MOHAMED KHIDER UNIVERSITY - BISKRA
FACULTY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING
Ref: ....

جامعة محمد خيضر - بسكرة
كلية العلوم والتكنولوجيا
قسم الهندسة الكهربائية
المرجع: ....

Thesis submitted to the department of electrical engineering in candidacy for the

Degree of Doctorate in Electrical Engineering

Specialty: Automatic Control

Option: Modeling and Control of Dynamic Systems

Control of Nonlinear Systems by Visual Servoing:

Visual object tracking for a quadrotor using image-based visual servoing

Presented by:

DJAMEL EDDINE TOUIL

Discussed publicly : 05/03/2020

In front the jury consists of:

President:	**Pr. Zine-Eddine BAARIR**	**Prof**	**University of Biskra**
Supervisor:	**Pr. Nadjiba TERKI**	**Prof**	**University of Biskra**
Examiner:	**Pr. Abdelkarim OUAFI**	**Prof**	**University of Biskra**
Examiner:	**Pr. Messaoud RAMDANI**	**Prof**	**University of Annaba**
Examiner:	**Dr. Yahia KOURD**	**MCA**	**University of Souk-Ahras**
Examiner:	**Dr. Athmane ZITOUNI**	**MCA**	**University of Biskra**

Dedication

I dedicate this modest work to:

My beloved parents for their encouragement, affection, advice and sacrifice.

My brother and sisters for their precious assistant and encouragement.

All my family.

All my friends and colleagues.

All my teachers.

ACKNOWLEDGEMENTS

First of all Praise be to **ALLAH** with all my heart for giving me the courage and patience that allowed me to accomplish this modest job.

I wish to express my deep thanks to my supervisor **Pr. TERKI Nadjiba** for her support and guidance throughout this work.

My thanks also go to the jury members for having agreed to review and evaluate this work:

Pr. Zine-Eddine BAARIR, professor at the University of Biskra and the jury president.

Pr. Abdelkarim OUAFI, professor at the University of Biskra.

Pr. Messaoud RAMDANI, professor at the University of Annaba.

Dr. Yahia KOURD, professor at the University of Souk-Ahras.

Dr. Athmane ZITOUNI, Professor at the University of Biskra.

I also extend my sincere thanks and appreciation to all the teachers especially **Dr. Medouakh Saadia**, who have supported and guide me in my work with a critical and wise eye.

Finally, I thank my family and my friends for their patience, encouragement, and support which were very useful to me during my work.

Abstract

Visual servoing has undergone major developments in recent years in the field of robotics. Since digital cameras are less expensive, lighter, and can potentially provide the robot equipments with a lot of information, which potentially promotes visual object tracking methods to be very promising for implementation in real-time applications.

Through this thesis, we focus on the development system for a real tracking application basing on a visual servoing approach and of new visual tracking methods. To reach this goal, we draw two essential sub-objectives distributed into two parts.

The first part is dedicated to the presentation of two object tracking methods that can determine the object's trajectory over a sequence of images, whatever its shape. Although many object tracking methods have been developed over the last few years, some difficulties such as occlusion, fast motion, scale variation, and illumination variation can lead these methods to fall into tracking failure cases. We are particularly interested in improving the correlation filter-based method, which is one of the most effective tracking methods for real-time applications thanks to its simplicity. The main principle of these methods relies on the learning of the filter model in the frequency domain by using the Fourier transform to obtain a high precision with a reduced calculation complexity. However, Fourier transformation can cause undesirable effects, which can degrade the tracking quality. This makes these methods unable to distinguish between the target and its background.
In this context, the first proposed method treats this limit by learning models of correlation filters via a convolutional neural network (CNN) features in the spatial domain using the PSO algorithm. In addition, we propose the HSV-based energy condition that has enriched the learning task by combining the RGB and HSV color bases.

The second method learns the filter's model in the frequency domain using the histogram of gradient (HOG) features to enable their implementation in real-time. We treat the drawback of boundaries in the frequency domain by applying a cosine window on feature channels. Furthermore, we adopt a re-detection module to improve the quality of tracking aginst the precited challenging cases The two proposed methods are validated on three reference datasets, OTB-50, VOT 2016 in the case of short-term tracking and UAV20L in the case of long-term tracking.

The second part is dedicated to the design of an image-based controller taking into account both the quadrotor's dynamics and the target's motion during the tracking process, to accurately preserve it in the field of view (FOV). We adopt the concept of image-based visual servoing (IBVS) in which the computation of the control law is achieved by minimizing the error between the desired visual primitives and current primitives extracted from the image information.

Lastly, we demonstrate through a virtual application that the proposed HOG-based tracking method can work reliably using the quadrotor's camera to track two targets of variable sizes and distances, also the efficiency of our system in dealing with occlusion, and scale variation, demonstrating the applicability of the IBVS-HOG system.

Keywords: Visual object tracking, visual servoing, HSV, BA, PSO, IBVS, quadrotor.

RÉSUMÉ

L'asservissement visuel a connu des développements majeurs ces dernières années dans le domaine de la robotique. Étant donné que les appareils photo numériques coûtent moins cher, légers, et peuvent potentiellement fournir aux équipements robotiques beaucoup d'informations, ce qui favorise potentiellement les méthodes de suivi visuel d'objets d'etre très prometteuses pour la mise en œuvre dans des applications en temps réel.

Dans cette thèse, nous nous concentrons sur le développement d'un système de vision destiné à une application de suivi dont se basant sur une approche d'asservissement visuel et sur de nouvelles méthodes de suivi visuel. Pour atteindre cet objectif, nous avons ciblé deux sous-objectifs essentiels répartis en deux parties.

La première partie est consacrée à la présentation de deux méthodes de suivi d'objet permettant de déterminer la trajectoire de l'objet dans le temps dans une séquence d'images, quelle que soit sa forme. Malgré des nombreuses méthodes de suivi d'objet qui ont été développées au cours des dernières années, certaines difficultés telles que l'occlusion, les mouvements rapides, les variations d'échelle et les variations d'éclairage peuvent amener ces méthodes à tomber dans des cas d'échec de suivi. Nous sommes intéressés particulièrement par l'amélioration de la méthode basée sur le filtre de corrélation, qui est l'une des méthodes de suivi les plus efficaces pour les applications en temps réel, et ce grâce à sa simplicité. Le principe de base de ces méthodes repose sur l'apprentissage du modèle de filtre dans le domaine fréquentiel en utilisant la transformée de Fourier pour obtenir une précision élevée avec une complexité de calcul réduite. Cependant, la transformation de Fourier peut provoquer des effets indésirables, susceptibles de dégrader la qualité de suivi, et ce qui empêche ces méthodes de distinguer la cible de son arrière-plan.
Dans ce contexte, la première méthode proposée traite cette limite en apprenant des modèles de filtres de corrélation via un réseau de neurones à convolution (CNN) dans le domaine spatial à l'aide de l'algorithme PSO. De plus, nous proposons la condition d'énergie basée sur le HSV qui a enrichi la tâche d'apprentissage en combinant les bases de couleurs RVB et HSV.

