UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Fixed-wing UAV tracking in outdoor scenarios for autonomous landing

Nuno Alexandre Antunes Martins Pessanha Santos

Supervisor: Doctor Alexandre José Malheiro Bernardino

Co-Supervisor: Doctor Victor José de Almeida e Sousa Lobo

Thesis approved in public session to obtain the PhD Degree in

Electrical and Computer Engineering

Jury final classification: Pass with Distinction

2020

UNIVERSIDADE DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Fixed-wing UAV tracking in outdoor scenarios for autonomous landing

Nuno Alexandre Antunes Martins Pessanha Santos

Supervisor: Doctor Alexandre José Malheiro Bernardino
Co-Supervisor: Doctor Victor José de Almeida e Sousa Lobo

Thesis approved in public session to obtain the PhD Degree in
Electrical and Computer Engineering
Jury final classification: Pass with Distinction

Jury

Chairperson: Doctor Mário Alexandre Teles de Figueiredo, Instituto Superior Técnico, Universidade de Lisboa
Members of the Committee:
Doctor Antonis Argyros, School of Sciences & Engineering, University of Crete
Doctor José Alberto Rosado dos Santos Vitor, Instituto Superior Técnico, Universidade de Lisboa
Doctor Jorge dos Santos Salvador Marques, Instituto Superior Técnico, Universidade de Lisboa
Doctor Alexandre José Malheiro Bernardino, Instituto Superior Técnico, Universidade de Lisboa
Doctor Jorge Nuno de Almeida e Sousa Almada Lobo, Faculdade de Ciências e Tecnologia, Universidade de Coimbra

2020

Abstract

This thesis aims to develop a monocular vision system to track an Unmanned Aerial Vehicle (UAV) pose (3D position and orientation) relative to the camera reference frame during its landing on a ship. The vast majority of accidents and incidents occur during take-off or landing since, in the vast majority of systems, an external pilot takes control. Having less human intervention increases system reliability and alleviates the use of certified pilots. Due to UAV size and weight, take-off is easily performed by hand, so the main focus will be in the landing maneuver. The vision system is located on the ship's deck, which reduces demands on the UAV's processing power, size, and weight. The proposed architecture is based on an Unscented Particle Filter (UPF) scheme with two stages: (i) pose boosting, and (ii) tracking. In the pose boosting stage, we detect the UAV on the captured frame using Deep Neural Networks (DNNs) and initialize a set of pose hypotheses that are likely to describe the true pose of the target using a pre-trained database indexed by bounding boxes. In the tracking stage, we use a UPF based approach to obtain an online estimate of the true pose of the target. On contrary to many vision-based particle filters that sample particles from a distribution that is based solely on predictions from the previous frames, in this work, we also use information from the current frame to improve the convergence of the filter. We fuse information from current and previous time steps with Unscented Transform (UT) filters, and use, for the first time in this type of problem, the Bingham and Bingham-Gauss distributions to model the dynamics and noise of the orientation in its natural manifold. These filters depend on the computation of importance weights that use sub-optimal approximations to the likelihood function. We evaluate different similarity metrics that compute a distance measure between an artificial rendered image with the hypothetic state of the system and the captured frame. Since we are approximating the likelihood function, we enrich the filter with additional refinement steps to abridge its sub-optimality. We have developed a "realistic" simulator for a quantitative analysis of the results. The entire description and experimental analysis of the system is based on the tracking error and processing time. When analyzing a landing sequence with a real sky gradient filled with clouds, we have obtained approximately 81% less rotation error using the Unscented Bingham Filter (UBiF) and the Unscented Bingham-Gauss Filter (UBiGaF) when compared to the simple Unscented Kalman Filter (UKF) without considering the use of pose optimization. When we use pose optimization, we can decrease the obtained rotation error in more than 50%.

Keywords: *Computer Vision, Model-Based Pose Estimation, Model-Based Tracking, Autonomous Vehicles.*

Resumo

Esta tese tem como objetivo desenvolver um sistema de visão monocular para fazer seguimento da pose (posição 3D e orientação) de um Veículo Aéreo Não Tripulado (VANT) referente ao referencial da câmara durante a aterragem num navio. A grande maioria dos acidentes e incidentes ocorre durante a descolagem ou aterragem, já que na grande maioria dos sistemas um piloto externo assume o controlo. Ter menos intervenção humana aumenta a fiabilidade do sistema, deixando de haver necessidade de utilizar pilotos certificados. Devido ao tamanho e peso do VANT, a descolagem é facilmente realizada à mão sendo que o foco principal será a manobra de aterragem. O sistema de visão está localizado no convés do navio, o que reduz o poder de processamento, tamanho e peso do VANT necessários. A arquitetura proposta baseia-se num esquema de Filtro de Partículas *Unscented* (FPU) com duas fases: (i) *boosting* de pose, e (ii) seguimento. Na fase de *boosting* de pose, detetamos o VANT na imagem usando Redes Neurais Profundas (RNPs) e inicializamos o seguimento com um conjunto de hipóteses que descrevem a pose real do alvo usando uma base de dados pré-treinada. Na fase de seguimento, usamos uma abordagem baseada num FPU para obter uma estimativa em tempo real da verdadeira pose do alvo. Ao contrário de muitos filtros de partículas baseados em visão que amostram partículas utilizando uma distribuição baseada somente nas imagens anteriores, neste trabalho, usamos também a informação da imagem atual para melhorar a convergência do filtro. É fundida informação do instante actual e dos anteriores utilizando filtros de Transformação *Unscented* (TU) e usamos, pela primeira vez neste tipo de problema, as distribuições de Bingham e Bingham-Gauss para representar a dinâmica do sistema e o ruído de orientação. Estes filtros dependem do cálculo de pesos que usam uma aproximação sub-óptima da função de verossimilhança. Nós avaliamos diferentes métricas de semelhança que calculam a distância entre uma imagem sintética criada com a hipótese do estado e a imagem capturada. Uma vez que estamos a aproximar a função de verossimilhança, enriquecemos o filtro com passos adicionais de refinamento para reduzir a sua sub-optimalidade. Foi desenvolvido um simulador "realista" para uma análise quantitativa dos resultados. Os resultados experimentais apresentados são baseados no erro de seguimento e no tempo de processamento. Ao analisar uma sequência de aterragem com um gradiente de céu real cheio de nuvens, obtivemos aproximadamente 81% menos erro de rotação usando o Filtro de Bingham *Unscented* (FBiU) e o Filtro de Bingham-Gauss *Unscented* (FBiGaU) quando comparado com o simples uso do Filtro de Kalman *Unscented* (FKU) sem considerar o uso da fase de otimização. Quando usamos otimização de pose, conseguimos reduzir o erro de rotação obtido em mais de 50%.

Palavras-chave: *Visão artificial, Estimativa de Pose Baseada em Modelo, Seguimento Baseado em Modelo, Veículos Autónomos.*

Acknowledgments

In this space, I will thank all the persons that have contributed during the preparation and conclusion of this thesis. I will not thank in order of importance because, for me, all were important in some way - *All pieces of a puzzle are essential to be able to finish it.*

First, I would like to express my gratitude to Paulo Monteiro, João Trindade, Tiago Rento, Hugo Rodrigues, Joana Rodrigues, Mariana Sousa, Rui Colaço, and Hélder Ferreira from the Portuguese Navy for their support throughout my Ph.D. studies. Working every day and being a Ph.D. student in my vacations and free time is not easy, especially working as a military in the Portuguese Navy without the availability and time of a different type of worker or a full-time student. These fellows gave me the confidence and support to keep going without giving up.

Second, I would also want to thank some of my *VisLab* colleagues, specially Athira Nambiar, Pedro Vicente, and Rui Figueiredo, for their support during my P.h.D courses. I need to give a special thanks to my *VisLab* colleague and friend Nuno Monteiro, the first guy that I met at the very beginning of my Ph.D. studies and that was essential for me to understand the IST working mode and helped me a lot in the courses that we had together.

Next, I want to express my sincere gratitude to my supervisor Doctor Alexandre Bernardino that I can say is like a *father* for me in the computer vision area and had an immense contribution to my Ph.D. enroll and during my Ph.D. years. He had helped me not only in this thesis but also guaranteeing with is guidance and support that I was always developing valuable work during the Ph.D. courses. For sure, this thesis would not have been concluded without him since he is a fantastic person, a friend, and a mentor with vast knowledge.

I am sincerely grateful to my co-supervisor, Doctor Victor Lobo, for all the support, being the third time he was my co-supervisor. He was essential for me since he believed in me around fifteen years ago when I was a Naval Academy student without any given proofs, and we started the *GEBA* project. We have started a team of friends, and I have started working, and improving myself, increasing my knowledge and working capability. I can say for sure that without him, I did not have the confidence to enroll and the needed belief in finishing my Ph.D. studies.

Nuno Alexandre Antunes Martins Pessanha Santos ♂, July 2020.

List of Acronyms

AAG Advanced Arresting Gear.

ABC Approximate Bayesian Computation.

AdaBoost Adaptative Boosting.

API Application Programming Interface.

AR Aspect Ratio.

AUC Area Under the Curve.

AuRe Augmented Reality.

AUTOLAND AUTOnomous LANDing.

BB Bounding Box.

Bi Bingham.

BiGa Bingham-Gauss.

BPF Boosted Particle Filter.

CAD Computer-Aided Design.

CNN Convolutional Neural Network.

COBYLA Constrained Optimization BY Linear Approximations.

COG Center Of Gravity.

COTS Commercial-Off-The-Shelf.

CPU Central Processing Unit.

CUDA Compute Unified Device Architecture.

CV Computer Vision.

DCM Direction Cosine Matrix.

DNN Deep Neural Network.

DOF Degrees Of Freedom.

DT Distance Transform.

EKF Extended Kalman Filter.

ESTOL Extreme Short Take-Off and Landing.

FaR-CNN Faster Region-based Convolutional Neural Network.

FAST Features from Accelerated Segment Test.

FM Feature Map.

FPB Fast Patrol Boat.

FPN Feature Pyramid Network.

FPS Frames Per Second.

FR-CNN Fast Region-based Convolutional Neural Network.

GAbF Genetic Algorithm based Framework.

GaBi Gauss-Bingham.

GCS Ground Control Station.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

GPU Graphics Processing Unit.

IEKF Iterated Extended Kalman Filter.

IFOLS Improved Fresnel Optical Landing System.

ILS Instrument Landing System.

ILSVRC Imagenet Large Visual Recognition Challenge.

IMU Inertial Measurement Unit.

IR Infrared Radiation.

KF Kalman Filter.

LIDAR Light Detection And Ranging.

MAE Mean Absolute Error.

MAP Maximum A Posteriori.

MLE Maximum Likelihood Estimation.

MPF Mixture Particle Filter.

OBB Oriented Bounding Box.

OpenGL Open Graphics Library.

PCGM Partially-Conditioned Gaussian Mixtures.

PDF Probability Density Function.

PF Particle Filter.

PFO Particle Filter Optimization.

PRC Precision-Recall Curve.

PSO Particle Swarm Optimization.

PTOL Point Take-Off and Landing.

PTZ Pan-Tilt-Zoom.

R-CNN Region-based Convolutional Neural Network.

RGB Red, Green, and Blue.

RMSE Root Mean Square Error.

ROI Region Of Interest.

RPN Region Proposal Network.

SD Standard Deviation.

SPARS Shipboard Pioneer Arresting System.

SSD Single Shot Detector.

STOL Short Take-Off and Landing.

TLS Transponder Landing System.

UAV Unmanned Aerial Vehicle.

UBiF Unscented Bingham Filter.

UBiGaF Unscented Bingham-Gauss Filter.

UKF Unscented Kalman Filter.

UPF Unscented Particle Filter.

UT Unscented Transform.

VTOL Vertical Take-Off and Landing.

WL Weak Learner.

YOLO You Only Look Once.

Table of Contents

Chapter 1

Introduction

All that we are is the result of what we have thought.

Gautama Buddha

Chapter contents

This chapter presents the motivation and context, the objectives, the challenges, the proposed methodology, the original contributions, the published work, and the outline of the document.

1.1 Motivation and Context

Portugal has to monitor approximately 50957 km^2 of territorial waters[1]. Fast Patrol Boats (FPBs) are extensively used in patrolling (Figure 1.1) the maritime traffic to ensure that all the applicable laws are being respected. Their efficiency can be significantly improved by the support of Unmanned Aerial Vehicles (UAVs) in extending their surveillance range e.g. by transmitting georeferenced video in real-time to the FPB.

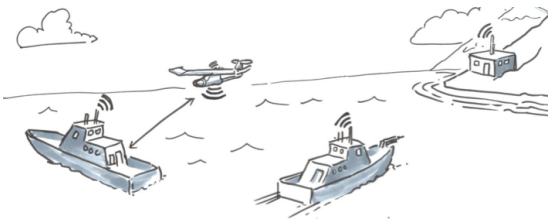

Figure 1.1: UAV patrolling mission illustration.

[1] We have approximately 16460 km^2 on the continent, 23663 km^2 on Azores, and 10834 km^2 on Madeira.

The vast majority of the accidents and incidents with UAVs occur during take-off or landing (the most challenging maneuvers) as described in Williams [2004]; Wild *et al.* [2016], since in the vast majority of the systems during this operations an external pilot takes control. Having less human intervention in these operations increases system reliability and alleviates the use of trained and certified UAV pilots. Landing on a ship is a tough task, especially in small ships that are very sensitive to the weather conditions and have a small area available for landing. Our UAV landing area has a size of around 5×6 meters (Figure 1.2 *right*). We use a net-based retention system that guarantees the safe UAV landing without disturbing the essential FPB function (Figure 1.3).

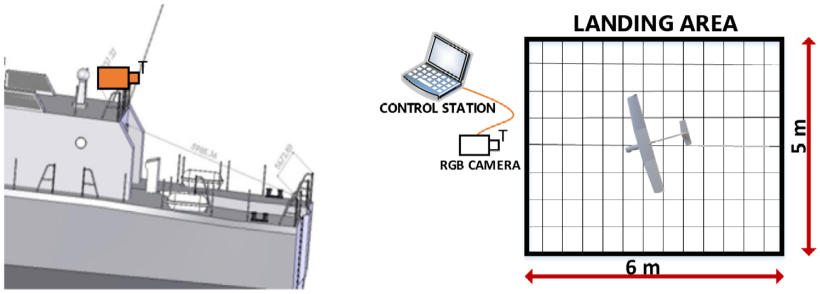

Figure 1.2: Camera location (*left*) and FPB available landing area (*right*) illustrations.

Figure 1.3: FPB 5×6 meters landing area (*red rectangle*) and net-based retention system.

A fixed-wing UAV presents a simpler structure, can carry greater payloads[2] for longer distances using less power, and have a larger endurance when compared with rotary-wing UAVs. The main disadvantages are the need for a launcher to take-off and a larger landing area. The standard mission profile for a fixed-wing UAV is described in Figure 1.4 *left*, where we can see that we have the take-off and respective climb until it reaches the mission envelope. After completing the mission, the UAV starts a descend trajectory until it performs loiters around a specific position. In this position, the UAV trajectory is chosen to take into account a state machine, as described in Figure 1.4 *right*. In the loiter stage, the UAV makes circular loiters in a predefined position waiting to be detected by the ground-based vision system. The stage transition happens when the UAV pose begins to be detected. In the approach stage, the UAV proceeds in a straight line to the landing cone[3]. In the land stage, the UAV begins the landing sequence[4]. If the UAV position is lost during the landing stage, a go-around strategy is adopted[5], and the loiter stage begins again. When the UAV is almost arriving at the landing

[2] Communications, electronics (autopilot and actuators control), sensors, and actuators.
[3] $d = 50$ meters from the landing area (Figure 1.5).
[4] Decreasing altitude and performing a predefined trajectory to the center of the landing area.
[5] Increase altitude and turn right.

area, we enter in a no return stage where we cannot avoid the landing even if we lost the UAV pose estimation. All the approximation trajectory must be performed inside a predefined landing cone that depends on the landing platform. In the landing trajectory, the UAV must be between the maximum apparent descendent angle (α_1) and the minimum apparent descendent angle (α_2), and the orientation horizontal angle (β_1) must be inside the opening horizontal angle (β_2), as described in Figure 1.5. The trajectory inside the landing cone must also be chosen taken into account the existing wind vortices and the ship's superstructures. Due to the UAV size and weight, the take-off could be made by hand (Figure 1.6 *left*), but the landing requires a simple and reliable system. The used UAV platform characteristics [Xia *et al.* , 2014; Ajaj *et al.* , 2014, 2016; Beaverstock *et al.* , 2013; Morais *et al.* , 2015] are described in Table 1.1 and the used UAV Computer-Aided Design (CAD) model is described in Figure 1.6 *right*.

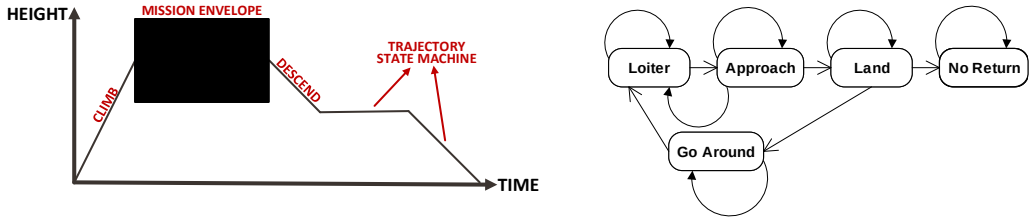

Figure 1.4: Standard mission profile (*left*) and trajectory state machine (*right*).

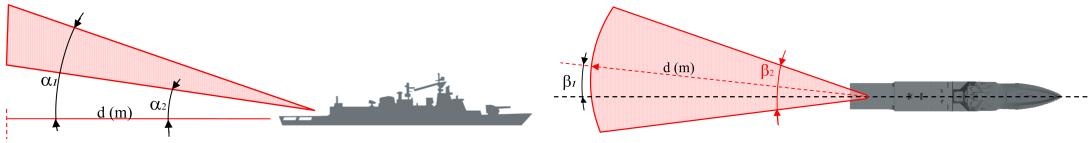

Figure 1.5: Approximation cone: lateral (*left*) and top (*right*) views.

Table 1.1: Used UAV platform characteristics.

Length:	1.2 m	**Airfoil:**	NACA 65-410
Wingspan (b):	1.8 m	**Maximum Take-Off Weight:**	5 kg
Chord (c):	0.24 m	**Cruising speed (v_{cr}):**	15 m/s
Wing area (S):	0.432 m^2	**Maximum speed (v_{max}):**	20 m/s

Some of the techniques currently used in autonomous landing are based on radar or in the utilization of the Global Positioning System (GPS) [Cho *et al.* , 2007; Smit, 2013; Xu *et al.* , 2013; Inc, 2016]. Since GPS and radar are vulnerable to jamming[6], and the radar requires large payload[7], we will explore vision-based techniques to perform this task. A vision system can increase the system autonomy and provide an alternative means for landing [Chowdhary *et al.* , 2013; Grant *et al.* , 2009; Wu *et al.* , 2013; Yang *et al.* , 2016; Zhou *et al.* , 2017]. The majority of the research made in this field is based on UAV onboard sensors and computation, using markers on the landing area to facilitate Computer Vision (CV) [Cesetti *et al.* , 2010; Lange *et al.* , 2008; Lin *et al.* , 2016; Morais *et al.* , 2015; Saripalli, 2009; Saripalli *et al.* , 2002;

[6] Intentional emission of radio frequency signals to block the signal reception.
[7] Our UAV has a maximum payload of 5 kg (Table 1.1).

Sharp *et al.* , 2001; Wenzel *et al.* , 2011; Wu *et al.* , 2013; Xiang *et al.* , 2012; Xu *et al.* , 2009; Zhao & Pei, 2013]. Instead, we propose a ground-based vision system [Hazeldene *et al.* , 2004; Kong *et al.* , 2013; Martinez *et al.* , 2009; Moore *et al.* , 2009; Kong *et al.* , 2015] installed on the ship. This allows more processing power, as well as the use of Commercial-Off-The-Shelf (COTS) UAVs with standard autopilots. The ground-based vision system computes the relative pose of the UAV concerning the landing platform from captured images, and then the Ground Control Station (GCS) [Klimkowska *et al.* , 2016] sets the trajectory of the UAV via radio communications (Figure 1.7).

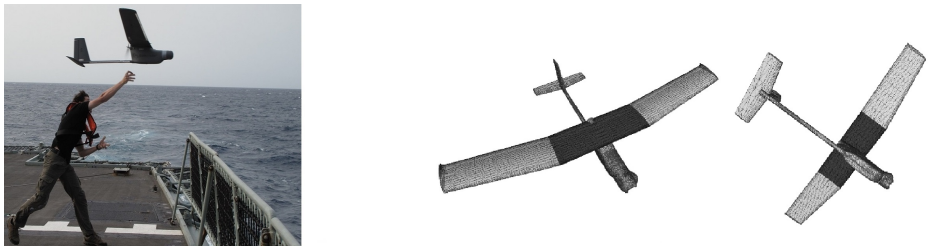

Figure 1.6: Real UAV take-off onboard a ship (*left*) and the used UAV CAD model (*right*).

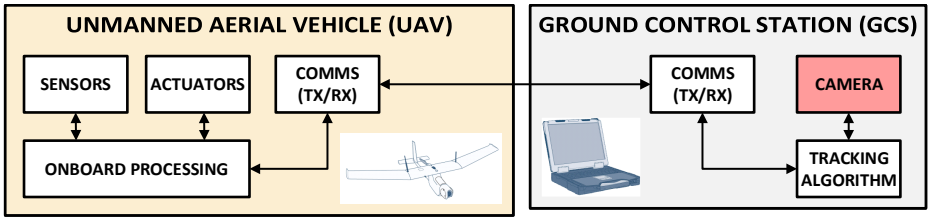

Figure 1.7: General scheme illustration.

1.2 Objectives

In this thesis, we aim to develop a monocular Red, Green, and Blue (RGB) ground-based vision system that estimates the UAV pose concerning the camera reference frame to perform tracking. The system should be able to operate in outdoor scenarios guaranteeing an error compatible with the automatic landing requirements. A simplified control structure is described in Figure 1.8, where we highlight the contribution of the thesis: (i) the camera as a sensor (to capture images), and (ii) a tracking algorithm. No other UAV or ground sensor information will be used in the estimation.

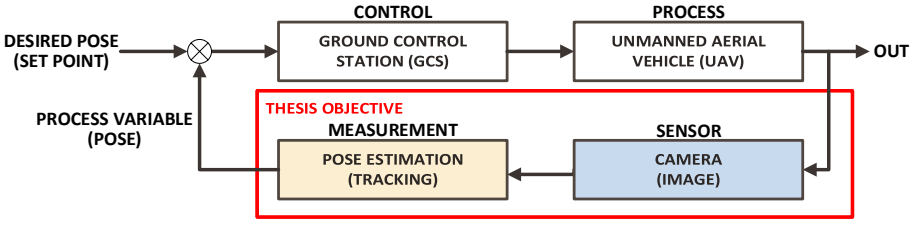

Figure 1.8: Simplified control structure.

1.3 Challenges

In our approach, it is used a monocular RGB camera located on the ship (Figure 1.2 *left*) with a processing station to perform the needed CV processing tasks (Figure 1.7). The system observes and obtains the UAV pose and sends that information to the GCS that computes the needed control commands (Figure 1.8) to guide the UAV via radio to perform autonomous landing using a net-based retention system (Figure 1.3). There are many challenges when we are trying to automate this complex operation in a real-world scenario. The initial UAV detection is difficult since the search area is vast, and we are operating in an outdoor environment with external conditions that we cannot control (Figure 1.9). In addition to that, we are using a small size UAV, making it difficult to detect at far distances. We also have a small landing area of 5×6 meters (Figure 1.2 *right*). Our UAV has 1.8 meters of wingspan (Table 1.1), which is almost 40% of the landing area width. We further need to deal with changing backgrounds and motion blur since we have a moving landing platform. The landing platform operates in a maritime environment where we can have fog and high humidity. To tackle these issues, we need to have a robust system that exploits the maximum image information possible to be able to estimate the UAV pose with low error. A summary of the main challenges can be seen in Figure 1.10.

Figure 1.9: Outdoor captured real UAV images.

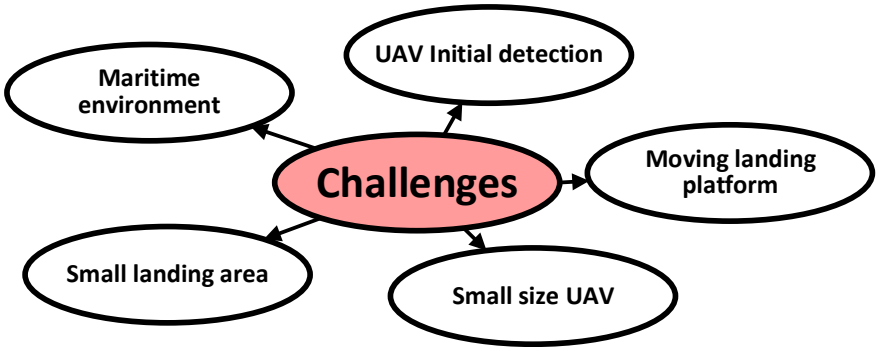

Figure 1.10: Main challenges.

1.4 Methodology

The proposed method is divided into two major parts (Figure 1.11):

- **Pose boosting** (Chapter 4) - In the pose boosting stage, we detect the UAV on each image frame and generate pose (3D position and orientation) hypotheses using a pre-trained database. In the detection, we have to address the challenges of detection in an outdoor scenario using a moving platform. We retrieve multiple pose hypotheses from the database (one-to-many relation) to be able to reduce the existing ambiguities (due to the UAV model geometry) and obtain a low estimate error. These hypotheses will be used for tracking in a filtering structure to reduce even more the existing ambiguities by using temporal information;
- **Tracking** (Chapter 5) - In this stage, we use the UAV CAD model combined in a filtering structure to perform UAV pose tracking. We explore multiple filter variants, including the use of distributions to better represent the motion and observation models, decreasing the estimated error between iterations. To be able to decrease even more the error due to the sub-optimal adopted filtering method, we also perform local optimization to search in the state space for better estimates.

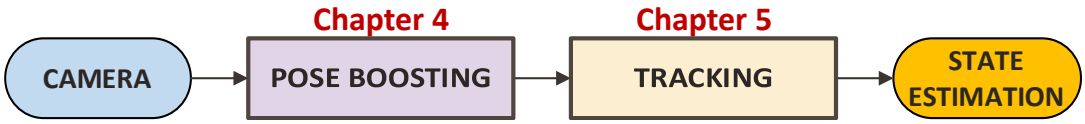

Figure 1.11: System architecture (*simplified*).

1.5 Original contributions

As a result of the developed work, the following contributions were made:

- ***Development and performance analysis of a ground-based vision system architecture to perform UAV tracking using a 3D model-based approach*** - Our first direct contribution is the proposal of a novel ground-based pose tracking framework for UAV landing using a 3D model-based approach. 3D model-based algorithms are good estimation methods but require large processing power. This processing power is easily available in a ground-based system since it does not present any payload restrictions when compared to the UAV. To be able to validate the developed system architecture, it is essential to compare it with more traditional methods. The tests performed show that the developed scheme has the best performance among all the tested approaches;
- ***The validation in a developed "realistic" simulator environment and real images*** - Since in the real images captured until now we do not have ground truth data, we have developed a "realistic" simulator to be able to validate the developed system stages and the overall architecture. The validation in real images is also performed, being able to infer the target detection performance and present a qualitative analysis of the overall system performance;

- ***The development and analysis of alternative similarity metrics used to approximate the observation likelihood function*** - Since we are tackling a very complex problem (UAV pose estimation), the likelihood function is hard to obtain. One of the solutions is to approximate it using a similarity metric as a proxy. However, these approximations are coarse and may be insufficient to capture all aspects of the true likelihood. We have developed similarity metrics that use the UAV CAD model to generate a computer graphics image on the UAV hypothetical pose and compare it with the real image with some robustness to background clutter, occlusion. These metrics were analyzed in a "realistic" simulator environment to demonstrate their strengths and weaknesses;
- ***Development of a pose boosting method that uses current image frame information to obtain a rough pose estimate*** - The original Boosted Particle Filter (BPF) [Okuma *et al.* , 2004] implementation uses a detector to improve and add diversity in the proposal generation. In our approach, we also use a detector to obtain the target position on the frame but extend it with rough estimates of depth and pose using a pretrained database of images indexed by bounding box properties. A key contribution was the fast appearance-based initialization using the information given by a corner detection algorithm. When using a corner detection algorithm, we can deal with illumination changes frequent in outdoor environments;
- ***Training of Deep Neural Networks (DNNs) for object detection using a synthetically generated dataset for transfer learning*** - Since there is no publicly available database, a training dataset generation scheme was developed using the UAV CAD model and real background images, varying the pose randomly around a predefined interval. The annotations are automatically generated to be able to train the network and perform transfer learning to real images. This data generation framework can be applied to any UAV provided its CAD model. The tests performed with a real dataset show the effectiveness of this approach;
- ***The use of directional statistics distributions in the UAV tracking, to improve the obtained orientation estimation*** - In the developed pose tracking framework, we have studied the combination of two different directional statistics distributions: (i) the Bingham (Bi), and (ii) the Bingham-Gauss (BiGa). In a periodic domain like the manifold of orientations in a 3D space, the Gaussian model is not a good approximation, especially in the presence of strong noise. The Bi distribution can be used in a filtering structure to model the periodic nature of rotations better. However, the Bi noise is specific to the angular position component and does not capture its correlation with the angular velocity. We introduce BiGa noise to model the full rotational noise, both in its angular position and velocity components. Until now, no other UAV tracking system or 3D model-based tracking system addressed the use of these distributions in a filtering structure. The proposed framework has been tested in a "realistic" simulator and significantly improves orientation estimation accuracy;
- ***The use of a pose optimization step to improve the estimate*** - A pose optimization step is used to improve the estimate in the time between measurements. Since we are using approximated observation likelihood functions (similarity distance metrics), and we use a limited number of particles, a local optimization step can decrease the estimation error. A novel particle pose optimization stage named Genetic Algorithm based Frame-

work (GAbF) based on the evolution strategies present in the genetic algorithms have also been proposed. When using the GAbF modified *crossover* and *mutation* operators, we obtain superior performance in the pose estimation task. Several pose optimization approaches were tested, and results show significant improvements in the overall system accuracy;

- ***Graphics Processing Unit (GPU) implementation and test*** - To be able to decrease the processing time and increase the real-time capability of the system, the frame radial and tangential distortion correction, the target detection, UAV rendering, and similarity metric calculation were implemented entirely in the GPU. The UAV CAD model was also simplified to increase the rendering speed without loss of accuracy. It was possible to decrease the needed processing time, increasing the real-time capability of the system.

1.6 Published work

This thesis document was based on the written conference and journal articles, as described in Figure 1.12. A complete summary of the published work and developed projects is given in Appendix F.

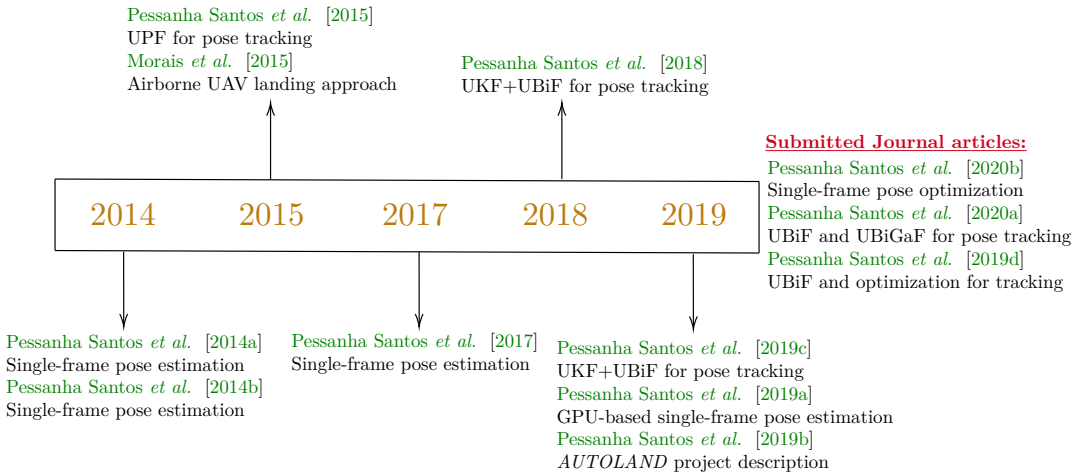

Figure 1.12: Published work.

Until now, eight articles have already been submitted and accepted at conferences [Pessanha Santos *et al.* , 2014a; Morais *et al.* , 2015; Pessanha Santos *et al.* , 2015, 2017, 2018, 2019c,a,b]. In Pessanha Santos *et al.* [2014a], is used a Particle Filter (PF) based approach for single-frame UAV pose estimation. In Morais *et al.* [2015], the UAV trajectory was estimated relative to the landing area using the UAV RGB camera to detect markers (lights) located on the ship's deck. In Pessanha Santos *et al.* [2015, 2017, 2018, 2019c], a vision system based on a standard RGB camera was used to track a UAV landing aboard a ship. In Pessanha Santos *et al.* [2015], is used a PF for pose estimation and an Unscented Kalman Filter (UKF) for filtering while Pessanha Santos *et al.* [2017] used a novel resampling step based on the evolution strategies found in the genetic algorithms. A comparison of the developed approach

with ten traditional resampling schemes showed the best performance among those tested. In Pessanha Santos *et al.* [2018], is used a PF for pose estimation combined with a UKF for the translational motion filtering and an Unscented Bingham Filter (UBiF) for the rotational motion filtering. In Pessanha Santos *et al.* [2019c], is also used a PF for pose estimation combined with a UKF for the translational motion filtering and a UBiF for the rotational motion filtering but showing new results that illustrate the effectiveness of the approach. In Pessanha Santos *et al.* [2019a], is analyzed the computational performance of a GPU-based approach for real-time single-frame UAV pose estimation. In Pessanha Santos *et al.* [2019b], the developed CV landing strategies during the AUTOnomous LANDing (AUTOLAND) project are presented and illustrated.

We have three journal articles accepted and published [Pessanha Santos *et al.* , 2014b, 2020b,a], and one submitted under review [Pessanha Santos *et al.* , 2019d] that will be improved, taking into account the reviewers comments. In Pessanha Santos *et al.* [2014b], it is presented a RGB monocular ground-based vision system for single frame UAV pose estimation. In Pessanha Santos *et al.* [2020b], the single frame pose estimation approach is explored as an optimization problem showing new methods and results. In Pessanha Santos *et al.* [2020a], we explore the use of directional statistics for UAV tracking, developing a new filter based on a directional statistic distribution that correlates attitude and angular velocity. In Pessanha Santos *et al.* [2019d], due to the sub-optimality of the adopted filter, we explored the use of refinement steps between observations to decrease the obtained UAV tracking error.

1.7 Thesis outline

In Chapter 2, we will present the related work concerning the UAV take-off and landing, retention systems, landing guidance systems, vision-based landing systems, object detection, 3D model-based pose estimation and pose tracking, including UAV applications. In Chapter 3, the problem formulation and the used methodologies are explained. In Chapter 4, the target detection method and the used pre-trained database for pose boosting are explored. In Chapter 5, the applied tracking architecture is described in detail. In Chapter 6, the obtained experimental results are presented and evaluated regarding pose estimation error and processing time. Finally, in Chapter 7, we present a summary of the research detailing its conclusions and suggest directions for future research in this area.

Chapter 2

Related work

I hear, I know. I see, I remember. I do, I understand.

Confucius

Chapter contents

This chapter makes a review of the related work both on the general goal of the thesis (UAV take-off and landing, retention systems, landing guidance systems, and vision-based UAV landing systems) and on the components of the proposed methodology (object detection, 3D model-based pose estimation, and pose tracking).

2.1 UAV take-off and Landing

The first Vertical Take-Off and Landing (VTOL) UAV autonomous landing onboard a ship was performed by the helicopter *MQ-8 Fire Scout* [Petrescu *et al.* , 2017; Lin *et al.* , 2017a]. *Fire Scout* is currently in use by the US Navy and can autonomously take-off and land from an aviation capable ship. *Scorpion* is a remotely piloted UAV with Extreme Short Take-Off and Landing (ESTOL) capability [Ro *et al.* , 2007], being able to tilt the thrust vector without changing the attitude of the aerodynamic surfaces [Ahmed *et al.* , 2015; Valavanis & Vachtsevanos, 2015; Barton, 2012]. This design confers additional flexibility and resistance against turbulence and stall, which is especially helpful for launch and recovery from ships in rough seas. *Fanwing* is a family of Short Take-Off and Landing (STOL) UAVs featuring a new lift concept and unique design based on a cross-flow fan along the wingspan [Li, 2013; Seyfang,

2012, 2011]. *Fanwing* was designed for high maneuverability, providing a short take-off length (less than 3 meters).

Due to the used UAV model size and weight (Figure 1.6), the take-off is easily performed by hand (Figure 2.1 *left*), so the main focus will be in the landing maneuver (Figure 2.1 *right*). The final system should be able to automatically perform the UAV landing with minimal human supervision and without structural changes in the existing platform.

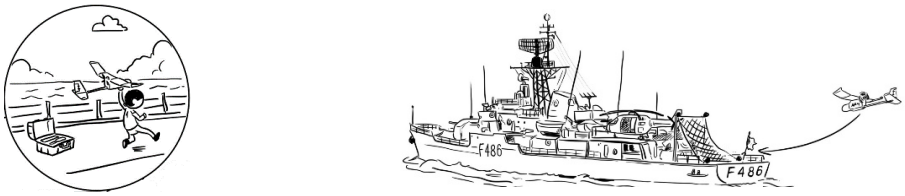

Figure 2.1: UAV take-off (*left*) and landing (*right*) illustrations.

2.2 Retention systems

The traditional retention system's objective is to decelerate a UAV as it lands rapidly. The Advanced Arresting Gear (AAG) is a modular, integrated tailhook retention system [Ma, 2003; Mendoza *et al.* , 2007]. During normal operation in a carrier deck, the tailhook engages on a wire, and the kinetic energy is transferred to the deck systems. Point Take-Off and Landing (PTOL) [Yoffe, 2017] refers to a type of landing where there is no requirement for a landing area. The *Skyhook* system [Eldridge *et al.* , 2009; Klausen *et al.* , 2016; Sørbø, 2016] uses a hook with a shock cord to be able to perform an abrupt UAV stop. Shipboard Pioneer Arresting System (SPARS) [Reuter & Greenstadt, 1988] is a net-based retention system used for shipboard operations. It has been used with the *Pioneer* UAV system by the US Navy [Gleason & Fahlstrom, 2010].

In our approach, we are using a net-based retention system (Figure 2.2), that guarantees the UAV safe landing without disturbing the essential FPB function.

Figure 2.2: Net-based retention system (*experimental test example*).

2.3 Landing guidance systems

Section contents

There are two main types of landing guidance system strategies: (i) non-cooperative (Section 2.3.1), and (ii) cooperative (Section 2.3.2). In the first type, the environment is not prepared to assist in the landing maneuver, and the UAV must have the ability to land without any exterior assistance. In the second type, the environment is prepared, including dedicated systems.

2.3.1 Non-cooperative guidance

In unknown environments, complex algorithms are required to recognize patterns in sensor data to be able to perform the UAV landing. We can use an Improved Fresnel Optical Landing System (IFOLS) [Gajjar & Zalewski, 2004] to give glide path information in the final landing phase or CV to measure the UAV angular velocity [Wang *et al.* , 2007].

2.3.2 Cooperative guidance

Cooperative approach landing guidance systems are typically based on radio-frequency communication between the ground and the UAV, providing information on the position of the UAV compared with the desired approach route. Some examples of radio-frequency systems are Instrument Landing System (ILS) [McLees *et al.* , 2018; Chisholm, 1989], Transponder Landing System (TLS) [Winner & Kuehn, 2002], and Global Navigation Satellite System (GNSS) [Stempfhuber & Buchholz, 2011].

In the developed approach, we use a cooperative method employing a monocular RGB ground-based system using the UAV CAD model, as will be described in Section 3.5.

2.4 Vision-based UAV landing systems

The vast majority of the vision-based landing systems [Kong *et al.* , 2014] use the onboard camera and external markers [Lange *et al.* , 2008; Merz *et al.* , 2006; Saripalli, 2009; Saripalli *et al.* , 2002, 2003; Sharp *et al.* , 2001; Wenzel *et al.* , 2011; Xiang *et al.* , 2012; Zhao & Pei, 2013; Yang & Tsai, 1998; Huh & Shim, 2010; Sereewattana *et al.* , 2015] to waive the use of GPS. The autonomous landing of a small fixed-wing UAV into a net using a ground-based vision system without GPS was successfully tested by Kim *et al.* [2013]. Other systems automatically detect existing runways using Infrared Radiation (IR) lamps [Gui *et al.* , 2013] or use an onboard database of known runways to perform image registration and control the UAV orientation [Miller *et al.* , 2008; Williams & Crump, 2012] e.g. for emergency landing [Fitzgerald *et al.* , 2005; Hubbard *et al.* , 2007; Mejias *et al.* , 2009].

Some methods are based on the use of the onboard camera to detect the “H” international landing mark and estimate the relative pose [Lin *et al.* , 2016; Saripalli, 2009; Saripalli *et al.* , 2002; Wenzel *et al.* , 2011; Zhao & Pei, 2013; Saripalli *et al.* , 2003; Yang & Tsai, 1998; Mondragón *et al.* , 2010; Lin *et al.* , 2015] or introducing a “T” form artificial mark with high emissivity[1] combined with a IR camera to detect the relative pose to the UAV [Xu *et al.* , 2009].

[1] The ratio of energy radiated from a material’s surface to that radiated from a perfect emitter at the same wavelength, temperature, and viewing conditions.

In our approach, it is used a monocular RGB camera located on the ship with a processing station to perform the needed CV processing tasks. The system observes the UAV and computes the control commands to send to the UAV via radio to perform autonomous landing using a net-based retention system (Figure 2.3). Since a ship is a moving platform at sea, we have continuously changing backgrounds (sea and clouds).

Figure 2.3: Landing area (*orange*), RGB camera (*yellow*), and communications (*green*).

2.5 Object detection

Object detection is one of the most critical tasks in CV. We need to estimate the UAV location on the captured frame before estimate and track its pose. This task is very challenging since we have a wide variation of possible UAV poses (3D position and orientation). The boosted cascade classifier of Viola & Jones [2001], was the first system to obtain a high detection accuracy in real-time. A boosted cascade classifier consists of stages, each with an ensemble of Weak Learners (WLs). A WL is a learning algorithm that produces a classifier that can label data with an accuracy above chance[2] [Ferreira & Figueiredo, 2012; Freund & Schapire, 1995; Vaghela *et al.* , 2009]. Its performance profits from an efficient classifier structure and the use of fast-to-compute *hand-crafted* features. Most of the entries in Imagenet Large Visual Recognition Challenge (ILSVRC) [Russakovsky *et al.* , 2015] until 2012 are based on *hand-crafted* feature extraction and classification for object detection [Felzenszwalb *et al.* , 2008; Fidler *et al.* , 2013]. In 2012, the ILSVRC winner achieved half the error of previous entries using a DNN [Krizhevsky *et al.* , 2012]. After 2012, almost all the entries on ILSVRC for object detection use DNNs [Sermanet *et al.* , 2013]. In this thesis, we will explore DNNs that are the current state-of-the-art in classification and object detection.

Convolutional Neural Networks (CNNs) are a specific type of DNNs explicitly designed to deal with the variability of 2D shapes that are showing great results [Krizhevsky *et al.* , 2012; LeCun *et al.* , 1998; Szegedy *et al.* , 2015]. Approaches like Region-based Convolutional Neural Network (R-CNN), Fast Region-based Convolutional Neural Network (FR-CNN), Faster Region-based Convolutional Neural Network (FaR-CNN), Single Shot Detector (SSD) and You Only Look Once (YOLO) use a CNN framework [Girshick, 2015; Girshick *et al.* , 2014; He *et al.* , 2014; Ren *et al.* , 2015; Redmon & Farhadi, 2016; Liu *et al.* , 2016b; Redmon & Farhadi, 2018] (Table 2.1). With R-CNN and FR-CNN, the object region proposals are extracted using

[2] A simple *rule-of-the-thumb* classifier with $error < 0.5$ in the binary case.

selective search[3] [Uijlings *et al.* , 2013; Wang *et al.* , 2015] and only these regions are processed by the network, which saves important computational resources. FaR-CNN does the detection in a single forward pass using a Region Proposal Network (RPN) [Ren *et al.* , 2015]. SSD uses a single network (unified network) to generate objects presence scores in a discrete space of default Bounding Boxes (BBs) at different Aspect Ratios (ARs) and scales for each Feature Map (FM)[4]. This information is combined with predictions from multiple resolutions FMs to be able to adjust the BBs to represent the object shape better. It also benefits from extensive use of data augmentation (e.g. random crop strategy and color distortion as described in Liu *et al.* [2016b]) to reduce overfitting[5]. From the current existing real-time object detection implementations, the YOLO (Table 2.1) is faster than other detection systems and can run in multiple image sizes with a trade-off between speed and accuracy [Redmon & Farhadi, 2016, 2018]. This unified network uses the same data augmentation scheme used by the SSD network.

Table 2.1: Examples of CNNs for object detection.

Name:	Object region proposals:	Notes:
R-CNN [Girshick *et al.* , 2014]	Selective search [Girshick *et al.* , 2014]	The first solution for object detection, slow training and inference
FR-CNN [Girshick, 2015]	External algorithm	Training in a single stage, horizontal flipping as data augmentation scheme
FaR-CNN [Ren *et al.* , 2015]	RPN [Ren *et al.* , 2015]	Replace of the slow selective search by RPN (neural network)
SSD [Liu *et al.* , 2016b]	———	Unified framework and extensive use of data augmentation
YOLO [Redmon & Farhadi, 2018]	———	Unified framework, data augmentation similar to SSD and a faster detection than the other described networks

Depending on the problem, acquiring ground truth data is time-consuming and expensive, and the use of synthetic data can be a solution, especially in CV problems. The generated synthetic data should be as realistic as possible with an accurate annotation for training. It has been successfully used in object detection [Tremblay *et al.* , 2018; Hartwig & Ropinski, 2019], pose estimation [Tremblay *et al.* , 2018; Jalal *et al.* , 2019; Kiru Park & Vincze, 2017; Xing *et al.* , 2017], stereo vision [Zhang *et al.* , 2016a], and to obtain the disparity/flow [Mayer *et al.* , 2016]. In our implementation, we are using a synthetically generated training database using the UAV CAD model to train DNNs for UAV detection, as will be described in Section 4.2.

2.6 3D model-based pose estimation

Section contents

Pose estimation is the problem of determining the position and orientation of a rigid object relative to a fixed reference frame. The class of methods that use the object or environment CAD model to generate features and perform pose estimation are called 3D model-based pose

[3] Selective search is one alternative to exhaustive search with a sliding window, starting with over-segmentation merging similar regions and produce region proposals [Girshick *et al.* , 2014].
[4] The output of different units (neurons) creates a FM or activation map. Units in a convolutional layer share the same weights and bias between them.
[5] The classifier will perform well on the training data but poorly in the presence of new data not being able to generalize.

estimation. The main applications of 3D model-based pose estimation are Augmented Reality (AuRe) [Reitmayr & Drummond, 2006; Skrypnyk & Lowe, 2004; Seo *et al.* , 2011; Wuest *et al.* , 2005], robot manipulation [Vicente *et al.* , 2016; Choi & Christensen, 2010], robot navigation [Li-Chee-Ming & Armenakis, 2015], autonomous robots [Lwin *et al.* , 2019], object pose tracking [Kyrki & Kragic, 2011; Chliveros *et al.* , 2013; Seo *et al.* , 2013; Tsai *et al.* , 2015; Lourakis & Zabulis, 2013; Pauwels *et al.* , 2013; Cao *et al.* , 2016], human pose estimation [Pons-Moll & Rosenhahn, 2011], hand pose estimation [de La Gorce *et al.* , 2011], face reconstruction [Jiang *et al.* , 2018], among others. Some of the works to perform 3D model-based pose estimation are based on simple features such as e.g. edges [Lowe *et al.* , 1991; Klein & Murray, 2006; Klein & Drummond, 2003; Seo *et al.* , 2011; Wuest *et al.* , 2005], key points [Skrypnyk & Lowe, 2004; Vacchetti *et al.* , 2004b; Artieda *et al.* , 2009], contour extraction [Kosaka & Nakazawa, 1993; Lowe, 1992; Azad *et al.* , 2011], the combination between contours and edges [Chliveros *et al.* , 2013], or the combination between edges and texture information [Vacchetti *et al.* , 2004a]. The 3D model-based pose estimation methods can be divided into [Lepetit *et al.* , 2005; Zhong & Zhang, 2019]: (i) feature-based, (ii) edge-based, (iii) direct, and (iv) region-based. In the feature-based methods, we need to have a textured object to be able to obtain sufficient key points for correspondence [Artieda *et al.* , 2009], and in the edge-based, we perform 3D edge correspondence in the image usually obtaining a high sensitivity to noise leading to several local minima [Dambreville *et al.* , 2010]. The direct methods focus on the pixel values relying on the image gradient [Seo & Wuest, 2016; Zhong *et al.* , 2018], and the region-based methods focus on image statistics [Tjaden *et al.* , 2017]. Image statistics are not reliable in complex scenes but are more robust to illumination change when compared with the direct pixel value analysis [Zhong & Zhang, 2019]. Some more recent applications of direct methods e.g. model the illumination under the Lambert assumption [Seo & Wuest, 2016] or use a texture model for online template matching using pixel information [Zhong *et al.* , 2018]. Most of the region-based methods are based on the real-time *PWP3D* algorithm [Prisacariu & Reid, 2012] that defines a pixel-based posterior energy function that performs better when compared with pixel-based likelihoods. Other region-based methods use e.g. a monocular camera to obtain local color histograms and perform template matching [Tjaden *et al.* , 2017]. The two main existing approaches in the UAV field are divided into: (i) airborne (Section 2.6.1), and (ii) ground-based systems (Section 2.6.2).

2.6.1 Airborne systems

The vast majority of the current airborne 3D model-based pose estimation UAV systems use the CAD model of known objects. We can use the RGB UAV onboard camera and the 3D wireframe model of the environment to perform feature matching using a moving edges tracker algorithm [Li-Chee-Ming & Armenakis, 2015] and even combine this information with sensor data to get the velocities needed to perform control [Teuliere *et al.* , 2010]. These methods can be combined with temporal filtering techniques such as the PF to improve accuracy [Teuliere *et al.* , 2015] or with CNNs to detect dynamic objects and minimize the estimation error [Buyval *et al.* , 2017].

2.6.2 Ground-based systems

We did not find any ground-based vision system that estimates the UAV pose using its CAD model. The vast majority of the ground-based systems are developed for tracking or UAV control without using CAD information. By using the CAD model, we can represent our knowledge about the problem (*a priori* information) and use it to estimate the UAV pose. Some developed ground-based systems for UAVs are based on RGB stereo vision and perform image segmentation using a recursive algorithm to obtain the UAV range and altitude for landing [Hazeldene *et al.* , 2004] or combine a Chan-Vese segmentation algorithm [Chan & Vese, 2001; Osher *et al.* , 2004] for the UAV detection and then fuse that information with sensor data using an Extended Kalman Filter (EKF) to perform control [Tang *et al.* , 2016]. Others use IR stereo vision and combine an active-contour based algorithm with mean-shift to detect and track the UAV [Kong *et al.* , 2013] or use Trinocular cameras to apply a color-based algorithm based on probability distributions to extract four different color landmarks located on the UAV to provide visual feedback to the flight controller [Martinez *et al.* , 2009].

Nowadays, all UAVs have their 3D CAD model available, so their pose can be estimated using a class of methods for 3D model-based pose estimation. We propose a 3D model-based ground-based vision system [Hazeldene *et al.* , 2004; Kong *et al.* , 2013; Martinez *et al.* , 2009; Moore *et al.* , 2009] with high processing capability, which allows reducing the UAV size, weight, and power requirements. This approach makes it also possible to use standard UAVs equipped with COTS autopilots. Using a ground-based system, we can estimate the UAV pose using single frame information (Section 5.5) or use a filtering scheme to decrease the obtained estimate error and perform tracking (Section 3.5).

