lindsten, Jonas Callmer, Henrik Ohlsson, David Törnqvist, Thomas B Schön, and Fredrik Gustafsson. Geo-referencing for uav navigation using environmental classification. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1420–1425. IEEE, 2010.

[100] Tianmiao Wang, Chaolei Wang, Jianhong Liang, Yang Chen, and Yicheng Zhang. Vision-aided inertial navigation for small unmanned aerial vehicles in gps-denied environments. *International Journal of Advanced Robotic Systems*, 10(6):276, 2013.

[101] Adapteva Inc. Epiphany sdk reference. `http://adapteva.com/docs/epiphany_sdk_ref.pdf`, 2013.

[102] Brown Deer Technology LLC. The coprthr® primer. `http://www.browndeertechnology.com/docs/coprthr_primer-1.6.0.pdf`,

2014.

[103] Jonathan Tompson and Kristofer Schlachter. An introduction to the opencl programming model. *Digital version*, 2012.

[104] Karl K Pingle. Visual perception by a computer. *Automatic interpretation and classification of images*, pages 277–284, 1969.

[105] Raymond Serway and John Jewett. *Physics for scientists and engineers with modern physics*. Cengage learning, 2013.

[106] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, pages 591–611, 1965.

[107] Ankur Handa, Richard A Newcombe, Adrien Angeli, and Andrew J Davison. Real-time camera tracking: When is high framerate best? In *European Conference on Computer Vision*, pages 222–235. Springer, 2012.

[108] *NAIP Digital Ortho Photo Image*. USDA-FSA-APFO Aerial Photography Field Office, Salt Lake City, Utah, 2009,2010,2012,2014.

[109] Gerasimos G Rigatos. Nonlinear kalman filters and particle filters for integrated navigation of unmanned aerial vehicles. *Robotics and Autonomous Systems*, 60(7):978–995, 2012.

[110] John Von Neumann. 13. various techniques used in connection with random digits. 1951.

[111] Andy W Marshall. The use of multistage sampling schemes in monte carlo computations. 1954.

[112] Dieter Fox. Kld-sampling: Adaptive particle filters. In *Advances in neural information processing systems*, pages 713–720, 2001.

[113] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[114] Karl Pearson. Mathematical contributions to the theory of evolution. iii. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 187:253–318, 1896.

[115] Jae-Chern Yoo and Tae Hee Han. Fast normalized cross-

correlation. *Circuits, systems and signal processing*, 28(6):819, 2009.

[116] Arthur Goshtasby, Stuart H Gage, and Jon F Bartholic. A two-stage cross correlation approach to template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (3): 374–378, 1984.

[117] A Ardeshir Goshtasby. Similarity and dissimilarity measures. In *Image registration*, pages 7–66. Springer, 2012.

[118] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[119] Arie Nakhmani and Allen Tannenbaum. Particle filtering with region-based matching for tracking of partially occluded and scaled targets. *SIAM journal on imaging sciences*, 4(1):220–242, 2011.

[120] FJ Richards. A flexible growth function for empirical use. *Journal of experimental Botany*, 10(2):290–301, 1959.

[121] Dieter Fox. Kld-sampling: Adaptive particle filters. In *Advances in neural information processing systems*, pages 713–720, 2002.

Notes

Rokas Jurevičius

INVESTIGATION OF PARTICLE FILTER BASED VISUAL LOCALIZATION FOR UNMANNED AERIAL VEHICLE FLIGHTS AT LOW-ALTITUDE

Doctoral Dissertation
Technological Sciences, Informatics Engineering T 007

Editor Indgrida Verbickaitė

Vilniaus universiteto leidykla
Saulėtekio al. 9, LT-10222 Vilnius
El. p. info@leidykla.vu.lt,
www.leidykla.vu.lt
Tiražas 20 egz.