La deuxième méthode apprend le modèle du filtre dans le domaine fréquentiel en utilisant les caractéristiques d'histogramme des gradients orientés (HOG) pour permettre leur implémentation en temps réel. Nous traitons l'inconvénient des limites dans le domaine fréquentiel en appliquant une fenêtre cosinus sur les canaux des caractéristiques. De plus, nous adoptons un module de re-détection pour améliorer la qualité du suivi des cas difficiles précités. Les deux méthodes proposées sont validées sur trois bases de données de référence, OTB-50, VOT 2016 dans le cas du suivi à court terme et l'UAV20L dans le cas du suivi à long terme.

La deuxième partie est consacrée à la conception d'un contrôleur basé sur l'image prenant en considération à la fois la dynamique du quadrotor et le mouvement de la cible pendant le processus de suivi, et ce afin de le conserver avec haute précision dans le champ de vision (FOV). Pour ce fait, nous adoptons le concept d'asservissement visuel à base d'image (IBVS) dans lequel le calcul de la loi de commande est réalisé en minimisant l'erreur entre les primitives visuelles souhaitées et les primitives actuelles extraites de l'image.

En dernier, nous démontrons par une application virtuelle que la méthode de suivi proposée par HOG peut fonctionner d'une manière fiable en utilisant la caméra du quadrotor pour le suivi de deux cibles de tailles et de distances variables, aussi l'efficacité de notre système pour traiter à l'occlusion et la variation d'échelle, en démontrant l'applicabilité du système IBVS-HOG.

Mots clés: Suivi d'objet visuel, asservissement visuel, HSV, BA, PSO, IBVS, quadrotor.

الملخص

يمر موضوع الصيانة البصرية بتطورات كبيرة في السنوات الأخيرة في مجالات الروبوتات، ونظرًا لأن الكاميرات الرقمية يمكن أن تزود الروبوت بالكثير من المعلومات، فهي أقل تكلفة وخفيفة للغاية، ما يعزز طرق تتبع الأشياء المرئية ليكون واعدا للغاية ليطبق في الوقت الحقيقي.

من خلال هذه الأطروحة، نركز على تطوير نظام موجه لتطبيق تتبع حقيقي يستند إلى منهج التحكم المرئي وطرق تتبع بصرية جديدة. لتحقيق هذا الهدف، نرسم هدفين فرعيين أساسيين يتم توزيعهما في جزأين.

يخصص الجزء الأول لتقديم طريقتين لتتبع الكائنات يمكنهما تحديد مسار الكائن بمرور الوقت في سلسلة من الصور، أيا كان شكلها. على الرغم من أنه تم تطوير العديد من طرق تتبع الكائنات على مدار السنوات القليلة الماضية، إلا أن بعض الصعوبات مثل الانسداد، الحركة السريعة، تغيير المقياس وتغيير الإضاءة قد تؤدي إلى سقوط هذه الطرق في فشل التتبع. نحن مهتمون بشكل خاص بتحسين الطريقة القائمة على عامل تصفية الارتباط، والتي تعد واحدة من أكثر طرق التتبع فعالية للتطبيقات في الوقت الفعلي بفضل بساطتها. يعتمد جانب هذه الطرق على معرفة نموذج المرشح في مجال التردد باستخدام تحويل فورييه للحصول على دقة عالية مع تعقيد حساب منخفض. ومع ذلك، يمكن أن يتسبب تحويل فورييه في تأثيرات غير مرغوب فيها، والتي يمكن أن تتسبب في انخفاض جودة نموذج التتبع. هذا يجعل هذه الطرق غير قادرة على التمييز بين الهدف وخلفيته.

في هذا السياق، تتعامل الطريقة الأولى المقترحة مع هذا الحد من خلال تعليم نماذج لمرشحات الارتباط عبر ميزات الشبكة العصبية التلافيفية (CNN) في المجال المكاني باستخدام خوارزمية PSO. بالإضافة إلى ذلك، نقترح أن شرط الطاقة المستندة إلى HSV قد ساعدت في مهمة التعلم من خلال الجمع بين قواعد الألوان RGB وHSV.

الطريقة الثانية تعلم النموذج المرشح في مجال التردد باستخدام الرسم البياني لميزات التدرج (HOG) لاستغلالها في الوقت الحقيقي. نتعامل مع عيوب الحدود في مجال التردد من خلال تطبيق نافذة جيب التمام على قنوات الخصائص. علاوة على ذلك، فإننا نعتمد وحدة إعادة اكتشاف لتحسين جودة التتبع بين الحالات الصعبة المحددة. يتم التحقق من صحة الطريقتين المقترحتين باستخدام ثلاث مجموعات بيانات مرجعية، OTB-50، VOT 2016 في حالة التتبع على المدى القصير وUAV20L في حالة التتبع طويل الأجل.

الجزء الثاني، مخصص لتصميم وحدة تحكم قائمة على الصور مع مراعاة كل من ديناميكيات الطائرة رباعية الدوار وحركة الهدف أثناء عملية التتبع، للحفاظ عليها بدقة في مجال الرؤية. نعتمد مفهوم التحكم البصري المرتكزة على الصور والذي يتحقق فيه حساب قانون التحكم عن طريق تقليل الخطأ بين الخصائص المرئية المرغوبة والخصائص المرئية الحالية المستخرجة من معلومات الصورة.

في الأخير، نوضح من خلال تطبيق افتراضي أن طريقة التتبع المعتمدة على خصائص الرسم البياني المتدرج يمكن أن تعمل بشكل موثوق على الطائرة رباعية المحرك ذات الكاميرا الثابتة لتتبع هدفين بأحجام و مسافات متغيرة، مما يدل على مدى قوة نظامنا في التعامل مع التغطية واختلاف أبعاد الشيء، مما يثبت قابلية تطبيق وفعالية النظام المقترح.

الكلمات المفتاحية: تتبع الكائنات البصرية، التحكم البصري، HSV، BA، PSO، IBVS، رباعي.

Publications in journals

- **D. E. Touil**, N. Terki, and S. Medouakh, "Learning spatially correlation filters based on convolutional features via pso algorithm and two combined color spaces for visual tracking," Applied Intelligence, vol. 48, no. 9, pp. 2837–2846, 2018.
- **D. E. Touil**, N. Terki, and S. Medouakh, "Hierarchical convolutional features for visual tracking via two combined color spaces with svm classifier," Signal, Image and Video Processing, vol. 13, no. 2, pp. 359–368, 2019.