2.7 Pose tracking

Given several measures over time, we can use target-specific dynamic models to filter the sensor data using a Kalman Filter (KF) [Lefferts *et al.* , 1982; Shuster, 1989; Humpherys *et al.* , 2012], a EKF [Markley *et al.* , 1994; Humpherys *et al.* , 2012], a UKF [Crassidis & Markley, 2003; Kraft, 2003; VanDyke *et al.* , 2004; Wan & Van Der Merwe, 2000] or a PF [Cheng & Crassidis, 2004; Oshman & Carmi, 2004; Kantas *et al.* , 2015; Abdelali *et al.* , 2015; Ng & Delp, 2009]. PFs in CV were applied to visual tracking tasks such as ball tracking [Taiana *et al.* , 2008; Xia & Wu, 2015], spherical pendulum tracking [Myhre & Egeland, 2015], human face tracking [Chang & Ansari, 2005], articulated object tracking [Gonzales & Dubuisson, 2015], vehicle tracking [Chan *et al.* , 2012], object tracking [Bohyung Han *et al.* , 2004; Chang *et al.* , 2005; Sugandi *et al.* , 2011], or arbitrarily shaped 3D objects pose (rotation and translation) estimation [Azad *et al.* , 2011]. A variation of the generic PF is the Mixture Particle Filter (MPF) where each component (*mode*) is modeled with an individual PF [Vermaak *et al.* , 2003]. The BPF extends the mixture PF to enrich the proposal distribution with new detections from Adaptative Boosting (AdaBoost) [Okuma *et al.* , 2004]. The combination of a PF with a UKF, known as an Unscented Particle Filter (UPF) [Van Der Merwe *et al.* , 2001; Rui & Chen, 2001a; Mohammadi & Asif, 2011; Birsan, 2005], is described in Li *et al.* [2003] for visual contour tracking, and in Guo & Qin [2007] for ground maneuvering target tracking. This approach generates better proposal distributions for the PF, taking into account the current

observations in a well know UKF structure [Guo *et al.* , 2007]. PFs can be combined with other types of filters to generate better proposal distributions e.g. the Iterated Extended Kalman Filter (IEKF) that computes the updated state as a Maximum A Posteriori (MAP) estimate [Bar-Shalom *et al.* , 2004; Liang-Qun *et al.* , 2005; Kyrki & Kragic, 2011]. It is also possible to use a local optimization algorithm where we apply multiple PF iterations at the same time instant to fine-tune the pose estimate. This approach is commonly known as Particle Filter Optimization (PFO) [Zhou & Chen, 2013; Liu *et al.* , 2016a; Zhang *et al.* , 2007]. We can also use a different approach and combine a PF with Particle Swarm Optimization (PSO) [Krzeszowski *et al.* , 2010; Zhang *et al.* , 2015; Eberhart & Kennedy, 1995; Nedjah & de Macedo Mourelle, 2006; Shi *et al.* , 2001; Trelea, 2003] to perform local optimization, where each particle updates its state vector, taking into account its history and its neighbors. It has been applied e.g. to human motion tracking [Saini *et al.* , 2014; Zheng & Meng, 2007], image registration [Khan & Nystrom, 2010], object tracking [Zheng & Meng, 2007], and multi-object tracking [Kwolek, 2013] using CV. When using a PF based approach, we have to obtain the likelihood function between iterations. That is often not possible since we may not know its analytical expression or it can be computationally hard to obtain. To solve this issue, we can use a likelihood-free approach [Sigges *et al.* , 2017; Owen *et al.* , 2015; Marjoram *et al.* , 2003; Flury & Shephard, 2011; Liu & West, 2001a], where the particle weights are approximated (employing a distance metric) using a simulation scheme to model the system parameters. The increase of the computational processing capability allows the approximation of very complex models. This can be seen as a derivation from the Approximate Bayesian Computation (ABC) algorithm [Pritchard *et al.* , 1999].

As initially described in Section 1.4, we have developed a pose tracking architecture based on a UPF. This structure has the objective of combining the strengths of some existing approaches and allowing the development of new ones. Our main contributions until now (Section 1.5) were the developed 3D model-based ground-based vision system architecture for UAV tracking, the use of a pre-trained database for pose boosting (Section 4.1), the inclusion of a pose optimization stage to improve the estimate in the time between measurements (Section 5.5), the use of a new methodology named GAbF (Section 5.5.4) to perform local optimization, the use of directional statistics distributions to improve the orientation estimation (Appendix C), the comparison with more traditional methods (Chapter 6), the validation on simulation and real images (Section 6.2), and a GPU-based implementation to decrease the processing time and increase the real-time capability of the system (Section 6.8).

2.8 Directional statistics

When using Gaussian filtering techniques (e.g. EKF [Markley *et al.* , 1994]) for attitude estimation, it is typically considered a small angle assumption in the state and observation noises [Crassidis & Markley, 2003; Darling & DeMars, 2016b; Markley & Crassidis, 2014; Pessanha Santos *et al.* , 2015] so that a Gaussian distribution can well approximate the posterior distribution. For large angular variations, we should use a directional statistic to represent better the true probability distribution [Kurz *et al.* , 2013, 2014a]. In 1D applications are typically used the wrapped normal or the von Mises distribution [Mardia & Jupp, 2000; Jammalamadaka & Sengupta, 2001].

The Bi distribution [Bingham, 1974] is defined directly on the unit multidimensional hypersphere [Gilitschenski *et al.* , 2014]. It has been successfully used in a filtering structure to: (i) estimate the attitude of a ping pong ball [Glover & Kaelbling, 2014], (ii) to predict objects pose using point cloud data [Glover *et al.* , 2012], (iii) for estimation of orientations in the 3D space with unit quaternions [Shuster, 1993] using a predict-update framework [Kurz *et al.* , 2014b], and (iv) using a UKF based approach using deterministic sampling [Gilitschenski *et al.* , 2016]. Some mixtures and combinations of distributions have also been made to quantify the correlation between *Euclidean* states (e.g. angular velocities) and the attitude on its manifold. Some of these implementations are the Partially-Conditioned Gaussian Mixtures (PCGM) [Darling & DeMars, 2016b], the Gauss-Bingham (GaBi) [Darling & DeMars, 2015a,b] and the BiGa [Darling & DeMars, 2016a]. These approaches allow us to correlate the attitude and angular velocity uncertainties.

As described in Section 1.5, until now, no other UAV tracking system uses directional statistics distributions (Appendix C). Since the Bi distribution (Section C.1) is defined directly in the unit multidimensional hypersphere, we have used it in a filtering structure to be able to quantify the existing attitude uncertainty on its manifold (Section 5.3.1). We have also used the BiGa distribution (Section C.2) in a filtering structure to correlate the estimated UAV angular velocity with the attitude to decrease the obtained attitude error (Section 5.3.2).

Chapter 3

Problem formulation and Methodologies

Anyone who has never made a mistake has never tried anything new.

Albert Einstein

Chapter contents

This chapter presents the problem formulation, the filtering techniques used to develop the proposed tracking architecture (PF, BPF, and UPF), and describes the overall system proposal.

3.1 Problem formulation

Section contents

As initially described in Section 1.2, the main objective is to perform UAV tracking, using image information, with respect to the camera reference frame. In the system design, we have made the following assumptions:

- The UAV follows a constant velocity model (Section 3.1.4);
- The FPB and wind perturbations are not explicitly modeled;
- The UAV CAD model is available (Figure 1.6 *right*);

- The intrinsic camera parameters are known (Section 3.1.7).

To be able to formulate the problem at hand, we will define the used reference frames for position and orientation representation (Section 3.1.1), the adopted state model (Section 3.1.2), introduce the state estimation framework (Section 3.1.3), the motion models (Section 3.1.4), the state transition model (Section 3.1.5), the observation model (Section 3.1.6), and the adopted vision system camera model (Section 3.1.7).

3.1.1 Reference frames

The UAV reference frame (Figure 3.1), is based on the following assumptions [Beard & McLain, 2012; Dobrokhodov, 2015; Klein & Morelli, 2006; Zheng *et al.* , 2017; Zhu *et al.* , 2011]:

- The UAV is considered a rigid body;
- The UAV mass and mass distribution remains constant during operation;
- The UAV reference frame origin is located in its Center Of Gravity (COG).

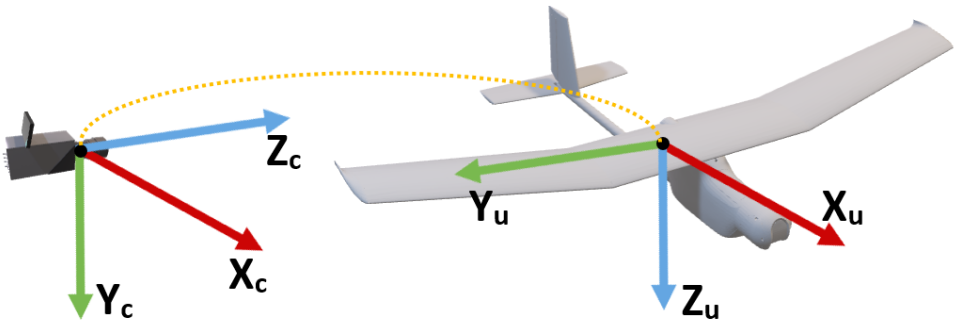

Figure 3.1: Camera and UAV reference frames.

We can correlate the reference frames shown in Figure 3.1, using a spatial rotation and translation [Lepetit *et al.* , 2005; Josef, 2006; Cyganek & Siebert, 2011]. A rotation matrix (spatial rotation) has the following properties [Goldstein *et al.* , 2002; Murray, 1994]:

- The matrix is orthogonal;
- The determinant is unity;
- A product of rotation matrices can represent successive rotations;
- The rotation matrix is not commutative.

In part of the work, we adopt *Euler* angles[1] where the following Direction Cosine Matrix (DCM) gives the transformation for a Z-Y-X ($\gamma-\beta-\alpha$) rotation sequence according to [Rogers, 2007; Lepetit *et al.* , 2005; Bigun, 2006; Zhu *et al.* , 2011]:

$$R(\gamma,\beta,\alpha) = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & sin(\alpha) \\ 0 & -sin(\alpha) & cos(\alpha) \end{bmatrix}}_{R_x(\alpha)} \underbrace{\begin{bmatrix} cos(\beta) & 0 & -sin(\beta) \\ 0 & 1 & 0 \\ sin(\beta) & 0 & cos(\beta) \end{bmatrix}}_{R_y(\beta)} \underbrace{\begin{bmatrix} cos(\gamma) & sin(\gamma) & 0 \\ -sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_z(\gamma)} \quad (3.1)$$

[1] *Euler* angle α represents the rotation around X, β represents the rotation around Y, and γ represents the rotation around Z (Figure 3.1).

For other parts of the work, it is more convenient to adopt a unit quaternion based representation. For a unit quaternion $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$ (3.7), the DCM matrix is given by [Pervin & Webb, 1982; Lepetit *et al.* , 2005; Cyganek & Siebert, 2011]:

$$R(\mathbf{q}) = \begin{bmatrix} (q_4^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1 q_2 + q_4 q_3) & 2(q_1 q_3 - q_4 q_2) \\ 2(q_1 q_2 - q_4 q_3) & (q_4^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2 q_3 + q_4 q_1) \\ 2(q_1 q_3 + q_4 q_2) & 2(q_2 q_3 - q_4 q_1) & (q_4^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{3.2}$$

Using the relation (3.1) or (3.2) the rotation can be related between reference frames. The orientation error can be obtained according to:

$$\delta(R_1, R_2) = \sqrt{\frac{\| \, log_m \left(R_1^T R_2\right) \, \|_F^2}{2}} \frac{180}{\pi} \quad [deg] \tag{3.3}$$

where $\{R_1, R_2\}$ are rotation matrices. If we use a unit quaternion (3.7) based representation, the orientation error quaternion $\mathbf{q}_e$, that expresses the error between two unit quaternions $\mathbf{q}_u$ and $\mathbf{q}_p$, can alternatively be obtained according to:

$$\mathbf{q}_e = \mathbf{q}_u \otimes \bar{\mathbf{q}}_p \tag{3.4}$$

where $\otimes$ represents unit quaternion multiplication (the composition of orientations) and $\bar{\mathbf{q}}_p$ corresponds to the conjugate of $\mathbf{q}_p$ [Finkelstein *et al.* , 1962; Conway, 1937].

3.1.2 State model

The UAV state is represented according to the camera reference frame and contains linear and angular positions and velocities:

$$\mathbf{x}_t = [\mathbf{t}_t^T, \mathbf{r}_t^T]^T \quad \text{with} \quad \mathbf{t}_t^T = [\mathbf{u}_t^T, \mathbf{v}_t^T] \quad \text{and} \quad \mathbf{r}_t^T = [\mathbf{q}_t^T, \boldsymbol{\omega}_t^T] \tag{3.5}$$

where $\mathbf{u}_t^T = [X, Y, Z]$ is the linear position, $\mathbf{v}_t^T = [v_x, v_y, v_z]$ is the linear velocity, $\boldsymbol{\omega}_t^T = [\omega_x, \omega_y, \omega_z]$ is the angular velocity. Representing the orientation directly in the space of *Euler* angles is difficult because of the existing singularities[2]. To deal with this, we use a unit orientation quaternion $\mathbf{q}_t$ (3.5) defined as:

$$\mathbf{q}_t = [\boldsymbol{\varrho}^T, q_4]^T \tag{3.6}$$

where $\mathbf{q}_t \in S^3 \subset \mathbb{R}^4 : \| \, \mathbf{q} \, \| = 1$ with:

$$\boldsymbol{\varrho}^T = [q_1, q_2, q_3] = \hat{\mathbf{e}} \sin\left(\frac{\rho}{2}\right) \quad \text{and} \quad q_4 = \cos\left(\frac{\rho}{2}\right) \tag{3.7}$$

where $\hat{\mathbf{e}}$ is the axis of rotation, and ρ is the angle of rotation. The UAV pose (3D position and orientation), concerning the camera reference frame, is given by $\mathbf{p}_t = [\mathbf{u}_t{}^T, \mathbf{q}_t^T]^T$. The captured camera image at each time instant t is given by $\mathbf{y}_t$.

Our main objective will be to estimate the posterior Probability Density Function (PDF) $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ (3.10), and for that, we need to perform tracking to filter the existing noise. The state estimation of dynamical systems is introduced in Section 3.1.3.

[2] Loss of one degree of freedom, commonly known as *gimbal lock* [Oh & Vadali, 1988].

3.1.3 State estimation

The state estimation of dynamical systems is based on a transition model (Section 3.1.5) that describes how the system evolves and an observation model (Section 3.1.6) that explains how the measurements are related to the state. In general, a discrete state-space model can be characterized by [Arulampalam *et al.* , 2002; Challa, 2011; Haug, 2012; Thrun *et al.* , 2005; Van Der Merwe *et al.* , 2001]:

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, \boldsymbol{\xi}_t) \tag{3.8}$$

$$\mathbf{y}_t = \mathbf{H}(\mathbf{x}_t, \boldsymbol{\eta}_t) \tag{3.9}$$

where $\mathbf{F}(.)$ is the system function, $\mathbf{H}(.)$ is the measurement function, t is a time index, $\mathbf{x}_t$ is the state of the model (not directly observable), $\mathbf{y}_t$ is the observation, $\boldsymbol{\xi}_t$ and $\boldsymbol{\eta}_t$ are respectively the system and observation noise. In probabilistic state estimation, the primary objective is to estimate the PDF of the state given the past observations $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. The transition model (3.8) can be represented by the state transition PDF $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ and the observation model (3.9) can be represented by the likelihood PDF $p(\mathbf{y}_t \mid \mathbf{x}_t)$. Given the *Markov process* assumption for the state propagation and conditional independence for the observations, we can represent the posterior PDF $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ in a recursive way as [Arulampalam *et al.* , 2002; Challa, 2011; Doucet *et al.* , 2001; Doucet & Johansen, 2009; Forsyth & Ponce, 2002; Haug, 2012; Thrun *et al.* , 2005; Gelfand *et al.* , 2010]:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t \mid \mathbf{x}_t)p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})}{\underbrace{\int p(\mathbf{y}_t \mid \mathbf{x}_t)p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})d\mathbf{x}_t}_{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}} \propto p(\mathbf{y}_t \mid \mathbf{x}_t)p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) \tag{3.10}$$

with prediction $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ given by the *Chapman-Kolmogorov* equation [Anderson & Moore, 1979; Iltis, 1990; Ross, 2010]:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1})p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})d\mathbf{x}_{t-1} \tag{3.11}$$

where $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ is the state distribution at the previous time step.

3.1.4 Motion models

We consider that the UAV follows a constant velocity model, i.e. it suffers small accelerations between two consecutive time steps t and $t+1$. Furthermore, we consider that linear and angular motions are independent [Haug, 2012; Bazin *et al.* , 2010; Antone & Teller, 2000]. Thus, the full model is composed of two separate dynamical systems. The motion (rotation and translation) decoupling has been applied with success in multiple fields such as e.g. motion estimation in catadioptric vision [Bazin *et al.* , 2010], image-based visual servoing motion control [Deguchi, 1998], visual odometry [Kim *et al.* , 2018; Scaramuzza & Siegwart, 2008; Zhang *et al.* , 2016b], camera pose estimation from matched feature points [Fathian *et al.* , 2017], indoor mapping [Liu *et al.* , 2017], extrinsic calibration of a camera and a 3D Light Detection And Ranging (LIDAR) [Zhou *et al.* , 2018], vision-based robot control [Tahri & Chaumette, 2005; Andreff *et al.* , 2002], and two-point cloud rotation and translation estimation

[Ma *et al.* , 2016]. In the UAV field, it has been applied with success e.g. to automatic landing using the airborne camera [Azinheira & Rives, 2008], using stereo vision to estimate altitude, attitude and motion [Eynard *et al.* , 2012], vision-based rotation estimation [Bazin *et al.* , 2012], tracking with a monocular camera [Chen & Dawson, 2006], and UAV controller design [Lee *et al.* , 2010; Metni *et al.* , 2005; Guenard *et al.* , 2008]. This decoupling simplifies the formulation and the UAV control law since the translation and rotation can be controlled independently [Atkins *et al.* , 2016; Kingston *et al.* , 2003; Eubank *et al.* , 2009; Gautam *et al.* , 2014].

3.1.5 State transition model

In discrete-time, the state transition model (3.8) is given by [Kraft, 2003; Pessanha Santos *et al.* , 2015, 2018, 2019c]:

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, \boldsymbol{\xi}_t) = \begin{bmatrix} \mathbf{F}^l(\mathbf{t}_t, \boldsymbol{\xi}_t^l) \\ \mathbf{F}^r(\mathbf{r}_t, \boldsymbol{\xi}_t^r) \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \mathbf{I}_{3\times3} & \Delta t \cdot \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \mathbf{t}_t + \boldsymbol{\xi}_t^l \\ \begin{bmatrix} \mathbf{q}_t \otimes \delta\mathbf{q}_t^\omega \otimes \delta\mathbf{q}_t^r \\ \boldsymbol{\omega}_t + \boldsymbol{\xi}_t^r \end{bmatrix} \end{bmatrix} \tag{3.12}$$

where $\boldsymbol{\xi}_t^l \sim \mathcal{N}(0, \mathbf{Q}_t^l)$ is a Gaussian noise random variable with zero mean and covariance $\mathbf{Q}_t^l$, $\otimes$ represents unit quaternion multiplication (orientations composition), $\boldsymbol{\xi}_t^r \sim \mathcal{N}(0, \mathbf{Q}_t^r)$ is a Gaussian noise random variable with zero mean and covariance $\mathbf{Q}_t^r$, and $\delta\mathbf{q}_t^\omega$ and $\delta\mathbf{q}_t^r$ are quaternions representing the integration of the effect of the angular velocity and rotation noise assumed constant during a sampling interval Δt:

$$\delta\mathbf{q}_t^\omega = \boldsymbol{\Omega}(\boldsymbol{\omega}_t) \quad \text{and} \quad \delta\mathbf{q}_t^r = \boldsymbol{\Omega}(\boldsymbol{\xi}_t^r) \tag{3.13}$$

with:

$$\boldsymbol{\Omega}(\mathbf{b}) = \left[\frac{\mathbf{b}}{\| \mathbf{b} \|} \sin\left(\frac{\| \mathbf{b} \| \Delta t}{2} \right) , \cos\left(\frac{\| \mathbf{b} \| \Delta t}{2} \right) \right] \tag{3.14}$$

When using the UBiF (Section 5.3.1) or the Unscented Bingham-Gauss Filter (UBiGaF) (Section 5.3.2), the noise of the rotational components of the state transition model is assumed Bi distributed (Section C.1) and BiGa distributed (Section C.2) respectively.

3.1.6 Observation model

In the developed approach, we use two different observation models: (i) a linear and Gaussian model (Section 3.1.6.1), and (ii) a likelihood model approximation using image information (Section 3.1.6.2). The linear and Gaussian model is the classical model and is a computationally inexpensive rough approximation to the real observation model. The likelihood approximation is a computationally expensive non-linear and non-Gaussian model obtained using image information. Both models are essential in the final algorithm implementation, the first allows a rough and fast pose estimative, and the second concentrates all the computational efforts in the most promising zones.

3.1.6.1 Linear and Gaussian model

As initially described in Section 3.1.4, we consider that the UAV follows a constant velocity model with independent motions. The linear and Gaussian model for the implemented motion filtering (Section 5.3) is given by [Kraft, 2003; Pessanha Santos *et al.* , 2015, 2018, 2019c]:

$$\mathbf{z}_t = \mathbf{H}(\mathbf{x}_t, \boldsymbol{\eta}_t) = \begin{bmatrix} \mathbf{H}^l(\mathbf{t}_t, \boldsymbol{\eta}_t^l) \\ \mathbf{H}^r(\mathbf{r}_t, \boldsymbol{\eta}_t^r) \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \mathbf{t}_t + \boldsymbol{\eta}_t^l \\ \begin{bmatrix} \mathbf{I}_{4\times4} & \mathbf{0}_{3\times3} \end{bmatrix} \mathbf{r}_t \otimes \delta\mathbf{q}_t^\eta \end{bmatrix} = \begin{bmatrix} \mathbf{u}_t + \boldsymbol{\eta}_t^l \\ \mathbf{q}_t \otimes \delta\mathbf{q}_t^\eta \end{bmatrix} \tag{3.15}$$

where $\boldsymbol{\eta}_t^l \sim \mathcal{N}(0, \mathbf{R}_t)$ is a Gaussian noise random variable with zero mean and covariance matrix $\mathbf{R}_t$ and $\delta\mathbf{q}_t^\eta$ is a quaternion representing the integration of the effect of the observation rotation noise in a similar way to (3.13). When using the UBiF (Section 5.3.1) or the UBiGaF (Section 5.3.2), the noise of the rotational components of the observation model is assumed Bi distributed (Section C.1).

3.1.6.2 Likelihood approximation model

The probability density function $p(\mathbf{y}_t \mid \mathbf{x}_t)$ (Section 3.1.3) expresses the probability mass of a particular image $\mathbf{y}_t$, given a particular state hypothesis $\mathbf{x}_t$. We can approximate samples $\hat{\mathbf{y}}_t$ from this density by rendering synthetic images of the UAV CAD model at state $\mathbf{x}_t$ using a graphics engine $\hat{\mathbf{y}}_t = g(\mathbf{x}_t)$. This is a coarse approximation to the true density because many effects of the image formation process are not modeled (e.g. background texture, image noise, motion blur, or occlusions). The likelihood function $L(\mathbf{x}_t, \mathbf{y}_t)$ is the likelihood of the state $\mathbf{x}_t$ given an observation $\mathbf{y}_t$ and is proportional to $p(\mathbf{y}_t \mid \mathbf{x}_t)$, considering $\mathbf{y}_t$ fixed. To compute an approximated value to this likelihood (up to scale), we use a similarity metric between $\mathbf{y}_t$ and the synthetic images rendered for the assumed states $\mathbf{x}_t, d(\mathbf{y}_t, g(\mathbf{x}_t))$, as will be described in Section 5.4.1. The similarity metric will be used as observation model $p(\mathbf{y}_t \mid \mathbf{x}_t)$ in the implemented UPF based approach, as will be described in Section 3.5.

3.1.7 Vision system camera model

We are using the pinhole camera model (Figure 3.2), that describes the real-world projection in an image plane using intrinsic[3] and extrinsic[4] parameters [Jähne *et al.* , 1999; Prince, 2012; Radke, 2013; Hartley & Zisserman, 2003]. The relation between 3D coordinates and camera coordinates is given by [Ma *et al.* , 2012; Hartley & Zisserman, 2003]:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} fs_x & S_\theta & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic matrix}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}}_{\text{Extrinsic matrix}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.16}$$

where λ is an arbitrary positive scalar, (u, v) represents the image coordinates (in pixels), (X, Y, Z) are 3D coordinates, f is the focal length that represents the distance between the image plane and the optical center, s_x is the horizontal pixel number by meter (pixel/m), s_y is the vertical pixel number by meter (pixel/m), S_θ is the skew factor proportional to the angle

[3] Allow a mapping between camera coordinates and pixel coordinates in the image frame.
[4] Define the coordinate system transformations from the 3D world to 3D camera coordinates.

between the u and v pixel axes[5], $c = (c_x, c_y)$ are the coordinates (in pixels) of the center of the image, R is a rotation matrix and T is the translation matrix to camera coordinates.

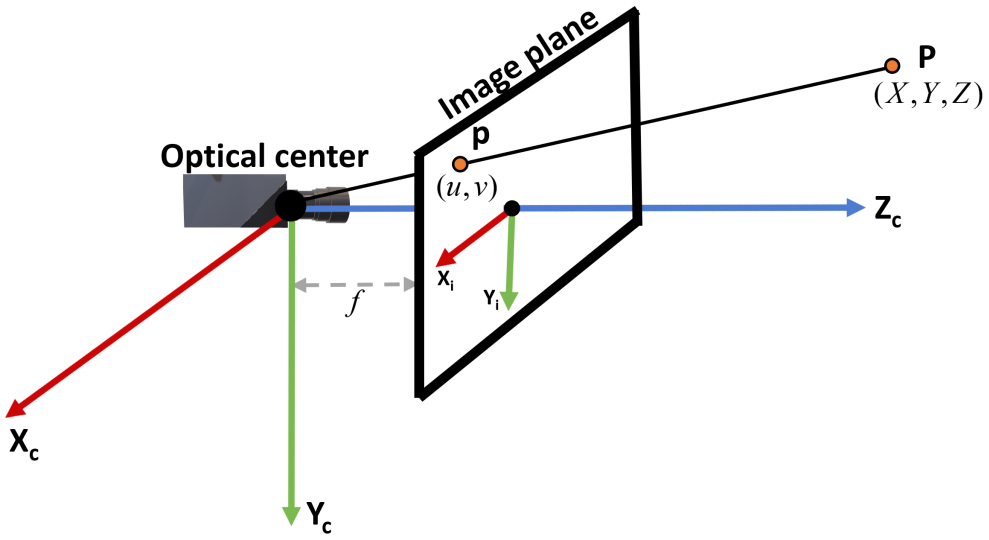

Figure 3.2: Pinhole camera model illustration.

In the real world, the camera lens present distortions, being the most relevant the radial[6] and tangential[7] [Cyganek & Siebert, 2011; Prince, 2012; Radke, 2013]. The tangential distortion component presents less influence on the final result. The distortion can be represented by [Hartley & Zisserman, 2003; Lepetit *et al.* , 2005; Ma *et al.* , 2012; Szeliski, 2010]:

$$\begin{aligned} x_{distortion} &= \underbrace{x(1 + K_1 r^2 + K_2 r^4 + \overbrace{\dots}^{Optional})}_{x_{radial}} + \underbrace{2p_1 xy + p_2(r^2 + 2x^2)}_{x_{tangential}} \\ y_{distortion} &= \underbrace{y(1 + K_1 r^2 + K_2 r^4 + \overbrace{\dots}^{Optional})}_{y_{radial}} + \underbrace{2p_2 xy + p_1(r^2 + 2y^2)}_{y_{tangential}} \end{aligned} \tag{3.17}$$

where $\{x, y\}$ are coordinates of the distorted points, $r^2 = x^2 + y^2$, $\{K_1, K_2\}$ are two constant values between $[-1, \dots, 1]$, and $\{p_1, p_2\}$ are two constant values between $[-0.1, \dots, 0.1]$. Normally $\{K_1, K_2\}$ are enough to obtain a good approximation, but if needed the polynomial can be expanded as shown in (3.17).

We have obtained the intrinsic matrix (3.16) and the distortion parameters (3.17) offline [Sturm & Maybank, 1999; Lepetit *et al.* , 2005], using a chessboard calibration pattern according to Bouguet [2008] (Figure 3.3). An example of a distortion-corrected image can be seen in Figure 3.4.

3.2 Particle Filter

Due to the non-linear nature of the rotation and to the multimodality of the likelihood function (due to UAV symmetries, occlusions, and background clutter), we cannot use a typical filtering

[5] The pixels in a sensor may not be square, resulting in a small distortion.
[6] Non-linear distortion (2D image deformation) that depends on the distance to the image center.
[7] It depends on the lens location relative to the image plane (lens not parallel to the image plane).

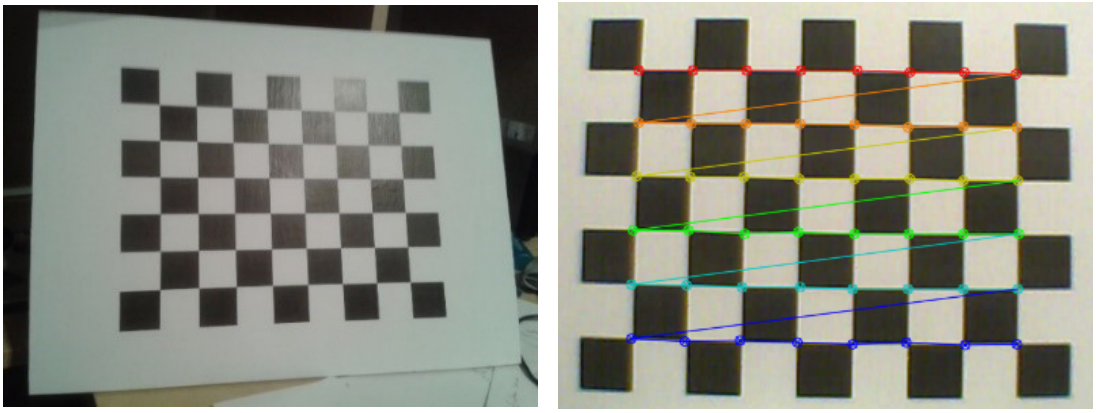

Figure 3.3: Used chessboard calibration pattern.

Figure 3.4: Real captured (*left*) and distortion corrected (*right*) images.

model (e.g. EKF or UKF) and the PF [Haug, 2012; Thrun *et al.* , 2005; Challa, 2011; Simon, 2006] is a *tool* that can represent this continuous distribution by discrete samples (Figure 3.5). Since we cannot sample directly from $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, the idea of this filtering technique is to represent the state $\mathbf{x}_t$ (3.8) using a set of weighted samples (particles) that approximate the posterior PDF. The set of possible system states are represented by the samples $\mathbf{x}_t^m$ with associated weights w_t^m originating $\{\mathbf{x}_t^m, w_t^m\}_{m=1}^M$. The posterior PDF is then approximated by a discrete weighted approximation of the true posterior according to:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \approx \sum_{m=1}^{M} w_t^m \, \delta(\mathbf{x}_t - \mathbf{x}_t^m) \tag{3.18}$$

where M is the number of particles, $\delta(.)$ is the *Dirac* function, and $\sum_{m=1}^{M} w_t^m = 1$. The generic PF is composed of four main steps [Wang *et al.* , 2016; Van Der Merwe *et al.* , 2001; Wang *et al.* , 2012; Boli *et al.* , 2004; Chan *et al.* , 2012; Gordon *et al.* , 1993]: (i) initialization, (ii) importance sampling, (iii) importance weighting, and (iv) resampling, as described in Algorithm 1. After resampling, an additional move step can be used to decrease the probability of occurring degeneracy[8].

Typically in CV, it is used the transition prior[9] $q(\mathbf{x}_t^m \mid \mathbf{x}_{t-1}^m)$ [Arulampalam *et al.* , 2002; Ristic *et al.* , 2004; Rui & Chen, 2001b] as proposal distribution (Section 3.1.3) not incorporating the actual measurement in its generation (Section 5.2). The actual measurement is used

[8] Only a few of the particles will have significant weight.
[9] In the importance sampling step, as described in Algorithm 1.

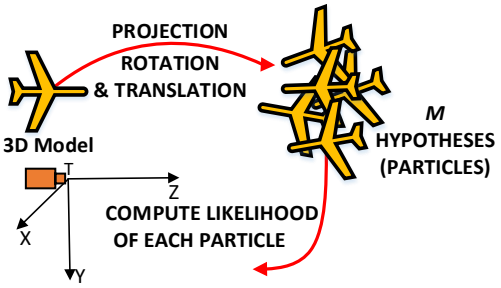

Figure 3.5: Multiple pose hypotheses illustration.

only in the likelihood calculation (importance weights), as described in Algorithm 1. This approach can be improved by using current frame information in the proposal distribution, e.g. using a UKF (Section 3.4). Another different approach is to enrich the proposal distribution injecting new particles from the current image using a detector (Section 3.3). As will be described in Section 3.5, our approach uses both methods to better approximate the true posterior and decrease the obtained estimation error.

Algorithm 1 Particle Filter (PF)

▷ **Initialization**:
1. Draw M initial particles from the prior $p(\mathbf{x}_0)$ and initialize the weights with $\frac{1}{M}$ creating $\{\mathbf{x}_0^m, w_0^m\}_{m=1}^M$.
■ **Importance sampling** ($t \geq 1$):
1. Sample M particles $\hat{\mathbf{x}}_t^m \sim q(\mathbf{x}_t^m \mid \mathbf{x}_{0:t-1}^m, \mathbf{y}_{0:t})$, where q is a proposal distribution which is easy to sample from and is used to approximate the posterior PDF.
■ **Importance weighting** ($t \geq 1$):
1. Obtain the unnormalized importance weights:
$\tilde{w}_t^m \propto w_{t-1}^m \frac{p(\mathbf{y}_t|\hat{\mathbf{x}}_t^m)\, p(\hat{\mathbf{x}}_t^m|\mathbf{x}_{t-1}^m)}{q(\hat{\mathbf{x}}_t^m|\mathbf{x}_{t-1}^m, \mathbf{y}_{1:t})}$
2. Normalize the obtained importance weights:
$w_t^m = \frac{\tilde{w}_t^m}{\sum_{i=1}^M \tilde{w}_t^i}$
■ **Resampling** ($t \geq 1$):
1. Eliminate particles with low importance weights and replicate particles having high importance weights to obtain M random samples with uniform weights creating $\{\hat{\mathbf{x}}_t^m, \hat{w}_t^m\}_{m=1}^M \rightarrow \left\{\mathbf{x}_t^m, w_t^m = \frac{1}{M}\right\}_{m=1}^M$.
■ **Move step - *Optional*** ($t \geq 1$):
1. Use a move step to increase the diversity of the particles after the resampling step [Havangi *et al.* , 2013; Haykin *et al.* , 2001]. This step can be performed by adding Gaussian noise to the particle state [Kotecha & Djuric, 2003; Kotecha & Djuric, 2001; Haug, 2012].
▷ **Output:** $\{\mathbf{x}_t^m, w_t^m\}_{m=1}^M$

3.3 Boosted Particle Filter

The MPF [Vermaak *et al.* , 2003] is a variation of the generic PF, designed to track multiple targets, where each target is modeled with an individual PF. The BPF [Okuma *et al.* , 2004] extends this application using AdaBoost [Viola & Jones, 2001] to incorporate current observations and be able to detect objects leaving and entering in the analyzed scene. In the original BPF implementation, the proposal is given by [Okuma *et al.* , 2004]:

$$q_B(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) = \alpha q_{ada}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_t) + (1-\alpha)p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \tag{3.19}$$

where q_{ada} is the proposal from the AdaBoost detection, and α varies between zero and one defining the contribution of each proposal. When $\alpha = 0$, the implemented filter becomes a simple MPF. As will be described in Section 3.5, we also use a detector to obtain the target position on the frame (Section 4.2) but extend it with rough estimates of depth and pose using a pre-trained database of synthetically generated images (Section 4.3).

3.4 Unscented Particle Filter

When we combine a PF with a UKF, we obtain a UPF [Rui & Chen, 2001a; Van Der Merwe *et al.* , 2001; Birsan, 2005], as initially described in Section 2.7. The UPF is composed of the typical four steps described in Section 3.2 for the PF: (i) initialization, (ii) importance sampling, (iii) importance weighting, and (iv) resampling, as described in Algorithm 2. The main difference, when compared with the generic PF, is in the importance sampling step, where we use a UKF (Appendix A) to integrate the current observation and generate a better proposal distribution [Rui & Chen, 2001a; Van Der Merwe *et al.* , 2001; Wang *et al.* , 2016; Haykin *et al.* , 2001; Zhou *et al.* , 2010; Birsan, 2005].

As will be described in Section 3.5, we extend this concept to include a directional noise model (Bi and BiGa distributions) to better cope with the periodic nature of the rotational motion component. We will use a UKF (Appendix A) for the translational motion filtering and a UBiF (Section 5.3.1) or a UBiGaF (Section 5.3.2) for the rotational motion filtering.

Algorithm 2 Unscented Particle Filter (UPF)

▷ **Initialization**:
1. Draw M initial particles from the prior $p(\mathbf{x}_0)$ and initialize weights with $\frac{1}{M}$ creating $\{\mathbf{x}_0^m, w_0^m\}_{m=1}^M$. Initialize the mean $\bar{\mathbf{x}}_0^m$ and covariance $\mathbf{P}_0^m$ for the UKF (Appendix A).
■ **Importance sampling** ($t \geq 1$):
1. Update the particles applying a UKF to each particle (Appendix A). From the UKF we obtain $\tilde{\mathbf{x}}_t^m$ (A.29) and $\tilde{\mathbf{P}}_t^m$ (A.30), and we can sample particles using the proposal distribution $\hat{\mathbf{x}}_t^m \sim q(\mathbf{x}_t^m \mid \mathbf{x}_{0:t-1}^m, \mathbf{y}_{0:t}) = \mathcal{N}\left(\tilde{\mathbf{x}}_t^m, \tilde{\mathbf{P}}_t^m\right)$ [Rui & Chen, 2001a; Van Der Merwe *et al.* , 2001].
■ **Importance weighting and Resampling** ($t \geq 1$):
1. These steps are the same as applied in the PF described in Section 3.2.
▷ **Output:** $\{\mathbf{x}_t^m, w_t^m\}_{m=1}^M$

3.5 Overall system proposal

Section contents

In the classical UPF, we combine a PF (Section 3.2) with a UKF (Appendix A). We extend this concept to include a directional noise model (Bi and BiGa distributions) to better cope with the periodic nature of the rotational motion component. The current observation inclusion on the proposal distribution outperforms the traditional PF transition prior based proposals

[Rui & Chen, 2001b]. As adopted in the BPF (Section 3.3), we also use a detector but we extend it to compute a coarse estimate of the pose of the UAV with the current observation. In summary, our system proposal (Figure 3.6) is inspired in a BPF where a modified UPF computes the proposal distribution to include directional noise models (Section 5.2) and is enriched with a pose optimization step (Section 5.5) to cope with the sub-optimality of the likelihood model (Section 5.4.1). The proposed system architecture (Figure 3.6) is described in detail in Algorithm 3, including the system assumptions, the inputs, the initialization, the pose boosting stage (Chapter 4), and the tracking stage (proposal, approximate weighting and resampling, and pose optimization) (Chapter 5). The pose boosting stage is initially shown in Section 3.5.1 and described in detail in Chapter 4. The tracking stage is initially specified in Section 3.5.2 and fully explained in Chapter 5.

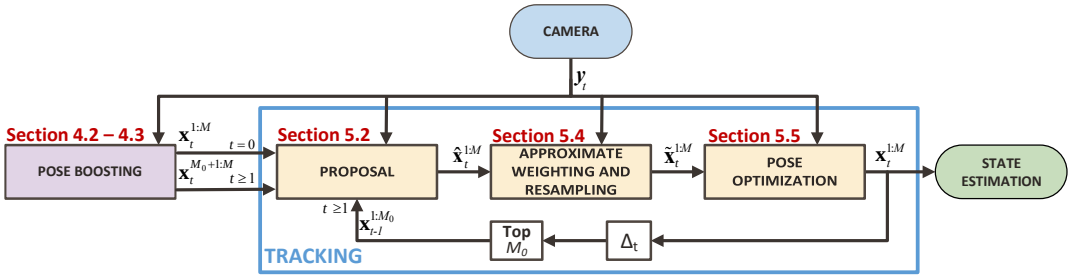

Figure 3.6: System architecture.

3.5.1 Pose boosting introduction

In the pose boosting stage, we apply a detector to obtain the ROI coordinates of objects with the appearance of the UAV (Section 4.2). The ROI with the highest confidence value is used to obtain a rough UAV pose estimate using a pre-trained database of UAV images generated artificially at different orientations (Section 4.3). Each particle represents the UAV state, as described in Section 3.1.2.

3.5.2 Tracking introduction

The tracking stage is composed of three modules (Figure 3.6): (i) proposal, (ii) approximate weighting and resampling, and (iii) pose optimization. The proposal module is responsible for generating a set of particles $\hat{\mathbf{x}}_t^{1:M}$ that reflect the distribution of the state of the system. This module is quite complex, and the adopted architecture variants will be described in detail in Section 5.2. The proposal stage takes the current set of particles $\mathbf{x}_t^{1:M}$ and generates samples from a proposal distribution similarly to the UPF, using as observation $\mathbf{z}_t$ which we approximate as the pose of the particle with the best likelihood for the acquired image $p(\mathbf{y}_t \mid \mathbf{x}_t)$. This maximum likelihood approximation $\mathbf{z}_t$ is a very coarse estimate of the actual pose but still effective in improving the proposal distribution (Section 5.5). After the proposal stage, we approximate the particle weights $w_t^{1:M}$ (approximate weighting) using an approximate likelihood function (Section 5.4.1). The resampling module is responsible for eliminating particles with low importance weights and replicate particles having high importance weights to improve

Algorithm 3 Proposed system (Figure 3.6) - *Pseudocode*

▷ **Assumptions**:
1. Constant UAV velocity model (Section 3.1.4);
2. FPB and wind perturbations are not explicitly modeled (Section 3.1).

▷ **Inputs**:
1. UAV CAD model (Section 3.1);
2. Camera parameters - radial and tangential factors (Section 3.1.7);
3. Pre-trained detector weights $\mathbf{w}_{trained}$ (Section 4.2);
4. Pre-trained pose samples database $\mathbf{D}_{database}$ (Section 4.3).

▷ **Initialization**:
1. Load UAV CAD model and camera parameters;
2. Load UKF (Appendix A), UBiF (Section 5.3.1), and UBiGaF (Section 5.3.2) parameters.

■ **Pose boosting**:
1. Capture a new frame $\mathbf{y}_t$;
2. Obtain Regions of Interest (ROIs) on the captured frame using a detector (Section 4.2):
   $\text{ROIs} = Detector(\mathbf{y}_t, \mathbf{w}_{trained})$
3. Obtain $M - M_0$ pose particles using the pre-trained database (Section 4.3):
   $\mathbf{x}_t^{M_0+1:M} = Hypotheses\_generation(\mathbf{D}_{database}, \text{ROIs})$
4. Combine $\mathbf{x}_t^{M_0+1:M}$ with the best M_0 particles from the previous time step $\mathbf{x}_{t-1}^{1:M_0}$ $(t > 1)$:
   $\mathbf{x}_t^{1:M} = \left\{\mathbf{x}_{t-1}^{1:M_0}, \mathbf{x}_t^{M_0+1:M}\right\}$

■ **Proposal** $(t \geq 1)$:
1. Obtain the current measurement $\mathbf{z}_t$ (Section 5.4.1):
   $\mathbf{z}_t = Coarse\_pose\_estimate(\mathbf{x}_t^{1:M}, F_t)$
2. Apply motion filtering to the set (Section 5.3):
   2.1. Case 1 - UKF (Appendix A):
   $\hat{\mathbf{x}}_t^{1:M} = \text{UKF}(\mathbf{x}_t^{1:M}, \mathbf{z}_t)$
   2.2. Case 2 - UKF (Appendix A) + UBiF (Section 5.3.1) or UBiGaF (Section 5.3.2):
   $$\mathbf{x}_t^{1:M} = \begin{cases} \text{UKF}(\mathbf{t}_t^{1:M}, \mathbf{z}_t) \\ \text{UBiF}(\mathbf{r}_t^{1:M}, \mathbf{z}_t) \text{ or } \text{UBiGaF}(\mathbf{r}_t^{1:M}, \mathbf{z}_t) \end{cases}$$

■ **Approximate weighting and Resampling** $(t \geq 1)$:
1. Evaluate the likelihood of the current particle set (Section 5.4.1):
   $w_t^{1:M} = Approximate\_weighting(\hat{\mathbf{x}}_t^{1:M}, \mathbf{z}_t)$
2. Apply a resampling strategy (Section 5.4.2):
   $\tilde{\mathbf{x}}_t^{1:M} = Resampling(\hat{\mathbf{x}}_t^{1:M}, w_t^{1:M})$

■ **Pose optimization** $(t \geq 1)$:
1. Apply an optimization stage to refine the pose estimate (Section 5.5):
   $\mathbf{x}_t^{1:M} = Optimization(\tilde{\mathbf{x}}_t^{1:M})$

▷ **Output**:
1. UAV state estimation (Section 3.5.2):
   $\mathbf{x}_t^* = State\_estimation(\mathbf{x}_t^{1:M})$

particle diversity, generating the set $\tilde{\mathbf{x}}_t^{1:M}$ (Section 5.4.2). Finally, to increase the accuracy of the result, we use a pose optimization stage to perform a fine local adjustment of the pose component of the particles, as described in Section 5.5. The optimized particle set is given by $\mathbf{x}_t^{1:M}$, from which we can use statistics (mean, mode, maximum, etc.) to obtain the state estimation $\mathbf{x}_t^*$ that will ultimately represent our UAV pose for control purposes. To generate particles for the next time step, we choose, from this set, the best M_0 particles $\mathbf{x}_{t-1}^{1:M_0}$. These particles will be mixed with new ones obtained using the current frame at time t.

Chapter 4

Pose boosting

We cannot teach people anything; we can only help them discover it within themselves.

Albert Einstein

Chapter contents

This chapter describes the proposed pose boosting stage, used to obtain a set of rough pose estimates from the captured image.

4.1 Proposed system structure

The pose boosting stage is inspired in the BPF (Section 3.3), which is an adapted MPF intended for multiple target tracking that uses AdaBoost to incorporate current observations (targets position on the image) in the proposal generation (3.19). In our implementation, we also use a detector, but we have adapted its architecture to single target tracking, improving it using a pre-trained database to be able to retrieve not only the UAV position but a set of rough pose estimates. As initially described in Chapter 3, the pose boosting stage (Figure 3.6) can be divided into two different stages (Figure 4.1): (i) detection (Section 4.2), and (ii) hypotheses generation (Section 4.3).

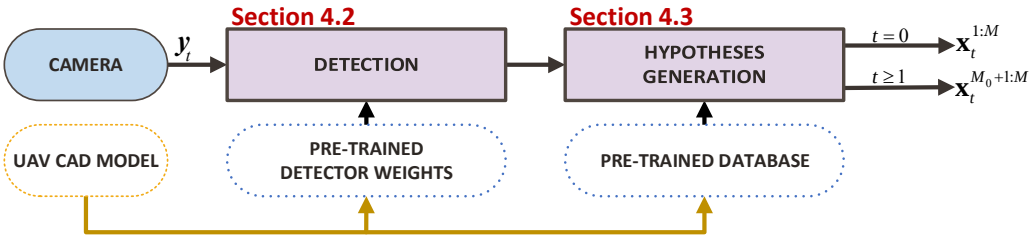

Figure 4.1: Pose boosting (*proposed structure*).

The UAV detection in the captured frame is essential to be able to apply the algorithms needed to estimate its pose. Since the search area is vast, and we do not use any UAV sensor information, we have trained detectors based on DNNs (Section 4.2). In the hypotheses generation stage, we compare the obtained detector ROI with the highest confidence value with a pre-trained database of UAV bounding boxes in multiple poses to obtain a rough pose estimate (Section 4.3). The UAV CAD model (Figure 1.6 *right*) plays a vital role in the detector training and database generation (Figure 4.1), as will be described in Section 4.2 and Section 4.3.

4.2 Target detection

Section contents

The UAV detection stage is critical (Section 2.5) since we are operating in an outdoor environment, and we need to have illumination invariance. The presence of other objects can affect system performance and reliability. The target detection phase consists of searching in the image for ROIs that may contain our UAV. The used detectors are described in Section 4.2.1 and the used synthetic images (Figure 4.2) dataset generation scheme is described in Section 4.2.2.

Figure 4.2: Synthetic training dataset (*example*).

4.2.1 YOLO and SSD

The UAV detection is made using YOLO v3 [Redmon & Farhadi, 2018] or SSD [Liu *et al.* , 2016b], which are state-of-the-art detectors methods without the need for external region proposals (Section 2.5). The YOLO v3 uses a variant of the original DarkNet-53 network [Redmon & Farhadi, 2016, 2017] incorporating a 106 layer fully convolutional [Redmon & Farhadi, 2018]. This network performs detection downsampling the input image at three different scales (32, 16, and 8) in a concept similar to Feature Pyramid Networks (FPNs) [Lin *et al.* , 2017b], improving its performance when detecting small objects. The SSD uses a VGG16 network [Simonyan & Zisserman, 2014] for feature extraction and then incorporates additional convolutional layers and filters to obtain the location of the object on the captured frame. The

implemented convolutional layers decrease in size progressively and have the objective of being able to predict at multiple scales (Section 2.5). The YOLO implementation is performed adapting the publicly available code [Redmon & Farhadi, 2019], and the SSD implementation is performed by adapting the publicly available *Tensorflow* object detection Application Programming Interface (API) [J. Huang & Zhu, 2017]. The detector will be trained using a synthetic images dataset (Figure 4.2), as described in Section 4.2.2. The detector performance was evaluated using 679 real captured test images, as described in Section 6.3.

4.2.2 Synthetic images dataset generation

Since there is no public UAV database for object detection and the real data acquisition is very costly, we decide to use a synthetically generated dataset to perform transfer learning to real images [Joseph Tan *et al.* , 2015; Varol *et al.* , 2017]. A synthetic image is created, rendering the UAV CAD model (Figure 1.6 *right*) on top of a dataset of real captured background images, as described in Figure 4.3. For each image is also generated an annotation file that contains the UAV ROI coordinates. The ROI coordinates on each image are obtained using four different steps (Figure 4.4): (i) UAV rendering, (ii) image binarization[1] [Sauvola & Pietikäinen, 2000] to detect the UAV area, (iii) the calculation of the Oriented Bounding Box (OBB)[2], and (iv) from the obtained OBB coordinates we calculate UAV ROI. We have used 335769 annotated images ($width \times height = 1280 \times 720$) for training, as described in Section 6.3.

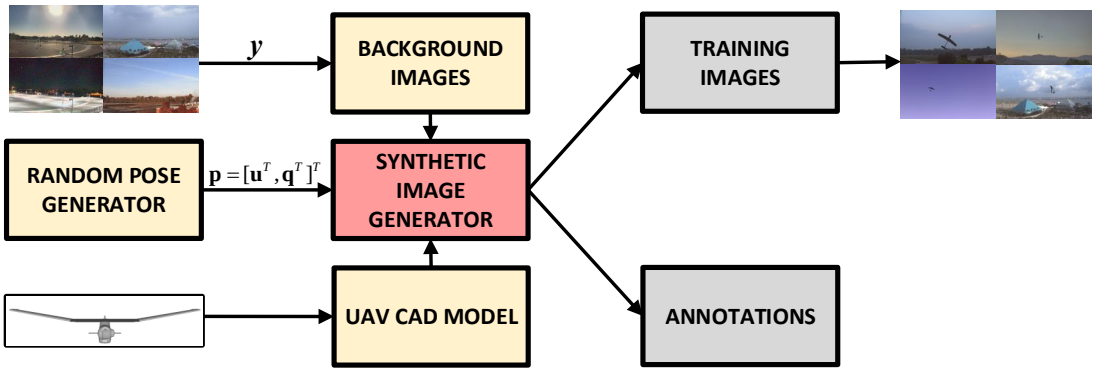

Figure 4.3: Synthetic images dataset generation scheme.