Publications in international conferences

- **D. E. Touil**, N. Terki, and K. Yahia, "Optimal target location estimation using metaheuristic and deep learning algorithms for visual tracking," in Third International Conference on Technological Advances in Electrical Engineering (ICTAEE'18.), December 10-12 2018, Skikda, Algeria.
- **D. E. Touil**, N. Terki, A. Aouina, and R. Ajgou, "Intelligent image-based-visual servoing for quadrotor air vehicle," in 2018 International Conference on Communications and Electrical Engineering (ICCEE). IEEE, pp. 1–7, El-Oued, Algeria.
- **D. E. Touil**, N. Terki, and S. B. Khelil, "Robust visual tracking based on deep learning and optimized correlation filters," in Second International Conference on Electrical Engineering ICEEB'2018, December 2-3 2018, Biskra, Algeria.
- **D. E. Touil**, N. Terki, R. Ajgou, and S. B. Khelil, "Optimum correlation filters for visual tracking via histogram of gradient features and svm classifier," in Third International Conference on Technological Advances in Electrical Engineering (ISTSID'19.), December 10-12 2019, El-Oued, Algeria.

TABLE OF CONTENTS

List of Tables

LIST OF FIGURES

List of Acronyms

BA	Bat Algorithm
CCCT	Colour-aware Complex Cell Tracker
CCOT	Continuous Convolution Operator Tracker
C-HOG	Circular Histograms Of Gradient
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DCF	Discriminative Correlation Filter
DFT	Discrete Fourier Transform
DOF	Degrees Of Freedom
DLT	Deep Learning Tracker
DNT	Dual Deep Network Tracker
DP	Distance Precision
DSST	Discriminative Scale Space Tracker
EAO	Expected average overlap
EBT	Edge Box Tracker
FFT	Fast Fourier Transform
FOV	Field Of View
GPS	Global Positioning System
GPU	Graphics Processing Unit
HCF	Hierarchical Convolutional Features

HALE	High-Altitude Long Endurance
HOG	Histogram Of Gradient
HVS	Hybrid Based Visual Servoing
HSV	Hue Saturation Value
IBVS	Image-Based Visual Servoing
IFFT	Inverse Fast Fourier Transform
IMU	Inertial Measurement Unit
KCF	Kernel Correlation Filter
Le-Net	LeCun Network
LSHT	Tracking via Locality Sensitive Histograms
MAV	Micro Unmanned Air Vehicle
MALE	Medium-Altitude Long-Endurance
MDNET	Multi-Domain Convolutional Neural Networks
MEEM	Multiple Experts using Entropy Minimization
MOSSE	Minimum Output Sum of Squared Error
MUAV	Mini Unmanned Air Vehicle
NAV	Nano Unmanned Air Vehicle
NMF	Non-negative Matrix Factorization
NN	Neural Networks
OPE	One Pass Evaluation
OS	Overlap Success
OTB	Online Tracking Benchmark
PBVS	Position-Based Visual Servoing
PCA	Principal Component Analysis
PD	Proportional-Derived

PTAV	Polytypic Output Sum of Squared Error
PTAV	Parallel Tracking and Verifying
PSO	Particle Swarm Optimization
RAM	Random Access Memory
R-HOG	Rectangular Histograms Of Gradient
RNN	Recurrent Neural Network
RGB	Red Green Blue
Res-Net	Residual Network
RELU	Rectified Linear Unit
SCM	Sparsity-based Collaborative Model
SDK	Software Development Kit
SIFT	Scale-Invariant Feature Transform
SRBT	Salient Region Based Tracker
SRDCF	Learning Spatially Regularized Correlation Filters
Staple	Sum of Template And Pixel-wise LEarners
Struck	Structured Output Tracking with Kernels
SVM	Support Vector Machines
TGPR	Transfer Learning Based Visual Tracking with Gaussian Processes Regression
TLD	Tracking-Learning-Detection
TUAV	Tactical Unmanned Air Vehicle
UAV20L	Unmanned Aerial Vehicle for Long-term tracking
UAV	Unmanned Aerial Vehicle
VGG-Net	Visual Geometry Group-Network
VOT	Visual Object Tracking
VTD	Visual Tracking Decomposition

VTOL	Vertical Take-Off and Landing
ZF-Net	Zeiler-Fergus Network

CHAPTER 1

INTRODUCTION

1.1 Context

Recently, the field of computer vision has known the emergence of great applications exploiting a vision-based system in robotics and autonomous systems. Visual object tracking is one of the most active topics. It deals with several issues such as the object's location estimation in a video. In this context, vision-based systems rely on a visual tracking method to follow any object such as vehicles, pedestrians, cyclists, etc. The diversity of objects makes the diversity of tracking approaches. The common procedure of these approaches uses a priori information of the target such as its initial position, height, and width or the box's coordinates that overlap the object. Then, estimating the following target's positions in the following frames. During the tracking process, the positions of the object are estimated from a certain type of input, including RGB images, RGB videos, camera images, and grayscale images. The output is often a high approximation of the actual positions of the object. Generally, the prior information of the tracked object is not available. This step is performed practically using a detector or manually by a supervisor. Visual tracking approaches use a bounding box for the first frame to create an object's appearance model used to find some similarity between the actual and previous images. In other words, this appearance model is used to separate the tracked object from its background in each input frame. The effectiveness of such visual tracker is based on the adaptive updating of the appearance model during the tracking. The strict issue of visual tracking is reflected in the dynamic variation of the object's appearance in the video, which complicates the modeling of its appearance. Several difficult cases cause this problem, including deformation, fast motion, background clutter, and occlusion. The smart solution is to learn a filter, which combines the object's patch in two successive images. The location of the object is estimated by finding a high correlation score. The conventional method is built using probabilistic technique [22, 23] or an appropriate optimization problem [13, 24]. Although the significant improvement made in recent years, the crucial cases, mentioned above, still limits the reference methods, which stimulates the quick search in the visual tracking topic. In this thesis, we address some of these difficult cases in the context of visual tracking [12, 25–28].

Over the past decade, UAV's commercial use has grown exponentially. thanks to the reduction of sensors, processing costs and the maturation of autopilot technology, which made it easy to possess these vehicles. It led to an increase in the activity in the UAV sector with various applications such as search, rescue, mapping, and parcel delivery. Among all these applications, the functionality that differentiates and the degree of autonomy of the vehicle. The autonomy of a UAV relies on its ability to act or react without the help of human intervention and with a little prior knowledge as possible about the environment in which it operates. The notion of autonomy is therefore an important issue in robotics, because it differentiates between the simplicity and the intelligence of the robot where the environment can change during the work of the robot. In particular, airial vehicles, which have evolved to be progressively or completely controlled, are used to perform certain applications autonomously such as moving, tracking, detecting and, avoiding obstacles. Although the UAV has already found widespread use, they suffered from two major limitations. The first is that they operate outside and rely on operator observation of GPS in addition to other on-board sensors for position control and navigation. A good GPS signal limits the wide and reliable use of quadrotors and it is one of the challenges that researchers are currently trying to solve. To reduce GPS dependence, several works in recent years has been devoted to imposing vision information as a major tool for localization, and even in the quest for the greater autonomy. The reasons of using the camera as visual sensor are double. First, the camera is light, commercial and versatile. Second, many UAVs had already been incorporated with cameras into their main mission, making their use possible as aids to alternative navigation by processing visual data. The second disadvantage is that UAVs have a short flight time, which limits the operational range. One optional solution is to land the drone and reload it on a mobile land robot.