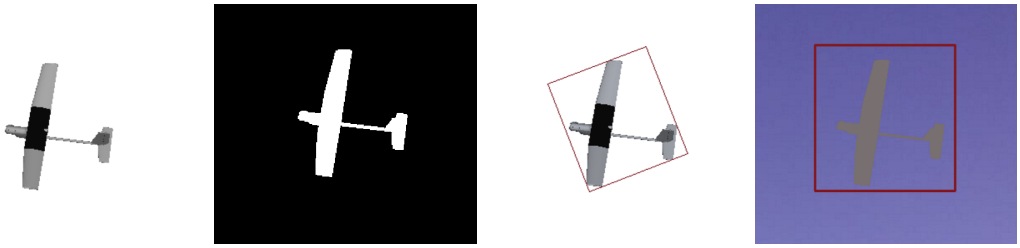

Figure 4.4: Rendered UAV (*left*), binarization (*center left*), OBB (*center right*), and ROI (*right*).

[1] Conversion between a grayscale image $[0, \ldots, 255]$ and a black and white binary image $\{0, 1\}$.
[2] The oriented rectangle of the minimum area enclosing the UAV

4.3 Hypotheses generation

Section contents

From the ROIs obtained in the target detection stage (Section 4.2), we cannot infer the UAV orientation (Section 4.3.2) but only its 3D position (Section 4.3.3). We will use the current image information[3] to generate hypotheses that are near the real UAV pose, using a pre-trained database. Using this database, we can create a relation one-to-many from the parameters of the OBB that contains the UAV corner points to a set of UAV poses (Figure 4.5) to add diversity to the filter proposal. Many methods can obtain such corner points. In our case, we used the Features from Accelerated Segment Test (FAST)[4] corner detector. The database generation scheme is illustrated in Section 4.3.1, the orientation hypotheses generation in Section 4.3.2, and the translation hypotheses generation in Section 4.3.3.

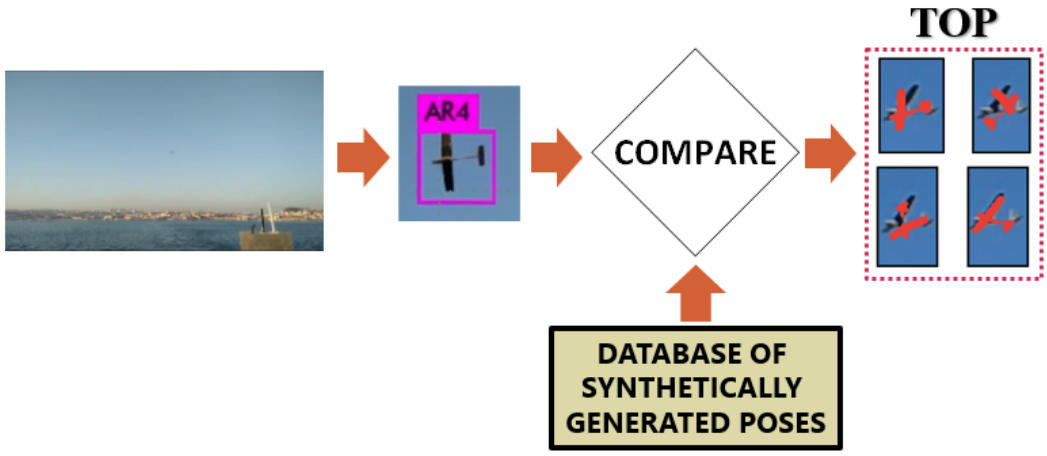

Figure 4.5: Hypotheses generation (*simplified scheme*).

4.3.1 Database generation

The database is created by rendering synthetic images of the UAV 3D CAD model at a fixed position, but varying the rotation according to a uniform distribution restricted in a specific interval concerning the camera reference frame[5]. This database is created offline and indexed efficiently for fast run-time access. For each generated possibility, it is obtained the OBB that better fits the projected object and is stored in a database indexed by the angle (θ) and AR (R). The database poses are given by:

[3] The ROI with the highest confidence value.
[4] A corner detection method that uses a *Bresenham* circle algorithm [Bresenham, 1977] (linear algorithm for discrete circle representation) of radius 3 to classify if a specific point is a corner or not. If K adjacent pixels in the circle are all brighter or all darker (plus or minus the threshold respectively) than the candidate pixel, it is considered a corner [Rosten & Drummond, 2006; Rosten *et al.* , 2010].
[5] *Euler* angle α represents the rotation around X, β represents the rotation around Y, and γ represents the rotation around Z (Figure 3.1).

$$\mathbf{D}^i = \left[X^i, Y^i, Z^i, \mathbf{q}^i\right] \tag{4.1}$$

where i is the pose index in the database. Each database pose has one associated OBB $\mathbf{B}^i$ given by:

$$\mathbf{B}^i = \left[\theta^i, x^i, y^i, w^i, h^i\right] \tag{4.2}$$

where θ^i is the OBB angle in degrees relative to the horizontal, $\left(x^i, y^i\right)$ is the OBB center coordinate, w^i is the OBB width and h^i is the OBB height. The AR R^i is obtained according to (Figure 4.6):

$$R^i = \frac{w^i}{h^i} \tag{4.3}$$

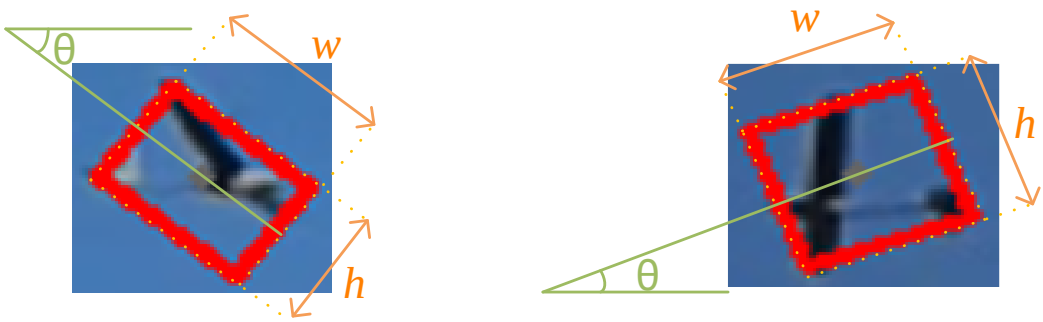

Figure 4.6: OBB angle (θ), width (w), and height (h) illustration.

4.3.2 Orientation hypotheses generation

From the captured frame, we will obtain $\mathbf{B} = [\theta, x, y, w, h]$ (Figure 4.7). The difference between θ and R of the observation and all θ^i and R^i from the database is calculated online using the *Euclidean* distance:

$$d\left(\theta, R, \theta^i, R^i\right) = \sqrt{(\theta - \theta^i)^2 + (R - R^i)^2} \tag{4.4}$$

The poses are ordered by its distance value $d\left(\theta, R, \theta^i, R^i\right)$, and the ones with the lower value will be used as samples (Figure 4.8). These samples represent the database hypotheses set that better describe our observation. In the first iteration, we will retrieve the M poses, and after the first iteration, we will use $M - M_0$ poses (Figure 4.1). Since in the database generation (Section 4.3.1), we have rendered the UAV at a fixed position, to obtain the 3D position, we apply a relation between the obtained OBBs, as will be described in Section 4.3.3.

4.3.3 Translation hypotheses generation

We have adopted the *weak-perspective* model [Carceroni & Brown, 1997; Lee & Park, 2006; Lee *et al.* , 2006; Bradski & Kaehler, 2008] since the UAV size is small when compared to the camera distance. This model assumes that the object points are all at the same depth,

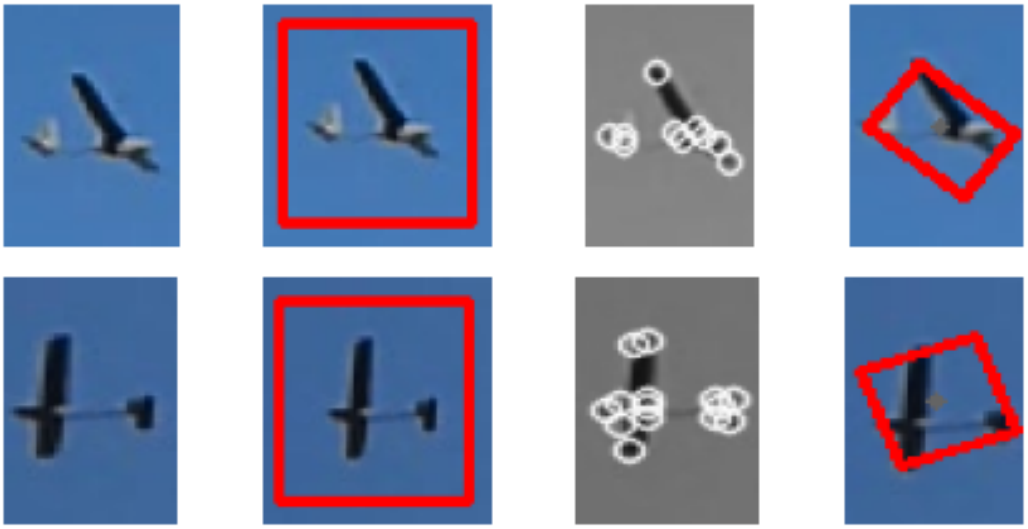

Figure 4.7: Observation frame OBB detection. From left to right: (*i*) original image, (*ii*) detected ROI, (*iii*) FAST features, and (*iv*) OBB.

Figure 4.8: Some selected possibilities from the best database matches.

projected in a plane parallel to the image. We also assume that the object size variations are only due to distance scaling. The Z coordinate can be approximated by the relationship between the OBB areas and depth according to:

$$Z = Z^i \sqrt{\frac{A^i}{A}} \tag{4.5}$$

where A^i corresponds to the OBB i area, and A corresponds to the observation OBB area. The X and Y coordinates for each particle are calculated by the relation between the coordinate of the center of the observed OBB (x, y), the obtained Z coordinate, and the intrinsic camera parameters (obtained by calibration) given by the focal length $f = (f_x, f_y)$ and the camera center coordinates $c = (c_x, c_y)$ (Section 3.1.7).

$$X = \frac{Z(x - c_x)}{f_x} \tag{4.6}$$

$$Y = \frac{Z(y - c_y)}{f_y} \tag{4.7}$$

An accurate camera calibration step [Bouguet, 2008] is essential to ensure precision in system performance (Section 3.1). The database performance evaluation will be described in Section 6.4.

Chapter 5

Tracking

Evolution is a process of constant branching and expansion.

Stephen Jay Gould

Chapter contents

This chapter presents the proposed tracking stage, detailing the developed proposal generator architecture variants, the adopted motion filtering techniques, the approximate weighting and resampling strategies, and the explored pose optimization schemes.

5.1 Proposed system structure

As described in Section 3.5, the tracking module (Figure 3.6) is divided into three different stages (Figure 5.1): (i) proposal (Section 5.2), (ii) approximate weighting and resampling (Section 5.4), and (iii) pose optimization (Section 5.5). In the proposal step, we test different variants (Section 5.2): (i) using the boosted particles solely on the current frame, or (ii) mixing them in a standard PF, or (iii) mixing them in a UPF using 3D model-based likelihood models. In the approximate weighting and resampling step, we have also used a 3D model-based likelihood model approximated by alternative image similarity metrics, evaluated in Section 5.4.1. The resampling strategy to reduce the discrepancy between the particle weights is explained in Section 5.4.2. In the pose optimization step, we have explored again 3D model-based strategies (Section 5.5) to search locally and abridge the sub-optimality of the filter.

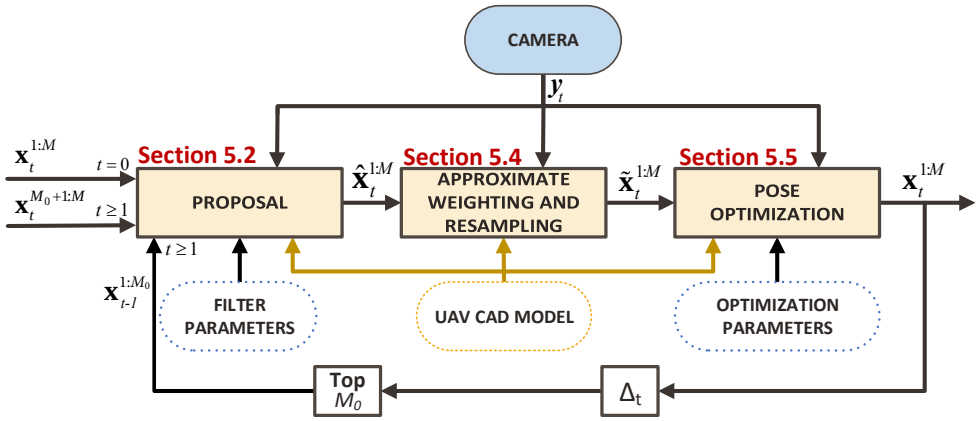

Figure 5.1: Tracking (*proposed structure*).

5.2 Proposal generator architecture variants

Section contents

The proposal distribution is fundamental in the architecture of PFs. Its role is to generate a set of particles from a proposal distribution that should be as close as possible to the true posterior distribution. Typically, it uses a prior based on a motion model applied to the information of the previous time step and blends it with likelihood information on the current time step. In our work, we have tested five different combinations: (i) neglect temporal information and use only pose boosting (Section 5.2.1), (ii) mix pose boosted particles with a PF based on a constant velocity model (Section 5.2.2), (iii) mix pose boosted particles with a standard UPF (Section 5.2.3), (iv) mix pose boosted particles with a UPF that uses UKF for translational motion and a UBiF for the rotational motion (Section 5.2.4), and (v) mix pose boosted particles with a UPF that uses UKF for translational motion and a UBiGaF for the rotational motion (Section 5.2.4).

5.2.1 Pose boosted proposal

The pose boosted proposal does not use previous frame information or motion filtering relying only on current frame information to generate the proposal particle set $\hat{\mathbf{x}}_t^{1:M} = \mathbf{x}_t^{1:M}$ (Figure 5.1), as described in Chapter 4.

5.2.2 Proposal with prediction

As described in Section 5.2.1, the pose boosting sampling only uses information from the current frame. The importance sampling using prediction (Figure 5.2), on the other hand, uses information from the past, encoded in a subset of the particles of the previous iteration $\mathbf{x}_{t-1}^{1:M_0}$ (the best M_0 particles). In the first iteration, all the particles of the proposal come from the pose boosting $\hat{\mathbf{x}}_t^{1:M} = \mathbf{x}_t^{1:M}$. After the first iteration, the proposal distribution will

be composed of particles coming from the previous iteration $\mathbf{x}_{t-1}^{1:M_0}$ and particles obtained from the pose boosting sampling $\mathbf{x}_t^{M_0+1:M}$ (M is the total number of particles). The adopted state transition model is described in Section 3.1.5. The PFs traditionally used in CV (e.g. condensation algorithm [Blake & Isard, 1997]) do not incorporate the current frame and only use the transition prior ($M_0 = M$) to generate hypotheses. The proposal with UKF explained below incorporate these observations more systematically (Section 5.2.3).

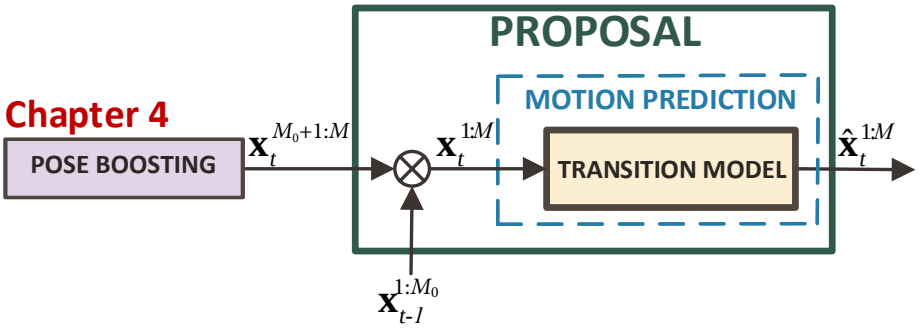

Figure 5.2: Proposal with prediction.

5.2.3 Proposal with UKF

To further improve the proposal distribution, we have applied a UKF to each particle (Appendix A) similar to what is performed in the UPF (Section 3.4). The translational and rotational motions $\mathbf{x}_t^{1:M}$ are filtered using this approach, as described in Figure 5.3. In the first iteration, all the particles of the proposal come from the pose boosting $\hat{\mathbf{x}}_t^{1:M} = \mathbf{x}_t^{1:M}$. After the first iteration, the particle set will be composed of particles coming from the previous iteration $\mathbf{x}_{t-1}^{1:M_0}$ and particles obtained from the pose boosting sampling $\mathbf{x}_t^{M_0+1:M}$ originating the set $\mathbf{x}_t^{1:M}$. The current measurement $\mathbf{z}_t$, is given by the particle with the highest weight using a similarity metric (Section 5.4.1). This is a coarse estimate of the pose but still effective in improving the proposal distribution.

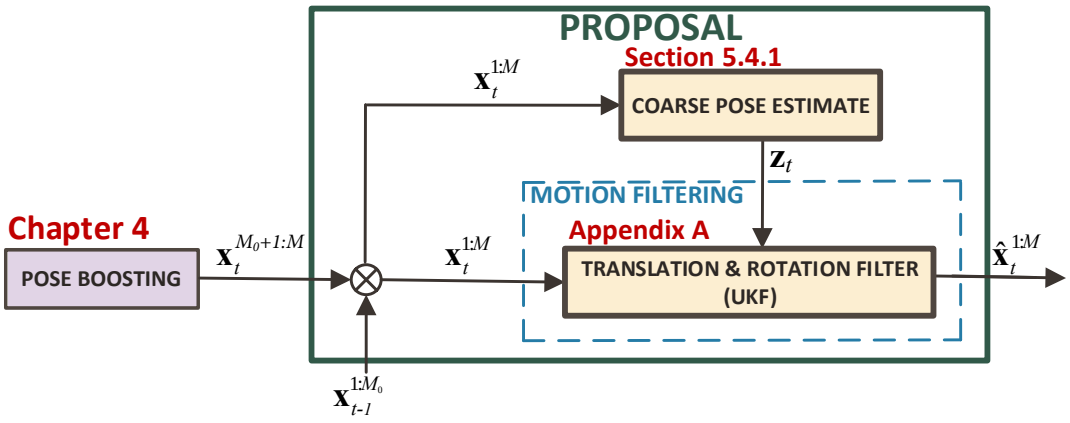

Figure 5.3: Proposal with UKF.

5.2.4 Proposal with UKF and UBi(Ga)F

When we use the UKF for the rotational motion $\mathbf{r}_t^{1:M}$ filtering (Appendix A), we have to consider a small angle assumption to quantify the existing uncertainty [Crassidis & Markley, 2003; Darling & DeMars, 2016b; Markley & Crassidis, 2014; Pessanha Santos *et al.* , 2015]. In a periodic domain like the manifold of orientations in a 3D space, the Gaussian approach is not a good approximation, especially in the presence of strong noise. We can use the Bi and BiGa distributions described in Appendix C in a filtering structure to obtain better orientation estimates. This formulation allows us to take into account the periodic nature of the rotation (Bi and BiGa) and even the existing angular velocity uncertainty in the angular pose estimation in its natural manifold (BiGa). We propose the UBiF and the UBiGaF to address this issue (Section 5.3). These filters use the Bi and the BiGa distributions in their formulation to better cope with the orientation manifold. The main difference between the filters is in the update step. The UBiF update step is adapted from the Bayes filter formulation, and the UBiGaF update step relies on a UKF structure. The remaining filtering steps are the same as described in Section 5.2.3. $\hat{\mathbf{x}}_t^{1:M}$ gives the final set of motion filtered particles (Figure 5.4).

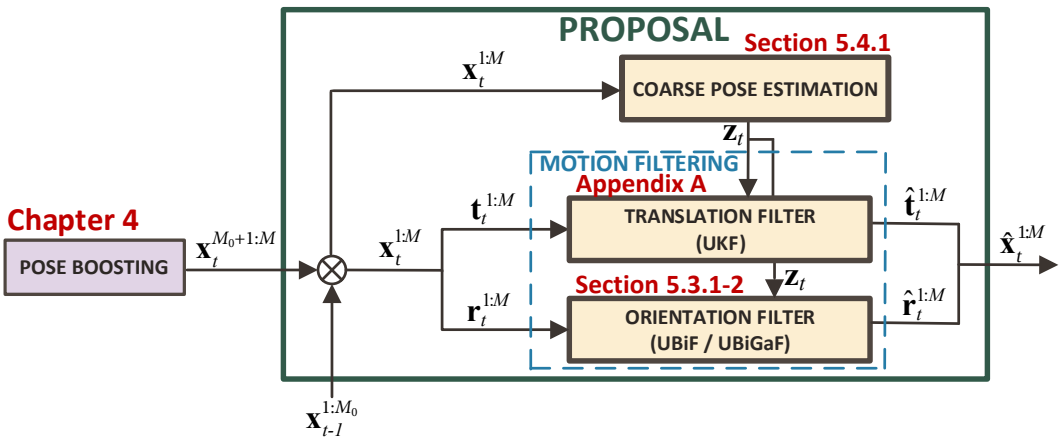

Figure 5.4: Proposal using a UKF with a UBiF or UBiGaF.

5.3 Motion filtering

Section contents

In the proposal with UKF (Section 5.2.3), we use a UKF for the translational and rotational motion filtering of all the particles in the set $\mathbf{x}_t^{1:M}$. As we are using a linear model for translation, a simple KF could be applied. To facilitate the transition between the linear and the angular filter formulations, we will use a discrete-time UKF [Cheon & Kim, 2007; Crassidis & Markley, 2003; Julier, 2002; Kraft, 2003; Van Der Merwe *et al.* , 2001; Zhou *et al.* , 2011], as described in Appendix A. In the proposal with UKF and UBiF or UBiGaF (Section 5.2.4), we

use a UKF for the translational motion filtering $\mathbf{t}_t^{1:M}$ and a UBiF (Section 5.3.1) or UBiGaF (Section 5.3.2) for the rotational motion filtering $\mathbf{r}_t^{1:M}$.

5.3.1 Unscented Bingham Filter (UBiF)

In a periodic domain like the manifold of orientations in a 3D space, the Gaussian model is not a good approximation, especially in the presence of strong noise. The Bi distribution [Bingham, 1974; Mardia & Jupp, 2000] (Appendix C) is an antipodally symmetric distribution that represents a zero-mean Gaussian distribution projected on the unit hypersphere (Section C.1). Since the product of two Bi distributions is closed under multiplication after renormalization (Section C.1.2), we can use the UBiF with an update step directly derived from the Bayes filter formulation [Ho & Lee, 1964; Thrun *et al.* , 2005]. We apply a UBiF to the orientation part of the state vector. As in other filtering frameworks, the used UBiF has two stages: (i) prediction (Section 5.3.1.1), and (ii) update (Section 5.3.1.2). The angular velocities will be obtained from the orientation difference between iterations. The UBiF schematic view is described in Figure 5.5. The performance evaluation of this filtering scheme will be made in Section 6.7.

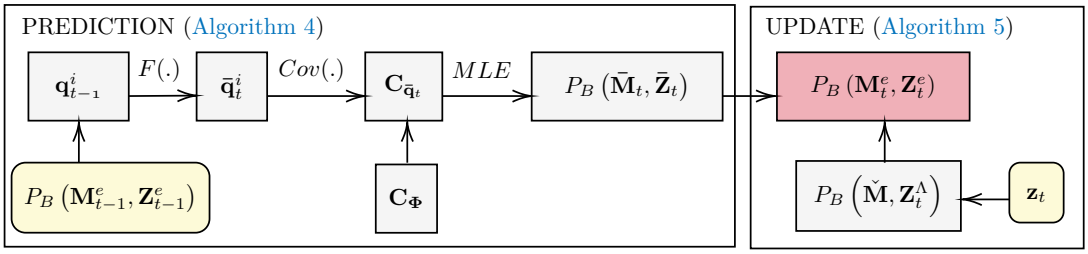

Figure 5.5: UBiF schematic view.

5.3.1.1 Prediction

The prediction step is described in Algorithm 4. For the rotation case, the system model is given by:

$$\mathbf{q}_t = \mathbf{F}(\mathbf{q}_{t-1}) \otimes \mathbf{\Phi}_{t-1} \tag{5.1}$$

where $\mathbf{q}_{t-1} \sim P_B\left(\mathbf{M}_{t-1}^e, \mathbf{Z}_{t-1}^e\right)$ is the orientation at time $t-1$ and $\mathbf{\Phi}_{t-1} \sim P_B(\mathbf{M}_{t-1}^\Phi, \mathbf{Z}_{t-1}^\Phi)$ is the Bi distributed system noise where $\mathbf{M}$ is an orthogonal matrix describing the orientation of the distribution and $\mathbf{Z}$ is the concentration matrix that controls the spread of the distribution around its mean (Section C.1). The system dynamic $\mathbf{F}(.)$ is given by (3.12).

After the prediction step (Algorithm 4), the predicted system state is described by the Bi distribution $P_B(\bar{\mathbf{M}}_t, \bar{\mathbf{Z}}_t)$ (Figure 5.5).

5.3.1.2 Measurement update

The measurement update step is described in Algorithm 5. The measurement model is represented as:

Algorithm 4 Unscented Bingham Filter (UBiF) - Prediction Step

▷ **Initialization:** $\mathbf{\Phi}_{t-1}$ (5.1)
▷ **Inputs:** $\mathbf{M}^e_{t-1}, \mathbf{Z}^e_{t-1}$ (5.1)
1. Approximate the current system state (Section C.1.5):
   $\mathbf{q}_{t-1} \sim P_B\left(\mathbf{M}^e_{t-1}, \mathbf{Z}^e_{t-1}\right)$
2. Obtain the sigma points $\mathbf{q}^i_{t-1}$ using deterministic sampling (Appendix D):
   $\mathbf{q}^i_{t-1} = \mathbf{M}^e_{t-1}\tilde{\mathbf{q}}^i_{t-1} \quad i = 1, \ldots, 7$
3. Propagate the sigma points $\mathbf{q}^i_t$ using the system model $\mathbf{F}(.)$ (3.12):
   $\bar{\mathbf{q}}^i_t = \mathbf{F}(\mathbf{q}^i_{t-1}) \quad i = 1, \ldots, 7$
4. Compute the covariance matrix $\mathbf{C}_{\bar{\mathbf{q}}_t}$ from the sigma points (C.8):
   $\mathbf{C}_{\bar{\mathbf{q}}_t} = Cov(\bar{\mathbf{q}}_t)$
5. Obtain the covariance matrix $\mathbf{C}_{\mathbf{\Phi}}$ from the Bi system noise $\mathbf{\Phi}_t$ (C.6):
   $\mathbf{C}_{\mathbf{\Phi}} = Cov(\mathbf{\Phi}_t)$
6. Obtain the covariance matrices composition $\mathbf{C}_{\bar{\mathbf{q}}'_t}$ (C.7):
   $\mathbf{C}_{\bar{\mathbf{q}}'_t} = Cov(\bar{\mathbf{q}}_t \otimes \mathbf{\Phi}_t)$
7. Estimate the Bi distribution from the obtained covariance (C.9):
   $P_B(\bar{\mathbf{M}}_t, \bar{\mathbf{Z}}_t) \sim MLE(\mathbf{C}_{\bar{\mathbf{q}}'_t})$
▷ **Outputs:** $\bar{\mathbf{M}}_t, \bar{\mathbf{Z}}_t$

$$\mathbf{z}_t = \mathbf{H}(\mathbf{q}_t) \otimes \mathbf{\Lambda}_t \tag{5.2}$$

where $\mathbf{z}_t \in \mathbb{S}^3$ is the measurement at time t and $\mathbf{\Lambda}_t \sim P_B\left(\mathbf{M}^\Lambda_t, \mathbf{Z}^\Lambda_t\right)$ is the Bi distributed measurement noise. The current observation $\mathbf{z}_t$ is given by the orientation quaternion of the approximate maximum likelihood particle, as described in Figure 5.4. Function $\mathbf{H}(.)$ relates the measurement $\mathbf{z}_t$ to the values of the orientation $\mathbf{q}_t$ (identity function in our study case). Choosing $\mathbf{M}^\Phi_t = \mathbf{M}^\Lambda_t = \mathbf{I}_{4\times 4}$ is equivalent to the concept of zero-mean noise in the *Euclidean* space [Gilitschenski *et al.* , 2016; Bingham, 1974].

Algorithm 5 Unscented Bingham Filter (UBiF) - Update Step

▷ **Initialization:** $\mathbf{\Lambda}_t$ (5.2)
▷ **Inputs:** $\bar{\mathbf{M}}_t, \bar{\mathbf{Z}}_t$ (Algorithm 4) and $\mathbf{z}_t$ (5.2)
1. Disturb the measure $\mathbf{z}_t$ using the Bi noise $\mathbf{\Lambda}_t$ (5.3):
   $P_B(\check{\mathbf{M}}, \mathbf{Z}^\Lambda_t)$
2. The measurement update can be derived from the *Bayes theorem*:
   $\underbrace{P(\bar{\mathbf{q}}'_t|\mathbf{z}_t)}_{P_B(\mathbf{M}^e_t, \mathbf{Z}^e_t)} = \mathrm{C} \cdot \underbrace{P(\mathbf{z}_t|\bar{\mathbf{q}}'_t)}_{P_B(\check{\mathbf{M}}, \mathbf{Z}^\Lambda_t)} \cdot \underbrace{P(\bar{\mathbf{q}}'_t)}_{P_B(\bar{\mathbf{M}}_t, \bar{\mathbf{Z}}_t)}$
   where C is a normalization constant;
3. The final estimate is then obtained from (C.3):
   $P_B(\mathbf{M}^e_t, \mathbf{Z}^e_t) = P_B(\check{\mathbf{M}}, \mathbf{Z}^\Lambda_t) \cdot P_B(\bar{\mathbf{M}}_t, \bar{\mathbf{Z}}_t)$
▷ **Outputs:** $\mathbf{M}^e_t, \mathbf{Z}^e_t$

To be able to apply the measurement update step (Algorithm 5), the noise is rotated (disturbed) according to the actual measurement $\mathbf{z}_t$ according to (C.4):

$$\begin{aligned} P(\mathbf{z}_t \,|\, \bar{\mathbf{q}}'_t) &= P_B(\bar{\mathbf{q}}'^{-1}_t) \otimes \mathbf{z}_t; \mathbf{M}^\Lambda_t, \mathbf{Z}^\Lambda_t) = P_B(diag(-1,-1,-1,1) \cdot \bar{\mathbf{q}}'_t \otimes \mathbf{z}_t; \mathbf{M}^\Lambda_t, \mathbf{Z}^\Lambda_t) = \\ &= P_B(\bar{\mathbf{q}}'_t; \underbrace{diag(-1,-1,-1,1) \cdot \mathbf{M}^\Lambda_t \otimes \mathbf{z}_t}_{\check{\mathbf{M}}}, \mathbf{Z}^\Lambda_t) = P_B(\check{\mathbf{M}}, \mathbf{Z}^\Lambda_t) \end{aligned} \tag{5.3}$$

where $P_B(\check{\mathbf{M}}, \mathbf{Z}^\Lambda_t)$ represents the Bi distributed measurement noise $P_B\left(\mathbf{M}^\Lambda_t, \mathbf{Z}^\Lambda_t\right)$ with a peak aligned with $\mathbf{z}_t$ and spread controlled by $\mathbf{Z}^\Lambda_t$.

The estimate is described by $P_B(\mathbf{M}^e_t, \mathbf{Z}^e_t)$, directly obtained by the product of the Bi dis-

tributed system state $P_B(\bar{\mathbf{M}}_t, \bar{\mathbf{Z}}_t)$ with the Bi disturbed measure $P_B(\check{\mathbf{M}}, \mathbf{Z}_t^{\Lambda})$ (Algorithm 5). The update quaternion $\mathbf{q}_t$ is obtained from the $P_B(\mathbf{M}_t^e, \mathbf{Z}_t^e)$ mode (Algorithm 5). The quaternion error is obtained by multiplying the previous quaternion $(\mathbf{q}_{t-1})$ with the conjugate of the estimated one $(\bar{\mathbf{q}}_t)$ [Finkelstein *et al.* , 1962; Conway, 1937]. The angular velocities are obtained converting to the angle-axis representation, according to:

$$\omega_x = \frac{2\cos^{-1}(q_4)}{\Delta t} \times \frac{q_1}{\| \mathbf{q}_e \|} \tag{5.4}$$

$$\omega_y = \frac{2\cos^{-1}(q_4)}{\Delta t} \times \frac{q_2}{\| \mathbf{q}_e \|} \tag{5.5}$$

$$\omega_z = \frac{2\cos^{-1}(q_4)}{\Delta t} \times \frac{q_3}{\| \mathbf{q}_e \|} \tag{5.6}$$

where Δt is the sampling interval and $\mathbf{q}_e = \mathbf{q}_{t-1} \otimes \bar{\mathbf{q}}_t = [q_1, q_2, q_3, q_4]^T$ (3.7).

As described in Section 5.3.1, the UBiF uses a Bi distribution to model the periodic nature of rotations better and decrease the estimate error. The performance evaluation of this filtering scheme will be made in Section 6.7.

5.3.2 Unscented Bingham-Gauss Filter (UBiGaF)

The UBiF (Section 5.3.1) does not quantify the uncertainty of the correlation between angular velocity $\boldsymbol{\omega}$ and the quaternion attitude $\mathbf{q}$ on their natural manifold. We will use the BiGa distribution [Darling & DeMars, 2016a; Jazwinski, 1970] (Appendix C) that allows capturing this correlation in a filtering structure (Section C.2). The BiGa is a distribution that consists in the product of a Bi distribution and a Gaussian distribution conditioned on the Bi distributed random variables (Section C.2). Thus, we propose the UBiGaF. As described in Section 5.3.1, the multiplication of two Bi distributions is closed under multiplication after renormalization (Section C.1.2), but the same did not happen using the BiGa distribution since we do not have a closed-form multiplication. To be able to incorporate it on a filtering structure, we developed an update step that was based on the Unscented Transform (UT) [Rui & Chen, 2001a; Julier, 2002; Li *et al.* , 2003] with a structure similar to the UKF (Appendix A). The UBiGaF is separated into: (i) prediction (Section 5.3.2.1), and (ii) update (Section 5.3.2.2). The UBiGaF schematic view is described in Figure 5.6. The performance evaluation of this filtering scheme will be made in Section 6.7.

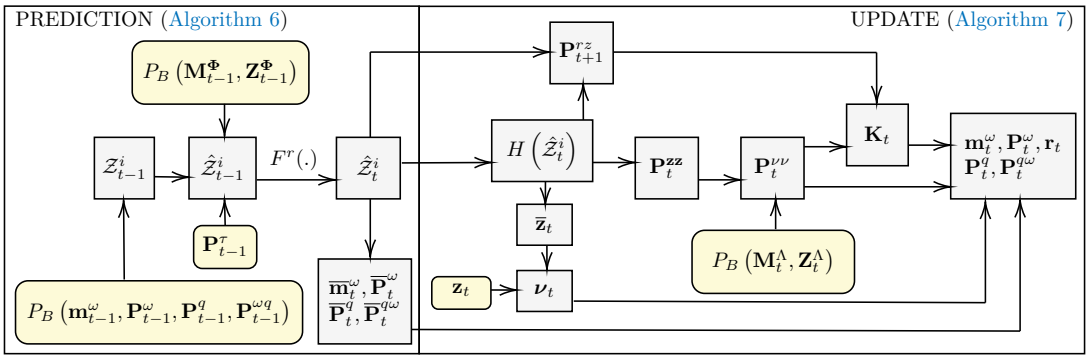

Figure 5.6: UBiGaF schematic view.

5.3.2.1 Prediction

The prediction step is described in Algorithm 6. The system state at time $t-1$ is given by:

$$\mathbf{r}_{t-1}^T = \left[\mathbf{q}_{t-1}^T, \boldsymbol{\omega}_{t-1}^T\right] \tag{5.7}$$

where $\mathbf{q}_{t-1}$ is the attitude quaternion and $\boldsymbol{\omega}_{t-1}$ is its angular velocity at time instant $t-1$. The state vector is assumed to be BiGa distributed with parameters defined by $\mathbf{m}_{t-1}^{\omega}$ (C.13), $\mathbf{P}_{t-1}^{\omega}$ (C.14), $\mathbf{P}_{t-1}^{q}$ (C.15) and $\mathbf{P}_{t-1}^{q\omega}$ (C.16) at time instant $t-1$. The system model is given by:

$$\mathbf{r}_t = \mathbf{F}(\mathbf{r}_{t-1}) \odot \boldsymbol{\Psi}_{t-1} \tag{5.8}$$

where $\boldsymbol{\Psi}_{t-1} \sim P_{BG}$ with Covariance $\mathbf{P}_{t-1}^{\tau}$ for the angular velocity part and Covariance $\boldsymbol{\Phi}_{t-1} \sim \mathbf{P}_{t-1}^{\Phi} \sim P_B(\mathbf{M}_{t-1}^{\boldsymbol{\Phi}}, \mathbf{Z}_{t-1}^{\boldsymbol{\Phi}})$ for the orientation part, $\mathbf{F}(.)$ is the motion model (3.12) and $\odot$ represents the BiGa composition obtained using the sigma points representation (Appendix E).

Algorithm 6 Unscented Bingham-Gauss Filter (UBiGaF) - Prediction Step

▷ **Initialization:** $\boldsymbol{\Psi}_{t-1}$ (5.8)
▷ **Inputs:** $\mathbf{m}_{t-1}^{\omega}, \mathbf{P}_{t-1}^{\omega}, \mathbf{P}_{t-1}^{q}, \mathbf{P}_{t-1}^{q\omega}$ (5.8)
1. Approximate the current system state (C.13, C.14, C.15, C.16):
   $\left[\mathbf{q}_{t-1}^T, \boldsymbol{\omega}_{t-1}^T\right] \sim P_{BG}\left(\left[\mathbf{q}_{t-1}^T, \boldsymbol{\omega}_{t-1}^T\right]; \mathbf{m}_{t-1}^{\omega}, \mathbf{P}_{t-1}^{\omega}, \mathbf{P}_{t-1}^{q}, \mathbf{P}_{t-1}^{q\omega}\right)$
2. Obtain the sigma points $\boldsymbol{\mathcal{Z}}_{t-1}^i$ using deterministic sampling (Appendix E);
3. Add uncertainty $\mathbf{P}_{t-1}^{\tau}$ to the angular velocity covariance:
   $\hat{\mathbf{P}}_{t-1}^{\omega} = \mathbf{P}_{t-1}^{\omega} + \mathbf{P}_{t-1}^{\tau}$
4. Add Bi noise $\boldsymbol{\Phi}_{t-1}$ to the orientation covariance (C.7):
   ${}^{\boldsymbol{\Phi}}\mathbf{P}_{t-1}^{q} = Cov\left(\mathbf{P}_{t-1}^{q} \otimes \boldsymbol{\Phi}_{t-1}\right)$
5. Add uncertainty $\hat{\mathbf{P}}_{t-1}^{\omega}$ to the sigma points angular velocity part and ${}^{\boldsymbol{\Phi}}\mathbf{P}_{t-1}^{q}$ to the quaternion part originating the sigma points $\hat{\boldsymbol{\mathcal{Z}}}_{t-1}^i$;
6. Propagate each one of the sigma points $\hat{\boldsymbol{\mathcal{Z}}}_{t-1}^i$ (3.12):
   $\hat{\boldsymbol{\mathcal{Z}}}_{t}^i = \mathbf{F}^r(\hat{\boldsymbol{\mathcal{Z}}}_{t-1}^i) = f_q(\hat{\boldsymbol{\mathcal{Z}}}_{t-1}^i) \otimes \mathbf{q}_M$
   where f_q is the quaternion part of the considered sigma point and $\mathbf{q}_M$ is the quaternion motion; given by the angular part $f_\omega(\hat{\boldsymbol{\mathcal{Z}}}_{t-1}^i)$ of each sigma point;
7. Compute $\bar{\mathbf{m}}_t^{\omega}$ (E.19), $\bar{\mathbf{P}}_t^{\omega}$ (E.20), $\bar{\mathbf{P}}_t^{q}$ (E.21) and $\bar{\mathbf{P}}_t^{q\omega}$ (E.22) from $\hat{\boldsymbol{\mathcal{Z}}}_t^i$.
▷ **Outputs:** $\bar{\mathbf{m}}_t^{\omega}, \bar{\mathbf{P}}_t^{\omega}, \bar{\mathbf{P}}_t^{q}, \bar{\mathbf{P}}_t^{q\omega}, \hat{\boldsymbol{\mathcal{Z}}}_t^i$

The quaternion motion $\mathbf{q}_M = [q_1, q_2, q_3, q_4]^T$ (3.7) used to propagate each one of the sigma points $\hat{\boldsymbol{\mathcal{Z}}}_{t-1}^i$ (Algorithm 6) is given by (3.14):

$$q_1 = \frac{\omega_x}{\parallel \boldsymbol{\omega} \parallel} \sin\left(\frac{\parallel \boldsymbol{\omega} \parallel \Delta t}{2}\right) \tag{5.9}$$

$$q_2 = \frac{\omega_y}{\parallel \boldsymbol{\omega} \parallel} \sin\left(\frac{\parallel \boldsymbol{\omega} \parallel \Delta t}{2}\right) \tag{5.10}$$

$$q_3 = \frac{\omega_z}{\parallel \boldsymbol{\omega} \parallel} \sin\left(\frac{\parallel \boldsymbol{\omega} \parallel \Delta t}{2}\right) \tag{5.11}$$

$$q_4 = \cos\left(\frac{\parallel \boldsymbol{\omega} \parallel \Delta t}{2}\right) \tag{5.12}$$

where $f_\omega(\hat{\boldsymbol{\mathcal{Z}}}_{t-1}^i) = [\omega_x, \omega_y, \omega_z]^T$ and Δt is the sampling interval. We calculate the BiGa parameters from the obtained sigma points after propagation $\hat{\boldsymbol{\mathcal{Z}}}_t^i$, as illustrated in Section E.3.

After the prediction step (Algorithm 6), the predicted system state is described by a BiGa distribution with parameters defined by $\bar{\mathbf{m}}_t^{\omega}$, $\bar{\mathbf{P}}_t^{\omega}$, $\bar{\mathbf{P}}_t^{q}$, $\bar{\mathbf{P}}_t^{q\omega}$ (Figure 5.6).

5.3.2.2 Measurement update

The measurement update step is described in Algorithm 7, and is similar to the used in the UBiF (Section 5.3.1.2). The measurement model is also represented as:

$$\mathbf{z}_t = \mathbf{H}(\mathbf{q}_t) \otimes \mathbf{\Lambda}_t \tag{5.13}$$

where $\mathbf{z}_t \in \mathbb{S}^3$ is the current observation (Figure 5.4) at time t and $\mathbf{\Lambda}_t \sim P_B\left(\mathbf{M}_t^{\Lambda}, \mathbf{Z}_t^{\Lambda}\right)$ is the Bi distributed measurement noise. Function $\mathbf{H}(.)$ relates the measurement $\mathbf{z}_t$ to the values of the orientation $\mathbf{q}_t$ (identity function in our study case).

Algorithm 7 Unscented Bingham-Gauss Filter (UBiGaF) - Update Step

▷ **Initialization:** $\mathbf{\Lambda}_t$ (5.13)
▷ **Inputs:** $\bar{\mathbf{m}}_t^{\omega}$, $\bar{\mathbf{P}}_t^{\omega}$, $\bar{\mathbf{P}}_t^{q}$, $\bar{\mathbf{P}}_t^{q\omega}$, $\hat{\boldsymbol{\mathcal{Z}}}_t^{i}$ (Algorithm 6) and $\mathbf{z}_t$ (5.13)
1. Predict measurement expected value $\bar{\mathbf{z}}_t$ (A.23) from $\hat{\boldsymbol{\mathcal{Z}}}_t^{i}$ angular part;
2. Predict measurement covariance $\mathbf{P}_t^{zz}$ (A.24) from $\hat{\boldsymbol{\mathcal{Z}}}_t^{i}$ angular part;
3. Obtain the innovation $\boldsymbol{\nu}_t$ (A.25);
4. Obtain the innovation covariance $\mathbf{P}_t^{\nu\nu}$ (C.7):
   $\mathbf{P}_t^{\nu\nu} = Cov(\boldsymbol{\nu}_t \otimes \mathbf{\Lambda}_t)$
5. Computation of the cross-correlation matrix $\mathbf{P}_t^{rz}$ (A.27);
6. Computation of the Kalman gain $\mathbf{K}_t$ (A.28):
   $\mathbf{K}_t = \mathbf{P}_t^{rz}\left(\mathbf{P}_t^{\nu\nu}\right)^{-1}$
7. Update of the *a posteriori* state estimate $\mathbf{r}_t$ (A.29);
8. Retrieve $\mathbf{m}_t^{\omega}$, $\mathbf{P}_t^{\omega}$, $\mathbf{P}_t^{q}$, $\mathbf{P}_t^{q\omega}$ from the obtained state covariance $\mathbf{P}_t^{r}$ (A.30).
▷ **Outputs:** $\mathbf{r}_t, \mathbf{m}_t^{\omega}$, $\mathbf{P}_t^{\omega}$, $\mathbf{P}_t^{q}$, $\mathbf{P}_t^{q\omega}$

Using the BiGa formulation, we can model the full rotational noise, both in its angular position and velocity components. As initially described in Section 5.3.2, we do not have a closed-form multiplication for the product of two BiGa distributions. To be able to deal with that, we have adopted a structure similar to the UKF (Appendix A). Using this filtering structure, we can obtain the *a posteriori* state estimate $\mathbf{r}_t$ and state covariance $\mathbf{P}_t^{r}$, as described in Algorithm 7. Contrarily to the UBiF, we do not need to estimate the angular velocities from the quaternion difference since we estimate them directly in the filter. The performance evaluation of this filtering scheme will be made in Section 6.7.

5.4 Approximate weighting and Resampling

Section contents

This section describes the explored similarity metrics used to approximate the likelihood function (Section 5.4.1), and the adopted resampling strategies (Section 5.4.2).

5.4.1 Pose evaluation

Nowadays, all UAVs have their CAD model available, so their pose can be estimated using 3D model-based methods. In our approach, we synthetically generate images of the UAV, with a certain rotation and translation, and compare it with the actual image pixel information using a similarity metric. The increase of computational capability allows the approximation of very complex methods and real-time particle evaluation. The similarity metric should be designed to be robust to illumination changes and background clutter. We have tested three different similarity metrics: (i) color [Pessanha Santos *et al.* , 2014b; Taiana *et al.* , 2010, 2008] (Section 5.4.1.1), (ii) contour [Choi & Christensen, 2011] (Section 5.4.1.2), and (iii) Distance Transform (DT) [Vicente *et al.* , 2016] (Section 5.4.1.3). Their performance was evaluated, taking into account realistic backgrounds and the needed processing time, as described in Section 6.6.

5.4.1.1 Color similarity metric

The color similarity metric [Pessanha Santos *et al.* , 2014b; Taiana *et al.* , 2010, 2008] measures the difference between the RGB color space histogram (8 bins for each color – $B = 24$) of two areas of the real captured image: (i) the inside, and (ii) the outside of the UAV boundary projected in the real image. This approach assumes that the UAV is all the same color, being distinct from the expected tracking backgrounds. The difference between them is calculated using the *Bhattacharyya* similarity metric [Gómez-Luna *et al.* , 2013] according to:

$$p(\mathbf{y}_t \mid \mathbf{x}_t) \propto d_{color} = 1 - \sum_{b=1}^{B} \sqrt{\mathbf{h}^{inner}(b) \cdot \mathbf{h}^{outer}(b)} \tag{5.14}$$

where $\mathbf{h}^{inner}$ is the inner histogram, $\mathbf{h}^{outer}$ is the outer histogram, and b is the respective histogram bin. One example of inner and outer histogram regions can be seen in Figure 5.7, where we have the inner histogram that represents the UAV hypothesis area and the outer histogram that represents the background area between the obtained OBB and the UAV.

Figure 5.7: An example of the inner (*black*) and outer (*blue*) regions.

5.4.1.2 Contour similarity metric

In the contour similarity metric, the set of visible edges (from the 3D CAD model) is projected onto the captured image plane according to the currently tested pose hypothesis. The edge line segments are identified, and a sample point in the middle is generated. Then a 1D perpendicular search (Figure 5.8) is made to match the sample points with the nearest edge. After calculating the matches, the contour metric is calculated as in [Choi & Christensen, 2011]:

$$p(\mathbf{y}_t \mid \mathbf{x}_t) \propto L_{contour} = \exp^{-\lambda_v \frac{(p_v - p_m)}{p_v}} \cdot \exp^{-\lambda_e \bar{e}} \tag{5.15}$$

where $\bar{e}$ is the arithmetic average distance between the sample points and the matched edge points, λ_v and λ_e are sensitivity terms used to tune the metric, p_v is the number of visible sample points and p_m is the number of matched sample points.

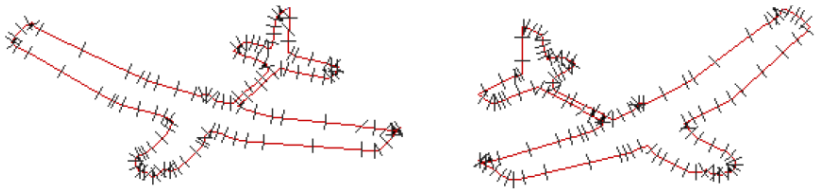

Figure 5.8: UAV sampled points with 1D search lines (*black*).

5.4.1.3 DT similarity metric

To compute the DT of an image, we apply one edge detector (e.g. *Canny* edge detector [Canny, 1986]) to the image, and then for each pixel, we compute its distance to the closest edge [Borgefors, 1986], as seen in Figure 5.9. Then, the DT metric is calculated as in [Vicente *et al.* , 2016]:

$$p(\mathbf{y}_t \mid \mathbf{x}_t) \propto L_{DT} = \exp^{-\sigma d} \tag{5.16}$$

where σ is a fine-tuning parameter and d is given by:

$$d = \frac{1}{k} \sum_{i=0}^{B} E(\bar{y}(i)) \cdot D(y(i)) \tag{5.17}$$

where k is the number of edge pixels on the hypothesis to compare, B is the total number of image pixels, y is the captured frame (Figure 1.9), $\bar{y}$ is the pose hypothesis image (Figure 5.9), $D(y)$ is the distance transform of the captured frame and $E(\bar{y})$ is the edge map of the pose hypothesis image to compare (Figure 5.9).

Figure 5.9: UAV image $\bar{y}$ (*left*), edge map $E(\bar{y})$ (*center*), and DT representation $D(\bar{y})$ (*right*).

5.4.2 Resampling

After we approximate the particle weights (Section 5.4.1), we apply a resampling step [Thrun *et al.* , 2005; Haug, 2012; Li *et al.* , 2015a; Pessanha Santos *et al.* , 2017]. The resampling consists in selecting new particle positions and weights such that the discrepancy between the resampled weights is reduced [Douc & Cappé, 2005; Coquelin *et al.* , 2009; Bréhard *et al.* , 2008], i.e. eliminating particles with low approximate weights and by replicating particles

having high approximate weights. We have tested ten traditional resampling schemes, namely the [Beadle & Djuric, 1997; Carpenter *et al.* , 1999; Gordon *et al.* , 1993; Li *et al.* , 2015b; Liu & Chen, 1998; So, 2003; Douc & Cappé, 2005; Bolic *et al.* , 2003; Budhiraja *et al.* , 2007; Crisan & Lyons, 1999; Jianping *et al.* , 2009; Liu *et al.* , 1998]: stratified, systematic, residual, residual systematic, optimal, reallocation, metropolis, minimum sampling, multinomial, and branching, as described in Appendix B. The analysis of the inclusion of the resampling step in the complete system formulation will be performed in Section 6.7.

5.5 Pose optimization

Section contents

The pose optimization objective is to perform a local search (refinement steps) in the particle neighborhood to optimize the used similarity metric using the current frame $\mathbf{y}_t$ information. This local optimization can help to reach a better pose estimate, especially if we are near the global optimum [Gall *et al.* , 2010; Liu *et al.* , 2016a]. All the implemented similarity metrics (Section 5.4.1) are noisy and non-monotonic[1] with pose (Section 6.5), and cannot be optimized with gradient-based methods[2]. We have tested four different algorithms for pose optimization: (i) PFO (Section 5.5.1), (ii) PSO (Section 5.5.2), (iii) modified PSO (Section 5.5.3), and (iv) GAbF (Section 5.5.4). The pose optimization performance evaluation will be made in Section 6.6.