1.2 Problematic

Through this thesis, our problem is divided into two parts to maintain the task of visual tracking of an object in an autonomous way for a quadrotor. The first part studies the improvement of the discriminative correlation filter (DCF) [29] for visual tracking to deal with the major problems of deformation, heavy occlusions, fast motion, scale and illumination variations, that limit most advanced tracking methods. Also, making it impossible to distinguish between two similar objects and/or between the target and its background since the tracked object may appear with low-quality images, which provide the missing information.

Unlike conventional correlation filter-based methods, we aim to improve the accuracy of correlation filter modeling to perform robustly the object's location estimation. Indeed, we focus on improving two main steps: feature extraction and correlation filter modeling. For the aim of performing the ability to detect the target object in the acquired image in the face of constraints already mentioned. Indeed, the use of visual information to develop robust tracking method is a scientific challenge. For this pupose, we propose two tracking methods to robustly accomplish the desired application.

In the second part, we seek to take a step towards a greater quadrotor autonomy by studying the possibility of performing object tracking automatically from visual information provided from a camera. We study the adoption of the 2-D approach of visual servoing, which aims at minimizing the error between the coordinates of the observed image's features and the desired one [30, 31].

1.3 Contributions

In the first part, we mainly study the improvement of the work of Ma et al [12]. They learn three discriminant correlation filter (DCF) models with the deep features extracted from the convolutional neural network (CNN) to precisely locate the tracked object. Their contribution is the combination between the first and the last layers of CNN. Fast Fourier Transform (FFT) is exploited in the learning part to reduce the computation time [13, 32]. However, they neglected the fact that the Fourier domain caused undesirable effects, which gradually deteriorates the tracking quality. Our main contributions are summarized as follows:

- We propose to use the particle swarm optimization algorithm (PSO) [33, 34] to learn three correlation filter models in the spatial domain using features extracted from three corresponding CNN's layers. We show that it is effective in handling the challenges of abrupt motion, deformation, and occlusion.

- We switch between two color spaces, RGB and HSV, through the HSV-energy condition for each RGB input frame, the idea is to classify each frame into high-level illumination or low-level illumination. Then, we choose which color base is more suitable to learn the DCF models. We propose also a second tracking method for a real-time application. We use the histogram of oriented gradient (HOG) features to learn two DCF models for both tasks of object position estimation and scale estimation.

- We evaluate our methods on three reference datasets of OTB-50 [35], VOT 2016 [3] in the short-term tracking and UAV20L [2] in the long-term tracking.

In the second part, we propose the image-based visual controller that stabilizes the altitude, attitude, and position of the quadrotor in space. The controller is to reduce the error between the desired and the current vector values of the image feature to generate the appropriate speed to the drone controllers. In this context, the Proportional-Derived (PD) linear controller is chosen to control the position of the quadrotor in space. Indeed, we use the bat algorithm (BA) [36] to optimize the PDs' gains to optimally control the position of the quadrotor in space.

Several simulations are performed in MATLAB, in which a vertical take-off and landing type (VTOL) unmanned aerial vehicle (UAV), called quadrotor, is implemented. The results obtained show that the use of the BA algorithm offers great flexibility in tracking the trajectory of the drone. It also guarantees the stability for performing the trajectory tracking for a quadrotor, which allows the vision-based controller to achieve the optimal desired positions in minimal time.

Finally, a real-time object tracking application is tested to prove the efficiency of the proposed HOG-based tracking method.

1.4 Structure

This thesis is organized as follows:

In Chapter 2, we present the state-of-the-art of visual tracking. First, we introduce previous works, and then we discuss the tracking attributes and their effects on visual tracking. Besides, we will summarize the most used databases in the literature. We also give all performance terms used to judge such a visual tracking method.

Chapter 3 presents in detail the main steps of our methods. Starting with feature extraction, then modeling correlation filters. Finally, the results and discussions in each database are given.

In Chapter 4, we give a detailed overview of the state-of-the-art of visual servoing. First of all, we introduce the previous works, as well as the fundamental notions, the basic tools necessary for understanding our works. Then, we will give a brief overview of the quadrotor type UAVs, as well as the different approaches of visual servoing used on this system.

In chapter 5, we present the model of the quadrotor used, as well as the bat algorithm used to optimize the parameters of the designed PD's control laws.

Chapter 6 presents a validation of the image-based control by adopting the object tracking technique based on the HOG-based method presented in part I.

The final chapter presents a conclusion on our work and a vision for future work.

Part I

Visual Object Tracking

Chapter 2

State-of-the-art

2.1 Introduction

Visual object tracking has been widely studied and implemented in several applications. The majority of the reference methods can not handle challenging cases including significant occlusion, deformation, fast motion, scale and illumination variations, which affect the tracking performance. In this chapter, we present a detailed study of visual object tracking. First, we discuss in Section 2.2 the methods of object tracking. This section mainly introduces a study of the three classes of visual tracking methods, namely discriminative, generative and hybrid methods. Next, we present the most used databases, including their characteristics, attributes, and evaluation metrics. We introduce in Section 2.4 the theory of discriminative correlation filters (DCF), which bases on the formulation of Bolme et al [29]. We also introduce in Section 2.5 the theory of convolutional neural networks (CNN) and the histogram of oriented gradient (HOG). We begin with CNN's architecture and we present the commonly used architectures of CNN. Then, the formulation of the HOG features is presented in detail.

2.2 Visual bject tracking methods

Due to the quick appearance of several visual tracking methods, each method relies on the main aspect such as detectors, classifiers, and segmentation. In [37], they study block-matching and classifiers to categorize visual tracking methods. As shown in figure 2.1, the visual object tracking problem can be categorized according to the number of tracked targets.