5.5.1 Particle Filter Optimization (PFO)

The PFO algorithm [Zhou & Chen, 2013; Liu *et al.* , 2016a; Zhang *et al.* , 2007] is based on the PF theory iterated on the same input (image frame), as described in Section 3.2. In the PFO, the posterior PDF at time t $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ (Section 3.1.3) is iteratively approximated on each one of the N sequential iterations k $(1 < k \leq N)$ by a set of particles (samples) $\{\bar{\mathbf{p}}_k^m, w_k^m\}_{m=1}^M$ at the same time instant t [Zhou & Chen, 2013; Liu *et al.* , 2016a; Zhang *et al.* , 2007]. In our PFO implementation, $\bar{\mathbf{p}}_k^m = [(\bar{\mathbf{u}}_k^m)^T, (\bar{\mathbf{q}}_k^m)^T]^T$ (3.5) represents the UAV pose samples that are initialized with the pose subpart of $\tilde{\mathbf{x}}_t^{1:M}$ (Figure 5.1) and the observation $\mathbf{y}_t$ is the captured image frame. Since we are performing optimization at the same time instant, the velocity component of the state is not affected $[\mathbf{v}_t^T, \boldsymbol{\omega}_t^T]^T$ (3.5), and the PFO does not use it. Although not used in the PFO implementation, the linear and angular velocity vector of each of the original particles is maintained to be used in the next system iteration (Figure 3.6).

The particle weights will be approximated on each iteration by a similarity metric, as described in Section 5.4.1. On each PFO iteration, we will also perform resampling (Section 5.4.2). The transition model adds some artificial dynamic noise between iterations [Pessanha Santos *et al.* , 2017]:

[1] Multimodal with more than one peak or *mode*.
[2] For instance, *Gradient descent* [Rumelhart *et al.* , 1985] or *Levenberg–Marquardt* [Moré, 1978].

$$\bar{\mathbf{x}}_{k+1}^{1:M} = \bar{\mathbf{x}}_{k}^{1:M} + \zeta \tag{5.18}$$

where ζ is typically a zero-mean Gaussian random variable that can have constant variance, a decreasing variance in time, or another kind of evolution developed for the specific implementation [Flury & Shephard, 2011; Kantas *et al.* , 2015; Liu & West, 2001b]. The amount of added noise must be tuned to decrease the filter convergence time. The artificial noise schemes applied in the experimental tests to generate the proposal distribution will be described in Section 6.6.

Algorithm 8 Particle Filter Optimization (PFO)

▷ **Inputs:** $\mathbf{x}_t^{1:M} = [(\mathbf{u}_t^m)^T, (\mathbf{q}_t^m)^T, (\mathbf{v}_t^{1:M})^T, (\boldsymbol{\omega}_t^{1:M})^T]^T$ (3.5) , N

▷ **Initialization**:

1. Initialize the particle set $\bar{\mathbf{p}}_0^m = [(\bar{\mathbf{u}}_0^m)^T, (\bar{\mathbf{q}}_0^m)^T]^T = [(\mathbf{u}_t^m)^T, (\mathbf{q}_t^m)^T]^T$ (3.5) and the weights w_0^m with $\frac{1}{M}$ creating $\{\bar{\mathbf{p}}_0^m, w_0^m\}_{m=1}^M$.

■ **Importance sampling** $(k = 1 : N)$:

1. Explore the parameter space (3.5) adding noise using a predefined strategy (Section 6.7):
   3D position → $\bar{\mathbf{u}}_k^m = \bar{\mathbf{u}}_{k-1}^m + \mathcal{N}(\mathbf{m}^{trans}, \boldsymbol{\Sigma}^{trans})$
   Orientation → $\bar{\mathbf{q}}_k^m = \bar{\mathbf{q}}_{k-1}^m \otimes \delta\mathbf{q}_r(\boldsymbol{\Sigma}^{rot})$ (3.12)

where $\mathbf{m}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix.

■ **Approximate weighting** $(k = 1 : N)$:

1. Approximate the unnormalized weights (Section 5.4.1):
   $\tilde{w}_k^m \propto \mathit{Approximate\_weighting}(\bar{\mathbf{p}}_k^m, \mathbf{y})$
2. Normalize the obtained approximate weights:
   $w_k^m = \frac{\tilde{w}_k^m}{\sum_{i=1}^M \tilde{w}_k^i}$

■ **Resampling** $(k = 1 : N)$:

1. Eliminate low approximate weights particles and replicate high approximate weights particles to obtain M random samples with uniform weights creating $\{\hat{\mathbf{p}}_k^m, w_k^m\}_{m=1}^M \rightarrow \left\{\bar{\mathbf{p}}_k^m, \frac{1}{M}\right\}_{m=1}^M$ (Section 5.4.2).

▷ **Outputs:** $\mathbf{x}_k^{1:M} = [\bar{\mathbf{p}}_k^{1:M}, (\mathbf{v}_t^{1:M})^T, (\boldsymbol{\omega}_t^{1:M})^T]^T$

5.5.2 Particle Swarm Optimization (PSO)

In this method, each pose sample $\bar{\mathbf{p}}_k^m = [(\bar{\mathbf{u}}_k^m)^T, (\bar{\mathbf{q}}_k^m)^T]^T$ is updated, taking into account its history and its neighbors. Since we are performing optimization at the same time instant, the velocity component of the state is not affected $[\mathbf{v}_t^T, \boldsymbol{\omega}_t^T]^T$ (3.5), and the PSO does not use it. After initializing the particle swarm with its respective pose (Section 4.3), each particle stores its best position (the highest obtained similarity metric) and the best position of its neighborhood found so far [Nedjah & de Macedo Mourelle, 2006; Zhang *et al.* , 2015]. The best neighborhood search topology influences the obtained results, and it depends on the problem at hand and must be analyzed individually. The adopted search topology is given by:

$$d^{ni} = \frac{\delta^{ni}}{180} \times \varsigma + \upsilon^{ni} \tag{5.19}$$

where ς is a relative sensibility term used to fine-tune the similarity metric, δ^{ni} is the obtained rotation error and υ^{ni} is the obtained translation error between particles n and i. δ^{ni} is given by:

$$\delta^{ni}(R_n, R_m) = \sqrt{\frac{\| \, log_m \, (R_n^T R_m) \, \|_F^2}{2}} \frac{180}{\pi} \quad [deg] \tag{5.20}$$

where R_n is the rotation matrix of the particle n and R_m is the rotation matrix of the particle i. The used PSO scheme is described in Algorithm 9.

Algorithm 9 Particle Swarm Optimization (PSO)

▷ **Inputs:** $\mathbf{x}_t^{1:M} = [(\mathbf{u}_t^m)^T, (\mathbf{q}_t^m)^T, (\mathbf{v}_t^{1:M})^T, (\boldsymbol{\omega}_t^{1:M})^T]^T$ (3.5) , N

▷ **Initialization**:

1. Initialize the particle set $\bar{\mathbf{p}}_0^m = [(\bar{\mathbf{u}}_0^m)^T, (\bar{\mathbf{q}}_0^m)^T]^T = [(\mathbf{u}_t^m)^T, (\mathbf{q}_t^m)^T]^T$ (3.5) and the weights w_0^m with $\frac{1}{M}$ creating $\{\bar{\mathbf{p}}_0^m, w_0^m\}_{m=1}^M$ (Section 4.3). The PSO velocity $\boldsymbol{\vartheta}_0^m$ vector is initialized with zero.

■ **Update each particle best position** $(k = 1 : N)$:

$\bar{\mathbf{p}}_{best}^m = f_{best}(\bar{\mathbf{p}}_{k-1}^m, w_k^m)$

■ **Update each particle best neighborhood pose** $(k = 1 : N)$:

$\bar{\mathbf{p}}_{neigh}^m = f_{neigh}(\bar{\mathbf{p}}_{k-1}^m, w_k^m)$

■ **Obtain each particle velocity** $(k = 1 : N)$:

1. The velocity $\boldsymbol{\vartheta}_k^m$ is obtained according to [Kwolek, 2013; Saini *et al.* , 2014; Zhang *et al.* , 2015]:

$$\boldsymbol{\vartheta}_k^m = \varpi \, \boldsymbol{\vartheta}_{k-1}^m + \underbrace{c_1 r_1 [\bar{\mathbf{p}}_{best}^m - \bar{\mathbf{p}}_{k-1}^m]}_{\text{Social component}} + \underbrace{c_2 r_2 [\bar{\mathbf{p}}_{neigh}^m - \bar{\mathbf{p}}_{k-1}^m]}_{\text{Cognitive component}}$$

where ϖ is the inertia weight that tunes the influence of the iteration $k-1$ velocity $\boldsymbol{\vartheta}_{k-1}^m$ [Shi & Eberhart, 1998], $\{c_1, c_2\}$ are constants that tune the balance between the cognitive and social component, $\{r_1, r_2\}$ are random variables between zero and one.

■ **Update each particle pose** $(k = 1 : N)$:

$\bar{\mathbf{p}}_k^m = \bar{\mathbf{p}}_{k-1}^m + \boldsymbol{\vartheta}_k^m$

▷ **Outputs:** $\mathbf{x}_k^{1:M} = [\bar{\mathbf{p}}_k^{1:M}, (\mathbf{v}_t^{1:M})^T, (\boldsymbol{\omega}_t^{1:M})^T]^T$

5.5.3 Modified PSO

We have also tested the existing variation of the PSO algorithm [Clerc & Kennedy, 2002; Khan & Nystrom, 2010] (Section 5.5.2) with $\boldsymbol{\vartheta}_k^m$ given by:

$$\boldsymbol{\vartheta}_k^m = \Gamma(\boldsymbol{\vartheta}_{k-1}^m + \underbrace{c_1 r_1 [\mathbf{x}_{best}^m - \mathbf{x}_{k-1}^m]}_{\text{Social component}} + \underbrace{c_2 r_2 [\mathbf{x}_{neigh}^m - \mathbf{x}_{k-1}^m]}_{\text{Cognitive component}}) \tag{5.21}$$

where Γ is a constriction coefficient. If the signal between $[\mathbf{x}_{best}^m - \mathbf{x}_{k-1}^m]$ and $[\mathbf{x}_{neigh}^m - \mathbf{x}_{k-1}^m]$ is the same in all dimensions, then $\{r_1, r_2\}$ are considered Gaussian random variables between zero and one, otherwise $\{r_1, r_2\}$ are considered uniform random variables between zero and one. The use of a Gaussian random distribution has the purpose of adding particle diversity to the solution since it promotes *jumps* in the particles. The constriction coefficient aims to control the velocity of the particle and ensure convergence. This coefficient is given by [Khan & Nystrom, 2010]:

$$\Gamma = \frac{2k}{|\, 2 - \varphi - \sqrt{\varphi(\varphi - 4)} \,|} \tag{5.22}$$

where $k \in [0, 1]$ and $\varphi = c_1 + c_2$. Small k values result in a fast convergence with local exploration and higher values lead to slow convergence but global exploitation. The value of one is enough for most applications [Clerc & Kennedy, 2002].

5.5.4 Genetic Algorithm based Framework (GAbF)

In this section, will be described an original approach based on the evolution strategies present on the genetic algorithms [Boli *et al.* , 2004; Kwok *et al.* , 2005; Cagnoni *et al.* , 2007; Park *et al.* , 2007, 2009], to avoid sample impoverishment [Simon, 2006; Li *et al.* , 2012]. Instead of artificial noise, *crossover* and *mutation* operators are adopted to perform the exploration of the pose space [Carmi *et al.* , 2009; Goldberg & Deb, 1991; Kwok *et al.* , 2005; Park *et al.* , 2009; Uosaki *et al.* , 2003; Xiaowei *et al.* , 2013]. The GAbF operates in three phases (Figure 5.10): (i) *bootstrap*, (ii) *coarse optimization*, and (iii) *fine optimization*, as described in Algorithm 10.

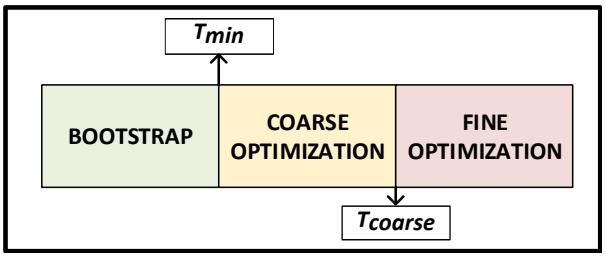

Figure 5.10: GAbF phases.

In the *bootstrap* phase, the best B pose samples coming from the approximate weighting and resampling stage (Figure 5.1) are collected in a list (Top B). The likelihood of each pose $\mathbf{p}_k^m$ is evaluated and stored in the list. The best A pose samples are stored in an auxiliary buffer (Top A). The pose samples with weight w very close to zero (below $\delta = 0.1$) are eliminated and replaced with a random particle selected from the Top A buffer, added to Gaussian noise of covariance $\mathbf{\Sigma}_{init}$. At this point, all pose samples have a likelihood above δ. Then, we perform up to ten improvement steps. In each step, all pose samples are evaluated and compared to those in the Top A (Figure 5.11) if the obtained weight is higher, the Top A is updated. If there are at least two pose samples in the Top A with likelihood bigger than Threshold min T_{min}, the *bootstrap* phase ends. Otherwise, each particle is perturbed with Gaussian noise with covariance $\mathbf{\Sigma}_{bootstrap}$. If after ten of these improvement steps, we do not have two pose samples above T_{min}, the *bootstrap* process is restarted up to a maximum of three restarts. In our experiments (choosing a proper T_{min}), we have noticed that the occurrence of restarts is infrequent. At any stage of the *coarse* and *fine optimization* steps, the best two pose samples have a significant role in the optimization process because they will provide the *chromosomes* for an approach inspired by genetic algorithms.

The *coarse optimization* phase begins when at least two pose samples have a weight higher than T_{min}. Each particle in the Top A list coming from the *bootstrap* process is analyzed. If the particle is the best one, it is perturbed with some Gaussian noise ($\mathbf{\Sigma}_{best_{coarse}}$). If the particle weight is smaller than the best two pose samples are combined using a *crossover* operation to create a new particle. The *crossover* operation consists of the random selection of attributes (X, Y, Z, α, β, γ in our case) of the original pose samples. To half of the pose samples generated by *crossover* is applied a soft *mutation* by adding Gaussian noise $\mathbf{\Sigma}_{coarse}$ to the result. Together these rules allow a focused particle diversity, simultaneously converging to the best solution and avoiding possible local maxima. The process stops when at least two of the pose samples are above the value Threshold T_{coarse}. If this does not happen in ten iterations,

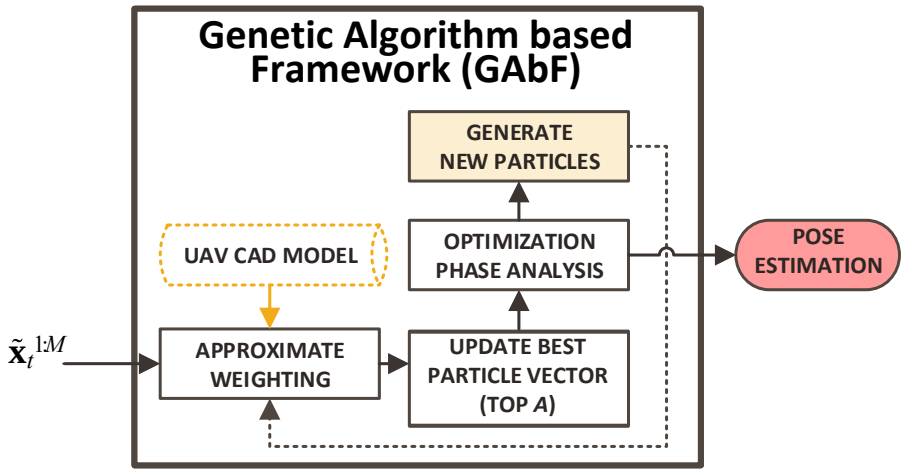

Figure 5.11: GAbF scheme.

the pose optimization filter returns to the *bootstrap* phase automatically.

The *fine optimization* phase is analogous to the coarse phase, but the Gaussian noise covariance applied $\mathbf{\Sigma}_{fine}$ in *mutation* is lower, to make a fine-tuning to the estimated pose. The *fine optimization* phase ends after five iterations. If during this process, the two best pose samples in the Top A became lower than T_{coarse} but higher than T_{min} the *coarse optimization* phase is restarted. In an extreme situation where the two best pose samples in the Top A became lower than T_{min} the *bootstrap* phase is restarted.

Algorithm 10 Genetic Algorithm based Framework (GAbF) - *Pseudocode*

▷ **Input:** Top B, w, $\mathbf{\Sigma}_B$, $\mathbf{\Sigma}_{init}$, $\mathbf{\Sigma}_{bootstrap}$, $\mathbf{\Sigma}_{coarse}$, $\mathbf{\Sigma}_{fine}$, T_{min}, T_{coarse}
▷ **Output:** Top A, Top B, w

loop
  w = Compute likelihood(Top B)
  (Top A, Phase) = Optimization phase(T_{min}, T_{coarse}, $\mathbf{w}$, step)
  if Phase = *bootstrap* and step = 0 **then**
    for All pose samples i in Top B **do**
      if $w(i) < \delta$ **then**
        Top $M(i)$ = Random(Top A) + noise($\mathbf{\Sigma}_{init}$)
      end if
    end for
  end if
  if Phase = *bootstrap* and step > 0 **then**
    Top B = Top B + noise($\mathbf{\Sigma}_{bootstrap}$)
  end if
  if Phase = *coarse* **then**
    Best Particle = Best Particle(Top B) + noise($\mathbf{\Sigma}_{best_{coarse}}$)
    Other pose samples = Crossover and Mutation(Top B, $\mathbf{\Sigma}_{coarse}$)
  end if
  if Phase = *fine* **then**
    Best Particle = Best Particle(Top B) + noise($\mathbf{\Sigma}_{best_{fine}}$)
    Other pose samples = Crossover and Mutation(Top B, $\mathbf{\Sigma}_{fine}$)
  end if
  step = step + 1
end loop

Chapter 6

Experimental results

If a tree falls in the forest and no one is there to hear it, does it make a sound?

Bishop George Berkeley

Chapter contents

This chapter shows the most relevant results obtained during the development of this work. All the developments were made on a 3.70 GHz Intel i7-8700K Central Processing Unit (CPU) using an NVIDIA Quadro P5000 GPU with bandwidth 288.5 GB/s and pixel rate 110.9 GPixel/s.

6.1 Introduction

To describe the overall system performance, we have analyzed landing sequences taking in three types of simulation environments described in Section 6.2: (i) normal, (ii) complex, and (iii) real background. The use of a simulated environment is justified by the need to generate ground truth sequences of UAV motions (six Degrees Of Freedom (DOF)), which would be hard to obtain in real imagery. The use of simulations with different complexities is essential to assess the robustness of the overall system and its individual components to a diversity of possible conditions. Although we do not have ground truth data from landing sequences, we have labeled real data in the target detection performance analysis (Section 6.3). We have also used real video sequences in a final system qualitative validation (Section 6.7.5). Some examples of the performed experimental tests, including the used experimental setup, can be seen in Figure 6.1 and Figure 6.2.

Figure 6.1: Performed experimental tests I (*some examples*).

In this chapter, we will describe the system modeling and simulation (Section 6.2), the target detection system evaluation (Section 6.3), the pose boosting performance (Section 6.4), the comparison of the developed similarity metrics (Section 6.5), evaluate the pose optimization (Section 6.6), the final system tracking performance (Section 6.7), and analyze the GPU implementation (Section 6.8). A summary of the performed experimental tests can be seen in Table 6.1.

Table 6.1: Experimental tests summary.

Tests:	**Objective:**	**Inputs:**	**Outputs:**
Target detection analysis (Section 6.3)	Obtain detector performance	Synthetic training dataset	Precision-Recall curves
		Real test dataset	Processing time
Pose boosting evaluation (Section 6.4)	Obtain hypotheses generation performance	Pre-trained database	Pose error
		Normal background dataset	
Comparison of similarity metrics (Section 6.5)	Obtain the metrics performance	Normal background dataset	Metrics value
		Complex background dataset	Processing time
Pose optimization evaluation (Section 6.6)	Obtain the pose optimization schemes performance	Normal background dataset	Pose error
			Processing time
Complete system analysis (Section 6.7)	Comparison between architecture variants (Section 6.7.1)	Normal background video	Pose error
		Complex background video	
	Particle number vs. Pose optimization (Section 6.7.2)	Complex background video	Effective number of particles
			Pose error
	Real background tracking analysis (Section 6.7.3)	Real background video	Pose error
	Pose boosting contribution analysis (Section 6.7.4)	Real background video	Pose error
	Real captured video quantitative analysis (Section 6.7.5)	Real background video	———
GPU performance analysis (Section 6.8)	Distortion correction evaluation (Section 6.8.2)	Real images dataset	Histogram similarity (error)
			Processing time
	Particle rendering & Model simplification (Section 6.8.3)	UAV CAD model	Rendering error
			Processing time
	GPU-based color similarity metric (Section 6.8.4)	UAV CAD model	Processing time
		Normal background dataset	

Figure 6.2: Performed experimental tests II (*some examples*).

6.2 System Modeling and Simulation

Section contents

To obtain ground truth data in landing sequences, we have developed a "realistic" simulator with three different test environments denoted: (i) normal, (ii) complex, and (iii) real, as illustrated in Figure 6.3. The simulated video environments allow a quantitative analysis of the results. The CAD models and image renders are made with Open Graphics Library (OpenGL) [Woo *et al.* , 1999; OpenGL, 2016]. The synthetic video generation scheme is described in Figure 6.4. The UAV motion is created using the computer keyboard as a joystick and the

rigid body dynamics[1] to perform the simulation of the UAV trajectory [Etkin & Reid, 1996; Valavanis & Vachtsevanos, 2015; Beard & McLain, 2012]. It was not considered the wind influence or any other external force in the developed simulation environment.

Figure 6.3: Video environments: Normal (*left*), complex (*center*) and real (*right*).

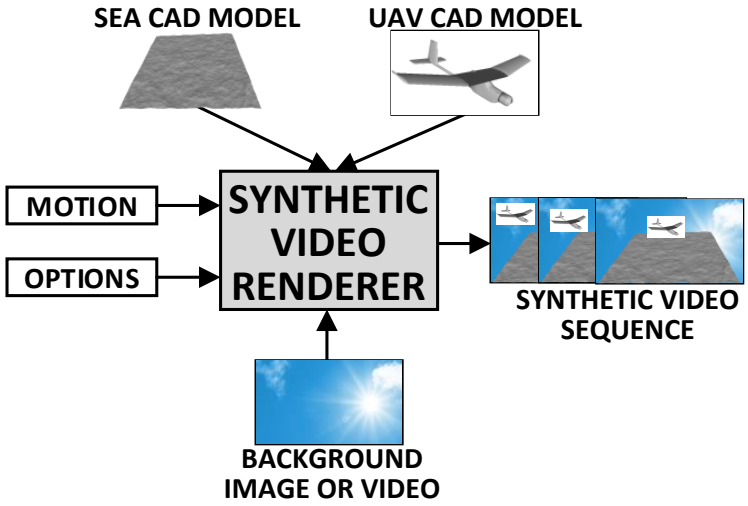

Figure 6.4: Synthetic video generation scheme.

6.2.1 Normal background video sequence

The simulator can be configured with the size of each frame on the video, the applied movement to the background image, the UAV and sea surface motion, image blur and noise (Figure 6.4). The sea surface is simulated using one additional CAD model representing a simple terrain model (Figure 6.4) that moves between frames with random displacement generated by a zero-mean Gaussian distribution. In the normal background video sequence, we use a real sky gradient obtained during experimental tests (Figure 6.5).

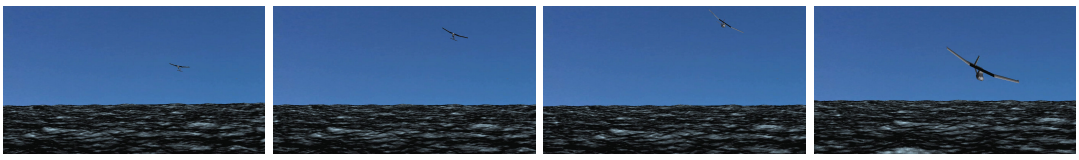

Figure 6.5: Created normal synthetic video sequence (*example*).

[1] Without considering aerodynamic forces.

6.2.2 Complex background video sequence

In the complex background video sequence (Figure 6.6), we apply a translation on the horizontal axis to a background filled with clouds to simulate the sky background motion. When the sky background image reaches the right border, it is mapped to the left border. It is important to note that in the complex sequence, we are analyzing one extreme case where the sky background is filled with clouds and exists a strong gradient variation due to the sunset. It reflects common issues in outdoors imagery[2] (Figure 6.6).

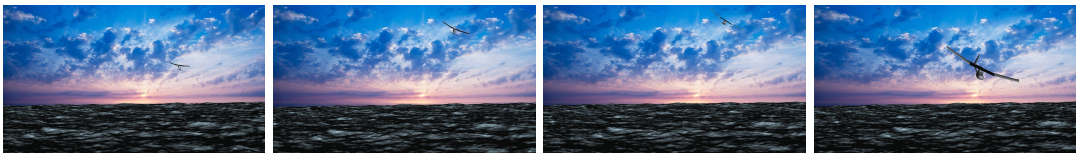

Figure 6.6: Created complex synthetic video sequence (*example*).

6.2.3 Real background video sequence

In the real background video sequence, we use a real captured FPB video and render the UAV CAD model on it to obtain a landing sequence with ground truth information (Figure 6.7). This sequence presents less clutter and noise in the background when compared with the complex sequence (Figure 6.6), but brings the advantage of using real data captured during field tests.

Figure 6.7: Created real synthetic video sequence (*example*).

6.3 Target detection evaluation

Section contents

The used target detection methods are based on supervised learning to train a model from a dataset with both the inputs (images) and the desired results (ROI containing our UAV) [Vaghela *et al.* , 2009]. We have tested YOLO [Redmon & Farhadi, 2018] and SSD [Liu *et al.* , 2016b] (Section 4.2). The output of the network is a vector of bounding boxes containing a confidence value for each one after processing the whole image. We use the GPU both for training and to perform detection.

[2] Non-uniform and textured background, space-variant illumination.

6.3.1 Training and Tests description

The training dataset was generated by projecting the UAV CAD model in various poses into real background in a total of 335769 images (Section 4.2.2). 25% of the entire dataset is used for validation. A test set independent of the training set is composed of 679 real full images ($width \times height = 1920 \times 1080$) containing our UAV, as shown in Figure 6.8.

Figure 6.8: Test samples (*dataset example*).

6.3.2 Performance metrics

Two different performance metrics were used: (i) precision[3], and (ii) recall[4] [Davis & Goadrich, 2006; Flach & Kull, 2015; Howse *et al.* , 2015]. The precision is given by:

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

where TP stands for true positive and FP for false positive. A true positive is one annotation that is also found as a detection, and a false positive is a detection for which no annotation exists. The recall defines how many of the objects in the image are found and is given by:

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

where FN stands for false negative. A false negative is an annotation for which no detection exists. The error evaluation will be made using Precision-Recall Curves (PRCs) [Davis & Goadrich, 2006] that describe the trade-off between precision and recall for different detection threshold values. The ideal method should map to a point in the graph at the upper-right corner where $Precision = Recall = 1$. In this case, all objects in the image are found, and there are no false positive detections. The Area Under the Curve (AUC) measures how good the detector is across all the threshold values, being 100% the ideal value [Boyd *et al.* , 2013; Buckland & Gey, 1994; Keilwagen *et al.* , 2014].

6.3.3 Real dataset results

As described in Figure 6.9, the YOLO obtained the best results on the test dataset with a PRC AUC of about 72%, which is 19% higher than the obtained using SSD. Some examples of detections for SSD can be seen in Figure 6.10 and for YOLO in Figure 6.11. From the analysis of Figure 6.9, we can see that we obtain for YOLO a high precision value with a

[3] How much of the found detections are actual objects.
[4] How many objects that are in the image are found.

few false positives (Figure 6.12) and a little lower recall since, in the vast majority of the test samples, the detections are true positives, but we obtain some false negatives (Figure 6.13). On a 1280×720 image, the YOLO detection takes about 0.034 seconds (≈ 30 Frames Per Second (FPS)) that is compatible with the real-time requirements.

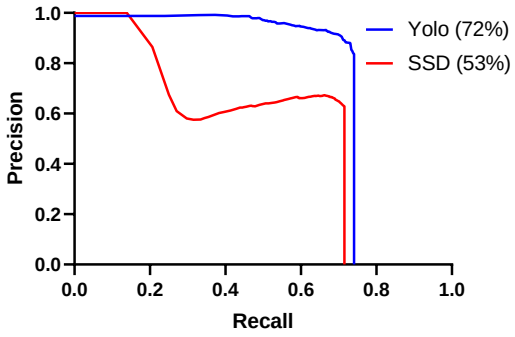

Figure 6.9: Obtained PRCs using YOLO (*blue*) and SSD (*red*).

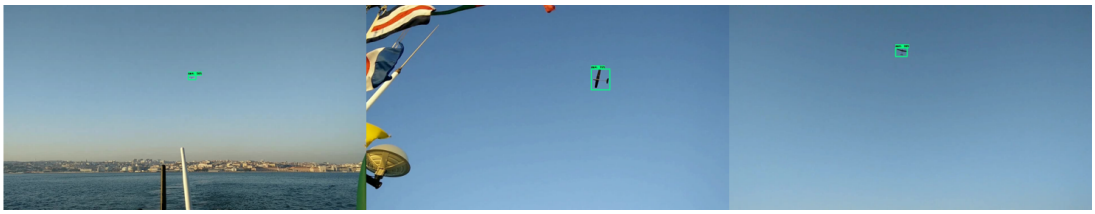

Figure 6.10: Example of UAV detections using SSD (*green rectangles*).

Figure 6.11: Example of UAV detections using YOLO (*pink rectangles*).

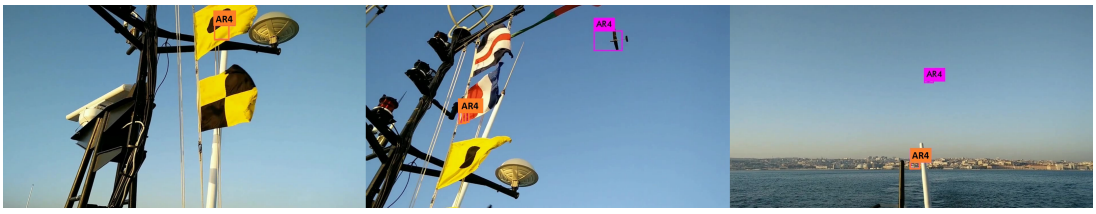

Figure 6.12: Example of false positives using YOLO (*orange rectangles*).

Figure 6.13: Example of false negatives using YOLO (*yellow rectangles*).

6.3.4 Conclusions

We will use YOLO in our approach since it presents the best compromise between accuracy and speed. Some examples of detections using YOLO can be seen in Figure 6.11.

6.4 Pose boosting evaluation

Section contents

A pre-trained database for hypotheses generation (Section 4.3) was created by rendering 10999 sample images of the UAV 3D CAD model at a fixed position ($X = 0$, $Y = 0$ and $Z = 4$), but varying the *Euler* angles[5] α, β and γ according to a uniform distribution for each angle independently with respect to the camera reference frame. In a real scenario, our UAV is cooperative during the landing and is approximating our position, so we focus our database on the front hemisphere $[-90°, 90°]$, as described in Figure 6.14.

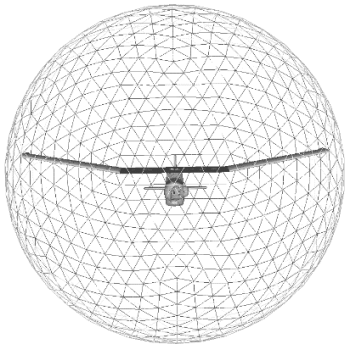

Figure 6.14: Database virtual sphere representation (*nominal pose illustration*).

6.4.1 Tests description

To test the method, we have rendered 15000 random poses for the UAV in the virtual scenario at 5, 15, and 30 meters distance. Then, we retrieve the top 10, 25, 50, and 100 pose samples

[5] *Euler* angle α represents the rotation around X, β represents the rotation around Y, and γ represents the rotation around Z (Figure 3.1).

from the database, using the scheme described in Figure 6.15, and analyze the obtained error on each coordinate independently. We have used a normal background without noise and blur (Section 6.2.1). The idea is to evaluate the ability of our method to generate hypotheses close to the correct solution to feed the tracking PFs, so we evaluate the method quality via the error of the closest generated pose.

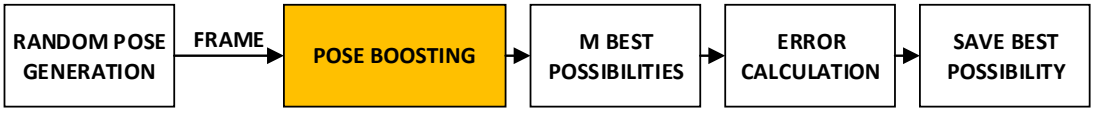

Figure 6.15: Pre-trained database initialization error analysis.

6.4.2 Performance metrics

The translation error between the selected pose and the ground truth pose was obtained using the *Euclidean* distance, and the orientation error was obtained according to (3.3):

$$\delta(R_g, R_r) = \sqrt{\frac{\| \, log_m \left(R_g^T R_r\right) \, \|_F^2}{2}} \frac{180}{\pi} \quad [deg] \tag{6.3}$$

where R_g corresponds to our ground truth rotation matrix and R_r corresponds to the retrieved rotation matrices. The ground truth and the hypothesis with the lower error were selected to evaluate the performance of our system (Figure 6.15). From the saved data, we have obtained the translation error directly from the coordinate difference and the rotation error according to (3.4):

$$\mathbf{q}_e = \mathbf{q}_g \otimes \bar{\mathbf{q}}_r \tag{6.4}$$

where $\otimes$ represents unit quaternion multiplication (the composition of orientations), $\mathbf{q}_g$ corresponds to our ground truth quaternion, and $\bar{\mathbf{q}}_r$ corresponds to the conjugate [Finkelstein *et al.* , 1962; Conway, 1937] of the obtained hypothesis quaternion. Each obtained error quaternion $\mathbf{q}_e$ is then converted to *Euler* angles to analyze each angle independently. It was also obtained the Standard Deviation (SD)[6], the Mean Absolute Error (MAE)[7], and the Root Mean Square Error (RMSE)[8] [Willmott & Matsuura, 2005] of the translation (coordinate difference) and orientation (*Euler* angles) errors.

6.4.3 Normal background

The obtained translation error decreases with the UAV proximity and is dependent on the number of pose samples used from the database. In Table 6.2 for the 5 meters case, we can see that the X and Y errors are very similar with a MAE of approximately 0.06 meters and a RMSE of approximately 0.08 meters for all the used database particles combination. The Z error is higher when compared to the X and Y but between $[-1.18, 1.27]$ meters in the

[6] A measure used to quantify the data dispersion.
[7] Measures the average magnitude of the error without considering their direction.
[8] A quadratic metric to obtain the average error, penalizing large variations.

worst case (10 database particles). We obtain a high number of outliers[9], especially in the Z coordinate.

Table 6.2: Translation error (meters) at 5 meters using 10, 25, 50 and 100 database particles.

	X				Y				Z			
	10	**25**	**50**	**100**	**10**	**25**	**50**	**100**	**10**	**25**	**50**	**100**
5% Percentile	-0.14	-0.13	-0.12	-0.12	-0.12	-0.12	-0.12	-0.12	-1.18	-1.22	-1.31	-1.36
25% Percentile	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.23	-0.20	-0.21	-0.22
Median	0.00	0.00	0.00	0.00	-0.01	-0.01	-0.01	-0.01	0.03	0.04	0.04	0.05
75% Percentile	0.05	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.30	0.28	0.27	0.28
95% Percentile	0.14	0.13	0.13	0.13	0.15	0.15	0.15	0.14	1.27	0.97	0.81	0.76
Outlier %	3.65	4.00	3.94	4.09	5.45	5.81	6.30	6.23	13.22	13.11	13.43	13.05
MAE	0.06	0.06	0.06	0.05	0.06	0.06	0.06	0.05	0.46	0.41	0.41	0.42
RMSE	0.08	0.08	0.07	0.07	0.08	0.08	0.07	0.07	0.67	0.61	0.59	0.60
SD	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.19	0.09	0.05	0.03

The obtained rotation error is very similar in all the tested distances and is dependent on the number of pose samples used from the database. In Table 6.3 for the 5 meters case, we can see that we obtain a high error for all angles with a MAE between $[61.81, 73.56]$ degrees and a RMSE between $[94.20, 97.54]$ degrees in all combinations for α and γ. We obtain a better performance in the β angle, with a MAE between $[28.66, 41.91]$ degrees and a RMSE between $[52.52, 61.08]$ degrees in all the tested combinations. We have some improvement in the obtained results when we use more than ten pose samples, obtaining e.g. a decrease in the obtained MAE of about 10% for α and γ and 20% for β when using 25 database pose samples.

Table 6.3: Rotation error (degrees) at 5 meters using 10, 25, 50 and 100 database particles.

	Alpha (α)				**Beta (β)**				**Gamma (γ)**			
	10	**25**	**50**	**100**	**10**	**25**	**50**	**100**	**10**	**25**	**50**	**100**
5% Percentile	-166.70	-170.80	-173.20	-174.80	-114.00	-101.40	-106.90	-105.20	-170.20	-172.20	-174.70	-175.60
25% Percentile	-59.88	-33.39	-20.11	-14.08	-25.21	-14.73	-9.82	-7.09	-54.36	-26.44	-17.65	-12.86
Median	-1.62	0.04	0.23	-0.18	-0.05	0.01	0.01	-0.05	0.01	-0.01	-0.03	-0.06
75% Percentile	53.95	33.71	21.16	13.83	22.88	13.35	9.79	6.74	55.70	29.28	17.89	11.73
95% Percentile	166.30	171.40	173.00	175.00	108.90	102.20	104.10	103.10	168.60	172.40	174.30	175.80
Outlier %	0.00	26.73	34.12	34.95	13.45	21.95	24.95	26.77	0.00	31.73	35.51	35.53
MAE	73.49	67.14	63.98	62.37	41.91	33.75	30.92	28.66	73.56	65.76	63.23	61.81
RMSE	94.85	94.20	95.14	96.56	61.08	54.26	53.63	52.52	97.54	94.84	95.60	96.86
SD	94.83	94.20	95.14	96.56	61.08	54.26	53.63	52.52	97.54	94.85	95.60	96.86

6.4.4 Conclusions

This method presents an overall low translation error, however, the rotation errors are significant. The large error and the high number of outliers obtained in the rotation are due to a representation of orientation with features of the OBB (Section 4.3), that present low discrimination of pose and some ambiguities. Future work should research image features with higher discriminative power. Notwithstanding, the presented method is fast and, as shown later, provides a good enough diversity of poses both for the initialization of the particle filters and for

[9] Values higher than 1.5 times the *Interquartile* range. The *Interquartile* range represents 50% of data, being the difference between the 75% and 25% percentiles.

the measurement of the unscented particle filters. Depending on our expected orientation, we could still improve results by using a particular database trained in a specific hemisphere area (Figure 6.14).

6.5 Comparison of similarity metrics

Section contents

The ideal similarity metric should have a global maximum at the correct pose and slowly decay for increasingly distinct poses (Section 5.4.1). This behavior would result in a large region of convergence[10].

6.5.1 Tests description

We have tested the color similarity metric (5.14) (color), the contour similarity metric (5.15) with a search distance of 15 and 25 pixels with $\lambda_v = \lambda_e = 1$ (Contour15 and Contour25), and the DT similarity metric (5.16) with σ equal to 25, 50 and 100 (DT25, DT50 and DT100). We have applied the developed similarity metrics (Section 5.4.1) in the normal and complex environments, as shown in Figure 6.3. We have performed six tests for each environment (Table 6.4), rendering the UAV at a nominal pose (Figure 6.14) and varying each variable (orientation and translation) independently.

Table 6.4: Pose evaluation tests description.

Number:	**Initial pose** $(X, Y, Z, \alpha, \beta, \gamma)$**:**	**Test:**	**Interval:**	**Sampling rate:**
1	(0,0,5,0,0,0)	Rotation around Z (angle γ)	$[-180, \ldots, 180]$	1 degree
2		Rotation around X (angle α)		
3		Rotation around Y (angle β)		
4		Translation around Z	$[3, \ldots, 7]$	0.01 meters
5		Translation around Y	$[-0.24, \ldots, 0.24]$	
6		Translation around X	$[-0.74, \ldots, 0.74]$	

6.5.2 Performance metrics

To decide which one to choose, we will evaluate the similarity metrics taking into account the possible existence of global maximum, local maxima, and analyze their behavior (sensibility and noise) when we vary (translation and orientation) the rendered possibility.

6.5.3 Normal background

The first tests were made, taking into account a normal background (Figure 6.5). After analyzing the variations of the angular dimensions in Figure 6.16, it is possible to see that all the

[10] The zone where the metric is monotonous for each side of the maximum.

adopted similarity metrics have peaks around the 180 degrees error. These peaks can lead to local maxima since we obtain a high value of distance due to the model symmetry (Figure 6.17). The contour similarity metric presents a very sensitive and noisy behavior, being almost impossible to use in the study scenario. The variation, along with translation for the remaining metrics (Figure 6.18), shows a good X and Y discrimination and better discrimination when the Z difference is lower than zero $Z < 5$.

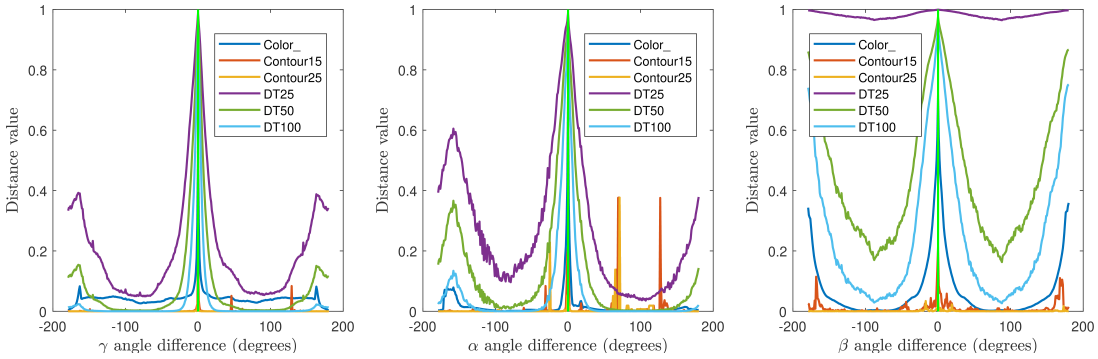

Figure 6.16: Test 1 - Rotation around Z (*left*), test 2 - Rotation around X (*center*), and test 3 - Rotation around Y (*right*) with normal background.

Figure 6.17: An example of 180 degrees variation on Z (*left*) and 180 degrees variation on Y (*right*) where the black color corresponds to the pose overlap.

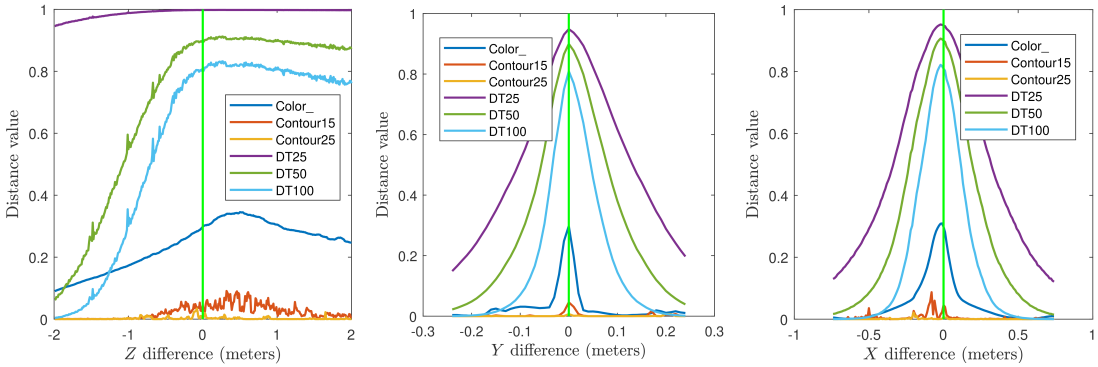

Figure 6.18: Test 4 - Translation around Z (*left*), test 5 - Translation around Y (*center*), and test 6 - Translation around X (*right*) with normal background.

6.5.4 Complex background

As expected, in a complex background (Figure 6.6), we get worse performances (Figure 6.19 and Figure 6.20). Since it is based on the pixel difference between two different areas, the color similarity metric is less affected by the background clutter than the metrics based on edges.

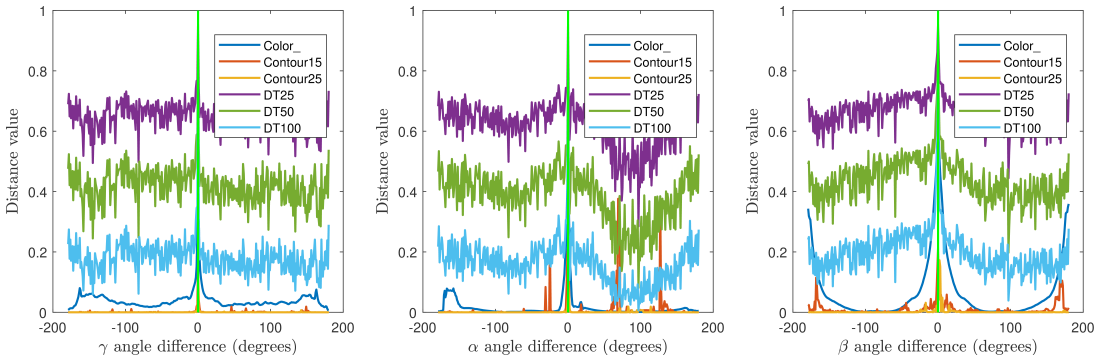

Figure 6.19: Test 1 - Rotation around Z (*left*), test 2 - Rotation around X (*center*), and test 3 - Rotation around Y (*right*) with complex background.

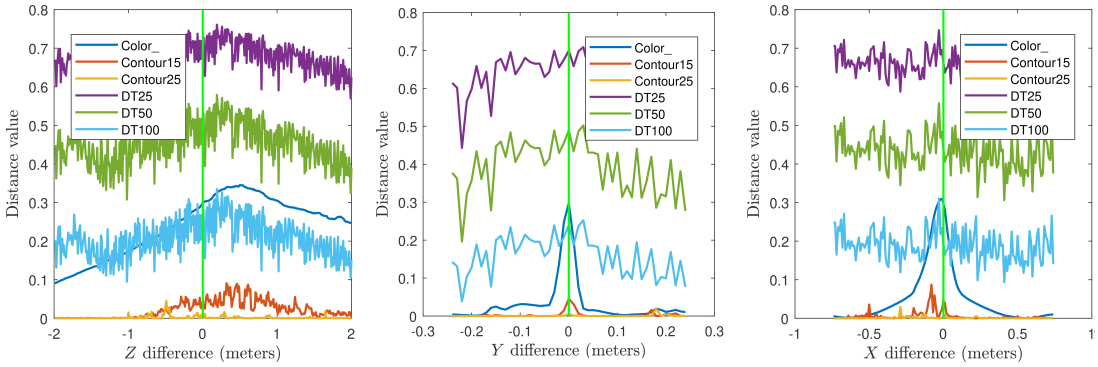

Figure 6.20: Test 4 - Translation around Z (*left*), test 5 - Translation around Y (*center*), and test 6 - Translation around X (*right*) with complex background.

6.5.5 Processing time analysis

The processing time for each similarity metric computation is essential, especially in a PF based framework where each hypothesis (particle) must be evaluated on each iteration at least once. As seen in the analysis of Table 6.5, the color similarity metric is much faster than the DT similarity metric. The extra time needed for the DT similarity metric is due to the extra needed computation to obtain the DT of the captured image and edge map of the pose hypothesis image to compare (Section 5.4.1). It is essential to take into account that if we are evaluating 100 pose samples on each iteration, we will have a frame rate of about two FPS using the color similarity metric. At the moment, the particle rendering is performed using the GPU, but the similarity metric calculation is performed in the CPU. A GPU-based color similarity metric will be evaluated in Section 6.8.4.

Table 6.5: Color and DT similarity metrics average processing time.

	5 meters		50 meters	
	Average time (ms):	**FPS:**	**Average time (ms):**	**FPS:**
Color similarity	4.9	203.3	3.2	309.3
DT similarity	19.2	52.1	19.4	51.6

6.5.6 Conclusions

Overall, the similarity metric that presents the best results in the non-cluttered scenario is the DT, with a slow decay around zero, guaranteeing good discrimination around that pose. However, in cluttered environments, the DT metric is very noisy and multimodal. Instead, the color similarity metric works well in all the tested scenarios, having high decay around the correct pose estimate as the principal disadvantage. If we are analyzing a clear sky, we can use the DT similarity metric (Section 5.4.1.3), but in the case of a high clutter environment, it is better to use the color similarity metric (Section 5.4.1.1).

6.6 Pose optimization evaluation

Section contents

A test set of 10000 synthetic frames[11] was created for the UAV distances 5, 15, 25, 35, and 45 meters (centered on the image using $X = 0$ and $Y = 0$) varying the rotation α, β and γ, according to a uniform distribution restricted in the interval $[-180°, 180°]$ with respect to the camera reference frame[12].

6.6.1 Tests description

The hypotheses generation scheme (Section 4.3), was used for the pose samples initialization. The final pose estimation was obtained from the most likely pose sample after ten pose optimization iterations ($N = 10$) using 100 pose samples ($M = 100$). As described in Section 5.5, we have tested four different pose optimization algorithms: (i) PFO (Section 5.5.1), (ii) PSO (Section 5.5.2), (iii) modified PSO (Section 5.5.3), and (iv) GAbF (Section 5.5.4). When using PFO, we have tested ten traditional resampling schemes, namely: stratified, systematic, residual, residual systematic, optimal, reallocation, metropolis, minimum sampling, multinomial, and branching, as described in Appendix B. The implemented artificial dynamic noise strategies (Section 5.5.1) were:

- **Constant variance noise (Noise)** - Between successive iterations is added noise with a constant variance;
- **Three discrete phases noise (3Phase)** - Each phase has a constant variance (that decreases between phases) and is executed n times;
- **Continuously decreasing variance noise (Iterative)** - The variance decreases after each iteration until it reaches a minimum value.

The added noise was Gaussian with mean $\mathbf{m}^{trans}$ and covariance $\mathbf{\Sigma}^{trans}$ for the translation (Table 6.6) and mean $\mathbf{m}^{rot}$ and covariance $\mathbf{\Sigma}^{rot}$ for the rotation (Table 6.7). In the 3Phase

[11] Using the normal background, as described in Section 6.4.
[12] *Euler* angle α represents the rotation around X, β represents the rotation around Y, and γ represents the rotation around Z (Figure 3.1).

case, we have used $n = 3$, running the third phase one extra time. In the Iterative case, $\mathcal{I}$ is the current iteration number.

Table 6.6: Translation added artificial noise (meters).

Artificial Noise:		**Translation (meters):**	
		Mean $\mathbf{m}^{trans} = [m_X, m_Y, m_Z]$	**Covariance** $\mathbf{\Sigma}^{trans} = diag[\Sigma_X, \Sigma_Y, \Sigma_Z]$
Noise		$[0, 0, 0]$	$diag[0.05, 0.05, 0.01]$
3Phase	**First**		$diag[0.05, 0.05, 0.01]$
	Second		$diag[0.035, 0.035, 0.005]$
	Third		$diag[0.0025, 0.0025, 0.0025]$
Iterative			$diag[0.005, 0.005, 0.005]$

Table 6.7: Rotation added artificial noise (degrees).