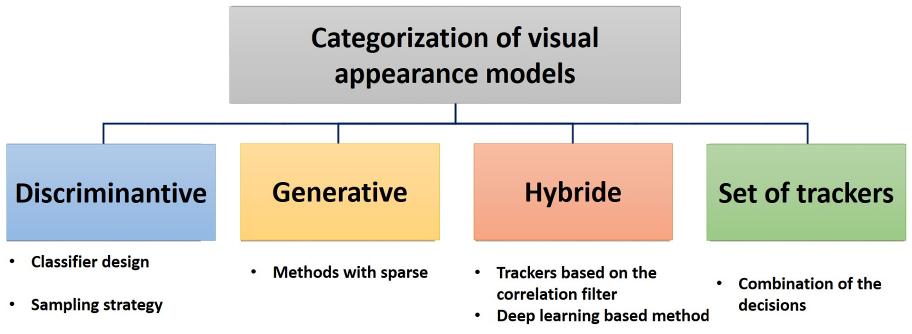

Categorization of visual appearance models
Discriminantive
Generative
Hybride
Set of trackers
Classifier design
Sampling strategy
Methods with sparse
Trackers based on the correlation filter
Deep learning based method
Combination of the decisions

Figure 2.1: Categories of visual tracking methods

Most of the visual tracking methods are divided into two groups: **Set of trackers** and **individual trackers**. The first group combines several trackers, while the second group relies on only one, which means that only one algorithm can search the object in the current image without limitation, basing on the correlation filter, the optical flow, and the block matching. The **set of trackers** group performs the object detection task by associating the extracted patches of the object with its appearance template [38]. Other studies [39, 40] show that the object's appearance model is the appropriate tool for classifying tracking methods. It allows distinguishing the target from its background. In this context, we discuss the categories of visual tracking methods based on the appearance model tool.

2.2.1 Discriminative methods

This category of methods aim to form an efficient classifier for ensuring an accurate separation between the target and its background. They exploit a conditional probability on the candidate samples extracted from the appearance representation of the object patch. In other words, the conditional probability that represents the confidence of the classifier allows estimating the appropriate candidate sample of the tracked object in the current frame [38]. On the other hand, this task requires a high-level representation of the object in the large feature space such as HOG [10, 41], SIFT [42]. Generally, the discriminative methods require a large dataset in order to achieve good performances. Several studies propose many tools to represent the object's appearance to learn robustly the classifier model, in which the common idea is to annotate the positive samples and the negative samples [43, 44]. Next, they look for the highest confidence score to locate the object, as shown in figure 2.2.

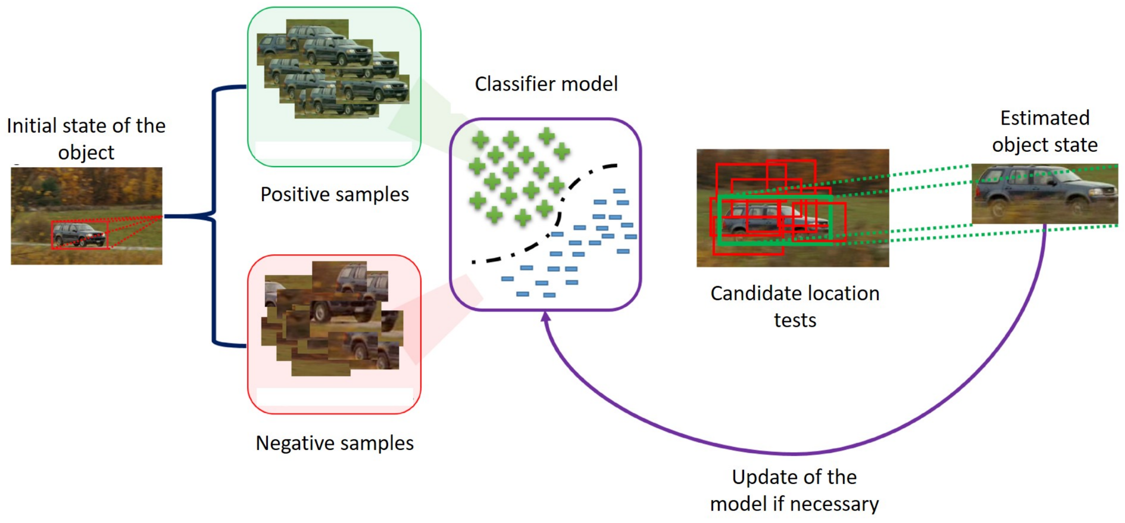


Figure 2.2: Illustration of discriminative visual tracking system.

In this context, Avidan et al [43], determine the position of the target using the mean shift algorithm. They proposed to train online an ensemble of weak classifiers to annotate each region of interest. Then, they fused them to a strong classifier to label pixels belonging to the object in the next frame. They even propose an online methodology for updating the classifier models. The best classifiers are trained on the next image, while the others are updated on all images. Bai et al [45], propose a Bayesian framework, where they exploit the posterior distribution to design weak classifiers. They extract features robustly to maintain the annotation of the used samples. In [46], Zhang et al adopted the bag of positive samples for learning the classifier via an online discriminative feature selection algorithm, so they outperform the algorithm proposed in [47], in terms of computation accuracy. In addition, Zhang et al [48] have focused on computational efficiency by adopting a compression theory for detection. The idea is to compress large features in a small, randomly selected space. They based on Haar features, as it is suitable for managing motion blur and rotating appearance changes. Other works avoid the combination and selection of weak classifiers. In [49], they improve the quality of tracking by proposing a partially structured model of SVM to model the unknown object's parts. Also, an SVM-based system is presented by Ning et al [50]. They present an effective dual linear SVM-based algorithm to enable fast learning and execution during tracking, it accurately processes large features, which allow achieving maximum real-time computational efficiency. Since deep learning provides high-level representation, recent studies have adopted it in many computer vision tasks such as the classification [51], object detection [52], and saliency detection [53]. Jin et al [54] learned a CNN model on the current and previous samples. The translation of the object is estimated by using a gaussian mask function. In [55], Hong et al adopted features extracted from a pre-formed CNN to learn an SVM model. They estimate the position of the target using a saliency map.

2.2.2 Generative methods

Generative methods study the appearance model of the tracked object to estimate its location in the current frame. It describes the object, even its background. These methods select appropriate instances to estimate the object's position. The selection of the appropriate instances is done by using various tools such as a particle filter framework. Figure 2.3 illustrates the generative visual tracking framework [38]. Like the discriminative methods, the generative methods exploit the image's features to represent the target object, such as the intensity of the image In general, they do not require a large dataset for training. Updating the appearance model of the object makes the difference between generative and discriminative tracking methods, where the discriminative methods update the appearance model when the location of the object is estimated, while the generative methods ignore this step when some cases of significant appearance changes have occurred as occlusion or deformation.

The optimization problem proposed in [56] is effectively improved by the optimization of the proximal gradient proposed in [57] in terms of real-time implementation. The idea is to converge the gradient rapidly using the summations of a smooth function and a non-differential function. Other trackers update the older appearance models with a new one in the current image. However, this idea remains limited compared to such variation of the appearance model. Xing et al [58] solve this problem by using short, medium, and long lifetimes to design the appearance model of the object. This strategy learns the sparse coefficients robustly and assures the accuracy of the appearance modeling. Another similar idea to the sparse constraint is adopted in the Non-negative Matrix Factorization (NMF) for the visual tracking [59]. It relies on the reconstruction

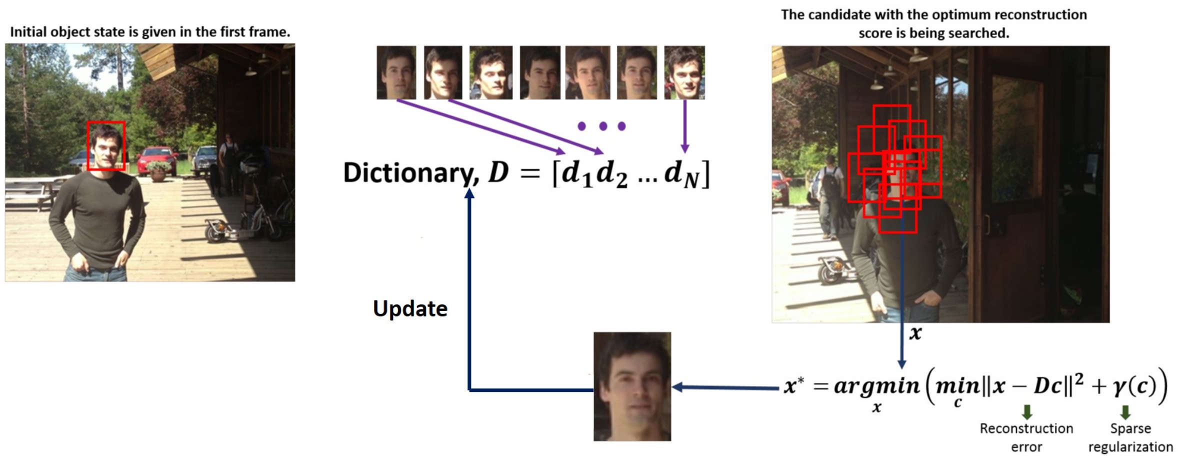

Initial object state is given in the first frame.
Dictionary, $D = [d_1 d_2 \dots d_N]$
Update
The candidate with the optimum reconstruction score is being searched.
x
$x^* = \underset{x}{argmin}\left(\underset{c}{min}\|x - Dc\|^2 + \gamma(c)\right)$
Reconstruction error
Sparse regularization

Figure 2.3: illustration of a generative visual tracking system.

of non-negative constraint coefficients to readjust the elements of the dictionary and focus on the parts of the tracked object. Nevertheless, the NMF algorithm is still unexplored in a real-time tracking system. Following these works, Zhang et al [60] forced the obtained sparse coefficients for different samples. They used some dictionary models to represent all the appearance models. The potential limitation in this approach is that a tracking failure can occur if the object model is misrouted. In [61], they adopted a CNN to extract features in the context of sparse optimization. Similar to [57], they use the $L1$ standard normalized cost function to reconstruct the foreground object model.

2.2.3 Hybrid methods

This category of method combines both discriminative and generative methods, and even exploits the two appearance models used in these two methods. In this subsection, we present the methods classified in the hybrid category.

2.2.3.1 Correlation filters based methods

These methods aim to find the appropriate model used to identify objects in a simplified scenario. It is based on the adaptive filter theory, in which the correlation operation is adopted between the candidate samples to obtain **the correlation map**. This map is then processed to identify the correlation score according to a peak-shaped signal. The high score is considered as the desired answer. The appropriate model is then considered as a correlation filter model, which gives an effective correlation score even for a uniform shift of the target. This leads to consider that correlation filter methods base on discriminant regressor models. Figure 2.4 illustrates the main steps of the correlation filter-based framework [38].

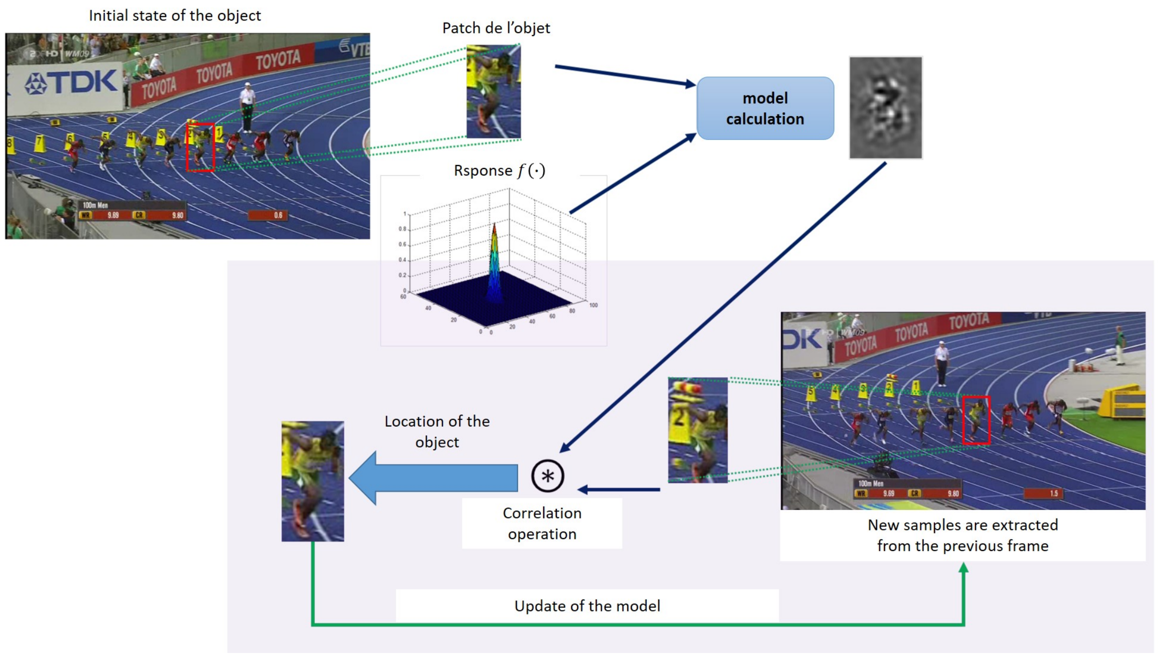


Figure 2.4: A representative illustration of a correlation filter-based visual tracking framework.

During the tracking process, the correlation operation requires the multiplication $\mathcal{O}(N^2)$ (N is the length of the signal), so that the circular correlation is adopted to handle the correlation operations for the following reasons: First, the Fourier transformation, which requires the multiplication $\mathcal{O}(N)$, which is an appropriate tool to perform the convolution operation as an element-by-element multiplication in the frequency domain [62]. Second, the correlation theory is based on a minimization problem between a correlation filter model of the candidate patch and the desired response, which is usually chosen as a gaussian function. Figure 2.4 illustrates perfectly that the accurate correlation response corresponds to the highest score of the appropriate object location.