Artificial Noise:		**Rotation (degrees):**	
		Mean $\mathbf{m}^{rot} = [m_\alpha, m_\beta, m_\gamma]$	**Covariance** $\mathbf{\Sigma}^{rot} = diag[\Sigma_\alpha, \Sigma_\beta, \Sigma_\gamma]$
Noise		$[0, 0, 0]$	$diag[5, 5, 5]$
3Phase	**First**		$diag[15, 15, 15]$
	Second		$diag[7, 7, 7]$
	Third		$diag[4, 4, 4]$
Iterative			$diag[18 - \mathcal{I}, 18 - \mathcal{I}, 18 - \mathcal{I}]$

When using PSO, the used distance metric between particles is given by (5.19) with $\varsigma = 5$ (Section 5.5.2). In the modified PSO, we obtain the constriction coefficient Γ using (5.22) with $k = 1$ (Section 5.5.3).

When using GAbF (Section 5.5.4), the used parameters are described in Table 6.8.

Table 6.8: GAbF used parameters.

Parameters:	**Value:**
δ	0.1
A	3
B	100
T_{min}	0.2
T_{coarse}	0.3
$\mathbf{\Sigma}^{trans}_{init}$	$diag[0.1, 0.1, 0.1]$
$\mathbf{\Sigma}^{rot}_{init}$	$diag[8, 8, 8]$

Parameters:	**Value:**
$\mathbf{\Sigma}^{trans}_{bootstrap}$	$diag[0.1, 0.1, 0.1]$
$\mathbf{\Sigma}^{rot}_{bootstrap}$	$diag[5, 5, 5]$
$\mathbf{\Sigma}^{trans}_{coarse}$	$diag[0.25, 0.25, 0.25]$
$\mathbf{\Sigma}^{rot}_{coarse}$	$diag[25, 25, 25]$
$\mathbf{\Sigma}^{trans}_{fine}$	$diag[0.15, 0.15, 0.15]$
$\mathbf{\Sigma}^{rot}_{fine}$	$diag[15, 15, 15]$

6.6.2 Performance metrics

The pose samples are evaluated using the color similarity metric (Section 5.4.1.1). The translation error was obtained from the coordinate difference, and the rotation error is obtained according to (6.4). It was also obtained the SD, MAE, and RMSE of the error, as applied in Section 6.4.2.

6.6.3 Normal background

The obtained results are shown in Table 6.9, Table 6.10 and Figure 6.21. In all tested algorithms, the translation error decreases with the UAV proximity. All estimation methods result in a better Z estimate (Table 6.9) when compared to the simple hypotheses generation scheme (Section 6.4). The lowest translation error is obtained using the GAbF algorithm, which presents a MAE below 30 centimeters.

Table 6.9: Pose optimization schemes translation error (meters) at 5 meters.

	PFO			Mod. PSO			GAbF		
	X	Y	Z	X	Y	Z	X	Y	Z
5% Percentile	-0.18	-0.19	-0.47	-0.18	-0.21	-0.67	-0.05	-0.06	-0.25
25% Percentile	-0.06	-0.06	-0.06	-0.04	-0.06	-0.16	-0.01	-0.03	-0.02
Median	0.00	0.00	0.20	0.00	0.00	0.09	0.00	0.01	0.11
75% Percentile	0.06	0.06	0.06	0.05	0.05	0.40	0.01	0.02	0.31
95% Percentile	0.22	-0.19	0.21	0.19	0.19	1.01	0.03	0.07	0.54
Outlier %	8.10	8.90	5.20	10.80	9.35	3.65	5.77	5.77	2.58
MAE	0.08	0.08	0.46	0.08	0.08	0.39	0.03	0.04	0.21
RMSE	0.13	0.12	0.67	0.13	0.13	0.54	0.09	0.06	0.27
SD	0.13	0.12	0.59	0.13	0.13	0.52	0.09	0.05	0.23

Table 6.10: Pose optimization schemes rotation error (degrees) at 5 meters.

	PFO			Mod. PSO			GAbF		
	α	β	γ	α	β	γ	α	β	γ
5% Percentile	-172.70	-128.70	-174.90	-174.50	-153.40	-175.70	-175.20	-151.80	-173.10
25% Percentile	-80.76	-17.10	-93.72	-85.72	-27.25	-106.4	-85.79	-8.58	-106.10
Median	1.70	1.47	-3.11	3.67	0.30	-0.65	6.59	1.31	2.71
75% Percentile	77.26	30.49	72.34	100.70	32.12	76.44	111.40	7.19	120.60
95% Percentile	172.60	158.70	170.80	174.70	161.90	173.30	176.00	112.30	174.10
Outlier %	0.00	24.29	0.00	0.00	26.29	0.00	0.00	30.49	0.00
MAE	106.49	64.53	111.76	110.54	68.83	112.75	106.26	38.95	122.44
RMSE	138.62	103.66	144.37	139.09	103.37	143.94	137.02	77.02	150.75
SD	107.90	75.07	109.40	112.40	84.36	112.00	112.00	58.31	117.60

Concerning the orientation error, we are generating hypotheses in the interval $[-180°, 180°]$, and have used the color similarity metric to approximate the particle weights. As we can see from the analysis of Figure 6.21, the histograms of the generated particle errors have one peak near zero degrees, with occurrences between zero and 180 degrees. When we apply pose optimization, the error begins to be concentrated near zero and 180 degrees. This happens mainly because of the color similarity metric (Section 5.4.1.1) behavior, where complementary poses are similar (Figure 6.17), which results in high data dispersion (Table 6.10). The best results are obtained again with GAbF, mainly due to threshold-based phases where the search space is continuously reduced when we are closer to the solution. The *crossover* operator between the best pose samples in the top A (Section 5.5.4) promotes samples on symmetric configurations near the detected local maxima. If we have several local maxima (ambiguous pose), the possible particle combinations allow us to determine which is the correct pose estimate (global maxima). If we are stuck in local maxima, the *mutation* operator allows us to add diversity to the search space improving the pose estimate.

6.6.4 Processing time analysis

The computational cost for the PSO is very similar to the obtained for the PFO since the most computationally expensive operation is the similarity metric computation that is performed for each particle once on each iteration (Section 6.5.5). The PSO methods have to obtain the best neighborhood particle on each iteration, but that is a fast numeric operation. The

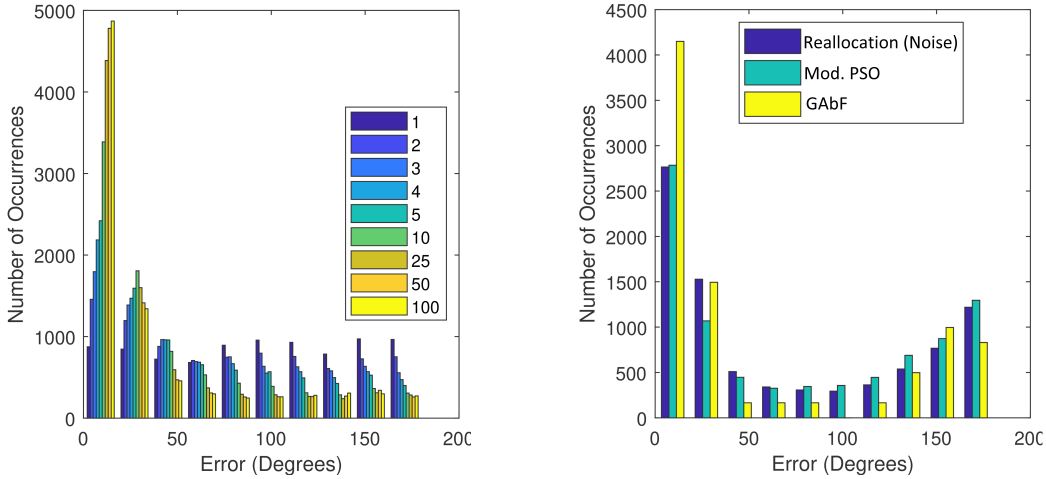

Figure 6.21: Rotation error histogram (degrees): Hypotheses generation scheme when changing the number of used database pose samples (*left*) and Pose optimization (*right*) using the color similarity metric.

computational cost for the GAbF method is higher since it is a three-phase[13] threshold method, and we have to obtain multiple times the particle similarity metric before we reach the third threshold phase and be able to retrieve a pose estimation (Section 5.5.4). Additionally, if the threshold value of the best particles did not increase in ten iterations, the filter returns to the previous phase increasing, even more, the needed processing time. As described in Section 5.5.4, we have noticed that the occurrence of restarts is infrequent.

6.6.5 Conclusions

When we perform local optimization, the results tend to get closer to the local maxima, and this only decreases the obtained error if we are at the global maximum or if we apply an adequate exploitation scheme. The GAbF presents a better performance but has a high processing time when compared with the other tested approaches. We will explore the use of pose optimization more extensively in the complete system analysis, as described in Section 6.7, where we will evaluate landing sequences in different environments (Figure 6.3) to quantify the real contribution of the scheme.

6.7 Complete system analysis

Section contents

In this section, we have analyzed the proposed system structure (Section 3.5) performing the following tests:

[13] *Bootstrap*, *coarse optimization*, and *fine optimization* phases (Figure 5.10).

- The comparison between the proposed proposal architecture variants, to be able to quantify the obtained error and the best method in the study scenario (Section 6.7.1);
- The analysis of the contribution of the pose optimization stage, comparing its results with the obtained when we increase the particle number to be able to quantify the real advantages of its use (Section 6.7.2);
- The implementation of the best-obtained combinations in a real simulated background sequence with ground truth given by the UAV CAD model rendering, to be able to characterize the possible system performance in a real scenario (Section 6.7.3);
- The analysis of the pose boosting stage contribution for the estimate, quantifying the decrease of performance obtained when we are not using it (Section 6.7.4);
- The qualitative analysis of real captured video sequences, being able to infer the possible system performance in the real world (Section 6.7.5).

6.7.1 Comparison between proposal architecture variants

In the created landing sequence (Figure 6.22 and Figure 6.23), the UAV is approaching the ship so that we will use the database trained in the interval $[-90°, 90°]$ (Figure 6.14). To be able to analyze the proposal architecture variants, we will use a constant number of particles.

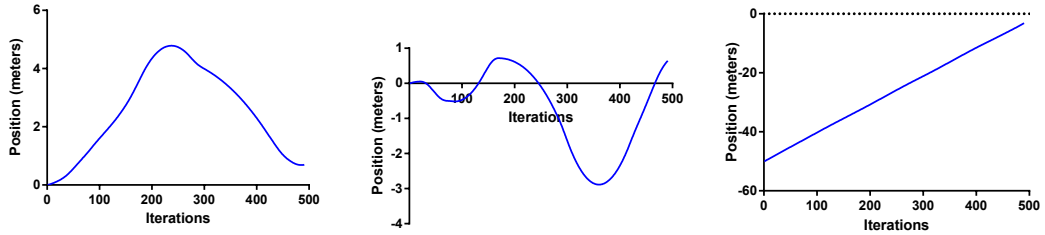

Figure 6.22: Tested landing sequence 1: X (*left*), Y (*center*) and Z (*right*).

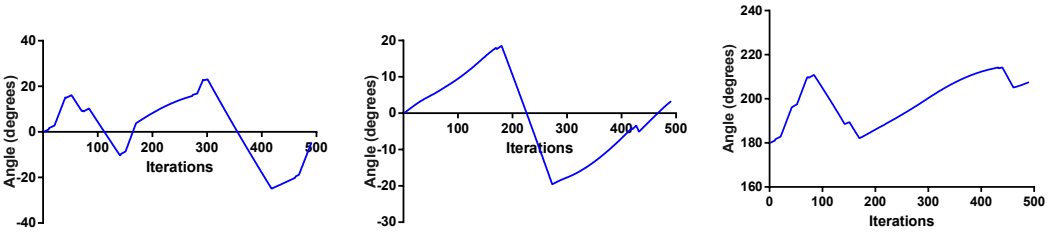

Figure 6.23: Tested landing sequence 1: α (*left*), β (*center*) and γ (*right*).

6.7.1.1 Tests description

We have tested five different filter combinations (Table 6.11), in the previously described normal and complex environments (as seen in Figure 6.3). The difference between the tested combinations is on the proposal step (Figure 3.6). In combination 1, is used only the pose boosting sampling (Section 5.2.1). In combination 2, we apply a mixture of pose boosting sampling and prediction sampling (Section 5.2.2). In combination 3, we apply a UKF for the rotational and translational motion filtering of the particles (Section 5.2.3). In combinations

4 and 5, we apply a UKF for the translational motion filtering and, respectively, a UBiF or UBiGaF for the rotational motion filtering (Section 5.2.4). In the first filter iteration, all the 100 particles (M=100) come from the pose boosting sampling. The velocities are initialized with zero. The particles are evaluated using the color similarity metric (Section 6.5). In combinations 2 to 5 every time we capture a new frame, we update the particle vector with 25 new particles coming from the pose boosting sampling to enrich the proposal distribution and be able to use the most recent observation while maintaining the particle diversity [Okuma *et al.* , 2004; Flury & Shephard, 2011; Kantas *et al.* , 2015; Liu & West, 2001b]. The rest of the particles in the set are obtained from the last iteration using the resampling reallocation (Section 6.6). As described in Section 6.6, the best pose optimization results are obtained using GAbF, but this algorithm presents a high processing time when compared with the other tested methods. The PFO, on the other hand, presents a more straightforward structure, and we can easily choose the number of repetitions N that we want to perform on each filter iteration (Section 5.5.1) presenting a better compromise between speed and accuracy. The PFO artificial noise (Section 5.5.1) is Gaussian of zero mean and covariance 0.1 meters for the translation in X and Y, 0.2 meters for the translation in Z and 2,62 rad/sec for the angular velocity (3.12). The time between iterations is $\triangle t \cong 0.034$ seconds. The temporal filters use a Bi process noise $P_B^{\mathbf{\Phi}}$ with $\mathbf{M}_t^{\mathbf{\Phi}} = \mathbf{I}_{4\times4}$ and $\mathbf{Z}_t^{\mathbf{\Phi}} = diag$(-250,-250,-250,0) and a Bi measurement noise $P_B^{\mathbf{\Lambda}}$ with $\mathbf{M}_t^{\mathbf{\Lambda}} = \mathbf{I}_{4\times4}$ and $\mathbf{Z}_t^{\mathbf{\Lambda}} = diag(-800, -800, -800, 0)$.

Table 6.11: Performed combinations.

					Pose optimization		
Combination	**Particle number (M)**	**Database particles**	**Proposal**	**Approximate weighting**	**Method**	**Resampling**	**Number of repetitions**
1	100	100	Pose boosting	Color similarity metric	PFO	Reallocation	N
2		25	Pose boosting + Prediction				
3			UKF				
4			UKF + UBiF				
5			UKF + UBiGaF				

6.7.1.2 Performance metrics

On each iteration, the state estimation is given by the particle with the highest approximate weight at the end of the filtering pipeline. The translation error was obtained using the *Euclidean* distance, and the rotation error according to (6.3) and (6.4). It was also obtained the SD, MAE, and RMSE of the error, as applied in Section 6.4.2.

6.7.1.3 Normal background

We describe in this section a total of nine tests using the normal background (Figure 6.5). Each of the combinations in Table 6.11 may have a different value of the number of repetitions in the pose optimization steps (N). We denote each test with notation $CxNy$, where x is the number of the combination in Table 6.11, and y is the number of repetitions of pose optimization iterations.

The mixture of pose boosting sampling and prediction sampling ($C2N0$) gives the worst translation estimate, where we have 90% of the error between $[0.54, 15.93]$ meters (Table 6.12). We obtain a slightly higher error when compared with the simple use of the pose boosting sampling ($C1N0$) since the use of a transition prior is not enough to place a reasonable number

of particles near the true UAV pose. When using $N = 0$ ($N0$), the best estimate is given by the use of a single UKF ($C3N0$). When we increase N (1 and 4), the best estimate is given by the use of the UKF + UBiF with $N = 1$ ($C4N1$). The increase of N does not necessarily decrease the obtained error e.g. we have a decrease in the estimated error in $C4N1$, but the error increases in the $C4N4$ case. A large number of iterations in the same frame makes it more likely to converge to a local maxima worsening the result.

Table 6.12: Normal background translation error (meters).

	C1N0	C2N0	C3N0	C4N0	C4N1	C4N4	C5N0	C5N1	C5N4
5% Percentile	0.29	0.54	0.24	0.24	0.10	0.11	0.18	0.14	0.17
25% Percentile	1.81	2.67	0.91	1.14	0.84	0.48	0.79	0.61	0.90
Median	3.91	4.62	1.90	2.52	2.40	1.46	1.60	1.74	2.24
75% Percentile	6.52	7.43	3.51	5.37	3.65	3.44	3.46	3.13	4.95
95% Percentile	12.99	15.93	5.97	11.88	5.34	9.31	6.34	7.34	9.37
Outlier %	3.88	10.00	2.65	5.31	3.67	8.37	4.08	5.51	2.24
MAE	1.76	2.28	0.98	1.44	1.04	1.07	0.89	0.87	1.27
RMSE	1.42	1.82	0.61	0.84	0.16	0.81	0.29	0.52	0.92
SD	4.00	4.79	2.56	3.67	2.84	3.47	2.51	2.72	3.18

The UKF implementation ($C3N0$) gives the worst rotation estimate, where we have 90% of the error between $[3.71, 147.3]$ degrees (Table 6.13). Here we can start seeing the advantages of the use of the UBiF ($C4$) and UBiGaF ($C5$) for the rotational motion filtering where we obtain a better estimate when compared with the other tested methods (Table 6.13). When using $N = 0$ ($N0$), the best estimate is given by the use of the UKF + UBiGaF ($C5N0$). When we increase N, the best estimate is given by the use of the UKF + UBiF with $N = 4$ ($C4N4$). Here is possible to see a distinct improvement in the estimate when increasing N.

Table 6.13: Normal background rotation error (degrees).

	C1N0	C2N0	C3N0	C4N0	C4N1	C4N4	C5N0	C5N1	C5N4
5% Percentile	4.00	4.11	3.71	2.29	1.06	0.64	2.54	1.59	1.17
25% Percentile	7.96	8.89	9.15	5.49	2.23	1.53	5.34	4.94	2.57
Median	15.21	15.29	17.46	8.94	4.11	2.97	9.24	9.98	4.50
75% Percentile	22.06	24.73	35.94	13.79	8.11	7.06	14.20	22.85	8.49
95% Percentile	43.26	51.72	147.30	23.81	25.14	12.04	13.6	44.60	31.71
Outlier %	4.90	6.73	13.88	2.86	9.18	1.84	3.88	4.08	12.65
MAE	8.80	9.34	17.11	4.91	3.29	2.24	5.26	7.41	4.11
RMSE	3.94	2.29	2.56	0.01	0.63	0.04	0.58	2.53	0.82
SD	15.15	14.51	42.51	6.93	7.50	4.81	7.97	15.33	10.91

6.7.1.4 Complex background

We analyze in this section, the same nine tests described in Section 6.7.1.3 using the complex background (Figure 6.6). We also perform, in this section, additional tests to the ones described before to characterize the real contribution of the pose optimization stage in the final result increasing N from zero to nine using the $C4$ and $C5$ combinations. These were the combinations that obtained the best tracking performance, as described in Section 6.7.1.3.

The worst translation estimate is given again by the mixture of pose boosting sampling and prediction sampling ($C2N0$), where we have 90% of the error between $[2.74, 48.40]$ meters (Table 6.14). The inefficiency of the simple prior's use becomes evident when using a complex background. Using $N = 0$ ($N0$), the best estimate is given again by the use of a single UKF ($C3N0$). When we increase N, the best estimate is given by the use of the UKF + UBiGaF with $N = 1$ ($C5N1$). The increase of N again does not necessarily decrease the obtained error e.g. we have an error decrease in the $C4N1$ estimate, but the error increases in the $C4N4$ case.

Table 6.14: Complex background translation error (meters).

	C1N0	C2N0	C3N0	C4N0	C4N1	C4N4	C5N0	C5N1	C5N4
5% Percentile	0.28	2.74	0.35	0.17	0.21	0.14	0.27	0.12	0.30
25% Percentile	1.46	9.82	1.23	0.69	1.09	1.19	1.01	0.80	1.85
Median	3.88	31.96	2.46	2.45	2.52	3.05	3.17	2.45	4.12
75% Percentile	6.31	44.01	4.29	5.34	4.32	4.75	4.86	4.45	7.09
95% Percentile	11.82	48.40	6.20	9.91	7.28	8.95	8.09	7.14	15.70
Outlier %	2.04	0.00	1.02	1.84	13.88	1.43	11.02	7.76	7.96
MAE	1.65	10.73	1.18	1.37	1.29	1.37	1.38	1.16	2.25
RMSE	1.28	10.51	0.80	0.97	0.41	0.19	0.24	0.20	1.71
SD	3.95	16.73	2.63	3.37	3.14	3.36	2.65	2.69	4.70

The mixture of pose boosting sampling and prediction sampling implementation gives the worst rotation estimate ($C2N0$), where we have 90% of the error between $[5.34, 178.4]$ degrees (Table 6.15). Here again, we can see the advantages of the use of the UBiF ($C4$) and UBiGaF ($C5$) for the rotational motion filtering where we obtain a better estimate when compared with the other tested methods (Table 6.15). Using $N = 0$ ($N0$), the best estimate is given by the use of the UKF + UBiF ($C4N0$). When we increase N, the best estimate is given by the use of the UKF + UBiF with $N = 4$ ($C4N4$).

Table 6.15: Complex background rotation error (degrees).

	C1N0	C2N0	C3N0	C4N0	C4N1	C4N4	C5N0	C5N1	C5N4
5% Percentile	4.15	5.34	3.38	2.95	1.62	1.05	1.41	1.96	1.47
25% Percentile	8.24	136.60	9.02	6.19	4.67	2.58	4.35	5.02	4.20
Median	15.26	162.70	28.39	11.02	7.85	4.92	7.22	8.88	9.46
75% Percentile	22.45	172.70	118.30	18.14	13.53	8.56	14.30	15.33	14.23
95% Percentile	67.34	178.40	167.80	31.11	19.00	15.09	34.38	33.74	31.36
Outlier %	10.00	23.06	0.00	1.84	1.02	1.43	11.02	7.76	7.96
MAE	10.73	50.60	28.01	6.05	4.53	2.99	5.64	5.83	5.76
RMSE	4.50	6.64	7.17	1.53	0.75	0.30	0.32	1.03	0.58
SD	23.36	65.91	59.33	8.92	5.59	4.26	10.31	9.95	10.80

To better describe the system performance, we have chosen the best combinations ($C4$ and $C5$ as described in Table 6.11) and tested their performance when increasing N between zero and nine to see if there is a distinct improvement that was not stated in the previous analysis. From the analysis of Table 6.16 and Table 6.17, it is possible to see that we will obtain better translation results for $C5$ when using higher values of N. For $N > 4$ in $C4$, we have a clear increase in error, but the same did not happen in $C5$ where the error decreases in some N

combinations.

Table 6.16: Complex background translation error for $C4$ when increasing N (meters).

	C4N0	C4N1	C4N2	C4N3	C4N4	C4N5	C4N6	C4N7	C4N8	C4N9
5% Percentile	0.17	0.21	0.28	0.77	0.14	0.74	0.30	1.25	1.25	1.03
25% Percentile	0.69	1.09	0.94	1.35	1.19	2.46	2.05	2.80	3.27	4.34
Median	2.45	2.52	2.13	2.12	3.05	5.11	3.46	6.55	6.95	6.99
75% Percentile	5.34	4.32	3.81	5.05	4.75	8.21	6.87	13.27	12.27	14.14
95% Percentile	9.91	7.28	9.43	9.38	8.95	13.17	16.27	20.94	20.55	22.26
Outlier %	1.84	13.88	7.14	3.27	1.43	2.65	8.16	0.20	0.82	0.82
MAE	1.37	1.29	1.27	1.45	1.37	2.24	2.11	3.51	3.49	2.89
RMSE	0.41	0.41	0.74	0.80	0.19	1.82	1.65	3.51	3.49	3.89
SD	3.37	3.14	3.44	3.45	3.36	3.99	4.59	6.52	6.17	6.67

Table 6.17: Complex background translation error for $C5$ when increasing N (meters).

	C5N0	C5N1	C5N2	C5N3	C5N4	C5N5	C5N6	C5N7	C5N8	C5N9
5% Percentile	0.20	0.12	0.21	0.35	0.36	1.00	0.41	0.27	0.58	0.51
25% Percentile	1.11	0.80	1.17	1.39	2.06	2.55	2.09	1.38	1.83	3.12
Median	2.42	2.45	2.64	2.39	5.10	5.11	3.78	2.90	3.83	5.18
75% Percentile	4.59	4.45	4.93	4.59	9.39	9.22	7.98	5.03	8.28	8.69
95% Percentile	11.30	7.14	8.33	8.44	16.07	15.65	12.92	9.33	16.01	16.94
Outlier %	11.02	7.76	1.63	2.45	7.96	4.29	1.63	2.86	1.22	5.31
MAE	1.38	1.16	1.35	1.34	2.25	2.02	2.07	1.44	2.26	2.70
RMSE	0.24	0.20	0.17	1.02	1.71	1.44	1.47	1.15	1.70	2.09
SD	3.24	2.69	3.01	2.94	4.78	10.07	4.78	2.87	4.80	4.80

From the analysis of Table 6.18 and Table 6.19, it is possible to see that we will obtain a higher rotation error in $C5$ when using $N < 4$ and the same did not happen in $C4$ where the best estimate is obtained using $C4N2$. In the vast majority of the combinations, the rotation error is higher in $C5$ than in $C4$. The best rotation estimate for $C5$ is obtained using $C5N5$. Analyzing the best-obtained filters ($C4N2$ and $C5N5$) is possible to see that the obtained error is low, mainly between $[-10, 10]$ degrees for all the angles. The obtained translation error for these combinations can be seen in Figure 6.24 and the rotation error in Figure 6.25. An example of the obtained tracking results using $C4N2$ can be seen in Figure 6.26.

Table 6.18: Complex background rotation error for $C4$ when increasing N (degrees).

	C4N0	C4N1	C4N2	C4N3	C4N4	C4N5	C4N6	C4N7	C4N8	C4N9
5% Percentile	2.95	1.53	1.61	0.99	1.05	1.29	1.02	1.26	1.62	1.22
25% Percentile	6.19	4.56	3.48	2.67	2.57	2.72	2.55	2.67	4.74	3.25
Median	11.02	7.81	5.43	4.93	4.91	6.17	5.57	6.01	13.98	6.81
75% Percentile	18.14	13.53	8.53	11.20	8.52	14.98	8.91	16.00	21.69	18.90
95% Percentile	31.11	19.00	14.58	20.17	15.09	27.86	21.06	21.89	27.79	26.76
Outlier %	1.84	1.02	3.88	1.22	1.43	0.82	8.57	0.00	0.00	0.82
MAE	6.05	4.53	3.21	3.69	2.99	4.85	3.66	4.35	6.38	4.37
RMSE	1.53	0.75	0.98	0.30	0.30	0.61	2.51	2.71	0.98	3.87
SD	8.92	5.61	3.99	6.14	4.26	8.57	7.69	7.48	8.90	9.40

Table 6.19: Complex background rotation error for $C5$ when increasing N (degrees).

	C5N0	C5N1	C5N2	C5N3	C5N4	C5N5	C5N6	C5N7	C5N8	C5N9
5% Percentile	2.49	1.96	1.96	1.43	1.36	1.00	1.15	1.21	1.23	1.57
25% Percentile	6.61	5.02	4.33	3.23	3.94	2.56	3.25	2.86	2.41	3.31
Median	13.27	8.88	8.21	6.82	8.72	5.17	6.99	5.60	4.86	9.46
75% Percentile	25.47	15.33	14.47	13.83	13.45	9.22	10.26	16.33	9.90	16.67
95% Percentile	51.28	33.74	48.60	31.66	20.99	15.65	24.55	27.66	18.98	27.57
Outlier %	11.02	7.76	12.65	6.12	7.96	5.10	7.14	0.00	3.88	0.82
MAE	5.64	5.83	6.58	5.19	5.76	3.76	3.85	4.69	3.30	5.18
RMSE	0.32	1.03	0.35	0.16	0.58	1.87	0.22	2.07	1.75	0.39
SD	16.21	9.95	13.76	9.62	6.56	10.07	6.86	8.79	5.69	8.69

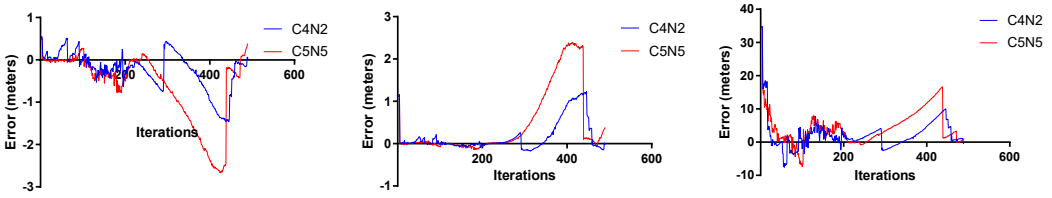

Figure 6.24: Obtained translation error ($C4N2$ and $C5N5$): X (*left*), Y (*center*) and Z (*right*).

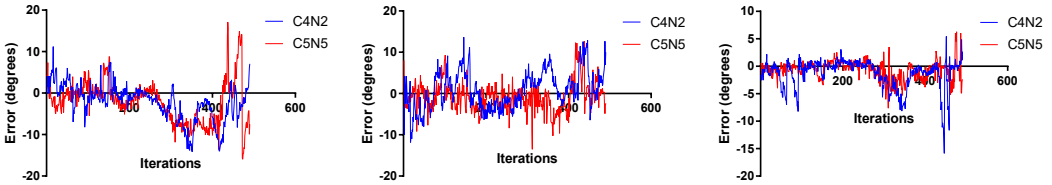

Figure 6.25: Obtained rotation error ($C4N2$ and $C5N5$): α (*left*), β (*center*) and γ (*right*).

6.7.1.5 Conclusions

For simple backgrounds, we recommend the use of the UKF + UBiGaF ($C5N0$) since it presents a good compromise between the obtained errors and the number of needed pose optimization iterations ($N = 0$). Both the filters show excellent performance in the complex background case, presenting the UBiF less variance in the estimate than the UBiGaF that contributes to the lower obtained error, as described before. Because of this, we have to use a higher value of N to obtain similar performance. The tested complex environment is a challenging environment since the background is filled with clutter and is moving. Nevertheless, the filters present overall good performance. The motion filtering is essential to obtain a low error, especially in the rotation case, where we have a clear advantage in its application. This advantage becomes evident when we analyze the simple use of the pose boosting sampling ($C1$) performance, where we obtain a high error even using the PFO, as described in Figure 6.27 and Figure 6.28.

6.7.2 Particle number vs. Pose optimization

It is crucial to quantify the real pose optimization stage contribution to the final result. For these tests, we have tested the $C4$ combination (UKF + UBiF) in the complex environment (Figure 6.6), using the landing sequence described in Figure 6.22 and Figure 6.23.

Figure 6.26: Complex background $C4N2$ tracking sequence (*estimate represented in red*).

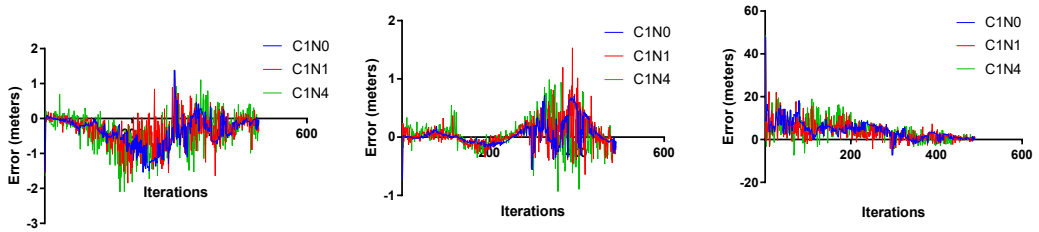

Figure 6.27: Obtained translation error ($C1N0$, $C1N1$ and $C1N4$): X (*left*), Y (*center*) and Z (*right*).

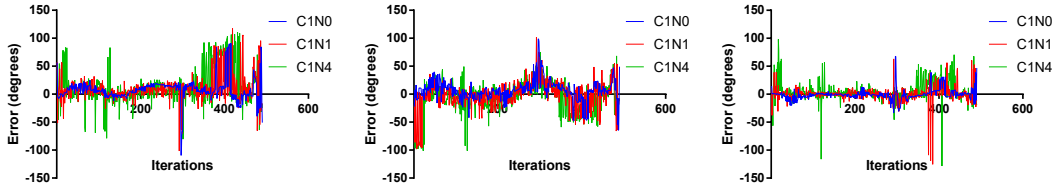

Figure 6.28: Obtained rotation error ($C1N0$, $C1N1$ and $C1N4$): α (*left*), β (*center*) and γ (*right*).

6.7.2.1 Tests description

In Section 6.7.1, we have seen that the combination $C4$ with pose optimization significantly increase the obtained results, but we have to choose the right N[14] value. By increasing N we are spending more computation. The computational cost is dominated by the number of pose renders. The number of pose renders in the pose optimization stage is $M \times N$, thus, in the overall algorithm is $M + M \times N$. Therefore, we compare the performance of an algorithm with N PFO steps with its version without PFO steps but with $M + M \times N$ particles. We describe in this section a total of eight different filter combinations for $C4$, to analyze if the increase of the particles on the set will have similar or even better results when compared with pose optimization. We have used the same number of database particles, similarity metric, resampling strategy, and noise, as described in Section 6.7.1.1.

6.7.2.2 Performance metrics

On each iteration, the state estimation is given by the particle with the highest approximate weight. The translation error was obtained using the *Euclidean* distance, and the rotation error according to (6.3) and (6.4). It was also obtained the SD, MAE, and RMSE of the error, as applied in Section 6.4.2. We have also obtained the effective number of particles $\hat{N}_{eff}$, to estimate how well the particle set approximates the true posterior according to [Stachniss *et al.* , 2005; Liu, 1996; Grisettiyz *et al.* , 2005]:

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{M} (w_t^i)^2} \tag{6.5}$$

where M is the particle number and w refers to the obtained approximated normalized weights (Section 5.4.1). If the value of $\hat{N}_{eff}$ is near its maximum number (M), we will have an excellent

[14] N is the number of repetitions performed in the pose optimization stage.

posterior approximation (3.10).

6.7.2.3 Complex background

From the analysis of Table 6.20, it is possible to see that we always obtain better results when performing local optimization when compared with the increase in the number of particles. From the analysis of Figure 6.29, we can see a clear increase in the obtained $\hat{N}_{eff}$ that becomes closer to its maximum value (M) when we increase N indicating a better true posterior approximation.

Table 6.20: $C4$ rotation error (degrees): Particle number vs. Pose optimization.

Particle number (M)	N	5% Percentile	25% Percentile	Median	75% Percentile	95% Percentile	MAE	RMSE	SD
100	1	1.26	3.52	7.87	15.28	19.95	4.53	0.53	7.17
200	0	1.99	5.35	9.75	15.73	37.79	6.44	2.88	12.80
100	2	1.16	2.92	5.11	9.10	21.98	3.73	0.10	7.57
300	0	1.75	5.27	10.53	15.87	56.99	6.96	3.14	14.02
100	3	0.98	2.20	4.20	7.94	14.41	3.01	0.53	7.31
400	0	2.12	5.18	10.42	16.93	36.40	6.55	2.84	12.15
100	4	1.00	3.10	7.42	11.22	28.15	4.38	0.21	8.44
500	0	1.72	5.12	9.89	17.07	36.85	6.21	2.74	12.11

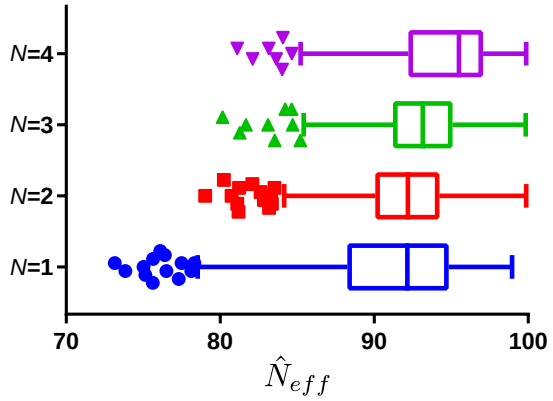

Figure 6.29: $C4$ $\hat{N}_{eff}$ with $M = 100$ (particle number) and $N = \{1, 2, 3, 4\}$.

6.7.2.4 Conclusions

As described in Section 6.7.1, the result did not improve every time we increase N since we can get stuck in local maxima and perform optimization steps that will lead to a wrong estimate. The obtained rotation error for $C4N0$ with $M = 500$ and $C4N4$ with $M = 100$ can be seen in Figure 6.30. We can see the advantage of using the pose optimization stage instead of increasing the particle number, as described in Table 6.20.

6.7.3 Real background tracking analysis

In this section, we have analyzed the proposed system structure (Section 3.5) in a real background sequence with ground truth created by the UAV CAD model rendering (Section 6.2.3). The analyzed landing sequence is represented in Figure 6.31 and Figure 6.32.

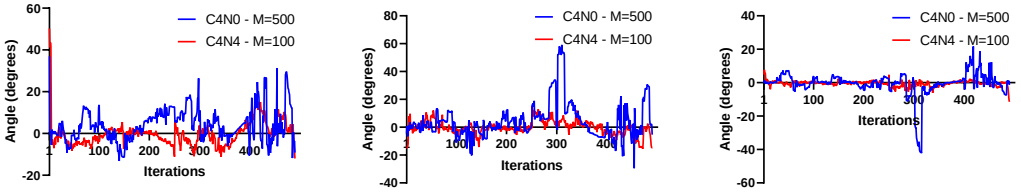

Figure 6.30: Complex background obtained rotation (degrees) error ($C4N0$ with $M = 500$ and $C4N4$ with $M = 100$): α (*left*), β (*center*) and γ (*right*).

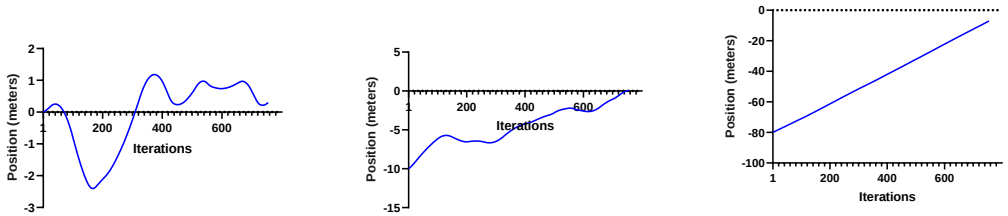

Figure 6.31: Tested landing sequence 2: X (*left*), Y (*center*) and Z (*right*).

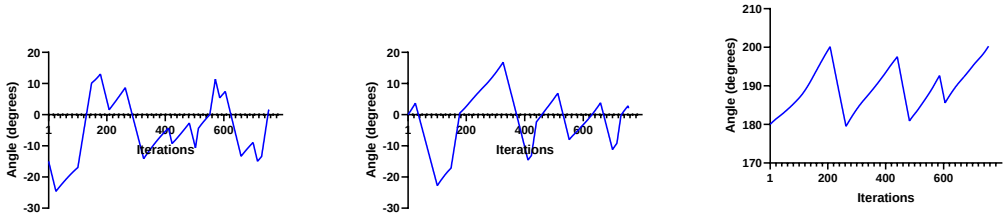

Figure 6.32: Tested landing sequence 2: α (*left*), β (*center*) and γ (*right*).

6.7.3.1 Tests description

The best results for the complex background analysis performed in Section 6.7.1.4 was obtained by the UKF + UBiF with $N = 2$ ($C4N2$) and UKF + UBiGaF with $N = 5$ ($C5N5$) combinations (Table 6.11) using 100 particles ($M = 100$). In this section, we have tested a different landing sequence with more angle variations in a real background sequence also using 100 particles (Section 6.7.1.4). We have used the same number of database particles, similarity metric, resampling and noise strategies, as described in Section 6.7.1.1.

6.7.3.2 Performance metrics

On each iteration, the state expectation is the particle with the highest approximate weight. The translation error was obtained using the *Euclidean* distance, and the rotation error according to (6.3) and (6.4). It was also obtained the SD, MAE, and RMSE of the error, as applied in Section 6.4.2.

6.7.3.3 Results

The best translation estimate is given by the $C5N5$ combination, where we have 90% of the error between $[0.31, 18.01]$ meters (Table 6.21). The best rotation estimate is given by the

$C4N2$ combination, where we have 90% of the error between $[1.28, 22.08]$ degrees (Table 6.22). When we compare the obtained results (Table 6.21 and Table 6.22), we can see that the error difference is very low between the selected combinations, as was also verified in the analysis performed in Section 6.7.1.

Table 6.21: Real sequence translation error for $C4N2$ and $C5N5$ (meters).

	5% Percentile	25% Percentile	Median	75% Percentile	95% Percentile	Outlier %	MAE	RMSE	SD
C4N2	0.19	1.29	4.85	9.00	18.40	4.11	2.29	2.20	5.99
C5N5	0.31	1.58	4.70	8.27	18.01	4.77	2.24	2.19	5.72

Table 6.22: Real sequence rotation error for $C4N2$ and $C5N5$ (degrees).

	5% Percentile	25% Percentile	Median	75% Percentile	95% Percentile	Outlier %	MAE	RMSE	SD
C4N2	1.28	3.50	6.73	11.59	22.08	3.71	3.87	2.10	5.86
C5N5	1.45	3.92	7.74	13.18	23.09	2.25	4.31	2.03	6.90

The analysis performed in Table 6.21 and Table 6.22 is incomplete and does not capture the true behavior of the combinations in the analyzed landing sequence (Figure 6.31 and Figure 6.32). As we can see in Figure 6.33, the translation error has very similar behavior in both methods and decrease with the UAV proximity since we are using a UKF with the same parameters. As we can see in Figure 6.34, the rotation error has a little bit less noise for the $C4N2$ combination but with very similar results decreasing with the UAV proximity. An example of the obtained tracking results using $C4N2$ can be seen in Figure 6.35.

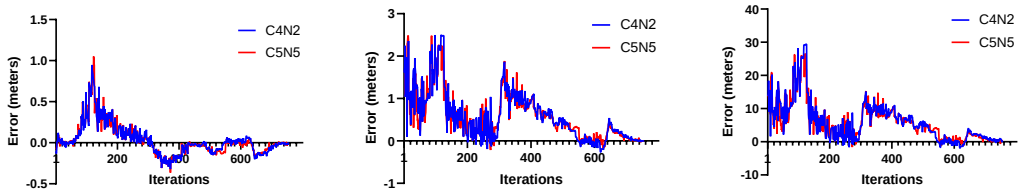

Figure 6.33: Real sequence obtained translation (meters) error ($C4N2$ and $C5N5$): X (*left*), Y (*center*) and Z (*right*).

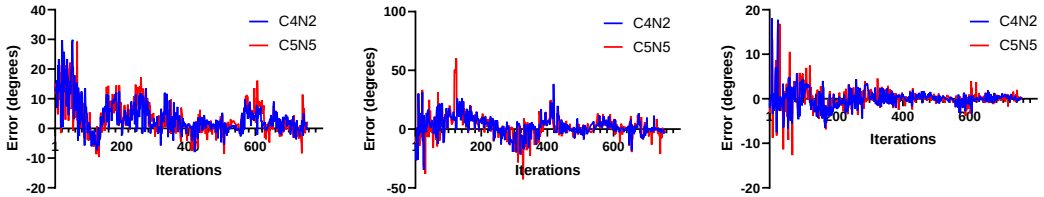

Figure 6.34: Real sequence obtained rotation (degrees) error ($C4N2$ and $C5N5$): α (*left*), β (*center*) and γ (*right*).

6.7.3.4 Conclusions

As described in Section 6.7.3.3, the obtained error between the tested combinations ($C4N2$ and $C5N5$) is very similar and decreases with the UAV proximity to the camera. As described

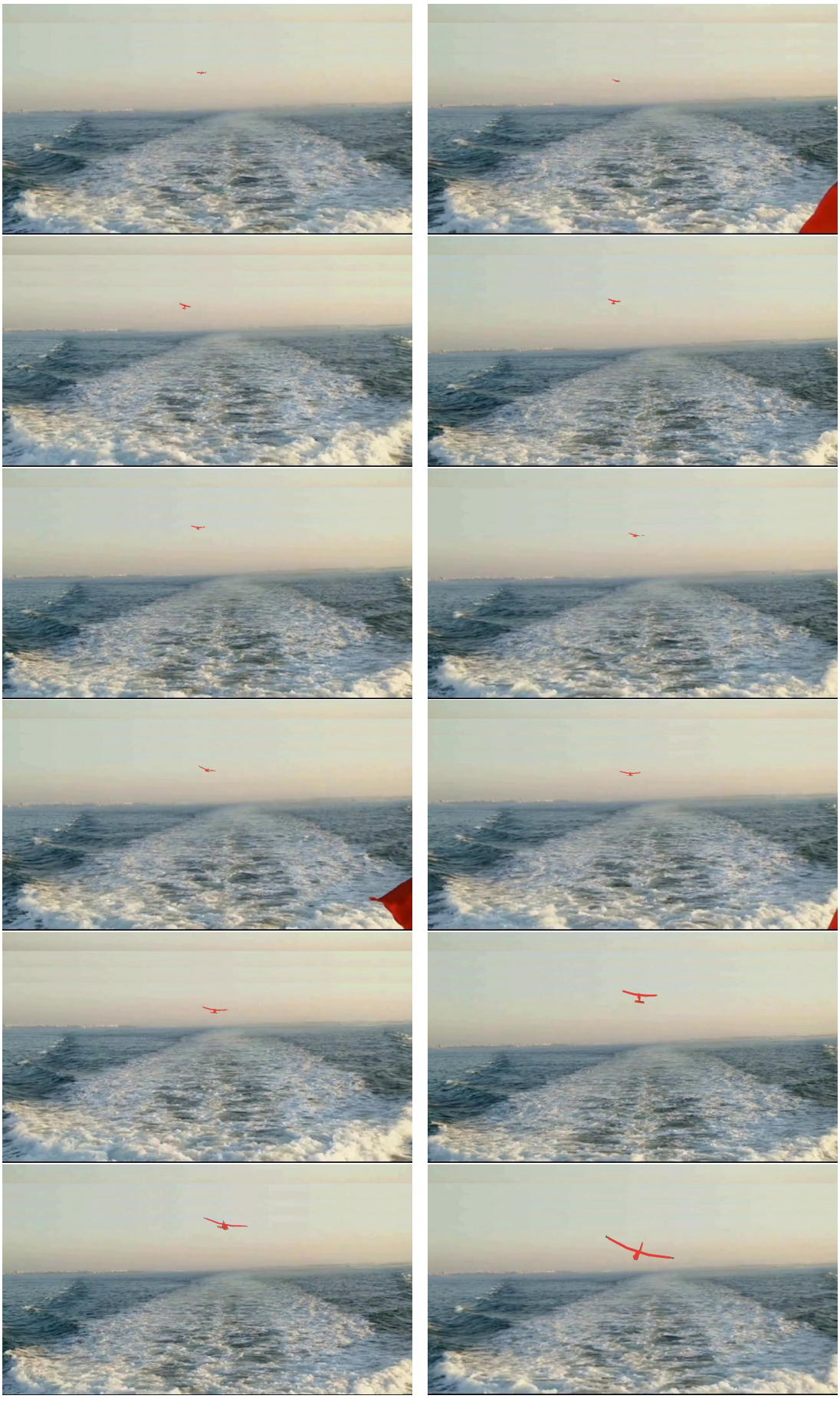

Figure 6.35: Real background $C4N2$ tracking sequence (*estimate represented in red*).

in Figure 6.33 and Figure 6.34, the obtained error is compatible with the automatic landing requirements.

6.7.4 Pose boosting contribution analysis

In this section, we have analyzed the proposed system structure (Section 3.5) in a real background (Figure 6.7) without using the pose boosting stage (Chapter 4). This test has the objective of analyzing the real contribution of this stage in the final result. The analyzed landing sequence is represented in Figure 6.31 and Figure 6.32.

6.7.4.1 Tests description

The best results for the complex background analysis performed in Section 6.7.1.4 was obtained by the UKF + UBiF with $N = 2$ ($C4N2$) and UKF + UBiGaF with $N = 5$ ($C5N5$) combinations (Table 6.11) using 100 particles. The pose boosting stage was only used in the first filter iteration (initialization). We have used the same number of database particles, similarity metric, resampling and noise strategies, as described in Section 6.7.1.1.

6.7.4.2 Performance metrics

On each iteration, the state expectation is the particle with the highest approximate weight. The translation error was obtained using the *Euclidean* distance, and the rotation error according to (6.3) and (6.4).

6.7.4.3 Real background

Both combinations present a very high error in translation and rotation, as described in Table 6.23 and Table 6.24. When we get stuck in a local maxima, the system does not receive information that can add particles close to the current pose of the target. The error in both combinations is very high, as described in Figure 6.36 and Figure 6.37. An example of the obtained tracking results for both combinations can be seen in Figure 6.38 and Figure 6.39.

Table 6.23: Real sequence translation error for $C4N2$ and $C5N5$ (meters).

	5% Percentile	25% Percentile	Median	75% Percentile	95% Percentile	Outlier %	MAE	RMSE	SD
C4N2	17.52	32.09	50.46	69.23	84.16	0.00	19.35	19.08	21.37
C5N5	17.45	31.85	50.10	68.85	83.79	0.00	19.37	19.08	21.28

Table 6.24: Real sequence rotation error for $C4N2$ and $C5N5$ (degrees).

	5% Percentile	25% Percentile	Median	75% Percentile	95% Percentile	Outlier %	MAE	RMSE	SD
C4N2	23.83	33.42	43.14	125.00	148.30	0.00	30.55	9.80	48.89
C5N5	23.41	48.56	72.36	96.50	165.50	4.50	41.06	6.92	48.79

Comparing the results obtained in this section with the ones obtained in Section 6.7.3, we can see that the translation error is about four times higher (Table 6.21 and Table 6.23) and the rotation error is about 6.5 times higher (Table 6.22 and Table 6.24) when not using the pose boosting stage.

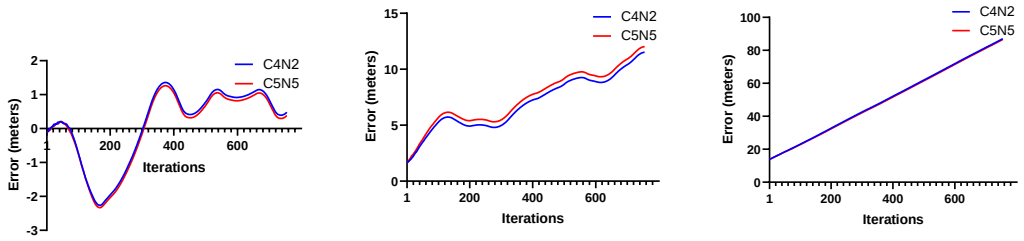

Figure 6.36: Real sequence obtained translation (meters) error ($C4N2$ and $C5N5$) without using the pose boosting stage: X (*left*), Y (*center*) and Z (*right*).

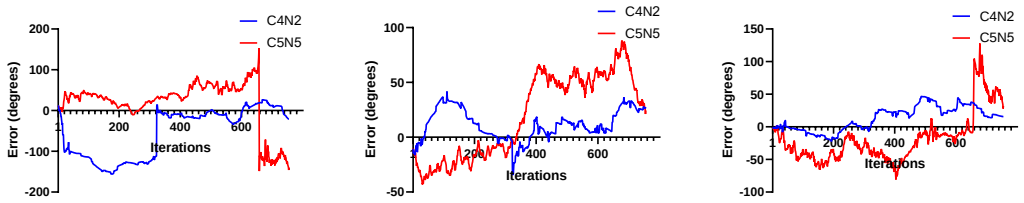

Figure 6.37: Real sequence obtained rotation (degrees) error ($C4N2$ and $C5N5$) without using the pose boosting stage: α (*left*), β (*center*) and γ (*right*).

6.7.4.4 Conclusions

The pose boosting stage is essential in the filtering to be able to add particle diversity and decrease the possibility of being stuck in local maxima. As described in Figure 6.36 and Figure 6.37, the obtained error is too high, not fulfilling the automatic landing requirements.

6.7.5 Real captured video sequence qualitative analysis

We have applied the combination UKF + UBiF with $N = 2$ ($C4N2$) to real UAV sequence videos for a qualitative analysis since we do not have ground-truth information from the captured video sequences. We have obtained outstanding results, as we can see from the qualitative analysis of Figure 6.40 and Figure 6.41.