The first work based on the correlation filters was done by Mahalanobis et al [63] and Réfrégier et al [64]. They learn a set of correlation filter models by a set of samples. However, this idea requires a single correlation result for a centered filter. The following works such as in [29] handle this disadvantage. They proposed the Minimum Output Sum of Squared Error (MOSSE). The idea is to design a discriminant correlation filter using the sum of squared errors to minimize a cost function. This minimization problem requires a correlation response of the filter model and the desired correlation response for each training example. Other works propose to use another type of information provided in the form of raw image intensity, such as, but not limited to, outlines, oriented gradients, and color channels. In [13], they adopted the correlation filter model designed in [29] for a linear regression problem. They use a multi-channel histogram of oriented gradient (HOG) in the form of a kernel correlation filter (KCF).

Zuo et al [65] increase the efficiency of multiple channels by integrating the support vector machine (SVM) framework. Until now, the adoption of multi-channel filters was intended to estimate the target location. However, several studies have adopted them in scale estimation. In [66], they estimate the appropriate scale value using an iterative optimization method. In addition, Danelljan et al proposed the Discriminative Scale Space Tracker (DSST) [5]. This method exploits multiple channels and the correlation response to maintain the multiscreen search in the frequency domain. In [67], they used a multi-scale search to estimate the best target scale based on the best correlation response. Ma et al [12] proposed to learn three correlation filters using hierarchical features. Their main contributions are summarized in two key points. First, they combine hierarchically the layers of a preformed CNN model. They note that the first layers are used to encode the approximation of the object, while the last ones allow encoding its details. Second, they used a coarse-to-fine strategy to estimate the location of the object. Other methods deal with the weakest problem of the correlation filter, which is represented by the boundary problem. Danelljan et al [68] have proposed to learn a model of correlation filter by minimizing a cost function in the spatial domain to penalize the resulting limits. They proposed also an adaptive decontamination system [69]. The idea is to learn the model of the correlation filter with the weights of the learning samples in order to eliminate the unsuitable scales. As a result, they outperform the tracker presented in [68]. Besides, they presented the Continuous Convolution Operator Tracker (CCOT) [24], which aims to learn models of correlation filters using convolutional layers to make them more compatible with different feature's sizes in the spatial domain. Their formulation surpasses other methods in the VOT 2016 database [3].

2.2.3.2 Methods based on deep learning

In recent years, several methods based on deep learning have emerged. These methods adopt the deep features in several visual tracking tasks. In [70], they exploited the Siamese network, which is a model of a neural network. The output gives similar features for different samples. The tracking mechanism is as follows. They extracted the feature vector from the learned model, then compared it to the candidate appearances of the object to find the similarity between them. The disadvantage of this method is that the evaluation of these candidates' models is computationally expensive. Also, Bertinetto et al [27] designed a CNN model with the same convolutional layers. They estimated the correlation between the appearance model and the candidate sample using a sliding window approach. Among the neural networks-based methods, the most widespread is the recurrent neural network (RNN) [71]. The work of Cui et al exploited the RNN to spatially model the relationship between the target and its background [72]. This effectively estimates the target's confidence map. In [44, 73], they used the RNN to spatially learn the relations between the input images. The goal is to estimate the target location directly. However, they did not achieve the performance required for the visual object tracking reference databases VOT [3, 35, 74] and OTB [4, 75].

2.2.4 Set of trackers

As discussed in section 2.1, there is a second type of tracking method, which is based on a combination of trackers. For example, Tokola et al [76] worked on detecting target location by exploiting the POSSE tracker, which is a generalization of the MOSSE tracker. In addition, Li et al [23] exploited the KCF tracker [13] for visual tracking. They specifically employ the response map of correlation filters to facilitate the trackable score function. In [14], they proposed to construct an expert ensemble, where the best expert is selected to restore tracker in the case of undesirable model updates using an entropy minimization (MEEM) for visual tracking. This method used an SVM classifier to select the appropriate expert. There are also hybrid methods [60, 77], which combined generative and discriminant trackers.

2.3 Databases and attributes

2.3.1 Attributes

To analyze any visual object tracker, it is inevitable to take into account and understand the challenges that can lead them to drift. It depends on the consecutive frames in which the object may undergo a strict appearance change. In the literature, these appearance changes occur due to several crucial problems, called **attributes** [7]. These attributes are presented as follows:

- **Background clutter:** This case can occur when the features of the object and its background are similar, which means that any small change in the appearance of the object may cause the tracker to consider that the background is much like the target than the target itself.

- **Deformation:** This case occurs when all parts of the tracked object move and rotate depending on each other, where the tracker considered them as a single rigid part.

- **Fast motion:** This is another critical case of visual object tracking. It presents a fast or a large movement of the object and/or the camera between two consecutive input images. This will lead the tracker to drift.

- **Occlusion:** It is a potentially difficult case. It may appear in any consecutive frames when part of the target or its complete parts is occluded by its background or other objects. In this case, the update of the appearance model will necessarily affect the position's estimation of the target in the consecutive input frames.

- **Blur motion:** It is the occurrence of a mess stained in the appearance of the object caused by fast motion.

- **Scale variation:** This case often occurs due to the close or distant movement of the camera from the target. Resizing the input frame is the basic tool used in visual tracking to address this problem.
- **Illumination variation:** The tracked object can be affected directly or indirectly by the illuminations of the material or the environment. This problem occurs when the lights are flashing or when there are moving projectors, which certainly affects the efficiency of extracting image features. Several methods deal with this problem by using invariant features to light variations [10, 41].
- **In-plane-rotation:** Videos can show rotation that occurs in a two-dimensional image plane. The best-known example is seeing the side of a motorcycle driver who has rotated backward.
- **Out-plane-rotation:** It differs from the rotation in the plane, which may be out of the image plane. It could lead to the disappearance of certain parts of the target.
- **Out of view:** Many tracking methods fail to recover the target as it moves in the border of the image. In this case, the target can also disappear or move in the video.
- **Low resolution:** A low resolution is another critical attribute, which is summarized as a reduction of the information provided from the target's patch, thus reducing the accuracy of the location estimation task. In general, this is due to the use of a low-resolution camera or the considerable distance that separates it from the object.

2.3.2 Databases

In recent years, several applications have appeared in the subject of visual tracking. They were evaluated using subjective assessments or designed metrics to check the tracker's performance. In order to manage these unfair subjective assessments, many studies had introduced complex databases, such as OTB-50 [4], VOT 2016 [3] and UAV20L [2], as shown in figure 2.5. The attributes discussed above are included in these databases. Our proposed methods are evaluated on these three databases.