6.8 GPU performance analysis

Section contents

We need to render the 3D CAD model for each tested possibility (Figure 6.42) to compute the similarity metric (Section 5.4.1). This operation is very time-consuming, and we can use the GPU capabilities to get parallel processing and increase the real-time capability of the system. We will use the Compute Unified Device Architecture (CUDA), that is a parallel computing platform and programming model developed by the NVIDIA company [Sanders & Kandrot, 2010; Wilt, 2013; Cheng *et al.* , 2014].

Figure 6.38: Real background $C4N2$ without pose boosting (*estimate represented in red*).

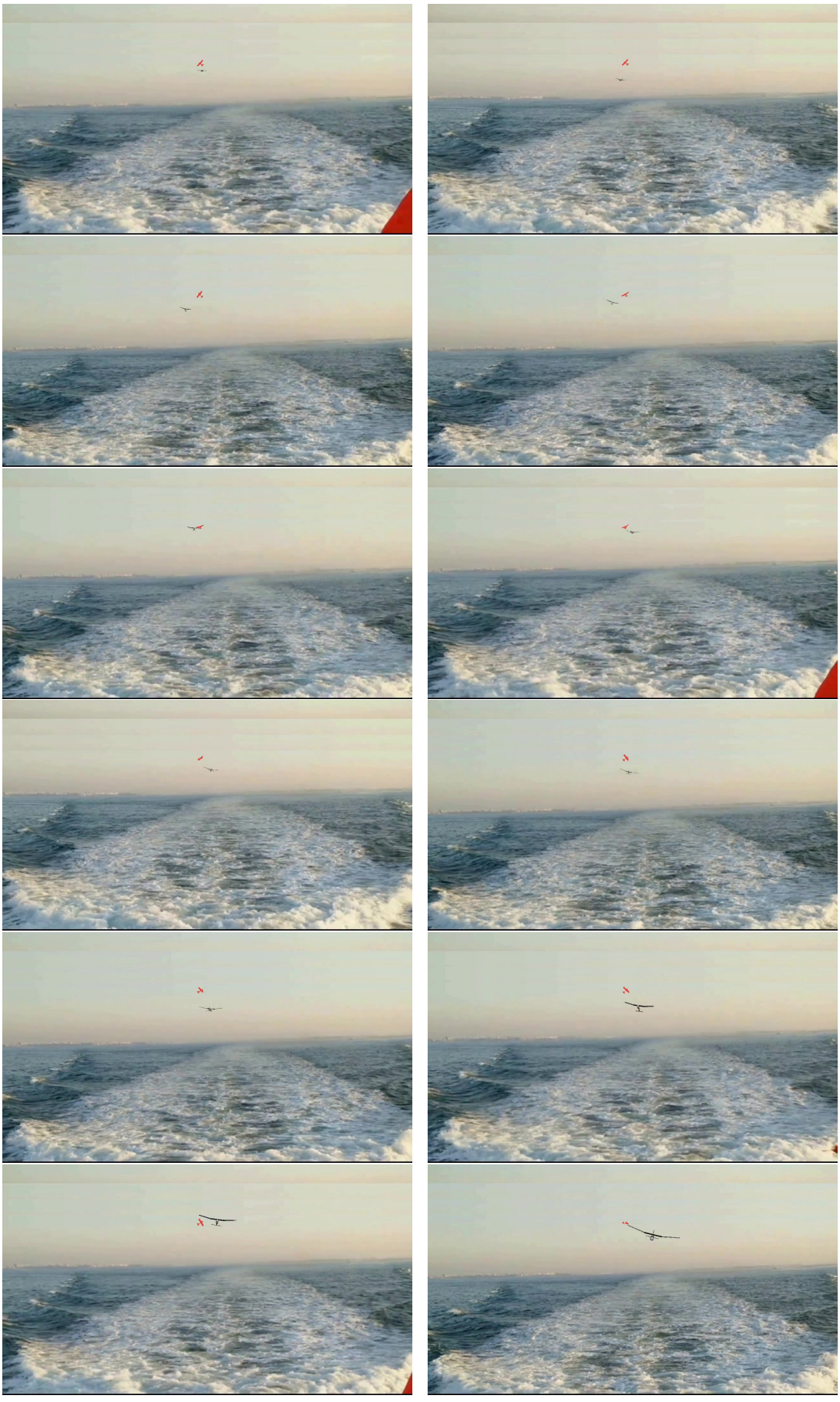

Figure 6.39: Real background $C5N5$ without pose boosting (*estimate represented in red*).

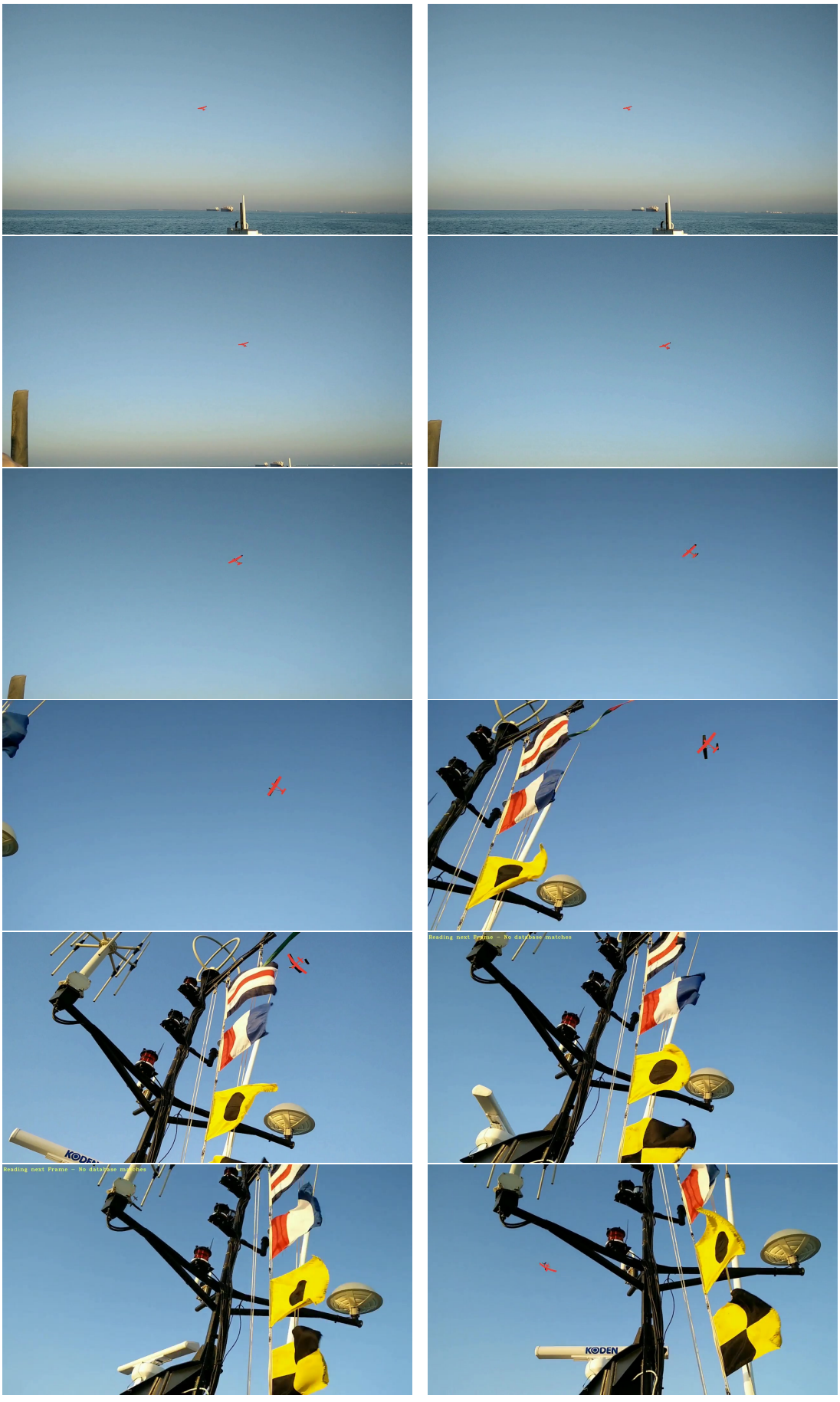

Figure 6.40: *C4N2* tracking sequence I (*estimate represented in red*).

Figure 6.41: $C4N2$ tracking sequence II (*estimate represented in red*).

Figure 6.42: OpenGL rendering example.

6.8.1 Tests description

One of the major bottlenecks of the CPU/GPU interaction is the transfer throughput between them [Farber, 2011; Cook, 2012]. The tested CPU/GPU scheme is described in Figure 6.43, where are illustrated the different stages and data transfer between the CPU and the GPU. We will analyze the obtained error and processing time of the distortion correction[15] (Section 6.8.2), the pose rendering (Section 6.8.3), and a GPU-based color similarity metric (Section 6.8.4).

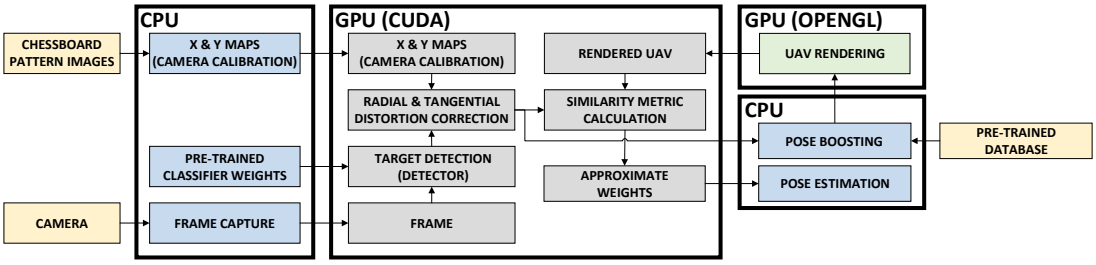

Figure 6.43: CPU/GPU tested scheme.

6.8.2 Distortion correction evaluation

The captured frame is sent to the GPU once per iteration and takes about 0.24 ms. The distortion correction is made using a preloaded remap matrix. This map is obtained using the camera calibration parameters (Section 3.1.7) and takes approximately 115 ms to calculate using the CPU. The transfer between the CPU and the GPU takes about 0.94 ms. Two different remap interpolation methods were applied: (i) linear, and (ii) bilinear [Jain *et al.* , 1995; Szeliski, 2010]. Using 1280x720 pixel images, we obtain the correspondent GPU and CPU corrected frames. They were compared, obtaining the *Bhattacharyya* similarity metric between histograms [Gómez-Luna *et al.* , 2013] and registering the execution time. The remap in the GPU is about 38 times faster than in the CPU for the linear and bilinear interpolation, without losing the accuracy of the result, as described in Table 6.25.

[15] The distortion correction can also be compensated directly in the particle rendering using OpenGL (Figure 6.44), but here we are analyzing its use.

Table 6.25: Remap computational time (ms) and histogram distance.

Interpolation	Local	Time (ms)	Histogram similarity (*error*)
Linear	CPU	2	0.997
Linear	GPU	0.054	
Bilinear	CPU	3	0.966
Bilinear	GPU	0.078	

6.8.3 Particle rendering and Model simplification evaluation

The particle rendering is performed using OpenGL, transferring the obtained particle to a CUDA (Figure 6.44) compatible format (*device to device memory transfer*). The used model, in this case, has 38478 vertices and 57272 faces (*Reference Mesh*). We can decrease the complexity of the used CAD model (vertices and faces), ensuring a compromise between error and speed. We must guarantee a simplification maintaining the object appearance, minimizing the obtained error. Several tools were used, getting the best simplification results using the *Blender* software [Blender.org, 2019] with the option *decimate*.

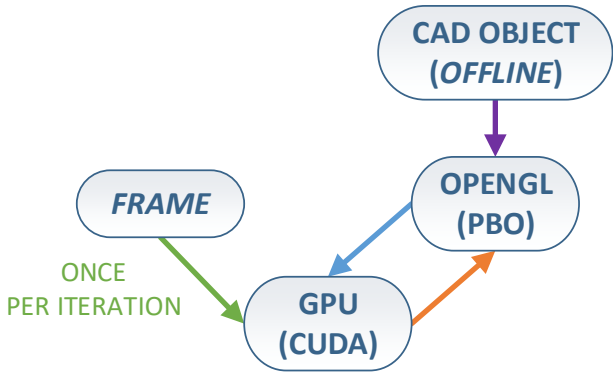

Figure 6.44: OpenGL and CUDA interaction.

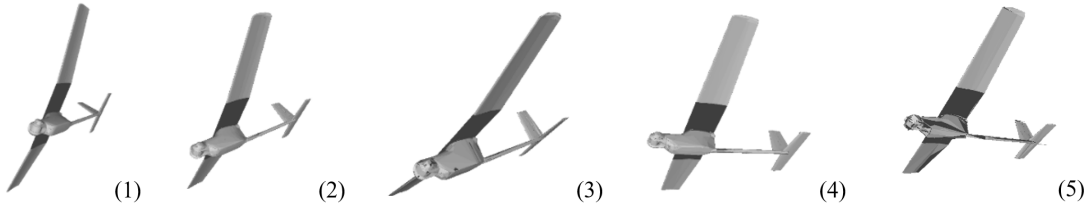

Figure 6.45: Model simplification: Test 1 to 5.

Table 6.26: Particle creation computational time (FPS).

Test	Vertices	Faces	Rendering speed (FPS)
Reference Mesh	57272	38478	2376
1	21860	28635	2421
2	12480	14317	2470
3	6789	7158	2644
4	7788	8589	2684
5	3511	2863	2704

When analyzing Figure 6.45, it is possible to see that when we increase the model simpli-

fication, some areas that need more detail will become poorly represented. This simplification is the primary source of error since we cannot get much object detail using fewer vertices and faces, especially in this kind of geometric objects with lots of discontinuities. The particle creation computational time is represented in Table 6.26, where we can see the increase of performance from the model simplification. The rendering error is obtained fixing the model projection in a central position and varying the projection angles using a uniform distribution restricted in the interval $[-180°, 180°]$ comparing it with the *reference mesh* in 100000 random poses. From the analysis of Table 6.26, Table 6.27 and Figure 6.46, we can see that the obtained error is small with a median value of 3.38% and rendering speed of 2704 FPS for test 5. If we need to render the UAV 100 times per filter iteration, we will obtain approximately 27 FPS, which is suitable for a real-time application.

Table 6.27: Model simplification error (%).

Test	Error (%)	
	Mean value	Median value
1	0.08	0.07
2	0.34	0.29
3	1.02	0.89
4	0.54	0.44
5	3.66	3.38

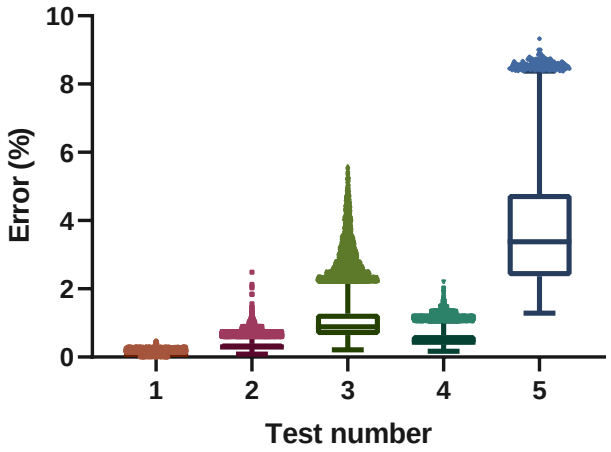

Figure 6.46: Model simplification error (%).

6.8.4 GPU-based color similarity metric evaluation

After the generation of particles, the similarity metric is the most time-consuming step since we need to test several poses for each captured frame. Using the CPU, as described in Section 6.5, the color similarity metric takes about 5 ms at 5 meters. Adapting the similarity metric to the GPU implementation, the color similarity metric (Figure 6.47) calculation takes about 0.71 ms (1408 FPS) after the particle rendering (Table 6.28) at 5 meters (approximately seven times faster).

Figure 6.47: Inner (*gray*) and outer (*blue*) regions using GPU.

Table 6.28: GPU-based color similarity metric processing time (ms).

Task	Time(ms)
Compute inner histogram and obtain outer histogram region	0.30
Compute outer histogram	0.40
between histograms	0.01
	Total ≅ 0.71

6.8.5 Conclusions

The GPU implementation is essential to obtain a real-time processing capability, and for that, we need to simplify the used UAV CAD model and use a GPU-based similarity metric. The distortion correction can be compensated directly in the particle rendering, decreasing the processing time. For each particle, and performing the distortion correction directly in the particle generation, we need 0.37 ms for the particle rendering (using test 5 as described in Table 6.26) and 0.71 ms for the GPU-based color similarity metric evaluation (Table 6.28). If we use a GPU with better characteristics, we can easily decrease the needed processing time.

Chapter 7

Conclusions and Future work

If you can look, see. If you can see, notice.

José Saramago

Chapter contents

This chapter presents the conclusions and describes future work.

7.1 Conclusions

This thesis introduces and describes a 3D model-based monocular vision system for UAV pose tracking. The presented architecture was based on a UPF scheme, combining some of the existing approaches and developing new ones to improve system performance (Section 3.5). The developed algorithm features a UAV detection method based on DNNs, a pose boosting methodology with a pre-trained database, motion filtering using a UKF combined with a UBiF or UBiGaF, and a pose optimization step to refine the estimate due to the approximated nature of image-based similarity metrics (as sub-optimal approximations to the true observation likelihood function). We have used a decoupled motion model to simplify the formulation (Section 3.1.4) that showed good results in our problem.

Acquiring real ground truth data is time-consuming and expensive. Instead, we created a "realistic" simulation environment (Section 6.2) that allowed us to quantify the performance of the system and was essential to analyze each of the components separately.

For target detection (Section 6.3), we have trained YOLO and SSD using a synthetic database and performed transfer learning to real data. The performance of the tested detectors was analyzed using 679 real captured images, obtaining a AUC of 72% for YOLO and a AUC of 53% when using SSD. YOLO also allows real-time image processing since we can perform detections at ≈ 30 FPS.

Inspired by the BPF [Okuma *et al.* , 2004], we have developed a novel pose boosting methodology that can generate suitable UAV pose hypotheses to feed the PFs. Pose boosting

applied to the UAV detections can achieve, by itself, a MAE of approximately 0.06 meters for X and Y and an error between $[-1.18, 1.27]$ meters for Z in the worst case (using ten database particles). The obtained rotation error was very similar in all the tested distances, obtaining a MAE of approximately 73.60 degrees for α and γ and 41.91 degrees for β in the worst case (using ten database particles). The obtained results showed that the pose boosting presents an overall low translation error, but must be combined with additional methods to be able to decrease the obtained rotation error. The pose boosting stage also proved to be essential to recover from local maxima and obtain errors compatible with the automatic landing requirements (Section 6.7.4).

To approximate the observation likelihood function required for the PFs, we have developed and tested three different similarity metrics (color, contour, and DT, as described in Section 5.4.1) in the developed environments (Section 6.2). The contour similarity metric presented a very sensitive and noisy behavior. The color and the DT similarity metrics have, due to the UAV geometric model symmetry, peaks around 180 degrees error that can lead to local maxima. Close to the correct pose, the DT similarity metric showed a good behavior in the non-clutter scenario (normal environment) but a noisy behavior in the high clutter environment. The color similarity metric worked well in all the tested scenarios and is very selective around the correct pose estimate. The color similarity metric also has the advantage of being four times faster than the DT similarity metric when using the CPU.

For the last stage of our filtering pipeline, to compensate for the sub-optimality of the similarity metrics, we have tested four different intra-frame pose optimization algorithms (PFO, PSO, modified PSO, and GAbF, as described in Section 6.6). All methods significantly improve the quality of the final estimate, in particular in the orientation dimension, where the error gets concentrated near zero and 180 degrees. This error happens due to the symmetry of the UAV geometry, where complementary poses achieve similar weights. The lowest error was obtained using the GAbF scheme, but since it is quite computationally intense, we have decided to use the PFO for its better compromise between speed and accuracy.

Finally, we have performed a complete system analysis on several landing sequences in a diversified set of scenarios (from simple to complex). Overall, the best results were obtained with the UKF combined with a UBiF or a UBiGaF. For a simple background, the best compromise between speed and accuracy was obtained using the UKF + UBiGaF without pose optimization. The real contribution of the pose optimization scheme became evident in the complex background obtaining an increase of performance in more than 50%. Both filters showed an excellent performance in the complex background, but the best performance was obtained when combining the UKF + UBiF with two pose optimization iterations and the UKF + UBiGaF with five pose optimization iterations. We have concluded that the pose optimization stage is essential to obtain good results, even when comparing to an equivalent increase of the number of particles in the algorithms without the pose optimization stage (Section 6.7.2).

To improve the real-time capability of the system, we have explored a complete GPU based implementation (Section 6.8), for distortion correction, particle rendering, and similarity metric evaluation. In the GPU, we can perform distortion correction using the bilinear interpolation 38 times faster and evaluate particles seven times faster than in the CPU. The UAV CAD model simplification can increase the rendering speed up to 12% with a median error of 3.38%.

The achieved tracking precision levels are suitable for automated landing. There are, how-

ever, some improvements that still can be made to decrease the error and increase the real-time capability of the system, as described in Section 7.2.

7.2 Future work

The developed work was described throughout the document detailing the system architecture and the obtained performance. The proposed tracking system architecture is robust and can easily be expanded and adapted to different applications. We have tackled many issues, but there are still some that could not be explored during this thesis period. The main identified issues are (Figure 7.1):

- ◇ **Camera control** - If we use a Pan-Tilt-Zoom (PTZ) camera, it is essential to develop a control system that maintains the UAV on the captured frame and compensates the ship balance;
- ◇ **Motion and Observation models** - The filter transition model accuracy can be increased by including the PTZ camera and GCS commands. Additionally, the filter observation model accuracy can also be increased by using a PTZ camera combined with an Inertial Measurement Unit (IMU);
- ◇ **Real-time system capability** - To be possible to implement the developed system architecture in real-time, we have to decrease the needed processing time for certain proposed stages. This processing time decrease can be done using other algorithms or better hardware (CPU and GPU);
- ◇ **Real field tests** - It is essential to validate the entire system with more real video sequences in different operation scenarios. During these tests, it is crucial to obtain the UAV ground truth using the UAV sensors. This would allow to:
  - ✓ Train the UAV detector using real data;
  - ✓ Analyze the ideal camera position, resolution, and FPS;
  - ✓ Improve our knowledge about the ideal relative landing speed, UAV motion model, and the landing trajectory.
- ◇ **Target detection and Pose boosting** - We can improve the target detection and pose boosting stages using different algorithms or use DNNs to obtain the initial UAV pose estimation directly instead of using the pre-trained database.

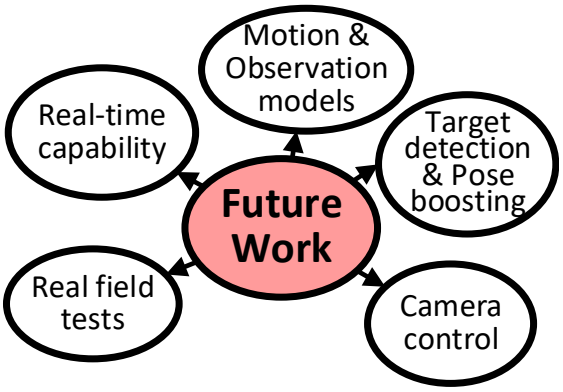

Figure 7.1: Future work.

Appendix A

Unscented Kalman Filter

Appendix contents

To filter along time the measurements obtained by our system, we will use a discrete-time UKF [Kraft, 2003; Crassidis & Markley, 2003; Van Der Merwe *et al.* , 2001; Zhou *et al.* , 2011; Cheon & Kim, 2007]. As described in Section 3.1.4, we consider that the UAV follows a constant velocity model and that linear and angular motions are independent. The adopted translational model is described in Section A.1, and the rotational model in Section A.2. The adopter filter state transition and observation models are described in Section A.3, the unscented sigma points creation in Section A.4, the filter prediction step in Section A.5, and the measurement update in Section A.6.

A.1 Translational model

Given the stated assumptions (Section 3.1.4), the linear motion state vector is defined as (Section 3.1.2):

$$\mathbf{t}_t^T = [\mathbf{u}_t^T, \mathbf{v}_t^T] \tag{A.1}$$

where $\mathbf{u}_t^T = [X, Y, Z]$ is the linear position and $\mathbf{v}_t^T = [v_x, v_y, v_z]$ is the linear velocity. The time evolution of the state for the linear dynamics is (3.12):

$$\mathbf{t}_{t+1} = \mathbf{F}^l(\mathbf{t}_t, \boldsymbol{\xi}_t^l) = \begin{bmatrix} \mathbf{I}_{3\times3} & \Delta t \cdot \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \mathbf{t}_t + \boldsymbol{\xi}_t^l \tag{A.2}$$

where $\boldsymbol{\xi}_t^l \sim \mathcal{N}(0, \mathbf{Q}_t^l)$ is a Gaussian noise random variable with zero mean and covariance $\mathbf{Q}_t^l$.

A.2 Rotational model

The angular velocity is represented by $\boldsymbol{\omega}_t^T = [\omega_x, \omega_y, \omega_z]$ (Section 3.1.2). Its time evolution is modeled as:

$$\boldsymbol{\omega}_{t+1} = \mathbf{F}^{\omega}(\boldsymbol{\omega}_t, \boldsymbol{\xi}_t^{\omega}) = [\boldsymbol{\omega}_t + \boldsymbol{\xi}_t^{\omega}] \tag{A.3}$$

where $\boldsymbol{\xi}_t^{\omega} \sim \mathcal{N}(0, \mathbf{Q}_t^{\omega})$ is a Gaussian noise random variable with zero mean and covariance $\mathbf{Q}_t^{\omega}$.

To represent the absolute orientation, we use a unit quaternion (3.6). The time evolution of the angular position can be written as (Section 3.1.5):

$$\mathbf{q}_{t+1} = \mathbf{q}_t \otimes \delta\mathbf{q}_t^{\omega} \otimes \delta\mathbf{q}_t^{r} = \mathbf{q}_t^T \otimes \boldsymbol{\Omega}(\boldsymbol{\omega}_t) \otimes \boldsymbol{\Omega}(\boldsymbol{\xi}_t^{r}) \tag{A.4}$$

where $\otimes$ represents unit quaternion multiplication (orientations composition), $\delta\mathbf{q}_t^{\omega}$ and $\delta\mathbf{q}_t^{r}$ are quaternions representing the integration of the effect of the angular velocity and rotation noise as described in (3.13) and $\boldsymbol{\Omega}(.)$ is obtained according to (3.14). The effect of the noise vector $\boldsymbol{\xi}_t^{r}$ is considered as a random angular velocity disturbance and has a covariance matrix $\mathbf{Q}_t^{r}$.

Because quaternions (3.6) are not a minimal representation of orientation, it is not straightforward to represent the state covariance. To address this issue, we represent the orientation dynamics in terms of error-quaternions with respect to the current state [Crassidis & Markley, 2003; Kraft, 2003; Pessanha Santos *et al.* , 2015]. A local error quaternion is defined as $\mathbf{e}^T = [\delta\boldsymbol{\varrho}^T, \delta q_4]$ (3.6), but a minimal representation is adopted using a vector of generalized *Rodrigues* parameters [Jiang & Ma, 2005; Crassidis & Markley, 2003]:

$$\mathbf{d} = R(\mathbf{e}) = f\frac{\delta\boldsymbol{\varrho}}{a + \delta q_4} \tag{A.5}$$

where $a = 1$ and f is a scale factor. We choose $f = 2(a+1)$ so that $\| \mathbf{d} \| = \rho$ (ρ is the angle of rotation (3.7)) for small angles [Crassidis & Markley, 2003]. The inverse transformation from $\mathbf{d}$ to $\mathbf{e}$ denoted $R^{-1}(\mathbf{d})$ is given by:

$$\delta q_4 = \frac{-a \| \mathbf{d} \|^2 + f\sqrt{f^2 + (1-a^2) \| \mathbf{d} \|^2}}{f^2 + \| \mathbf{d} \|^2} \tag{A.6}$$

$$\delta\boldsymbol{\varrho} = f^{-1}(a + \delta q_4)\mathbf{d} \tag{A.7}$$

with this parameterization, the incremental rotation dynamics is given by [Crassidis & Markley, 2003]:

$$\mathbf{d}_{t+1} = \mathbf{F}^{r}(\mathbf{d}_t, \mathbf{q}_t, \boldsymbol{\omega}_t, \boldsymbol{\xi}_t^{r}) = R(\mathbf{q}_t \otimes R^{-1}(\mathbf{d}_t) \otimes \boldsymbol{\Omega}(\boldsymbol{\omega}_t) \otimes \boldsymbol{\Omega}(\boldsymbol{\xi}_t^{r})) \tag{A.8}$$

Finally, using this representation, the state vector for the angular dynamics is defined as:

$$\mathbf{r}_t^T = [\mathbf{d}_t^T, \boldsymbol{\omega}_t^T] \quad \text{with covariance} \quad \mathbf{Q}_t^{a} = \begin{bmatrix} \mathbf{Q}_t^{r} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{Q}_t^{\omega} \end{bmatrix} \tag{A.9}$$

At each time step, the error vector $\mathbf{d}_t$ is reset to zero. After its update, $\mathbf{d}_t$ is accumulated to the absolute orientation quaternion $\mathbf{q}_t = \mathbf{q}_t \otimes R^{-1}(\mathbf{d}_t)$.

A.3 State transition and Observation models

The state transition model is represented as (Section 3.1.5):

$$\mathbf{x}_{t+1} = \mathbf{F}^i(\mathbf{x}_t, \boldsymbol{\xi}_t) \qquad i = 1, 2 \tag{A.10}$$

where $\boldsymbol{\xi}_t \sim \mathcal{N}(0, \mathbf{Q}_t)$ is a Gaussian noise random variable with zero mean and covariance $\mathbf{Q}_t$. For the translational motion, we have $\mathbf{x}_t = \mathbf{t}_t$, $\mathbf{F}^1 = \mathbf{F}^l$ (A.2), and $\mathbf{Q}_t = \mathbf{Q}_t^l$ (Section A.1). For the angular motion, we have $\mathbf{x}_t = \mathbf{r}_t$, $\mathbf{F}^2 = [\mathbf{F}^r, \mathbf{F}^\omega]$ (A.8, A.3), and $\mathbf{Q}_t = \mathbf{Q}_t^a$ (Section A.2).

The observation model is represented as (Section 3.1.6):

$$\mathbf{z}_t = \mathbf{H}(\mathbf{x}_t, \boldsymbol{\eta}_t) \tag{A.11}$$

where $\boldsymbol{\eta}_t \sim \mathcal{N}(0, \mathbf{R}_t)$ is a Gaussian noise random variable with zero mean and covariance matrix $\mathbf{R}_t$. For the translational motion, we have $\mathbf{z}_t = \mathbf{u}_t + \boldsymbol{\eta}_t$ and $\mathbf{R}_t = \mathbf{R}_t^l$. For the angular motion, our observation is the orientation error quaternion, encoded by incremental *Rodrigues* parameters $\mathbf{z}_t = R(\mathbf{q}_t \otimes \delta\mathbf{q}_t^\eta)$ where $\mathbf{q}_t$ is given by the coarse pose estimate, as described in Section 5.2.

A.4 Sigma points

In the UKF, a Gaussian approximation to the distributions of the n-dimensional state and process noises are used to generate a set of points (sigma points) that are sufficient to represent their statistics using a UT [Rui & Chen, 2001a; Julier, 2002; Li *et al.* , 2003]. The process noise covariance $\mathbf{Q}_{t-1}$ ($n \times n$ matrix) and the state covariance $\mathbf{P}_{t-1}$ ($n \times n$ matrix) are transformed into a $2n$ set of points $\delta\mathbf{x}_{t-1}(i)$ that represent perturbations to the current state according to:

$$\delta\mathbf{x}_{t-1}(i) = \pm \left(\sqrt{\iota \cdot (\mathbf{P}_{t-1} + \mathbf{Q}_{t-1})}\right)_i \qquad i = 1, ..., 2n \tag{A.12}$$

the parameter ι is a scaling parameter given by:

$$\iota = \alpha^2(n + k) \tag{A.13}$$

where α is a positive real ($0 \leq \alpha \leq 1$) parameter that controls the high order effects resulting from the existing nonlinearity, k is another real parameter ($k \geq 0$) that will control the distance between the sigma points and their average [Doucet *et al.* , 2000]. The matrices $\mathbf{P}_{t-1}$ and $\mathbf{Q}_{t-1}$ are symmetric and positive definite, so it is possible to use the *Cholesky* decomposition to compute $\sqrt{\iota \cdot (\mathbf{P}_{t-1} + \mathbf{Q}_{t-1})}$ [Higham, 1990]. The computation of the sigma points $\mathcal{X}_i$ is now done by adding directly $\delta\mathbf{x}_t(i)$ to the mean value of the state vector $\mathbf{x}_t$ according to:

$$\mathcal{X}_i = \mathbf{x}_{t-1} + \delta\mathbf{x}_{t-1}(i) \qquad i = 1, ..., 2n \text{ and } \mathcal{X}_0 = \mathbf{x}_t \tag{A.14}$$

A.5 Prediction

The process model $\mathbf{F}(.)$ is then applied to the obtained sigma points $\mathcal{X}_i$, generating the transformed sigma points $\mathcal{X}_i'$:

$$\mathcal{X}'_i = \mathbf{F}(\mathcal{X}_i, 0) \qquad i = 0, ..., 2n \tag{A.15}$$

No additional noise is considered at this step because the noise was already added at the sigma point's creation step (A.14). The *a priori* state estimate is obtained calculating the mean of the transformed sigma points $\mathcal{X}'_i$ according to:

$$\bar{\mathbf{x}}_t = \sum_{i=0}^{2n} W_i^m \mathcal{X}'_i \tag{A.16}$$

The weights are given by [Rui & Chen, 2001a; Haykin *et al.* , 2001]:

$$W_0^m = \frac{\lambda}{n+\lambda} \qquad \text{and} \qquad W_i^m = W_i^c = \frac{1}{2(n+\lambda)} \tag{A.17}$$

with λ given by:

$$\lambda = \alpha^2(n+k) - n \tag{A.18}$$

To estimate the *a priori* state covariance each propagated sigma point is removed from its mean to create the set of error vectors:

$$\delta\bar{\mathbf{x}}_t(i) = \mathcal{X}'_i - \bar{\mathbf{x}}_t \tag{A.19}$$

then:

$$\mathbf{P}_t^{xx} = \sum_{i=0}^{2n} W_i^c \, \delta\bar{\mathbf{x}}_t(i) \, \delta\bar{\mathbf{x}}_t(i)^T \tag{A.20}$$

where the scaling weights W_i^c are given by (A.17), except W_0^c alternatively given by [Cheon & Kim, 2007]:

$$W_0^c = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \tag{A.21}$$

where β is a non-negative term which incorporates knowledge of the higher-order moments(the chosen α and β determine the accuracy of third and higher-order moments for non-Gaussian inputs [Haykin *et al.* , 2001]) of the distribution [Doucet *et al.* , 2000].

The transformed sigma points are now projected into the measurement space according to:

$$\mathcal{Z}_i = \mathbf{H}(\mathcal{X}'_i, 0) \tag{A.22}$$

The measurement expected value is computed as:

$$\bar{\mathbf{z}}_t = \sum_{i=1}^{2n} W_i^m \mathcal{Z}_i \tag{A.23}$$

A.6 Measurement update

The measurement covariance estimate $\mathbf{P}_t^{zz}$ is given by:

$$\mathbf{P}_t^{zz} = \sum_{i=0}^{2n} W_i^c \left[\mathcal{Z}_i - \bar{\mathbf{z}}_t\right] \left[\mathcal{Z}_i - \bar{\mathbf{z}}_t\right]^T \tag{A.24}$$

The innovation vector $\boldsymbol{\nu}_t$ is obtained comparing the actual measurement $\mathbf{z}_t$ to the measurement estimate $\bar{\mathbf{z}}_t$:

$$\boldsymbol{\nu}_t = \mathbf{z}_t - \bar{\mathbf{z}}_t \tag{A.25}$$

The innovation covariance $\mathbf{P}_t^{\nu\nu}$ is obtained adding the measurement noise $\mathbf{R}_t$ to the measurement covariance $\mathbf{P}_t^{zz}$:

$$\mathbf{P}_t^{\nu\nu} = \mathbf{P}_t^{zz} + \mathbf{R}_t \tag{A.26}$$

The cross-correlation matrix $\mathbf{P}_t^{xz}$ is obtained from $\mathcal{Z}_i$ and $\mathcal{X}_i'$, according to:

$$\mathbf{P}_t^{xz} = \sum_{i=0}^{2n} W_i^c \left[\mathcal{X}_i' - \bar{\mathbf{x}}_t\right] \left[\mathcal{Z}_i - \bar{\mathbf{z}}_t\right]^T \tag{A.27}$$

The Kalman gain is then computed from:

$$\mathbf{K}_t = \mathbf{P}_t^{xz} (\mathbf{P}_t^{\nu\nu})^{-1} \tag{A.28}$$

Finally, the *a posteriori* state estimate is obtained according to:

$$\mathbf{x}_t = \bar{\mathbf{x}}_t + \mathbf{K}_t \boldsymbol{\nu}_t \tag{A.29}$$

and the state covariance $\mathbf{P}_t$ is given by:

$$\mathbf{P}_t = \mathbf{P}_t^{xx} - \mathbf{K}_t \mathbf{P}_t^{\nu\nu} \mathbf{K}_t^T \tag{A.30}$$

The UKF schematic view is described in Figure A.1.

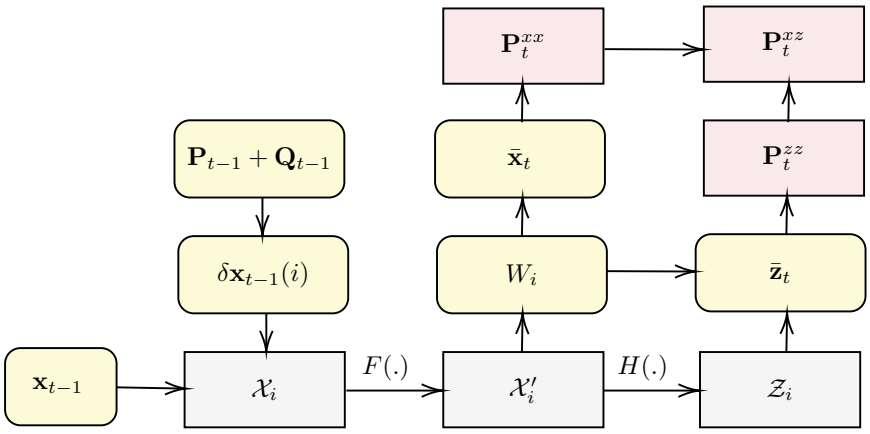

Figure A.1: UKF schematic view.

Appendix B

Resampling strategies

Appendix contents

One of the objectives of the resampling strategy is to avoid particle degeneracy[1], and it is essential that the random measure approximate the original distribution and prevent the existence of bias [Ristic & Clark, 2012; Verma *et al.* , 2003; Hall, 1985]. The tested traditional resampling strategies are described in Section B.1, and the tested traditional resampling variations in Section B.2.

B.1 Traditional strategies

The implemented traditional resampling strategies were [Beadle & Djuric, 1997; Carpenter *et al.* , 1999; Gordon *et al.* , 1993; Li *et al.* , 2015b; Liu & Chen, 1998; So, 2003; Douc & Cappé, 2005]:

- **Multinomial** - Generates N independently distributed random numbers u_t^j (t is the time instant and j is the random number index) from one standard uniform distribution over $(0, 1]$ and use them to select particles from the state vector[2] $\mathbf{x}_t^m$ (m is the particle state vector index). The particle is chosen when the following condition is met $\sum_{n=0}^{m-1} w_t^n < u_t^j \leq \sum_{n=0}^{m} w_t^n$ (w_t^n are the weights assigned to the particles);
- **Stratified** - Divides the particles into subgroups called *strata*. The particles are separated by N disjoint intervals, and the random numbers u_t^j are drawn independently from each one of these intervals according to $u_t^j \sim U\left(\frac{j-1}{N}, \frac{j}{N}\right]$ with $j = 1, 2, \ldots, N$;
- **Systematic** - Also divides the particles into subgroups called *strata*. The random number u_t^1 is drawn from $u_t^1 \sim U\left(0, \frac{1}{N}\right]$ and the rest are obtained according to $u_t^j = u_t^1 + \frac{j-1}{N}$ with $j = 2, 3, \ldots, N$;
- **Residual** - Consists in two stages, in the first stage, is performed a deterministic replication of each particle with weight w_t^j bigger than $\aleph = \frac{1}{N}$. The number of replicated

[1] Only a few of the particles will have significant weight.
[2] In our study, the state vector characterizes the UAV pose in a specific time instant (Section 3.1.2).

particle in this stage is given by $N_t = \sum_{m=1}^{N} \lfloor N\, w_t^m \rfloor$ (N is the total particle number). In the second stage, is applied a random sampling (e.g. multinomial resampling) with probability p equal to the remaining of the particle weights *residuals*, the total number of replicated particles in this stage is given by $R_t = N - N_t$. The residual of the weight is obtained according to $\hat{w}_t^m = w_t^m - \frac{N_t^m}{N}$. The first stage is a deterministic replication, and the number of times a particle is resampled is given by the second stage. Using e.g. multinomial resampling, a particle is resampled between $\lfloor N w_t^m \rfloor$ and $\lfloor N w_t^m \rfloor + R_t$.

B.2 Traditional strategies variations

Some variations of the traditional resampling strategies were also implemented, such as [Bolic *et al.* , 2003; Budhiraja *et al.* , 2007; Crisan & Lyons, 1999; Jianping *et al.* , 2009; Liu & Chen, 1998; Liu *et al.* , 1998]:

- **Residual systematic** - This resampling approach accumulates the fractional contribution of each particle in the searching sequence until it is large enough to generate a sample;
- **Branch-Kill** - The number of replicated particles $\mathbf{x}_t^m$ is given by $N_t^m = \lfloor N w_t^m \rfloor$ with probability $1-p$ or given by $N_t^m = \lfloor N w_t^m \rfloor + 1$ with probability p with $p = N w_t^m - \lfloor N w_t^m \rfloor$;
- **Optimal** - It automatically sets a threshold value $\aleph$, and all particles whose weights w_t^j are above this threshold are preserved rather than replicated. The other particles are resampled with probability equal to their weights and assigned a weight $\aleph$;
- **Reallocation** - It is based on a fixed threshold $\aleph = \frac{1}{N}$ where N is the sample size. The particles with weight w_t^j larger than $\aleph$ are replicated $N w_t^j$ times with weights given by $\frac{w_t^j}{\lfloor N w_t^j \rfloor}$, and the particles with smaller weight than $\aleph$ are resampled with probability $N w_t^j$ with weights $\aleph$;
- **Metropolis** - It uses a *Metropolis-Hastings* [Hastings, 1970; Chib & Greenberg, 1995] move step for searching in a particle set for a particle with a large weight to replace the current particle. The depth of the search S is predefined, and it is desirable that the number of times a particle be sampled to be proportional to its weight w_t^j;
- **Minimum Sampling** - Consists in two stages, in the first each particle is resampled $\lfloor N w_t^i \rfloor$ leading to a total of (M particles $M = \sum_{i=1}^{p} \lfloor N w_t^i \rfloor$) and in the second step the particles with relatively large weight residual (top $N - M$) will be further sampled one more each.

Appendix C

Directional statistics distributions

Appendix contents

When we use traditional filtering techniques (e.g. UKF [Crassidis & Markley, 2003; Kraft, 2003]) for attitude estimation, we have to consider a small angle assumption to quantify the existing uncertainty. To overcome this, we have explored the Bi (Section C.1) and the BiGa (Section C.2) distributions from the directional statistics field. The use of these distributions in a filtering structure is described in Section 5.3.

C.1 Bingham

Section contents

The Bi distribution is an antipodally symmetric distribution[1] that represents a zero-mean Gaussian distribution in $\mathbb{R}^d$ projected on the unit hypersphere S^{d-1} [Gilitschenski *et al.* , 2016; Fallaize & Kypraios, 2016; Bingham, 1974]. The PDF for the Bi distribution is obtained by [Bingham, 1974; Mardia & Jupp, 2000]:

$$P_B(\mathbf{q};\mathbf{M},\mathbf{Z}) = \frac{1}{F(\mathbf{Z})}\exp^{(\mathbf{q}^T\mathbf{M}\mathbf{Z}\mathbf{M}^T\mathbf{q})} \tag{C.1}$$

[1] Opposite points on S have equal probability.

where $\mathbf{q} \in S^{d-1} \subset \mathbb{R}^d : ||\mathbf{q}|| = 1$ is a unit vector[2], $\mathbf{M} \in \mathbb{R}^{d \times d}$ is an orthogonal matrix[3] describing the orientation of the distribution, $F(\mathbf{Z})$ is the normalization constant and $\mathbf{Z} = diag(z_1, z_2, ..., z_{d-1}, 0)$ with nondecreasing negative diagonal elements is the concentration matrix that controls the spread of the distribution around its mean (Figure C.1). Adding a multiple of the identity matrix to[4] $\mathbf{Z}$ or changing the order of a column of $\mathbf{M}$ and the corresponding $\mathbf{Z}$ columns does not change the distribution [Bingham, 1974], so we can force the last entry of $\mathbf{Z}$ to be zero for computational simplicity, and because of this, the last column of $\mathbf{M}$ represents the distribution mode [Bingham, 1974; Kurz *et al.* , 2013, 2014b].

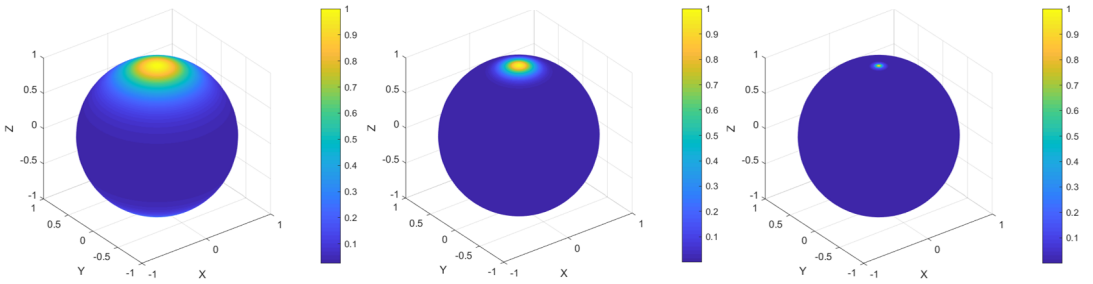

Figure C.1: Bi PDF with: $\mathbf{Z} = diag(-5, -5, 0) \wedge \mathbf{M} = \mathbf{I}_{3\times3}$ (*left*), $\mathbf{Z} = diag(-25, -25, 0) \wedge \mathbf{M} = \mathbf{I}_{3\times3}$ (*center*) and $\mathbf{Z} = diag(-500, -500, 0) \wedge \mathbf{M} = \mathbf{I}_{3\times3}$ (*right*).

C.1.1 Normalization constant

The main difficulty in the utilization of the Bi distribution consists in the computation of the normalization constant since the distribution must integrate to one over its domain. The normalization constant is obtained by:

$$F(\mathbf{Z}) = \int_{S^{d-1}} \exp^{(\mathbf{q}^T \mathbf{M}\mathbf{Z}\mathbf{M}^T \mathbf{q})} \, d\mathbf{q} = \int_{S^{d-1}} \exp^{(\mathbf{q}^T \mathbf{Z}\mathbf{q})} \, d\mathbf{q} \tag{C.2}$$

where $F(\mathbf{Z})$ does not depend on the matrix $\mathbf{M}$ since the orientation of the distribution peaks does not change its value. Since we are using this distribution in a real-time approach, we choose to interpolate tabulated values from a precomputed lookup table [Gilitschenski *et al.* , 2014; Niezgoda *et al.* , 2016; Glover *et al.* , 2012; Srivatsan *et al.* , 2017] for computational efficiency.

C.1.2 Product

The product of two Bi distributions is closed under multiplication after renormalization and is given by [Glover & Kaelbling, 2014]:

$$P_{B_1}(\mathbf{q}; \mathbf{M}_1, \mathbf{Z}_1) \cdot P_{B_2}(\mathbf{q}; \mathbf{M}_2, \mathbf{Z}_2) = \frac{1}{F(\mathbf{Z})} \exp^{(\mathbf{q}^T \mathbf{M}\mathbf{Z}\mathbf{M}^T \mathbf{q})} \tag{C.3}$$

[2] When using quaternions $d = 4$.
[3] A square matrix whose columns and rows are orthonormal vectors.
[4] The concentration matrix changes linearly and after the recalculation of $F(\mathbf{Z})$ to stay in the unit hypersphere, the obtained Bi distribution does not change.

where $F(\mathbf{Z})$ is the new normalization constant, $\mathbf{M}$ are the unit eigenvectors and $\mathbf{D}$ are the eigenvalues on the diagonal in ascending order of $(\mathbf{M}_1\mathbf{Z}_1\mathbf{M}_1^T + \mathbf{M}_2\mathbf{Z}_2\mathbf{M}_2^T)$, $\mathbf{Z} = \mathbf{D} - \lambda_{dd}\mathbf{I}_{d\times d}$ and λ_{dd} is the largest eigenvalue.

C.1.3 Rotation

When we are in S^3 is possible to change (rotate) the orientation of a Bi distribution $P_B(\mathbf{q}; \mathbf{M}, \mathbf{Z})$ (C.1) by a fixed quaternion $\mathbf{g} \in S^3$ according to [Glover & Kaelbling, 2013; Kurz *et al.* , 2014b; Gilitschenski *et al.* , 2016]:

$$P_B(\mathbf{r}; \mathbf{M} \otimes \mathbf{g}, \mathbf{Z}) \quad \text{when} \quad \mathbf{r} = \mathbf{q} \otimes \mathbf{g} \tag{C.4}$$

where $\otimes$ represents the composition of orientations, $\mathbf{M} \otimes \mathbf{g} \equiv [\mathbf{m}_1 \otimes \mathbf{g}, \mathbf{m}_2 \otimes \mathbf{g}, \mathbf{m}_3 \otimes \mathbf{g}, \mathbf{m}_4 \otimes \mathbf{g}]$ and $\mathbf{m}$ are the columns of $\mathbf{M}$. Since the quaternion multiplication is not commutative we have that $\mathbf{r} \sim P_B(\mathbf{g} \otimes \mathbf{M}, \mathbf{Z})$ when $\mathbf{r} = \mathbf{g} \otimes \mathbf{q}$.

C.1.4 Covariance

The covariance is a sufficient statistics[5] for the Bi distribution since the Bi distribution is the maximum entropy distribution[6] on the hypersphere, which matches the sample inertia matrix[7] [Mardia, 1975]. The covariance of the Bi PDF is given by [Bingham, 1974]:

$$Cov(\mathbf{q}) = E(\mathbf{q}\mathbf{q}^T) - (E(\mathbf{q}))^2 = E(\mathbf{q}\mathbf{q}^T) \tag{C.5}$$

where $(E(\mathbf{q}))^2 = 0$ is a consequence of the antipodal symmetry and $E(\mathbf{q}\mathbf{q}^T)$ is given by:

$$E(\mathbf{q}\mathbf{q}^T) = \mathbf{M} \cdot diag\left(\frac{\frac{d}{dz_1}F(\mathbf{Z})}{F(\mathbf{Z})}, \ldots, 1 - \frac{\sum_{i=1}^{d-1}\frac{d}{dz_i}F(\mathbf{Z})}{F(\mathbf{Z})}\right) \cdot \mathbf{M}^T \tag{C.6}$$

where the values of the gradient of F with respect to $\mathbf{Z}$ are precomputed and accessed by interpolation as made for the normalization constant. The covariance of the composition[8] of two Bi distributions can be obtained using the method of moments [Glover & Kaelbling, 2013; Prentice, 1984; Collins & Weiss, 1990]. The composition covariance $Cov(\mathbf{q}_1 \otimes \mathbf{q}_2)$ can be represented using the method of moments by [Glover & Kaelbling, 2013; Prentice, 1984; Collins & Weiss, 1990]:

[5] No other statistic that can be calculated provides any additional information.
[6] The most appropriate distribution to model the given set of data.
[7] The statistics used to estimate the covariance matrix, also known as scatter matrix.
[8] The composition is a directional analog to the addition of random vectors in a linear space.

$$\begin{bmatrix}
\begin{matrix}a_{11}b_{11} - 2a_{12}b_{12}\\ -2a_{13}b_{13} - 2a_{14}b_{14}\\ +a_{22}b_{22} + 2a_{23}b_{23}\\ +2a_{24}b_{24} + a_{33}b_{33}\\ +2a_{34}b_{34} + a_{44}b_{44}\end{matrix} &
\begin{matrix}a_{11}b_{12} + a_{12}b_{11} + a_{13}b_{14}\\ -a_{14}b_{13} - a_{12}b_{22} - a_{22}b_{12}\\ -a_{13}b_{23} - a_{23}b_{13} - a_{14}b_{24}\\ -a_{24}b_{14} - a_{23}b_{24} + a_{24}b_{23}\\ -a_{33}b_{34} + a_{34}b_{33} - a_{34}b_{44}\\ +a_{44}b_{34}\end{matrix} &
\begin{matrix}a_{11}b_{13} + a_{13}b_{11} - a_{12}b_{14}\\ +a_{14}b_{12} - a_{12}b_{23} - a_{23}b_{12}\\ -a_{13}b_{33} - a_{14}b_{34} - a_{34}b_{14}\\ +a_{23}b_{34} - a_{34}b_{23} + a_{24}b_{44}\\ -a_{44}b_{24}\end{matrix} &
\begin{matrix}a_{11}b_{14} + a_{12}b_{13} - a_{13}b_{12}\\ +a_{14}b_{11} - a_{12}b_{24} - a_{24}b_{12}\\ -a_{22}b_{23} + a_{23}b_{22} - a_{13}b_{34}\\ -a_{34}b_{13} - a_{23}b_{33} + a_{33}b_{23}\\ -a_{14}b_{44} - a_{24}b_{34} + a_{34}b_{24}\\ -a_{44}b_{14}\end{matrix} \\
\begin{matrix}a_{11}b_{12} + a_{12}b_{11} + a_{13}b_{14}\\ -a_{14}b_{13} - a_{12}b_{22} - a_{22}b_{12}\\ -a_{13}b_{23} - a_{23}b_{13} - a_{14}b_{24}\\ -a_{24}b_{14} - a_{23}b_{24} + a_{24}b_{23}\\ -a_{33}b_{34} + a_{34}b_{33} - a_{34}b_{44}\\ +a_{44}b_{34}\end{matrix} &
\begin{matrix}2a_{12}b_{12} + a_{11}b_{22}\\ +a_{22}b_{11} + 2a_{13}b_{24}\\ -2a_{14}b_{23} + 2a_{23}b_{14}\\ -2a_{24}b_{13} - 2a_{34}b_{34}\\ +a_{33}b_{44} + a_{44}b_{33}\end{matrix} &
\begin{matrix}a_{12}b_{13} + a_{13}b_{12} + a_{11}b_{23}\\ +a_{23}b_{11} - a_{12}b_{24} + a_{14}b_{22}\\ -a_{22}b_{14} + a_{24}b_{12} + a_{13}b_{34}\\ -a_{14}b_{33} + a_{33}b_{14} - a_{34}b_{13}\\ +a_{24}b_{34} + a_{34}b_{24} - a_{23}b_{44}\\ -a_{44}b_{23}\end{matrix} &
\begin{matrix}a_{12}b_{14} + a_{14}b_{12} + a_{11}b_{24}\\ +a_{12}b_{23} - a_{13}b_{22} + a_{22}b_{13}\\ -a_{23}b_{12} + a_{24}b_{11} - a_{14}b_{34}\\ +a_{34}b_{14} + a_{13}b_{44} + a_{23}b_{34}\\ -a_{24}b_{33} - a_{33}b_{24} + a_{34}b_{23}\\ -a_{44}b_{13}\end{matrix} \\
\begin{matrix}a_{11}b_{13} + a_{13}b_{11} - a_{12}b_{14}\\ +a_{14}b_{12} - a_{12}b_{23} - a_{23}b_{12}\\ -a_{13}b_{33} + a_{22}b_{24} - a_{24}b_{22}\\ -a_{33}b_{13} - a_{14}b_{34} - a_{34}b_{14}\\ +a_{23}b_{34} - a_{34}b_{23} + a_{24}b_{44}\\ -a_{44}b_{24}\end{matrix} &
\begin{matrix}a_{12}b_{13} + a_{13}b_{12} + a_{11}b_{23}\\ +a_{23}b_{11} - a_{12}b_{24} + a_{14}b_{22}\\ -a_{22}b_{14} + a_{24}b_{12} + a_{13}b_{34}\\ -a_{14}b_{33} + a_{33}b_{14} - a_{34}b_{13}\\ +a_{24}b_{34} + a_{34}b_{24} - a_{23}b_{44}\\ -a_{44}b_{23}\end{matrix} &
\begin{matrix}2a_{13}b_{13} + 2a_{14}b_{23}\\ -2a_{23}b_{14} + a_{11}b_{33}\\ +a_{33}b_{11} - 2a_{12}b_{34}\\ +2a_{34}b_{12} - 2a_{24}b_{24}\\ +a_{22}b_{44} + a_{44}b_{22}\end{matrix} &
\begin{matrix}a_{13}b_{14} + a_{14}b_{13} - a_{13}b_{23}\\ +a_{23}b_{13} + a_{14}b_{24} - a_{24}b_{14}\\ +a_{11}b_{34} + a_{12}b_{33} - a_{33}b_{12}\\ +a_{34}b_{11} + a_{23}b_{24} + a_{24}b_{23}\\ -a_{14}b_{44} - a_{22}b_{34} - a_{34}b_{22}\\ +a_{44}b_{12}\end{matrix} \\
\begin{matrix}a_{11}b_{14} + a_{12}b_{13} - a_{13}b_{12}\\ +a_{14}b_{11} - a_{12}b_{24} - a_{24}b_{12}\\ -a_{22}b_{23} + a_{23}b_{22} - a_{13}b_{34}\\ -a_{34}b_{13} - a_{23}b_{33} + a_{33}b_{23}\\ -a_{14}b_{44} - a_{24}b_{34} + a_{34}b_{24}\\ -a_{44}b_{14}\end{matrix} &
\begin{matrix}a_{12}b_{14} + a_{14}b_{12} + a_{11}b_{24}\\ +a_{12}b_{23} - a_{13}b_{22} + a_{22}b_{13}\\ -a_{23}b_{12} + a_{24}b_{11} - a_{14}b_{34}\\ +a_{34}b_{14} + a_{13}b_{44} + a_{23}b_{34}\\ -a_{24}b_{33} - a_{33}b_{24} + a_{34}b_{23}\\ -a_{44}b_{13}\end{matrix} &
\begin{matrix}a_{13}b_{14} + a_{14}b_{13} - a_{13}b_{23}\\ +a_{23}b_{13} + a_{14}b_{24} - a_{24}b_{14}\\ +a_{11}b_{34} + a_{12}b_{33} - a_{33}b_{12}\\ +a_{34}b_{11} + a_{23}b_{24} + a_{24}b_{23}\\ -a_{12}b_{44} - a_{22}b_{34} - a_{34}b_{22}\\ +a_{44}b_{12}\end{matrix} &
\begin{matrix}2a_{14}b_{14} - 2a_{13}b_{24}\\ +2a_{24}b_{13} + 2a_{12}b_{34}\\ -2a_{23}b_{23} - 2a_{34}b_{12}\\ +a_{11}b_{44} + a_{22}b_{33}\\ +a_{33}b_{22} + a_{44}b_{11}\end{matrix}
\end{bmatrix} \tag{C.7}$$

where $E(\mathbf{q}_1\mathbf{q}_1^T) = a_{ij}$ and $E(\mathbf{q}_2\mathbf{q}_2^T) = b_{ij}$ (4×4 matrices). Using this method, we can approximate the resulting composition covariance directly from their covariance matrices combination.

C.1.5 Inference

It is possible to estimate the parameters of a Bi distribution which approximates a set of samples [Bingham, 1974]. The inertia matrix for a set of M samples $\mathbf{q} = [\mathbf{q}_1, ..., \mathbf{q}_M]$ is given by [Bingham, 1974]:

$$\mathbf{S} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{q}_i\mathbf{q}_i^T \tag{C.8}$$

The Maximum Likelihood Estimation (MLE) $\hat{\mathbf{M}}$ for a set of samples is an eigenvalue problem since the columns of $\hat{\mathbf{M}}$ are eigenvectors $\boldsymbol{\kappa}$ of $\mathbf{S}$ [Bingham, 1974]. The MLE $\hat{\mathbf{Z}}$ can be found setting the partial log-likelihood function on $\mathbf{Z}$ to zero. This leads to:

$$\frac{\frac{d}{d\mathbf{z}_j}F(\mathbf{Z})}{F(\mathbf{Z})} = \frac{1}{M}\sum_{i=1}^{M}\left(\boldsymbol{\kappa}_j^T\mathbf{q}_i\right)^2 = \boldsymbol{\kappa}_j^T\mathbf{S}\boldsymbol{\kappa}_j = 0 \tag{C.9}$$

where $\boldsymbol{\kappa}_j$ are the eigenvectors of $\mathbf{S}$ (C.8). This calculation can be made using the Constrained Optimization BY Linear Approximations (COBYLA) algorithm [Powell, 1998].

C.1.6 Sampling

Is hard to sample directly from the Bi distribution because of the normalization constant. To solve this problem is used a *Metropolis-Hasting* sampler [Hastings, 1970; Chib & Greenberg, 1995] with proposal distribution given by a projected zero-mean Gaussian with covariance $\mathbf{S}$ (either from (C.6) or (C.8)) and target distribution provided by the Bi density [Glover & Kaelbling, 2013; Bingham, 1974].

C.2 Bingham-Gauss

Section contents

To quantify the correlation between the angular velocity (*Euclidean* space $\mathbb{R}^d$) and the attitude on the orientation manifold (S^{d-1}), we will use the BiGa distribution. The definition of conditional probability allows the distribution of two jointly distributed random vectors $\mathbf{x}$ and $\mathbf{y}$ to be written as the product of the distribution of $\mathbf{x}$ and the distribution of $\mathbf{y}$ conditioned on $\mathbf{x}$:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})\, p(\mathbf{y} \mid \mathbf{x}) \tag{C.10}$$

considering $\mathbf{x} = \mathbf{q}$ and $\mathbf{y} = \boldsymbol{\omega}$ (C.10) is possible to use this definition and construct a distribution that consists in the product of a Bi distribution and a Gaussian distribution conditioned on the Bi distributed random variables. The BiGa PDF is given by [Darling & DeMars, 2016a; Jazwinski, 1970]:

$$\begin{aligned} &P_{BG}\left(\mathbf{q}, \boldsymbol{\omega}; \mathbf{M}, \mathbf{Z}, \mathbf{m}_\omega, \mathbf{P}_q, \mathbf{P}_\omega, \mathbf{P}_{q\omega}\right) \\ &= P_G(\boldsymbol{\omega}; \mathbf{m}_\omega + \mathbf{P}_{\mathbf{q}\omega}^{\mathbf{T}} \mathbf{P}_{\mathbf{q}}^{-1} \mathbf{q}, \mathbf{P}_\omega - \mathbf{P}_{\mathbf{q}\omega}^{\mathbf{T}} \mathbf{P}_{\mathbf{q}}^{-1} \mathbf{P}_{\mathbf{q}\omega}) P_B(\mathbf{q}; \mathbf{M}, \mathbf{Z}) \end{aligned} \tag{C.11}$$

where $\mathbf{M}$ and $\mathbf{Z}$ are the orientation matrix and the matrix of concentration parameters of the Bi part defined by $\mathbf{P}_q$ (C.8) and P_G is a Gaussian PDF given by:

$$P_G(\boldsymbol{\omega}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) = \frac{1}{\sqrt{2\pi\boldsymbol{\sigma}^2}} e^{-\left(\frac{\omega-\mu}{\sigma}\right)^2} \tag{C.12}$$

with mean $\boldsymbol{\mu} = \mathbf{m}_\omega + \mathbf{P}_{\mathbf{q}\omega}^{\mathbf{T}} \mathbf{P}_{\mathbf{q}}^{-1} \mathbf{q}$ and variance $\boldsymbol{\sigma}^2 = \mathbf{P}_\omega - \mathbf{P}_{\mathbf{q}\omega}^{\mathbf{T}} \mathbf{P}_{\mathbf{q}}^{-1} \mathbf{P}_{\mathbf{q}\omega}$ as shown in (C.11) and described in Darling & DeMars [2016a]; Jazwinski [1970].

C.2.1 Distribution parameters

The parameters $\mathbf{m}_\omega$, $\mathbf{P}_\omega$, $\mathbf{P}_q$ and $\mathbf{P}_{q\omega}$ (C.11) are given by[9]:

$$\mathbf{m}_\omega = E_{P_{BG}}\left[\boldsymbol{\omega}\right] \in \mathbb{R}^r \tag{C.13}$$

$$\mathbf{P}_\omega = E_{P_{BG}}\left[(\boldsymbol{\omega} - \mathbf{m}_\omega)(\boldsymbol{\omega} - \mathbf{m}_\omega)^T\right] \in \mathbb{R}^{r\times r} \tag{C.14}$$

$$\mathbf{P}_q = E_{P_{BG}}\left[\mathbf{q}\mathbf{q}^T\right] \in \mathbb{R}^{d\times d} \tag{C.15}$$

$$\mathbf{P}_{q\omega} = E_{P_{BG}}\left[\mathbf{q}(\boldsymbol{\omega} - \mathbf{m}_\omega)^T\right] \in \mathbb{R}^{d\times r} \tag{C.16}$$

C.2.2 Antipodal symmetry

We have to guarantee that this distribution is antipodally symmetric with $\mathbf{q}$ and $-\mathbf{q}$ representing the same attitude. The PDF described in (C.11) needs to be divided into two hemispheres of the unit hypersphere to guarantee that condition [Darling & DeMars, 2016a]. The BiGa PDF becomes represented as:

[9] We have $r = 3$ and $d = 4$ in our study case.

$$P_{BG} = \begin{cases} P_{BG}\left(\mathbf{q}, \boldsymbol{\omega}; \mathbf{m}_{\boldsymbol{\omega}}, \mathbf{P}_{\boldsymbol{\omega}}, \mathbf{P}_{\mathbf{q}}, \mathbf{P}_{\mathbf{q}\boldsymbol{\omega}}\right) & \mathbf{q} \in \mathbb{S}^{+} \\ P_{BG}\left(\mathbf{q}, \boldsymbol{\omega}; \mathbf{m}_{\boldsymbol{\omega}}, \mathbf{P}_{\boldsymbol{\omega}}, \mathbf{P}_{\mathbf{q}}, -\mathbf{P}_{\mathbf{q}\boldsymbol{\omega}}\right) & \mathbf{q} \in \mathbb{S}^{-} \end{cases} \tag{C.17}$$

where $\mathbb{S}^{+}$ and $\mathbb{S}^{-}$ represent the hemispheres. For each $\mathbf{q}$ its position on the hypersphere is obtained analyzing its last nonzero element. If it is negative, the quaternion belongs to $\mathbb{S}^{-}$ otherwise belongs to $\mathbb{S}^{+}$.

Appendix D

UBiF sigma points creation

Appendix contents

The created sigma points follow the same principle as used in the UT applied in the UKF [Darling & DeMars, 2015a,b; Julier, 2002; Julier & Uhlmann, 1997, 2004] described in Appendix A. We need to use $4d-2$ samples that correspond in this case to fourteen samples ($d=4$). Since the distribution is antipodally symmetric, it is sufficient to consider only one pole (adapting the respective weights).

D.1 Canonical representation

The canonical[1] sigma points are given by [Gilitschenski *et al.* , 2016; Darling & DeMars, 2015b]:

$$\tilde{\mathbf{q}}^{1,2} = [\pm \sin\alpha_1, 0, 0, \cos\alpha_1]^T \tag{D.1}$$

$$\tilde{\mathbf{q}}^{3,4} = [0, \pm \sin\alpha_2, 0, \cos\alpha_2]^T \tag{D.2}$$

$$\tilde{\mathbf{q}}^{5,6} = [0, 0, \pm \sin\alpha_3, \cos\alpha_3]^T \tag{D.3}$$

$$\tilde{\mathbf{q}}^{7} = [0, 0, 0, 1]^T \tag{D.4}$$

where $\tilde{\mathbf{q}}^7$ is the sample located on the pole. For example, if we have $d=3$, we will need ten sigma points to approximate our Bi distribution, and the five corresponding to one pole will be located as shown in Figure D.1 with red circles. The covariance of the estimated Bi distribution is obtained by (C.6):

$$E_{P_B}\left\{\mathbf{x}_t\mathbf{x}_t^T\right\} = \mathbf{M}\cdot diag(f_1,f_2,f_3,f_4)\cdot\mathbf{M}^T \tag{D.5}$$

The deviation for each one of the canonical sigma points is obtained from α_i:

[1] The canonical distribution will be employed since it simplifies the needed mathematical approach because the parameters will be dimensionless [Darling & DeMars, 2015a,b].

$$\alpha_i = \sin^{-1}\left(\sqrt{\frac{f_i}{w_{B_i}}}\right) = \sin^{-1}\sqrt{\left(\frac{3f_i}{3f_i + \left(1 - \frac{1}{N}\right) f_4}\right)} \tag{D.6}$$

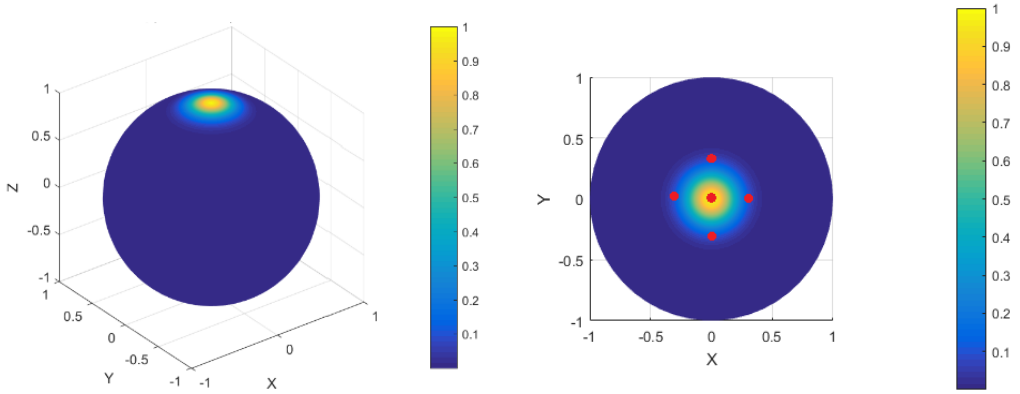

Figure D.1: An example of deterministic sampling with $d = 3$.

D.2 Weights calculation

The weights of the sigma points are given by:

$$w^{1,2} = \frac{w_{B_1}}{4} = \frac{f_1 + \frac{1-\frac{1}{N}f_4}{3}}{4} \tag{D.7}$$

$$w^{3,4} = \frac{w_{B_2}}{4} = \frac{f_2 + \frac{1-\frac{1}{N}f_4}{3}}{4} \tag{D.8}$$

$$w^{5,6} = \frac{w_{B_3}}{4} = \frac{f_3 + \frac{1-\frac{1}{N}f_4}{3}}{4} \tag{D.9}$$

where N is equal to the number of used sigma points. The weight for the central sigma point is obtained by:

$$w^7 = \frac{f_4}{N} \tag{D.10}$$

D.3 From canonical to the final sigma points

Each canonical sigma point $\tilde{\mathbf{q}}$ is multiplied by $\mathbf{M}$ (in the UBiF we use $\mathbf{M}_t^e\tilde{\mathbf{q}}$ as described in Section 5.3.1) originating the set of sigma points $\mathbf{q}$ that represent our P_B. The sigma points propagation is made adding a quaternion motion based on the angular velocities with some added noise to each one of the sigma points. The same principles are applied in the creation of the sigma points for the BiGa case (Appendix E), but in that case, we have to take into account the angular velocity part (*Euclidean* vector).

Appendix E

UBiGaF sigma points creation

Appendix contents

We follow the same approach to create the canonical sigma points as described in Appendix D for the Bi case but adding a *Euclidean* part describing the angular velocity.

E.1 Canonical representation

The canonical sigma points that represent the deviation for the *Euclidean* part are defined as [Darling & DeMars, 2015a,b]:

$$\tilde{\mathbf{r}}^{1,2} = [\overbrace{[0,0,0,1]^T}^{\tilde{\mathbf{q}}^{1,2}}, \overbrace{[\pm\delta,0,0]^T}^{\tilde{\boldsymbol{\omega}}^{1,2}}]^T \tag{E.1}$$

$$\tilde{\mathbf{r}}^{3,4} = [\overbrace{[0,0,0,1]^T}^{\tilde{\mathbf{q}}^{3,4}}, \overbrace{[0,\pm\delta,0]^T}^{\tilde{\boldsymbol{\omega}}^{3,4}}]^T \tag{E.2}$$

$$\tilde{\mathbf{r}}^{5,6} = [\overbrace{[0,0,0,1]^T}^{\tilde{\mathbf{q}}^{5,6}}, \overbrace{[0,0,\pm\delta]^T}^{\tilde{\boldsymbol{\omega}}^{5,6}}]^T \tag{E.3}$$

The angular deviations in the first three states of the attitude quaternion are introduced while the *Euclidean* part is held constant at zero to guarantee that the perturbed quaternion remains on the unit hypersphere according to:

$$\tilde{\mathbf{r}}^{7,8} = [\overbrace{[\pm\sin\alpha_1,0,0,\cos\alpha_1]^T}^{\tilde{\mathbf{q}}^{7,8}}, \overbrace{[0,0,0]^T}^{\tilde{\boldsymbol{\omega}}^{7,8}}]^T \tag{E.4}$$

$$\tilde{\mathbf{r}}^{9,10} = [\overbrace{[0,\pm\sin\alpha_2,0,\cos\alpha_2]^T}^{\tilde{\mathbf{q}}^{9,10}}, \overbrace{[0,0,0]^T}^{\tilde{\boldsymbol{\omega}}^{9,10}}]^T \tag{E.5}$$

$$\tilde{\mathbf{r}}^{11,12} = [\overbrace{[0,0,\pm\sin\alpha_3,\cos\alpha_3]^T}^{\tilde{\mathbf{q}}^{11,12}}, \overbrace{[0,0,0]^T}^{\tilde{\boldsymbol{\omega}}^{11,12}}]^T \tag{E.6}$$

The central sigma point is given by:

$$\tilde{\mathbf{r}}^{13} = [\overbrace{[0,0,0,1]^T}^{\tilde{\mathbf{q}}^{13}}, \overbrace{[0,0,0]^T}^{\tilde{\omega}^{13}}]^T \tag{E.7}$$

The parameters α_i and δ are given by:

$$\alpha_i = \sin^{-1}\left(\sqrt{\frac{f_i}{\mathbf{w_{B}}_i}}\right) \tag{E.8}$$

$$\delta = \sqrt{\frac{r}{\mathbf{w}_G}} \tag{E.9}$$

where $\mathbf{w_{B}}_i$ is described in (E.12), (E.13), (E.14), $\mathbf{w}_G$ is described in (E.10), and r is equal to three in the study case.

E.2 Weights calculation

The weights for the sigma points one to six are given by:

$$w^{1,\ldots,6} = \frac{\mathbf{w}_G}{4r} = \frac{2rf_{s+1}}{N4r} = \frac{2f_{s+1}}{4(2r+2s+1)} \tag{E.10}$$

where s is equal to three in the study case, N is equal to thirteen (the number of sigma points) and f_{s+1} is obtained analyzing the second moment of the canonical BiGa distribution (the zeroth and first moment is one and zero respectively) according to:

$$E_{p_{BG}}\left\{\tilde{\mathbf{q}}\tilde{\mathbf{q}}^T\right\} = diag\left[[f_1, f_2, f_3, f_4], [1,1,1]\right] \tag{E.11}$$

where the covariance can be obtained as described in (C.6) for the Bi part of the BiGa distribution alone. The weights for the sigma points seven to twelve are given by:

$$w^{7,8} = \frac{\mathbf{w_{B_1}}}{4} = \frac{f_1 + \frac{1-\frac{1}{N}-\frac{2r}{N}f_4}{s}}{4} \tag{E.12}$$

$$w^{9,10} = \frac{\mathbf{w_{B_2}}}{4} = \frac{f_2 + \frac{1-\frac{1}{N}-\frac{2r}{N}f_4}{s}}{4} \tag{E.13}$$

$$w^{11,12} = \frac{\mathbf{w_{B_3}}}{4} = \frac{f_3 + \frac{1-\frac{1}{N}-\frac{2r}{N}f_4}{s}}{4} \tag{E.14}$$

The weight for the central sigma point is given by:

$$w^{13} = \frac{\mathbf{w_C}}{2} = \frac{f_4}{2N} \tag{E.15}$$

E.3 From canonical to the final sigma points

Each sigma point is transformed from the canonical representation using the following relations [Darling & DeMars, 2016a]:

$$\mathbf{q} = M\mathbf{p} \tag{E.16}$$

$$\boldsymbol{\omega} = \sqrt{\mathbf{P}_{\omega} + \mathbf{P}_{q\omega}^{T}\mathbf{P}_{q}^{-1}\mathbf{P}_{q\omega}}\,\mathbb{Z} + \mathbf{P}_{q\omega}^{T}\mathbf{P}_{q}^{-1}M\mathbf{p} + \mathbf{m}_{\omega} \qquad \mathbf{q} \in \mathbb{S}^{s+} \tag{E.17}$$

$$\boldsymbol{\omega} = \sqrt{\mathbf{P}_{\omega} + \mathbf{P}_{q\omega}^{T}\mathbf{P}_{q}^{-1}\mathbf{P}_{q\omega}}\,\mathbb{Z} - \mathbf{P}_{q\omega}^{T}\mathbf{P}_{q}^{-1}M\mathbf{p} + \mathbf{m}_{\omega} \qquad \mathbf{q} \in \mathbb{S}^{s-} \tag{E.18}$$

where $\mathbf{p}$ corresponds to the quaternion part and $\mathbb{Z}$ correspond to the *Euclidean* part of the created canonical sigma points $\tilde{\mathbf{r}}$ originating the $\boldsymbol{\mathcal{Z}}$ sigma points. This transformation is similar to what is performed in the Bi case but now taking into account the *Euclidean* part of the sigma point vector as seen in (E.17) and (E.18).

The parameters $\mathbf{m}_{\omega}$, $\mathbf{P}_{\omega}$, $\mathbf{P}_{q}$ and $\mathbf{P}_{q\omega}$ as described in (C.13) to (C.16) can be obtained from the sigma points according to:

$$\mathbf{m}_{\omega} \approx 2\sum_{i=1}^{N} w^{i} f_{\omega}\left(\boldsymbol{\mathcal{Z}}^{i}\right) \tag{E.19}$$

$$\mathbf{P}_{\omega} \approx 2\sum_{i=1}^{N} w^{i}\left(f_{\omega}\left(\boldsymbol{\mathcal{Z}}^{i}\right) - \mathbf{m}_{\omega}\right)\left(f_{\omega}\left(\boldsymbol{\mathcal{Z}}^{i}\right) - \mathbf{m}_{\omega}\right)^{T} \tag{E.20}$$

$$\mathbf{P}_{q} \approx 2\sum_{i=1}^{N} w^{i} f_{q}\left(\boldsymbol{\mathcal{Z}}^{i}\right) f_{q}\left(\boldsymbol{\mathcal{Z}}^{i}\right)^{T} \tag{E.21}$$

$$\mathbf{P}_{\mathbf{q}\omega} \approx 2\sum_{i=1}^{N} w^{i} f_{q}\left(\boldsymbol{\mathcal{Z}}^{i}\right)\left(f_{\omega}\left(\boldsymbol{\mathcal{Z}}^{i}\right) - \mathbf{m}_{\omega}\right)^{T} \tag{E.22}$$

where $\boldsymbol{\mathcal{Z}}^{i}$ is the sigma point i, f_{ω} is the angular velocity part of the considered sigma point and f_{q} is the quaternion part of the considered sigma point.

Appendix F

Published work and Projects

This appendix describes the developed journal articles, book chapter, conference papers, and projects.

Journal articles:

- *Nuno Pessanha Santos, Victor Lobo and Alexandre Bernardino, "Unscented Particle Filters with Refinement Steps for UAV Pose Tracking", Robotics and Autonomous Systems, Elsevier, 2020* ***(Submitted - Under review)***;

- *Nuno Pessanha Santos, Victor Lobo and Alexandre Bernardino, "Directional statistics for 3D Model-Based UAV Tracking", IEEE Access, IEEE, 2020,* https://www.doi.org/10.1109/ACCESS.2020.2973970;

- *Nuno Pessanha Santos, Victor Lobo and Alexandre Bernardino, "Two-stage 3D Model-Based UAV Pose Estimation: A comparison of methods for optimization", Journal of Field Robotics, Wiley, 2020,* https://doi.org/10.1002/rob.21933;

- *Nuno Pessanha Santos, Fernando Melício, Victor Lobo and Alexandre Bernardino, "A ground-based vision system for UAV pose estimation", International Journal of Robotics and Mechatronics, 1(4), 138-144, 2014,* http://dx.doi.org/10.21535%2Fijrm.v1i4.180.

Conference papers:

- *Nuno Pessanha Santos, Victor Lobo and Alexandre Bernardino, "AUTOLAND project: Fixed-wing UAV Landing on a Fast Patrol Boat using Computer Vision", OCEANS 2019 - Seattle, 2019,* https://doi.org/10.23919/OCEANS40490.2019.8962869;

- *Nuno Pessanha Santos, Victor Lobo and Alexandre Bernardino, "3D Model-Based UAV Pose Estimation using GPU", OCEANS 2019 - Seattle, 2019,* https://doi.org/10.23919/OCEANS40490.2019.8962704;

- *Nuno Pessanha Santos, Victor Lobo and Alexandre Bernardino, "Unmanned Aerial Ve-*

hicle tracking using a Particle Filter based approach", 2019 IEEE Underwater Technology (UT) - Kaohsiung, 2019, https://doi.org/10.1109/UT.2019.8734465;

- *Nuno Pessanha Santos, Victor Lobo and Alexandre Bernardino, "3D model-based estimation for UAV tracking", OCEANS 2018 - Charleston, 2018, https://doi.org/10.1109/OCEANS.2018.8604539*;

- *Nuno Pessanha Santos, Victor Lobo and Alexandre Bernardino, "Particle filtering based optimization applied to 3D model-based estimation for UAV pose estimation", OCEANS 2017 - Aberdeen, 2017, https://doi.org/10.1109/OCEANSE.2017.8084783*;

- *Nuno Pessanha Santos, Victor Lobo and Alexandre Bernardino, "A ground-based vision system for UAV tracking", OCEANS 2015 - Genova, 2015, https://doi.org/10.1109/OCEANS-Genova.2015.7271349*;

- *Filipe Morais, Tiago Ramalho, Pedro Sinogas, Mario Monteiro Marques, Nuno Pessanha Santos and Victor Lobo, "Trajectory and Guidance Mode for autonomously landing an UAV on a naval platform using a vision approach", OCEANS 2015 - Genova, 2015, https://doi.org/10.1109/OCEANS-Genova.2015.7271423*;

- *Mario Monteiro Marques, Pedro Dias, Nuno Pessanha Santos, Victor Lobo, Ricardo Batista, D Salgueiro, A Aguiar, M Costa, J Estrela da Silva, A Sérgio Ferreira, J Sousa, Maria de Fátima Nunes, Elói Pereira, José Morgado, Ricardo Ribeiro, Jorge S Marques, Alexandre Bernardino, Miguel Griné and Matteo Taiana, "Unmanned Aircraft Systems in Maritime Operations: Challenges addressed in the scope of the SEAGULL project", OCEANS 2015 - Genova, 2015, https://doi.org/10.1109/OCEANS-Genova.2015.7271427*;

- *Nuno Pessanha Santos, Fernando Melício, Victor Lobo and Alexandre Bernardino, "A ground-based vision system for UAV pose estimation", The 10th International Conference on Intelligent Unmanned Systems - Montreal, 2014, http://ojs.unsysdigital.com/index.php/icius/article/view/295*.

Projects:

- **Voamais** - *"UAV computer vision operation in maritime and forest environments"*
  Financing: Portugal 2020 — Date: *Feb 2019 - now*

- **Seagull** - *"UAV based maritime situational awareness support systems"*
  Financing: National Strategic Reference Framework
  Project number: 34063 — Date: *Jun 2013 - Jun 2015*

- **Autoland** - *"Fixed-wing UAV automatic landing system on a moving platform"*
  Financing: National Strategic Reference Framework
  Project number: 23260 — Date: *Feb 2013 - Feb 2015*

Bibliography

Abdelali, Hamd Ait, Essannouni, Fedwa, Essannouni, Leila, & Aboutajdine, Driss. 2015. A new moving object tracking method using particle filter and probability product kernel. *Pages 1–6 of: 2015 Intelligent Systems and Computer Vision (ISCV)*. IEEE.

Ahmed, EA, Hafez, A, Ouda, AN, Ahmed, H, & Abd-Elkader, H. 2015. Modelling of a small unmanned aerial vehicle. *Advances in Robotics and Automation*, **4**(1), 1–11.

Ajaj, Rafic M, Beaverstock, Christopher S, & Friswell, Michael I. 2016. Morphing aircraft: the need for a new design philosophy. *Aerospace Science and Technology*, **49**, 154–166.

Ajaj, RM, Bourchak, M, & Harasani, W. 2014. Twist Morphing Using the Variable Cross Section Spar: Feasibility Study. *Journal of Aerospace Engineering*, **28**(6), 04014146.

Anderson, B. D., & Moore, J. B. 1979. *Optimal Filtering*. Prentice–Hall, Upper Saddle River, NJ.

Andreff, Nicolas, Espiau, Bernard, & Horaud, Radu. 2002. Visual servoing from lines. *The International Journal of Robotics Research*, **21**(8), 679–699.

Antone, Matthew E, & Teller, Seth. 2000. Automatic recovery of relative camera rotations for urban scenes. *Pages 282–289 of: Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, vol. 2. IEEE.

Artieda, Jorge, Sebastian, José M, Campoy, Pascual, Correa, Juan F, Mondragón, Iván F, Martínez, Carol, & Olivares, Miguel. 2009. Visual 3-d slam from uavs. *Journal of Intelligent and Robotic Systems*, **55**(4-5), 299.

Arulampalam, M Sanjeev, Maskell, Simon, Gordon, Neil, & Clapp, Tim. 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, **50**(2), 174–188.

Atkins, Ella Marie, Ollero, Aníbal, & Tsourdos, Antonios. 2016. *Unmanned aircraft systems*. John Wiley & Sons.

Azad, Pedram, Munch, David, Asfour, Tamim, & Dillmann, Rudiger. 2011. 6-DoF model-based tracking of arbitrarily shaped 3D objects. *Pages 5204–5209 of: Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE.

Azinheira, José Raul, & Rives, Patrick. 2008. Image-based visual servoing for vanishing features and ground lines tracking: Application to a uav automatic landing. *International Journal of Optomechatronics*, **2**(3), 275–295.

Bar-Shalom, Yaakov, Li, X Rong, & Kirubarajan, Thiagalingam. 2004. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons.

Barton, Jeffrey D. 2012. Fundamentals of small unmanned aircraft flight. *Johns Hopkins APL technical digest*, **31**(2), 132–149.

Bazin, Jean Charles, Demonceaux, Cédric, Vasseur, Pascal, & Kweon, In So. 2010. Motion estimation by decoupling rotation and translation in catadioptric vision. *Computer Vision and Image Understanding*, **114**(2), 254–273.

Bazin, Jean-Charles, Demonceaux, Cédric, Vasseur, Pascal, & Kweon, Inso. 2012. Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment. *The International Journal of Robotics Research*, **31**(1), 63–81.

Beadle, Edward R, & Djuric, Petar M. 1997. A fast-weighted Bayesian bootstrap filter for nonlinear model state estimation. *IEEE Transactions on Aerospace and Electronic Systems*, **33**(1), 338–343.

Beard, Randal W, & McLain, Timothy W. 2012. *Small unmanned aircraft: Theory and practice*. Princeton university press.

Beaverstock, C, Ajaj, RM, Friswell, MI, De Breuker, R, & Werter, NPM. 2013. Optimising mission performance for a morphing mav. *In: Proceedings of the Ankara International Aerospace Conference, Ankara, Turkey*, vol. 1113.

Bigun, Josef. 2006. *Vision with direction*. Springer.

Bingham, Christopher. 1974. An antipodally symmetric distribution on the sphere. *The Annals of Statistics*, 1201–1225.

Birsan, Marius. 2005. Unscented particle filter for tracking a magnetic dipole target. *Pages 1656–1659 of: OCEANS, 2005. Proceedings of MTS/IEEE*. IEEE.

Blake, Andrew, & Isard, Michael. 1997. The condensation algorithm-conditional density propagation and applications to visual tracking. *Pages 361–367 of: Advances in Neural Information Processing Systems*.

Blender.org. 2019. Blender.org - Home of the Blender Project. *URL https://www.blender.org/#hop*.

Bohyung Han, Comaniciu, D., Ying Zhu, & Davis, L. 2004 (June). Incremental density approximation and kernel-based Bayesian filtering for object tracking. *Pages I–I of: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1.

Boli, Miodrag, Djuri, Petar M., & Hong, Sangjin. 2004. Resampling algorithms for particle filters: a computational complexity perspective. *EURASIP J. Appl. Signal Process.*, **2004**, 2267–2277.

Bolic, Miodrug, Djuric, Petar M, & Hong, Sangiin. 2003. New resampling algorithms for particle filters. *In: Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP03). 2003 IEEE International Conference on*, vol. 2. IEEE.

Borgefors, Gunilla. 1986. Distance transformations in digital images. *Computer vision, graphics, and image processing*, **34**(3), 344–371.

Bouguet, Jean-Yves. 2008. Camera calibration toolbox for Matlab (2008). *URL http://www. vision. caltech. edu/bouguetj/calib_doc*, **1080**.

Boyd, Kendrick, Eng, Kevin H, & Page, C David. 2013. Area under the precision-recall curve: Point estimates and confidence intervals. *Pages 451–466 of: Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer.

Bradski, Gary, & Kaehler, Adrian. 2008. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".

Bréhard, Thomas, Coquelin, Pierre-Arnaud, Duflos, Emmanuel, & Vanheeghe, Philippe. 2008. Optimal policies search for sensor management: Application to the ESA radar. *Pages 1–8 of: 2008 11th International Conference on Information Fusion*. IEEE.

Bresenham, Jack. 1977. A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM*, **20**(2), 100–106.

Buckland, Michael, & Gey, Fredric. 1994. The relationship between recall and precision. *Journal of the American society for information science*, **45**(1), 12–19.

Budhiraja, Amarjit, Chen, Lingji, & Lee, Chihoon. 2007. A survey of numerical methods for nonlinear filtering problems. *Physica D: Nonlinear Phenomena*, **230**(1), 27–36.

Buyval, Alexander, Gavrilenkov, Mikhail, & Magid, Evgeni. 2017. A multithreaded algorithm of UAV visual localization based on a 3D model of environment: implementation with CUDA technology and CNN filtering of minor importance objects. *International Conference on Artificial Life and Robotics (ICAROB 2017)*.

Cagnoni, Stefano, Lutton, Evelyne, & Olague, Gustavo. 2007. *Genetic and evolutionary computation for image processing and analysis*. Vol. 8. Hindawi Publishing Corporation.

Canny, John. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 679–698.

Cao, Zhe, Sheikh, Yaser, & Banerjee, Natasha Kholgade. 2016. Real-time scalable 6DOF pose estimation for textureless objects. *Pages 2441–2448 of: 2016 IEEE International conference on Robotics and Automation (ICRA)*. IEEE.

Carceroni, Rodrigo L, & Brown, Christopher M. 1997. Numerical methods for model-based pose recovery. *University of Rochester, Computer Science Department, Technical Report*.

Carmi, Avishy, Godsill, Simon J, & Septier, Francois. 2009. Evolutionary MCMC particle filtering for target cluster tracking. *Pages 262–267 of: Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, 2009. DSP/SPE 2009. IEEE 13th*. IEEE.

Carpenter, James, Clifford, Peter, & Fearnhead, Paul. 1999. Improved particle filter for non-linear problems. *Pages 2–7 of: Radar, Sonar and Navigation, IEE Proceedings-*, vol. 146. IET.

Cesetti, A., Frontoni, E., Mancini, A., Zingaretti, P., & Longhi, S. 2010. A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks. *Journal of Intelligent and Robotic Systems*, **57**(1-4), 233–257.

Challa, Sudha. 2011. *Fundamentals of object tracking.* Cambridge University Press.

Chan, Tony F, & Vese, Luminita A. 2001. Active contours without edges. *IEEE Transactions on image processing*, **10**(2), 266–277.

Chan, Y-M, Huang, S-S, Fu, L-C, Hsiao, P-Y, & Lo, M-F. 2012. Vehicle detection and tracking under various lighting conditions using a particle filter. *IET intelligent transport systems*, **6**(1), 1–8.

Chang, Cheng, & Ansari, Rashid. 2005. Kernel particle filter for visual tracking. *IEEE signal processing letters*, **12**(3), 242–245.

Chang, Cheng, Ansari, Rashid, & Khokhar, Ashfaq. 2005. Multiple object tracking with kernel particle filter. *Pages 566–573 of: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE.

Chen, Jian, & Dawson, Darren M. 2006. UAV tracking with a monocular camera. *Pages 3873–3878 of: Proceedings of the 45th IEEE Conference on Decision and Control.* IEEE.

Cheng, John, Grossman, Max, & McKercher, Ty. 2014. *Professional CUDA c programming.* John Wiley & Sons.

Cheng, Yang, & Crassidis, John. 2004. Particle filtering for sequential spacecraft attitude estimation. *Page 5337 of: AIAA Guidance, Navigation, and Control Conference and Exhibit.*

Cheon, Yee-Jin, & Kim, Jong-Hwan. 2007. Unscented filtering in a unit quaternion space for spacecraft attitude estimation. *Pages 66–71 of: Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on.* IEEE.

Chib, Siddhartha, & Greenberg, Edward. 1995. Understanding the metropolis-hastings algorithm. *The American Statistician*, **49**(4), 327–335.

Chisholm, John P. 1989 (Sept. 12). *Advanced instrument landing system.* US Patent 4,866,450.

Chliveros, Georgios, Pateraki, Maria, & Trahanias, Panos. 2013. Robust multi-hypothesis 3D object pose tracking. *Pages 234–243 of: International Conference on Computer Vision Systems.* Springer.

Cho, Am, Kim, Jihoon, Lee, Sanghyo, Choi, Sujin, Lee, Boram, Kim, Bosung, Park, Noha, Kim, Dongkeon, & Kee, Changdon. 2007. Fully automatic taxiing, takeoff and landing of a UAV using a single-antenna GPS receiver only. *Pages 821–825 of: International Conference on Control, Automation and Systems ICCAS07.* IEEE.

Choi, Changhyun, & Christensen, Henrik I. 2010. Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. *Pages 4048–4055 of: Robotics and Automation (ICRA), 2010 IEEE International Conference on.* IEEE.

Choi, Changhyun, & Christensen, Henrik I. 2011. Robust 3D visual tracking using particle filtering on the SE (3) group. *Pages 4384–4390 of: Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE.

Chowdhary, Girish, Johnson, Eric N, Magree, Daniel, Wu, Allen, & Shein, Andy. 2013. GPS-denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft. *Journal of Field Robotics*, **30**(3), 415–438.

Clerc, Maurice, & Kennedy, James. 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, **6**(1), 58–73.

Collins, Robert T, & Weiss, Richard S. 1990. Vanishing point calculation as a statistical inference on the unit sphere. *Pages 400–403 of: Computer Vision, 1990. Proceedings, Third International Conference on.* IEEE.

Conway, Arthur William. 1937. Quaternion treatment of the relativistic wave equation. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences*, **162**(909), 145–154.

Cook, Shane. 2012. *CUDA programming: a developer's guide to parallel computing with GPUs.* Newnes.

Coquelin, Pierre-Arnaud, Deguest, Romain, & Munos, Rémi. 2009. Particle filter-based policy gradient in POMDPs. *Pages 337–344 of: Advances in Neural Information Processing Systems.*

Crassidis, John L, & Markley, F Landis. 2003. Unscented filtering for spacecraft attitude estimation. *Journal of guidance, control, and dynamics*, **26**(4), 536–542.

Crisan, Dan, & Lyons, Terry. 1999. A particle approximation of the solution of the Kushner–Stratonovitch equation. *Probability Theory and Related Fields*, **115**(4), 549–578.

Cyganek, Boguslaw, & Siebert, J Paul. 2011. *An introduction to 3D computer vision techniques and algorithms.* John Wiley & Sons.

Dambreville, Samuel, Sandhu, Romeil, Yezzi, Anthony, & Tannenbaum, Allen. 2010. A geometric approach to joint 2D region-based segmentation and 3D pose estimation using a 3D shape prior. *SIAM journal on imaging sciences*, **3**(1), 110–132.

Darling, Jacob, & DeMars, Kyle J. 2016a. The Bingham-Gauss Mixture Filter for Pose Estimation. *Page 5631 of: AIAA/AAS Astrodynamics Specialist Conference.*

Darling, Jacob E, & DeMars, Kyle J. 2015a. Analysis of the Gauss-Bingham Distribution for Attitude Uncertainty Propagation. *Pages 15–605 of: AAS/AIAA Space Flight Mechanics Meeting, No. AAS.*

Darling, Jacob E, & DeMars, Kyle J. 2015b. Rigid body attitude uncertainty propagation using the Gauss-Bingham distribution. *Pages 15–347 of: AAS/AIAA Space Flight Mechanics Meeting, No. AAS.*

Darling, Jacob E, & DeMars, Kyle J. 2016b. Uncertainty Propagation of correlated quaternion and Euclidean states using partially-conditioned Gaussian mixtures. *Pages 1805–1812 of: Information Fusion (FUSION), 2016 19th International Conference on.* IEEE.

Davis, Jesse, & Goadrich, Mark. 2006. The relationship between Precision-Recall and ROC curves. *Pages 233–240 of: Proceedings of the 23rd international conference on Machine learning.* ACM.

de La Gorce, Martin, Fleet, David J, & Paragios, Nikos. 2011. Model-based 3d hand pose estimation from monocular video. *IEEE transactions on pattern analysis and machine intelligence*, **33**(9), 1793–1805.

Deguchi, Koichiro. 1998. Optimal motion control for image-based visual servoing by decoupling translation and rotation. *Pages 705–711 of: Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, vol. 2. IEEE.

Dobrokhodov, Vladimir. 2015. *Kinematics and Dynamics of Fixed-Wing UAVs.* Springer. Pages 243–277.

Douc, Randal, & Cappé, Olivier. 2005. Comparison of resampling schemes for particle filtering. *Pages 64–69 of: ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.* IEEE.

Doucet, Arnaud, & Johansen, Adam M. 2009. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, **12**, 656–704.

Doucet, Arnaud, De Freitas, Nando, Murphy, Kevin, & Russell, Stuart. 2000. Rao-Blackwellised particle filtering for dynamic Bayesian networks. *Pages 176–183 of: Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc.

Doucet, Arnaud, De Freitas, Nando, & Gordon, Neil. 2001. *An introduction to sequential Monte Carlo methods.* Springer. Pages 3–14.

Eberhart, Russell, & Kennedy, James. 1995. Particle swarm optimization. *Pages 1942–1948 of: Proceedings of the IEEE international conference on neural networks*, vol. 4. Citeseer.

Eldridge, Mark, Harvey, James, Sandercock, Todd, & Smith, Ashleigh. 2009. Design and build a search and rescue uav. *Univ. Adelaide, Adelaide, Australia.*

Etkin, Bernard, & Reid, Lloyd Duff. 1996. *Dynamics of flight: stability and control.* Vol. 3. Wiley New York.

Eubank, Ryan, Atkins, Ella, & Macy, Daniel. 2009. Autonomous guidance and control of the flying fish ocean surveillance platform. *Page 2021 of: AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference.*

Eynard, Damien, Vasseur, Pascal, Demonceaux, Cédric, & Frémont, Vincent. 2012. Real time UAV altitude, attitude and motion estimation from hybrid stereovision. *Autonomous Robots*, **33**(1-2), 157–172.

Fallaize, Christopher J, & Kypraios, Theodore. 2016. Exact Bayesian inference for the Bingham distribution. *Statistics and Computing*, **26**(1-2), 349–360.

Farber, Rob. 2011. *CUDA application design and development*. Elsevier.

Fathian, Kaveh, Ramirez-Paredes, J Pablo, Doucette, Emily A, Curtis, J Willard, & Gans, Nicholas R. 2017. Quaternion based camera pose estimation from matched feature points. *arXiv preprint arXiv:1704.02672*.

Felzenszwalb, Pedro, McAllester, David, & Ramanan, Deva. 2008. A discriminatively trained, multiscale, deformable part model. *Pages 1–8 of: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE.

Ferreira, Artur J, & Figueiredo, Mario AT. 2012. *Boosting algorithms: A review of methods, theory, and applications*. Springer. Pages 35–85.

Fidler, Sanja, Mottaghi, Roozbeh, Yuille, Alan, & Urtasun, Raquel. 2013. Bottom-up segmentation for top-down detection. *Pages 3294–3301 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Finkelstein, David, Jauch, Josef M, Schiminovich, Samuel, & Speiser, David. 1962. Foundations of quaternion quantum mechanics. *Journal of mathematical physics*, **3**(2), 207–220.

Fitzgerald, Daniel L, Walker, Rodney A, & Campbell, Duncan A. 2005. A vision based emergency forced landing system for an autonomous uav. *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on. IEEE.*

Flach, Peter, & Kull, Meelis. 2015. Precision-Recall-Gain Curves: PR Analysis Done Right. *Pages 838–846 of: Advances in Neural Information Processing Systems*.

Flury, Thomas, & Shephard, Neil. 2011. Bayesian inference based only on simulated likelihood: particle filter analysis of dynamic economic models. *Econometric Theory*, **27**(05), 933–956.

Forsyth, David A, & Ponce, Jean. 2002. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.

Freund, Yoav, & Schapire, Robert E. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. *Pages 23–37 of: European conference on computational learning theory*. Springer.

Gajjar, Bhargav I, & Zalewski, Janusz. 2004. A07: On-ship landing and takeoff of Unmanned Aerial Vehicles (UAV'S). *IFAC Proceedings Volumes*, **37**(20), 42–46.

Gall, Juergen, Rosenhahn, Bodo, Brox, Thomas, & Seidel, Hans-Peter. 2010. Optimization and filtering for human motion capture. *International journal of computer vision*, **87**(1), 75–92.

Gautam, Alvika, Sujit, PB, & Saripalli, Srikanth. 2014. A survey of autonomous landing techniques for UAVs. *Pages 1210–1218 of: 2014 international conference on unmanned aircraft systems (ICUAS)*. IEEE.

Gelfand, Alan E, Diggle, Peter, Guttorp, Peter, & Fuentes, Montserrat. 2010. *Handbook of spatial statistics*. CRC press.

Gilitschenski, Igor, Kurz, Gerhard, Julier, Simon J, & Hanebeck, Uwe D. 2014. Efficient Bingham filtering based on saddlepoint approximations. *Pages 1–7 of: Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014 International Conference on*. IEEE.

Gilitschenski, Igor, Kurz, Gerhard, Julier, Simon J, & Hanebeck, Uwe D. 2016. Unscented orientation estimation based on the Bingham distribution. *IEEE Transactions on Automatic Control*, **61**(1), 172–177.

Girshick, Ross. 2015. Fast R-CNN. *Pages 1440–1448 of: Proceedings of the IEEE International Conference on Computer Vision*.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, & Malik, Jitendra. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *Pages 580–587 of: Proceedings of the IEEE conference on computer vision and pattern recognition*.

Gleason, Thomas J, & Fahlstrom, Paul G. 2010. Recovery of UAVs. *Encyclopedia of Aerospace Engineering*, 1–7.

Glover, Jared, & Kaelbling, Leslie Pack. 2013. Tracking 3-D rotations with the quaternion Bingham filter. *Computer Science and Artificial Intelligence Laboratory - Technical Report*.

Glover, Jared, & Kaelbling, Leslie Pack. 2014. Tracking the spin on a ping pong ball with the quaternion Bingham filter. *Pages 4133–4140 of: 2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Glover, Jared, Bradski, Gary, & Rusu, Radu Bogdan. 2012. Monte carlo pose estimation with quaternion kernels and the bingham distribution. *Page 97 of: Robotics: science and systems*, vol. 7.

Goldberg, David E, & Deb, Kalyanmoy. 1991. A comparative analysis of selection schemes used in genetic algorithms. *Urbana*, **51**, 61801–2996.

Goldstein, Herbert, Poole, Charles, & Safko, John. 2002. *Classical mechanics*.

Gómez-Luna, Juan, González-Linares, José María, Benavides, José Ignacio, & Guil, Nicolás. 2013. An optimized approach to histogram computation on GPU. *Machine Vision and Applications*, **24**(5), 899–908.

Gonzales, Christophe, & Dubuisson, Séverine. 2015. Combinatorial resampling particle filter: An effective and efficient method for articulated object tracking. *International Journal of Computer Vision*, **112**(3), 255–284.

Gordon, Neil J, Salmond, David J, & Smith, Adrian FM. 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Pages 107–113 of: Radar and Signal Processing, IEE Proceedings F*, vol. 140. IET.

Grant, Alan, Williams, Paul, Ward, Nick, & Basker, Sally. 2009. GPS jamming and the impact on maritime navigation. *Journal of Navigation*, **62**(02), 173–187.

Grisettiyz, Giorgio, Stachniss, Cyrill, & Burgard, Wolfram. 2005. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. *Pages 2432–2437 of: Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE.

Guenard, Nicolas, Hamel, Tarek, & Mahony, Robert. 2008. A practical visual servo control for an unmanned aerial vehicle. *IEEE Transactions on Robotics*, **24**(2), 331–340.

Gui, Yang, Guo, Pengyu, Zhang, Hongliang, Lei, Zhihui, Zhou, Xiang, Du, Jing, & Yu, Qifeng. 2013. Airborne Vision-Based Navigation Method for UAV Accuracy Landing Using Infrared Lamps. *Journal of Intelligent and Robotic Systems*, **72**(2), 197–218.

Guo, Rong-Hua, & Qin, Zheng. 2007. An unscented particle filter for ground maneuvering target tracking. *Journal of Zhejiang University-SCIENCE A*, **8**(10), 1588–1595.

Guo, Wenyan, Han, Chongzhao, & Lei, Ming. 2007. Improved unscented particle filter for nonlinear Bayesian estimation. *Pages 1–6 of: Information Fusion, 2007 10th International Conference on*. IEEE.

Hall, Peter. 1985. Resampling a coverage pattern. *Stochastic processes and their applications*, **20**(2), 231–246.

Hartley, Richard, & Zisserman, Andrew. 2003. *Multiple view geometry in computer vision*. Cambridge university press.

Hartwig, Sebastian, & Ropinski, Timo. 2019. Training Object Detectors on Synthetic Images Containing Reflecting Materials. *arXiv preprint arXiv:1904.00824*.

Hastings, W Keith. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**(1), 97–109.

Haug, Anton J. 2012. *Bayesian Estimation and Tracking: A Practical Guide*. John Wiley and Sons.

Havangi, Ramazan, Nekoui, Mohammad Ali, Taghirad, Hamid D, & Teshnehlab, Mohammad. 2013. An intelligent UFastSLAM with MCMC move step. *Advanced Robotics*, **27**(5), 311–324.

Haykin, Simon S, *et al.* . 2001. *Kalman filtering and neural networks*. Wiley Online Library.

Hazeldene, Adam, Sloan, Adam, Wilkin, Christopher, & Price, Andrew. 2004. In-flight orientation, object identification and landing support for an unmanned air vehicle. *Pages 13–15 of: Proceedings of the IEEE International Conference on Autonomous Robots and Agents*.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Pages 346–361 of: European Conference on Computer Vision*. Springer.

Higham, Nicholas J. 1990. Analysis of the Cholesky decomposition of a semi-definite matrix. *Manchester Institute for Mathematical Sciences School of Mathematics*.

Ho, Yu-Chi, & Lee, Robert CK. 1964. *A Bayesian approach to problems in stochastic estimation and control*. Tech. rept. Division of Engineering and Applied Physics, Harvard University, Cambridge, Massachusetts.

Howse, Joseph, Puttemans, Steven, Hua, Quan, & Sinha, Utkarsh. 2015. *OpenCV 3 Blueprints*. Packt Publishing Ltd.

Hubbard, David, Morse, Bryan, Theodore, Colin, Tischler, Mark, & McLain, Timothy. 2007. Performance evaluation of vision-based navigation and landing on a rotorcraft unmanned aerial vehicle. *Pages 5–5 of: Applications of Computer Vision, 2007. WACV07. IEEE Workshop on*. IEEE.

Huh, Sungsik, & Shim, David Hyunchul. 2010. A vision-based automatic landing method for fixed-wing UAVs. *Journal of Intelligent and Robotic Systems*, **57**(1-4), 217.

Humpherys, Jeffrey, Redd, Preston, & West, Jeremy. 2012. A fresh look at the Kalman filter. *SIAM review*, **54**(4), 801–823.

Iltis, Ronald A. 1990. Joint estimation of PN code delay and multipath using the extended Kalman filter. *IEEE Transactions on communications*, **38**(10), 1677–1685.

Inc, Insitu. 2016. ScanEagle. *URL https://insitu.com/information-delivery/unmanned-systems/scaneagle#3*.

J. Huang, V. Rathod, D. Chow C. Sun, & Zhu, M. 2017. *Tensorflow object detection API*.

Jähne, Bernd, Haussecker, Horst, & Geissler, Peter. 1999. *Handbook of computer vision and applications*. Vol. 2. Citeseer.

Jain, Ramesh, Kasturi, Rangachar, & Schunck, Brian G. 1995. *Machine vision*. Vol. 5. McGraw-Hill New York.

Jalal, Mona, Spjut, Josef, Boudaoud, Ben, & Betke, Margrit. 2019. SIDOD: A Synthetic Image Dataset for 3D Object Pose Recognition with Distractors. *In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Jammalamadaka, S Rao, & Sengupta, Ambar. 2001. *Topics in Circular Statistics*. Vol. 5. World Scientific.

Jazwinski, Andrew H. 1970. *Stochastic processes and filtering theory*. Academic Press, New-York.

Jiang, Lei, Wu, XiaoJun, & Kittler, Josef. 2018. Pose Invariant 3D Face Reconstruction. *arXiv preprint arXiv:1811.05295*.

Jiang, Xue-Yuan, & Ma, Guang-Fu. 2005. Spacecraft attitude estimation from vector measurements using particle filter. *Pages 682–687 of: 2005 International Conference on Machine Learning and Cybernetics*, vol. 2. IEEE.

Jianping, Zheng, Baoming, Bai, & Xinmei, Wang. 2009. Increased-diversity systematic resampling in particle filtering for BLAST. *Systems Engineering and Electronics, Journal of*, **20**(3), 493–498.

Josef, B. 2006. *Vision with Direction: A Systematic Introduction to Image Processing and Computer Vision.*

Joseph Tan, David, Tombari, Federico, Ilic, Slobodan, & Navab, Nassir. 2015. A versatile learning-based 3d temporal tracker: Scalable, robust, online. *Pages 693–701 of: Proceedings of the IEEE International Conference on Computer Vision.*

Julier, Simon J. 2002. The scaled unscented transformation. *Pages 4555–4559 of: American Control Conference, 2002. Proceedings of the*, vol. 6. IEEE.

Julier, Simon J, & Uhlmann, Jeffrey K. 1997. New extension of the Kalman filter to nonlinear systems. *Pages 182–193 of: AeroSense97*. International Society for Optics and Photonics.

Julier, Simon J, & Uhlmann, Jeffrey K. 2004. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, **92**(3), 401–422.

Kantas, Nikolas, Doucet, Arnaud, Singh, Sumeetpal S, Maciejowski, Jan, & Chopin, Nicolas. 2015. On particle methods for parameter estimation in state-space models. *Statistical science*, **30**(3), 328–351.

Keilwagen, Jens, Grosse, Ivo, & Grau, Jan. 2014. Area under Precision-Recall Curves for Weighted and Unweighted Data. *PLOS ONE*, **9**(3), 1–13.

Khan, M Khalid, & Nystrom, Ingela. 2010. A modified particle swarm optimization applied in image registration. *Pages 2302–2305 of: Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE.

Kim, H Jin, Kim, Mingu, Lim, Hyon, Park, Chulwoo, Yoon, Seungho, Lee, Daewon, Choi, Hyunjin, Oh, Gyeongtaek, Park, Jongho, & Kim, Youdan. 2013. Fully autonomous vision-based net-recovery landing system for a fixed-wing UAV. *IEEE/ASME Transactions On Mechatronics*, **18**(4), 1320–1333.

Kim, Pyojin, Coltin, Brian, & Kim, H Jin. 2018. Low-drift visual odometry in structured environments by decoupling rotational and translational motion. *Pages 7247–7253 of: 2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Kingston, Derek, Beard, Randal, McLain, Tim, Larsen, Michael, & Ren, Wei. 2003. Autonomous vehicle technologies for small fixed wing UAVs. *Page 6559 of: 2nd AIAA" Unmanned Unlimited" Conf. and Workshop & Exhibit.*

Kiru Park, Johann Prankl, Michael Zillich, & Vincze, Markus. 2017. Pose Estimation of Similar Shape Objects using Convolutional Neural Network trained by Synthetic data. *OAGM-ARW Joint Workshop.*

Klausen, Kristian, Moe, Jostein Borgen, van den Hoorn, Jonathan Cornel, Gomola, Alojz, Fossen, Thor I, & Johansen, Tor Arne. 2016. Coordinated control concept for recovery of a fixed-wing UAV on a ship using a net carried by multirotor UAVs. *Pages 964–973 of: 2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE.

Klein, Georg, & Drummond, Tom. 2003. Robust visual tracking for non-instrumented augmented reality. *Page 113 of: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society.

Klein, Georg, & Murray, David W. 2006. Full-3D Edge Tracking with a Particle Filter. *Pages 1119–1128 of: British Machine Vision Conference (BMVC)*.

Klein, Vladislav, & Morelli, Eugene A. 2006. *Aircraft system identification: theory and practice*. American Institute of Aeronautics and Astronautics Reston, Va, USA.

Klimkowska, A, Lee, I, & Choi, K. 2016. Possibilities of UAS for maritime monitoring. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, **41**, 885.

Kong, Weiwei, Zhang, Daibing, Wang, Xun, Xian, Zhiwen, & Zhang, Jianwei. 2013. Autonomous landing of an UAV with a ground-based actuated infrared stereo vision system. *Pages 2963–2970 of: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE.

Kong, Weiwei, Zhou, Dianle, Zhang, Daibing, & Zhang, Jianwei. 2014. Vision-based autonomous landing system for unmanned aerial vehicle: A survey. *Pages 1–8 of: 2014 international conference on multisensor fusion and information integration for intelligent systems (MFI)*. IEEE.

Kong, Weiwei, Zhang, Daibing, & Zhang, Jianwei. 2015. A ground-based multi-sensor system for autonomous landing of a fixed wing UAV. *Pages 1303–1310 of: 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE.

Kosaka, Akio, & Nakazawa, Goichi. 1993. Vision-Based Motion Tracking of Rigid Objects Using Prediction of Uncertainties. *Pages 2637–2637 of: IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1.

Kotecha, J. H., & Djuric, P. M. 2003. Gaussian particle filtering. *IEEE Transactions on Signal Processing*, **51**(10).

Kotecha, Jayesh H, & Djuric, Petar M. 2001. Gaussian sum particle filtering for dynamic state space models. *Pages 3465–3468 of: 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. Proceedings (ICASSP'01).*, vol. 6. IEEE.

Kraft, Edgar. 2003. A quaternion-based unscented Kalman filter for orientation tracking. *Pages 47–54 of: Proceedings of the Sixth International Conference of Information Fusion*, vol. 1.

Krizhevsky, Alex, Sutskever, Ilya, & Hinton, Geoffrey E. 2012. Imagenet classification with deep convolutional neural networks. *Pages 1097–1105 of: Advances in neural information processing systems*.

Krzeszowski, Tomasz, Kwolek, Bogdan, & Wojciechowski, Konrad. 2010. Articulated body motion tracking by combined particle swarm optimization and particle filtering. *Pages 147–154 of: International Conference on Computer Vision and Graphics*. Springer.

Kurz, Gerhard, Gilitschenski, Igor, & Hanebeck, Uwe D. 2013. Recursive nonlinear filtering for angular data based on circular distributions. *Pages 5439–5445 of: American Control Conference (ACC), 2013*. IEEE.

Kurz, Gerhard, Gilitschenski, Igor, & Hanebeck, Uwe D. 2014a. Nonlinear measurement update for estimation of angular systems based on circular distributions. *Pages 5694–5699 of: American Control Conference (ACC), 2014*. IEEE.

Kurz, Gerhard, Gilitschenski, Igor, Julier, Simon, & Hanebeck, Uwe D. 2014b. Recursive Bingham Filter for Directional Estimation Involving 180 Degree Symmetry. *Journal of Advances in Information Fusion*, **9**(2), 90–105.

Kwok, Ngai Ming, Fang, Gu, & Zhou, Weizhen. 2005. Evolutionary particle filter: re-sampling from the genetic algorithm perspective. *Pages 2935–2940 of: Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE.

Kwolek, Bogdan. 2013. Multi-object tracking using particle swarm optimization on target interactions. *Pages 63–78 of: Advances in Heuristic Signal Processing and Applications*. Springer.

Kyrki, Ville, & Kragic, Danica. 2011. Tracking rigid objects using integration of model-based and model-free cues. *Machine Vision and Applications*, **22**(2), 323–335.

Lange, Sven, Sunderhauf, Niko, & Protzel, Peter. 2008. Autonomous landing for a multirotor UAV using vision. *Pages 482–491 of: International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2008)*.

LeCun, Yann, Bottou, Leon, Bengio, Yoshua, & Haffner, Patrick. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

Lee, Bum-Jong, & Park, Jong-Seung. 2006. Fast vision-based camera tracking for augmented environments. *Pages 1018–1023 of: Intelligent Computing in Signal Processing and Pattern Recognition*. Springer.

Lee, Bum-Jong, Park, Jong-Seung, & Sung, Mee Young. 2006. Vision-based real-time camera matchmoving with a known marker. *Pages 193–204 of: International Conference on Entertainment Computing*. Springer.

Lee, Taeyoung, Leok, Melvin, & McClamroch, N Harris. 2010. Geometric tracking control of a quadrotor UAV on SE (3). *Pages 5420–5425 of: 49th IEEE conference on decision and control (CDC)*. IEEE.

Lefferts, Ern J, Markley, F Landis, & Shuster, Malcolm D. 1982. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, **5**(5), 417–429.

Lepetit, Vincent, Fua, Pascal, *et al.* . 2005. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision*, **1**(1), 1–89.

Li, Daizong. 2013. Design of a new VTOL UAV by combining cycloidal blades and fanwing propellers. *Pages 1–8 of: 2013 IEEE Aerospace Conference*. IEEE.

Li, Peihua, Zhang, Tianwen, & Pece, Arthur EC. 2003. Visual contour tracking based on particle filters. *Image and Vision Computing*, **21**(1), 111–123.

Li, Tian-cheng, Villarrubia, Gabriel, Sun, Shu-dong, Corchado, Juan M, & Bajo, Javier. 2015a. Resampling methods for particle filtering: identical distribution, a new method, and comparable study. *Frontiers of Information Technology & Electronic Engineering*, **16**(11), 969–984.

Li, Tiancheng, Sattar, Tariq Pervez, & Sun, Shudong. 2012. Deterministic resampling: Unbiased sampling to avoid sample impoverishment in particle filters. *Signal Processing*, **92**(7), 1637–1645.

Li, Tiancheng, Bolic, Miodrag, & Djuric, Petar M. 2015b. Resampling Methods for Particle Filtering: Classification, implementation, and strategies. *Signal Processing Magazine, IEEE*, **32**(3), 70–86.

Li-Chee-Ming, Julien, & Armenakis, Costas. 2015. A Feasibility Study on Using ViSPS 3d Model-Based Tracker for UAV Pose Estimation in Outdoor Environments. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, **40**(1), 329.

Liang-Qun, Li, Hong-Bing, Ji, & Jun-Hui, Luo. 2005. The iterated extended Kalman particle filter. *Pages 1213–1216 of: Communications and Information Technology, 2005. ISCIT 2005. IEEE International Symposium on*, vol. 2. IEEE.

Lin, Shanggang, Garratt, Matthew A, & Lambert, Andrew J. 2015. Real-time 6DoF deck pose estimation and target tracking for landing an UAV in a cluttered shipboard environment using on-board vision. *Pages 474–481 of: 2015 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE.

Lin, Shanggang, Garratt, Matthew A, & Lambert, Andrew J. 2016. Monocular vision-based real-time target recognition and tracking for autonomously landing an UAV in a cluttered shipboard environment. *Autonomous Robots*, 1–21.

Lin, Shanggang, Garratt, Matthew A, & Lambert, Andrew J. 2017a. Monocular vision-based real-time target recognition and tracking for autonomously landing an UAV in a cluttered shipboard environment. *Autonomous Robots*, **41**(4), 881–901.

Lin, Tsung-Yi, Dollár, Piotr, Girshick, Ross, He, Kaiming, Hariharan, Bharath, & Belongie, Serge. 2017b. Feature pyramid networks for object detection. *Pages 2117–2125 of: Proceedings of the IEEE conference on computer vision and pattern recognition*.

Liu, Bin, Cheng, Shi, & Shi, Yuhui. 2016a. *Particle Filter Optimization: A Brief Introduction*. Cham: Springer International Publishing. Pages 95–104.

Liu, Jane, & West, Mike. 2001a. Combined parameter and state estimation in simulation-based filtering. *Pages 197–223 of: Sequential Monte Carlo methods in practice*. Springer.

Liu, Jane, & West, Mike. 2001b. *Combined parameter and state estimation in simulation-based filtering*. Springer. Pages 197–223.

Liu, Jun S. 1996. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, **6**(2), 113–119.

Liu, Jun S, & Chen, Rong. 1998. Sequential Monte Carlo methods for dynamic systems. *Journal of the American statistical association*, **93**(443), 1032–1044.

Liu, Jun S, Chen, Rong, & Wong, Wing Hung. 1998. Rejection control and sequential importance sampling. *Journal of the American Statistical Association*, **93**(443), 1022–1031.

Liu, Wei, Anguelov, Dragomir, Erhan, Dumitru, Szegedy, Christian, Reed, Scott, Fu, Cheng-Yang, & Berg, Alexander C. 2016b. SSD: Single shot multibox detector. *Pages 21–37 of: European Conference on Computer Vision*. Springer.

Liu, Yuan, Wang, Jun, Song, Jingwei, & Song, Zihui. 2017. Globally Consistent Indoor Mapping via a Decoupling Rotation and Translation Algorithm Applied to RGB-D Camera Output. *ISPRS International Journal of Geo-Information*, **6**(11), 323.

Lourakis, Manolis, & Zabulis, Xenophon. 2013. Model-based pose estimation for rigid objects. *Pages 83–92 of: International conference on computer vision systems*. Springer.

Lowe, David G. 1992. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, **8**(2), 113–122.

Lowe, David G, *et al.* . 1991. Fitting parameterized three-dimensional models to images. *IEEE transactions on pattern analysis and machine intelligence*, **13**(5), 441–450.

Lwin, Khin Nwe, Myint, Myo, Mukada, Naoki, Yamada, Daiki, Matsuno, Takayuki, Saitou, Kazuhiro, Godou, Waichiro, Sakamoto, Tatsuya, & Minami, Mamoru. 2019. Sea Docking by Dual-eye Pose Estimation with Optimized Genetic Algorithm Parameters. *Journal of Intelligent & Robotic Systems*, 1–22.

Ma, Jason. 2003. Lacks significant S&T investment: Advanced arresting gear will be evolutionary, not revolutionary. *Inside the Navy*, **16**(35), 1–9.

Ma, Yanxin, Guo, Yulan, Zhao, Jian, Lu, Min, Zhang, Jun, & Wan, Jianwei. 2016. Fast and accurate registration of structured point clouds with small overlaps. *Pages 1–9 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Ma, Yi, Soatto, Stefano, Kosecka, Jana, & Sastry, S Shankar. 2012. *An invitation to 3-d vision: from images to geometric models*. Vol. 26. Springer Science & Business Media.

Mardia, Kanti V., & Jupp, Peter E. 2000. *Directional Statistics*. Vol. 494. John Wiley & Sons.

Mardia, KV. 1975. *Characterizations of directional distributions*. Springer. Pages 365–385.

Marjoram, Paul, Molitor, John, Plagnol, Vincent, & Tavaré, Simon. 2003. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, **100**(26), 15324–15328.

Markley, F Landis, & Crassidis, John L. 2014. *Fundamentals of spacecraft attitude determination and control*. Vol. 33. Springer.

Markley, FL, Berman, N, & Shaked, U. 1994. Deterministic EKF-like estimator for spacecraft attitude estimation. *Pages 247–251 of: American Control Conference, 1994*, vol. 1. IEEE.

Martinez, Carol, Campoy, Pascual, Mondragon, Ivan, & Olivares-Mendez, Miguel A. 2009. Trinocular ground system to control UAVs. *Pages 3361–3367 of: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on.* Ieee.

Mayer, Nikolaus, Ilg, Eddy, Hausser, Philip, Fischer, Philipp, Cremers, Daniel, Dosovitskiy, Alexey, & Brox, Thomas. 2016. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. *Pages 4040–4048 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

McLees, Robert E, Hooks, Andrew R, & Panyakeow, Prachya. 2018 (Oct. 2). *Flight control system with low-frequency instrument landing system localizer anomaly detection and method of use.* US Patent App. 10/089,892.

Mejias, Luis, Fitzgerald, Daniel L, Eng, Pillar C, & Xi, Liu. 2009. Forced landing technologies for unmanned aerial vehicles: towards safer operations. *Aerial Vehicles*, 415–442.

Mendoza, Edgar, Prohaska, John, Kempen, Cornelia, Bentley, Douglas, Murdock, Chad, Piatkowski, David, & White, Lonnie. 2007. Smart synthetic material arresting cable based on embedded distributed fiber optic sensors. *Page 66193Z of: Third European Workshop on Optical Fibre Sensors*, vol. 6619. International Society for Optics and Photonics.

Merz, Torsten, Duranti, Simone, & Conte, Gianpaolo. 2006. *Autonomous landing of an unmanned helicopter based on vision and inertial sensing.* Springer. Pages 343–352.

Metni, Najib, Hamel, Tarek, & Derkx, François. 2005. Visual tracking control of aerial robotic systems with adaptive depth estimation. *Pages 6078–6084 of: Proceedings of the 44th IEEE Conference on Decision and Control.* IEEE.

Miller, Andrew, Shah, Mubarak, & Harper, Don. 2008. Landing a UAV on a runway using image registration. *Pages 182–187 of: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on.* IEEE.

Mohammadi, Arash, & Asif, Amir. 2011. Consensus-based distributed unscented particle filter. *Pages 237–240 of: 2011 IEEE Statistical Signal Processing Workshop (SSP).* IEEE.

Mondragón, Iván F, Campoy, Pascual, Martinez, Carol, & Olivares-Méndez, Miguel A. 2010. 3D pose estimation based on planar object tracking for UAVs control. *Pages 35–41 of: 2010 IEEE International Conference on Robotics and Automation.* Ieee.

Moore, R, Thurrowgood, Saul, Bland, Daniel, Soccol, Dean, & Srinivasan, Mandyam V. 2009. A stereo vision system for UAV guidance. *Pages 3386–3391 of: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on.* IEEE.

Morais, F., Ramalho, T., Sinogas, P., Monteiro Marques, M., Pessanha Santos, Nuno, & Lobo, V. 2015. Trajectory and guidance mode for autonomously landing a UAV on a naval platform using a vision approach. *Pages 1–7 of: OCEANS 2015 - Genova.*

Moré, Jorge J. 1978. The Levenberg-Marquardt algorithm: implementation and theory. *Pages 105–116 of: Numerical analysis*. Springer.

Murray, Richard M. 1994. *A mathematical introduction to robotic manipulation*. CRC press.

Myhre, Torstein A, & Egeland, Olav. 2015. Parameter estimation for visual tracking of a spherical pendulum with particle filter. *Pages 116–121 of: 2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE.

Nedjah, Nadia, & de Macedo Mourelle, Luiza. 2006. *Swarm intelligent systems*. Vol. 26. Springer.

Ng, Ka Ki, & Delp, Edward J. 2009. New models for real-time tracking using particle filtering. *Page 72570B of: Visual Communications and Image Processing 2009*, vol. 7257. International Society for Optics and Photonics.

Niezgoda, Stephen R, Magnuson, Eric A, & Glover, Jared. 2016. Symmetrized Bingham distribution for representing texture: parameter estimation with respect to crystal and sample symmetries. *Journal of Applied Crystallography*, **49**(4), 1315–1319.

Oh, H. S., & Vadali, S. R. 1988. Feedback control and steering laws for spacecraft using single gimbal control moment gyros. *Page 3475 of: Guidance, Navigation and Control Conference*.

Okuma, Kenji, Taleghani, Ali, Freitas, Nando de, Little, James J, & Lowe, David G. 2004. A boosted particle filter: Multitarget detection and tracking. *Computer Vision-ECCV 2004*, 28–39.

OpenGL. 2016. Pixel Buffer Object. *URL https://www.opengl.org/wiki/Pixel_Buffer_Object*.

Osher, S, Fedkiw, R, & Piechor, K. 2004. *Level set methods and dynamic implicit surfaces*.

Oshman, Yaakov, & Carmi, Avishy. 2004. Estimating attitude from vector observations using a genetic algorithm-embedded quaternion particle filter. *Page 5340 of: AIAA Guidance, Navigation, and Control Conference and Exhibit*.

Owen, Jamie, Wilkinson, Darren J, & Gillespie, Colin S. 2015. Likelihood free inference for Markov processes: a comparison. *Statistical applications in genetics and molecular biology*, **14**(2), 189–209.

Park, S., Hwang, J., Rou, K., & Kim, E. 2007. A New Particle Filter Inspired by Biological Evolution: Genetic Filter. *International Journal of Electrical, Robotics, Electronics and Communications Engineering*, **9**, 1259 – 1263.

Park, Seongkeun, Hwang, Jae Pil, Kim, Euntai, & Kang, Hyung-Jin. 2009. A new evolutionary particle filter for the prevention of sample impoverishment. *Trans. Evol. Comp*, **13**(4), 801–809.

Pauwels, Karl, Rubio, Leonardo, Diaz, Javier, & Ros, Eduardo. 2013. Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. *Pages 2347–2354 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Pervin, Edward, & Webb, Jon A. 1982. *Quaternions in computer vision and robotics.* Tech. rept. Carnegie-Mellon University of Pittsburgh Department of Computer Science.

Pessanha Santos, Nuno, Melicio, Fernando, Lobo, Victor, & Bernardino, Alexandre. 2014a (September). A Ground-Based Vision System for UAV Pose Estimation. *In: ICIUS 2014: The 10th International Conference on Intelligent Unmanned Systems, Montreal, Quebec, Canada.*

Pessanha Santos, Nuno, Melicio, Fernando, Lobo, Victor, & Bernardino, Alexandre. 2014b. A Ground-Based Vision System for UAV Pose Estimation. *International Journal of Mechatronics and Robotics (IJMR) - UNSYSdigital International Journals*, **1**(4), 7.

Pessanha Santos, Nuno, Lobo, Victor, & Bernardino, Alexandre. 2015. A Ground-Based Vision System for UAV Tracking. *In: OCEANS 2015 - Genova.*

Pessanha Santos, Nuno, Lobo, Victor, & Bernardino, Alexandre. 2017. Particle Filtering based optimization applied to 3D model-based estimation for UAV pose estimation. *In: OCEANS 2017 - Aberdeen.*

Pessanha Santos, Nuno, Lobo, Victor, & Bernardino, Alexandre. 2018. 3D model-based estimation for UAV tracking. *In: OCEANS 2018 - Charleston.*

Pessanha Santos, Nuno, Lobo, Victor, & Bernardino, Alexandre. 2019a. 3D Model-Based UAV Pose Estimation using GPU. *In: OCEANS 2019 - Seattle.*

Pessanha Santos, Nuno, Lobo, Victor, & Bernardino, Alexandre. 2019b. *AUTOLAND* project: Fixed-wing UAV Landing on a Fast Patrol Boat using Computer Vision. *In: OCEANS 2019 - Seattle.*

Pessanha Santos, Nuno, Lobo, Victor, & Bernardino, Alexandre. 2019c. Unmanned Aerial Vehicle tracking using a Particle Filter based approach. *In: 2019 IEEE Underwater Technology (UT) - Kaohsiung.* IEEE.

Pessanha Santos, Nuno, Lobo, Victor, & Bernardino, Alexandre. 2019d. Unscented Particle Filters with Refinement Steps for UAV Pose Tracking. *Journal of Field Robotics (Submitted).*

Pessanha Santos, Nuno, Lobo, Victor, & Bernardino, Alexandre. 2020a. Directional statistics for 3D Model-Based UAV Tracking. *IEEE Access.*

Pessanha Santos, Nuno, Lobo, Victor, & Bernardino, Alexandre. 2020b. Two-stage 3D model-based UAV pose estimation: A comparison of methods for optimization. *Journal of Field Robotics.*

Petrescu, Relly Victoria, Aversa, Raffaella, Akash, Bilal, Berto, Filippo, Apicella, Antonio, & Petrescu, Florian Ion. 2017. Unmanned helicopters. *Journal of Aircraft and Spacecraft Technology*, **1**(4), 241–248.

Pons-Moll, Gerard, & Rosenhahn, Bodo. 2011. Model-based pose estimation. *Pages 139–170 of: Visual analysis of humans.* Springer.

Powell, MJD. 1998. Direct search algorithms for optimization calculations. *Acta numerica*, 287–336.

Prentice, Michael J. 1984. A distribution-free method of interval estimation for unsigned directional data. *Biometrika*, **71**(1), 147–154.

Prince, Simon JD. 2012. *Computer vision: models, learning, and inference*. Cambridge University Press.

Prisacariu, Victor A, & Reid, Ian D. 2012. PWP3D: Real-time segmentation and tracking of 3D objects. *International journal of computer vision*, **98**(3), 335–354.

Pritchard, Jonathan K, Seielstad, Mark T, Perez-Lezaun, Anna, & Feldman, Marcus W. 1999. Population growth of human Y chromosomes: a study of Y chromosome microsatellites. *Molecular biology and evolution*, **16**(12), 1791–1798.

Radke, Richard J. 2013. *Computer vision for visual effects*. Cambridge University Press.

Redmon, Joseph, & Farhadi, Ali. 2016. YOLO9000: Better, Faster, Stronger. *arXiv preprint arXiv:1612.08242*.

Redmon, Joseph, & Farhadi, Ali. 2017. *YOLOv2: Real-Time Object Detection*.

Redmon, Joseph, & Farhadi, Ali. 2018. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*.

Redmon, Joseph, & Farhadi, Ali. 2019. *YOLO: Real-Time Object Detection*.

Reitmayr, Gerhard, & Drummond, Tom. 2006. Going out: robust model-based tracking for outdoor augmented reality. *Pages 109–118 of: Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society.

Ren, Shaoqing, He, Kaiming, Girshick, Ross, & Sun, Jian. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. *Pages 91–99 of: Advances in neural information processing systems*.

Reuter, James D, & Greenstadt, Alan H. 1988 (June 28). *Shipboard air vehicle retrieval apparatus*. US Patent 4,753,400.

Ristic, Branko, & Clark, Daniel. 2012. Particle filter for joint estimation of multi-object dynamic state and multi-sensor bias. *Pages 3877–3880 of: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Ristic, Branko, Arulampalam, Sanjeev, & Gordon, Neil. 2004. Beyond the Kalman filter. *IEEE Aerospace and Electronic Systems Magazine*, **19**(7), 37–38.

Ro, Kapseong, Raghu, Kaushik, & Barlow, Jewel B. 2007. Aerodynamic characteristics of a free-wing tilt-body unmanned aerial vehicle. *Journal of Aircraft*, **44**(5), 1619–1629.

Rogers, Robert M. 2007. *Applied mathematics in integrated navigation systems*. American Institute of Aeronautics and Astronautics.

Ross, Sheldon M. 2010. *Introduction to probability models*. Academic press.

Rosten, Edward, & Drummond, Tom. 2006. Machine learning for high-speed corner detection. *Computer vision–ECCV 2006*, 430–443.

Rosten, Edward, Porter, Reid, & Drummond, Tom. 2010. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **32**(1), 105–119.

Rui, Yong, & Chen, Yunqiang. 2001a. Better proposal distributions: Object tracking using unscented particle filter. *Pages II–II of: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 2. IEEE.

Rui, Yong, & Chen, Yunqiang. 2001b. Better proposal distributions: Object tracking using unscented particle filter. *Pages 786–793 of: CVPR (2)*.

Rumelhart, David E, Hinton, Geoffrey E, & Williams, Ronald J. 1985. *Learning internal representations by error propagation*. Report. California Univ San Diego La Jolla Inst for Cognitive Science.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, & Bernstein, Michael. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, **115**(3), 211–252.

Saini, Sanjay, Bt Awang Rambli, Dayang Rohaya, Zakaria, M Nordin B, & Bt Sulaiman, Suziah. 2014. A review on particle swarm optimization algorithm and its variants to human motion tracking. *Mathematical Problems in Engineering*, **2014**.

Sanders, Jason, & Kandrot, Edward. 2010. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional.

Saripalli, Srikanth. 2009. Vision-based autonomous landing of an helicopter on a moving target. *In: Proceedings of AIAA Guidance, Navigation, and Control Conference, Chicago, USA*.

Saripalli, Srikanth, Montgomery, James F, & Sukhatme, Gaurav. 2002. Vision-based autonomous landing of an unmanned aerial vehicle. *Pages 2799–2804 of: Robotics and automation, 2002. Proceedings. ICRA02. IEEE international conference on*, vol. 3. IEEE.

Saripalli, Srikanth, Montgomery, James F, & Sukhatme, Gaurav. 2003. Visually guided landing of an unmanned aerial vehicle. *Robotics and Automation, IEEE Transactions on*, **19**(3), 371–380.

Sauvola, Jaakko, & Pietikäinen, Matti. 2000. Adaptive document image binarization. *Pattern recognition*, **33**(2), 225–236.

Scaramuzza, Davide, & Siegwart, Roland. 2008. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE transactions on robotics*, **24**(5), 1015–1026.

Seo, Byung-Kuk, & Wuest, Harald. 2016. A direct method for robust model-based 3d object tracking from a monocular rgb image. *Pages 551–562 of: European Conference on Computer Vision*. Springer.

Seo, Byung-Kuk, Park, Jungsik, & Park, Jong-Il. 2011. 3-D visual tracking for mobile augmented reality applications. *Pages 1–4 of: Multimedia and Expo (ICME), 2011 IEEE International Conference on*. IEEE.

Seo, Byung-Kuk, Park, Hanhoon, Park, Jong-Il, Hinterstoisser, Stefan, & Ilic, Slobodan. 2013. Optimal local searching for fast and robust textureless 3D object tracking in highly cluttered backgrounds. *IEEE transactions on visualization and computer graphics*, **20**(1), 99–110.

Sereewattana, Montika, Ruchanurucks, Miti, Rakprayoon, Panjawee, Siddhichai, Supakorn, & Hasegawa, Shoichi. 2015. Automatic landing for fixed-wing UAV using stereo vision with a single camera and an orientation sensor: A concept. *Pages 29–34 of: 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE.

Sermanet, Pierre, Kavukcuoglu, Koray, Chintala, Soumith, & LeCun, Yann. 2013. Pedestrian detection with unsupervised multi-stage feature learning. *Pages 3626–3633 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Seyfang, George R. 2011. Recent developments of the FanWing aircraft. *Pages 1–7 of: The International Conference of the European Aerospace Societies, CEAS*.

Seyfang, George R. 2012. FanWing–Developments and Applications. *Pages 1–9 of: 28th Congress of International Council of the Aeronautical Sciences, ICAS*. Citeseer.

Sharp, Courtney S, Shakernia, Omid, & Sastry, S Shankar. 2001. A vision system for landing an unmanned aerial vehicle. *Pages 1720–1727 of: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2. IEEE.

Shi, Yuhui, & Eberhart, Russell. 1998. A modified particle swarm optimizer. *Pages 69–73 of: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE.

Shi, Yuhui, *et al.* . 2001. Particle swarm optimization: developments, applications and resources. *Pages 81–86 of: Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, vol. 1. IEEE.

Shuster, Malcolm D. 1993. A survey of attitude representations. *The Journal of the Astronautical Sciences*, **41**(4), 439–517.

Shuster, Malcolmd. 1989. A simple Kalman filter and smoother for spacecraft attitude. *Journal of the Astronautical Sciences*, **37**(1), 89–106.

Sigges, Fabian, Baum, Marcus, & Hanebeck, Uwe D. 2017. A likelihood-free particle filter for multi-obiect tracking. *Pages 1–5 of: 2017 20th International Conference on Information Fusion (Fusion)*. IEEE.

Simon, D. 2006. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons.

Simonyan, Karen, & Zisserman, Andrew. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Skrypnyk, Iryna, & Lowe, David G. 2004. Scene modelling, recognition and tracking with invariant image features. *Pages 110–119 of: Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*. IEEE.

Smit, Samuel Jacobus Adriaan. 2013. *Autonomous landing of a fixed-wing unmanned aerial vehicle using differential GPS*. Thesis, Stellenbosch University.

So, Mike KP. 2003. Posterior mode estimation for nonlinear and non-Gaussian state space models. *Statistica Sinica*, 255–274.

Sørbø, Kjetil Hope. 2016. *Autonomous Landing of Fixed-Wing UAV in a Stationary Net-Path and Navigation System*. M.Phil. thesis, NTNU.

Srivatsan, Rangaprasad Arun, Xu, Mengyun, Zevallos, Nicolas, & Choset, Howie. 2017. Bingham distribution-based linear filter for online pose estimation. *In: Robotics: Science and Systems*.

Stachniss, Cyrill, Hähnel, Dirk, Burgard, Wolfram, & Grisetti, Giorgio. 2005. On actively closing loops in grid-based FastSLAM. *Advanced Robotics*, **19**(10), 1059–1079.

Stempfhuber, W, & Buchholz, M. 2011. A precise, low-cost RTK GNSS system for UAV applications. *Proc. of Unmanned Aerial Vehicle in Geomatics, ISPRS*.

Sturm, Peter F, & Maybank, Stephen J. 1999. On plane-based camera calibration: A general algorithm, singularities, applications. *Pages 432–437 of: Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 1. IEEE.

Sugandi, Budi, Kim, Hyoungseop, Tan, Joo Kooi, & Ishikawa, Seiji. 2011. Object Tracking Based on Color Information Employing Particle Filter Algorithm. *In: Object Tracking*. IntechOpen.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, & Rabinovich, Andrew. 2015. Going deeper with convolutions. *Pages 1–9 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Szeliski, Richard. 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media.

Tahri, Omar, & Chaumette, Francois. 2005. Point-based and region-based image moments for visual servoing of planar objects. *IEEE Transactions on Robotics*, **21**(6), 1116–1127.

Taiana, Matteo, Nascimento, Jacinto C, Gaspar, José António, & Bernardino, Alexandre. 2008. Sample-Based 3D Tracking of Colored Objects: A Flexible Architecture. *Pages 1–10 of: BMVC*.

Taiana, Matteo, Santos, Joao, Gaspar, J, Nascimento, J, Bernardino, Alexandre, & Lima, P. 2010. Tracking objects with generic calibrated sensors: An algorithm based on color and 3D shape features. *Robotics and autonomous systems*, **58**(6), 784–795.

Tang, Dengqing, Hu, Tianjiang, Shen, Lincheng, Zhang, Daibing, Kong, Weiwei, & Low, Kin Huat. 2016. Ground stereo vision-based navigation for autonomous take-off and landing of UAVs: a Chan-Vese model approach. *International Journal of Advanced Robotic Systems*, **13**(2), 67.

Teuliere, Celine, Eck, Laurent, Marchand, Eric, & Guenard, Nicolas. 2010. 3D model-based tracking for UAV position control. *Pages 1084–1089 of: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE.

Teuliere, Celine, Marchand, Eric, & Eck, Laurent. 2015. 3-D Model-Based Tracking for UAV Indoor Localization. *Cybernetics, IEEE Transactions on*, **45**(5), 869–879.

Thrun, Sebastian, Burgard, Wolfram, & Fox, Dieter. 2005. *Probabilistic robotics*. MIT press.

Tjaden, Henning, Schwanecke, Ulrich, & Schomer, Elmar. 2017. Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms. *Pages 124–132 of: Proceedings of the IEEE International Conference on Computer Vision*.

Trelea, Ioan Cristian. 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, **85**(6), 317–325.

Tremblay, Jonathan, To, Thang, & Birchfield, Stan. 2018. Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation. *Pages 2038–2041 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Tsai, Chi-Yi, Wang, Wei-Yi, Huang, Chi-Hung, & Shih, Bo-Ren. 2015. Cad modelbased 3d object pose estimation using an edge-based nonlinear model fitting algorithm. *Pages 59–62 of: Proceedings of the 3rd IIAE International Conference on Intelligent Systems and Image Processing*.

Uijlings, Jasper RR, van de Sande, Koen EA, Gevers, Theo, & Smeulders, Arnold WM. 2013. Selective search for object recognition. *International journal of computer vision*, **104**(2), 154–171.

Uosaki, Katsuji, Kimura, Yuuya, & Hatanaka, Toshiharu. 2003. Nonlinear state estimation by evolution strategies based particle filters. *Pages 2102–2109 of: Evolutionary Computation, 2003. CEC03. The 2003 Congress on*, vol. 3. IEEE.

Vacchetti, Luca, Lepetit, Vincent, & Fua, Pascal. 2004a. Combining edge and texture information for real-time accurate 3d camera tracking. *Pages 48–56 of: Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*. IEEE.

Vacchetti, Luca, Lepetit, Vincent, & Fua, Pascal. 2004b. Stable real-time 3d tracking using online and offline information. *IEEE transactions on pattern analysis and machine intelligence*, **26**(10), 1385–1391.

Vaghela, V. B., Ganatra, A., & Thakkar, A. 2009. Boost a Weak Learner to a Strong Learner Using Ensemble System Approach. *Pages 1432–1436 of: Advance Computing Conference, 2009. IACC 2009. IEEE International*.

Valavanis, Kimon P, & Vachtsevanos, George J. 2015. *Handbook of unmanned aerial vehicles*. Springer.

Van Der Merwe, Rudolph, Doucet, Arnaud, De Freitas, Nando, & Wan, Eric. 2001. The unscented particle filter. *Pages 584–590 of: Advances in neural information processing systems*.

VanDyke, Matthew C, Schwartz, Jana L, Hall, Christopher D, *et al.* . 2004. Unscented Kalman filtering for spacecraft attitude state and parameter estimation. *Advances in the Astronautical Sciences*, **118**(1), 217–228.

Varol, Gul, Romero, Javier, Martin, Xavier, Mahmood, Naureen, Black, Michael J, Laptev, Ivan, & Schmid, Cordelia. 2017. Learning from synthetic humans. *Pages 109–117 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Verma, Vandi, Thrun, Sebastian, & Simmons, Reid. 2003. Variable resolution particle filter. *Pages 976–984 of: IJCAI*.

Vermaak, Jaco, Doucet, Arnaud, & Perez, Patrick. 2003. Maintaining Multi-Modality through Mixture Tracking. *Page 1110 of: Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003)*, vol. 2.

Vicente, Pedro, Jamone, Lorenzo, & Bernardino, Alexandre. 2016. Robotic hand pose estimation based on stereo vision and GPU-enabled internal graphical simulation. *Journal of Intelligent and Robotic Systems*, **83**(3-4), 339–358.

Viola, Paul, & Jones, Michael. 2001. Rapid object detection using a boosted cascade of simple features. *Pages 511–518, Vol. 1 of: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE.

Wan, Eric A, & Van Der Merwe, Rudolph. 2000. The unscented Kalman filter for nonlinear estimation. *Pages 153–158 of: Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee.

Wang, Dong, Yang, Fangfang, Tsui, Kwok-Leung, Zhou, Qiang, & Bae, Suk Joo. 2016. Remaining useful life prediction of lithium-ion batteries based on spherical cubature particle filter. *IEEE Transactions on Instrumentation and Measurement*, **65**(6), 1282–1291.

Wang, Shigang, Li, Qian, & Guan, Baiqing. 2007. A computer vision method for measuring angular velocity. *Optics and Lasers in Engineering*, **45**(11), 1037–1048.

Wang, Xiaoyu, Yang, Ming, Zhu, Shenghuo, & Lin, Yuanqing. 2015. Regionlets for generic object detection. *IEEE transactions on pattern analysis and machine intelligence*, **37**(10), 2071–2084.

Wang, Zhengjie, Zhao, Xiaoguang, & Qian, Xu. 2012. Unscented particle filter with systematic resampling localization algorithm based on RSS for mobile wireless sensor networks. *Pages 169–176 of: Mobile Ad-hoc and Sensor Networks (MSN), 2012 Eighth International Conference on*. IEEE.

Wenzel, Karl Engelbert, Masselli, Andreas, & Zell, Andreas. 2011. Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle. *Journal of intelligent and robotic systems*, **61**(1-4), 221–238.

Wild, Graham, Murray, John, & Baxter, Glenn. 2016. Exploring civil drone accidents and incidents to help prevent potential air disasters. *Aerospace*, **3**(3), 22.

Williams, Kevin W. 2004. *A summary of unmanned aircraft accident/incident data: Human factors implications*. Tech. rept. Federal Aviation Administration Civil Aerospace Medical Institute.

Williams, Paul, & Crump, Michael. 2012. Intelligent landing system for landing UAVs at unsurveyed airfields. *In: Proceedings of the 28th International Congress of the Aeronautical Sciences*.

Willmott, Cort J, & Matsuura, Kenji. 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, **30**(1), 79–82.

Wilt, Nicholas. 2013. *The cuda handbook: A comprehensive guide to gpu programming*. Pearson Education.

Winner, Karl, & Kuehn, Benjamin R. 2002 (Oct. 22). *Transponder landing system*. US Patent 6,469,654.

Woo, Mason, Neider, Jackie, Davis, Tom, & Shreiner, Dave. 1999. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc.

Wu, Allen D, Johnson, Eric N, Kaess, Michael, Dellaert, Frank, & Chowdhary, Girish. 2013. Autonomous flight in GPS-denied environments using monocular vision and inertial sensors. *AIAA J. of Aerospace Information Systems (JAIS)*, **10**(4), 14.

Wuest, Harald, Vial, Florent, & Stricker, Didier. 2005. Adaptive line tracking with multiple hypotheses for augmented reality. *Pages 62–69 of: Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society.

Xia, Yu, & Wu, Xiao-Jun. 2015. Adaptive ball particle filter and its application to visual tracking. *IETE Technical Review*, **32**(6), 462–470.

Xia, Yuying, Ajaj, Rafic M, & Friswell, Michael I. 2014. Design and optimisation of composite corrugated skin for a span morphing wing. *Page 0762 of: 22nd AIAA/ASME/AHS Adaptive Structures Conference*.

Xiang, Wenhui, Cao, Yang, & Wang, Zengfu. 2012. Automatic take-off and landing of a quad-rotor flying robot. *Pages 1251–1255 of: Control and Decision Conference (CCDC), 2012 24th Chinese*. IEEE.

Xiaowei, Zhang, Hong, Liu, & Xiaohong, Sun. 2013. Object Tracking with an Evolutionary Particle Filter Based on Self-Adaptive Multi-Features Fusion. *Int J Adv Robotic Sy*, **10**(61).

Xing, Chang, Long, Chengjiang, Guo, Hao, Nie, Yongwei, Zhang, Yuan, Zhu, Dehai, Ma, Qin, & Tian, Mengxiao. 2017. How Does a Camera Look at One 3D CAD Object? *Pages 623–627 of: 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE.

Xu, Guili, Zhang, Yong, Ji, Shengyu, Cheng, Yuehua, & Tian, Yupeng. 2009. Research on computer vision-based for UAV autonomous landing on a ship. *Pattern Recognition Letters*, **30**(6), 600–605.

Xu, Guili, Qi, Xiaopeng, Zeng, Qinghua, Tian, Yupeng, Guo, Ruipeng, & Wang, Biao. 2013. Use of land's cooperative object to estimate UAV's pose for autonomous landing. *Chinese Journal of Aeronautics*, **26**(6), 1498–1505.

Yang, Tao, Li, Guangpo, Li, Jing, Zhang, Yanning, Zhang, Xiaoqiang, Zhang, Zhuoyue, & Li, Zhi. 2016. A ground-based near infrared camera array system for UAV auto-landing in GPS-denied environment. *Sensors*, **16**(9), 1393.

Yang, Zhi-Fang, & Tsai, Wen-Hsiang. 1998. Using parallel line information for vision-based landmark location estimation and an application to automatic helicopter landing. *Robotics and Computer-Integrated Manufacturing*, **14**(4), 297–306.

Yoffe, Meir. 2017 (May 2). *Point take-off and landing of unmanned flying objects*. US Patent 9,637,245.

Zhang, Yangyang, Ji, Chunlin, Malik, Wasim Q, Liu, Yi, O'Brien, Dominic C, & Edwards, David J. 2007. Joint antenna and user selection algorithm for uplink of multiuser mimo systems using sequential monte carlo optimization. *Pages 493–496 of: Statistical Signal Processing, 2007. SSP'07. IEEE/SP 14th Workshop on*. IEEE.

Zhang, Yi, Qiu, Weichao, Chen, Qi, Hu, Xiaolin, & Yuille, Alan. 2016a. UnrealStereo: A Synthetic Dataset for Analyzing Stereo Vision. *arXiv preprint arXiv:1612.04647*.

Zhang, Yudong, Wang, Shuihua, & Ji, Genlin. 2015. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, **2015**.

Zhang, Zichao, Rebecq, Henri, Forster, Christian, & Scaramuzza, Davide. 2016b. Benefit of large field-of-view cameras for visual odometry. *Pages 801–808 of: 2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Zhao, Yun Ji, & Pei, Hai Long. 2013. Improved Vision-Based Algorithm for Unmanned Aerial Vehicles Autonomous Landing. *Applied Mechanics and Materials*, **273**, 560–565.

Zheng, Yuhua, & Meng, Yan. 2007. Adaptive object tracking using particle swarm optimization. *Pages 43–48 of: Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. International Symposium on.* IEEE.

Zheng, Zewei, Jin, Zhenghao, Sun, Liang, & Zhu, Ming. 2017. Adaptive sliding mode relative motion control for autonomous carrier landing of fixed-wing unmanned aerial vehicles. *IEEE Access*, **5**, 5556–5565.

Zhong, Leisheng, & Zhang, Li. 2019. A Robust Monocular 3D Object Tracking Method Combining Statistical and Photometric Constraints. *International Journal of Computer Vision*, **127**(8), 973–992.

Zhong, Leisheng, Lu, Ming, & Zhang, Li. 2018. A direct 3D object tracking method based on dynamic textured model rendering and extended dense feature fields. *IEEE Transactions on Circuits and Systems for Video Technology*, **28**(9), 2302–2315.

Zhou, Dinale, Zhou, Jinglun, Zhang, Maojun, Xiang, Dao, & Zhong, Zhiwei. 2017. Deep learning for unmanned aerial vehicles landing carrier in different conditions. *Pages 469–475 of: 2017 18th International Conference on Advanced Robotics (ICAR).* IEEE.

Zhou, Enlu, & Chen, Xi. 2013. Sequential Monte Carlo simulated annealing. *Journal of Global Optimization*, **55**(1), 101–124.

Zhou, Junchuan, Knedlik, Stefan, & Loffeld, Otmar. 2010. INS/GPS tightly-coupled integration using adaptive unscented particle filter. *The Journal of Navigation*, **63**(3), 491–511.

Zhou, Junchuan, Yang, Yuhong, Zhang, Jieying, & Edwan, Ezzaldeen. 2011. Applying quaternion-based unscented particle filter on INS/GPS with field experiments. *Proceedings of the ION GNSS, Portland*, 1–14.

Zhou, Lipu, Li, Zimo, & Kaess, Michael. 2018. Automatic Extrinsic Calibration of a Camera and a 3D LiDAR using Line and Plane Correspondences. *Pages 5562–5569 of: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE.

Zhu, Zheng, Xia, Yuanqing, & Fu, Mengyin. 2011. Adaptive sliding mode control for attitude stabilization with actuator saturation. *IEEE Transactions on Industrial Electronics*, **58**(10), 4898–4907.