- **OTB-50**: The OTB-50 (OTB2013) database is the most commonly used database in the literature, which contains 50 fully annotated sequences that are collected from commonly used tracking sequences. The OTB sequences are annotated sequences according to previously discussed attributes. Through this dataset, the evaluation and analysis of the strength and weakness of tracking approaches are performed with two aspects [4]:
  - **Precision plot**: It is defined as the average Euclidean distance between the center locations of the tracked targets and the manually labeled ground truth. It shows the

percentage of frames whose estimated location is within the given threshold distance (threshold = 20 pixels) of the ground truth.

$$\triangle_\mu(\Lambda^G, \Lambda^P) = \frac{1}{N}\sum_{t=1}^{N} \|x_t^G - x_t^P\| \tag{2.1}$$

- **Success plot**: Another evaluation metric is the bounding box overlap. Given the tracked bounding box r_t and the ground truth bounding box r_a, the overlap score is defined as:

$$\Phi = \frac{|r_t \cap r_a|}{|r_t \cup r_a|} \tag{2.2}$$

  where $\cap$ and $\cup$ represent the intersection and union of two regions, respectively, and $|\cdot|$ denotes the number of pixels in the region. The number of successful frames is counted if the overlap Φ is larger than the given threshold of $t_o = 0.5$ for each frame.

- **UAV20L**: As its name indicates, the aim through this dataset is to capture videos from low-altitude UAVs, which is inherently different from video in popular tracking datasets. It includes 20 sequences meant for long-term aerial tracking. All sequences are fully annotated with the 12 discussed attributes. The performance evaluation of tracking approaches is performed via the same metrics in the OTB-50 dataset [2].

- **VOT 2016**: In 2013 the visual object tracking initiative was established to address the performance evaluation for short-term visual object trackers. It consists of 60 challenging videos that are automatically selected from the OTB, ALOV ++, PTR databases [3]. The VOT workshops proposed to evaluate the performance of any method using the following aspects [78]:

  - **Overlap (Accuracy)**: The accuracy metric measures the intersection over union (IoU) between the predicted bounding box and the ground truth bounding box for a particular sequence.
  - **Failure (Robustness)**: The robustness metric measures the number of failures when there is no overlap between the predicted bounding box and the ground truth bounding box for a particular sequence. The trackers can be ranked according to the robustness metric for each sequence. The overall ranking can be found by averaging the ranks for all the sequences.
  - **Expected Average Overlap (EAO)**: A single metric that measures both the accuracy and robustness of a tracker is the expected average overlap (EAO). Let Θ_i be the overlap between the ground truth bounding box and the predicted bounding box of the i^{th} frame of a sequence. Let the sequence under experimentation be a sequence of length N_s. The average overlap of this sequence of length N_s can be calculated as:

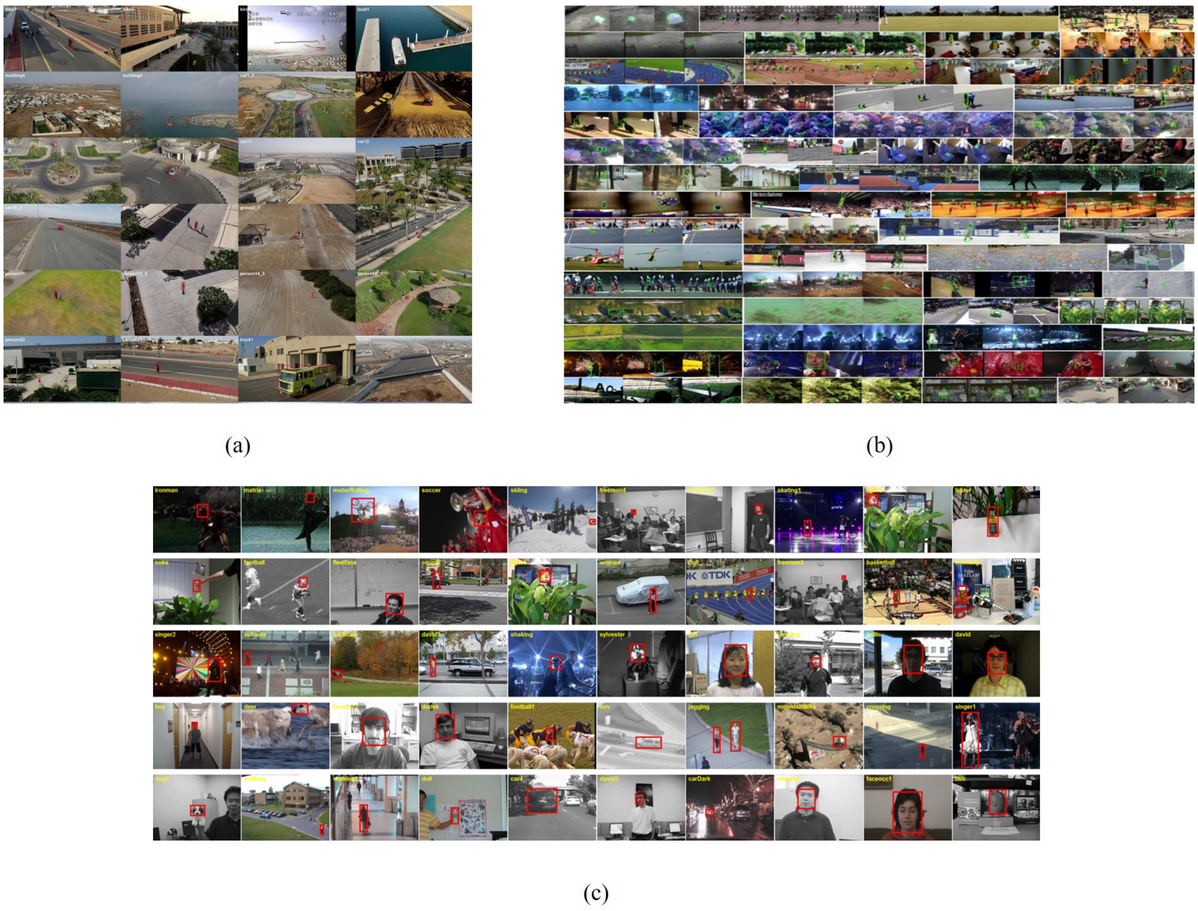


Figure 2.5: Databases (a) UVA20L [2], (b) VOT 2016 [3] and (c) OTB-50 [4].

$$\Theta_{N_s} = \frac{1}{N_s}\sum_{i=1}^{N_s}\Theta_i \tag{2.3}$$

The average of the above quantity can be found by:

$$\hat{\Theta}_{N_s} = \langle\Theta_{N_s}\rangle \tag{2.4}$$

The expected average overlap metric can be calculated by calculating the average of Θ_{N_s} over typical sequence lengths, from N_{lo} to N_{hi}:

$$\Theta = \frac{1}{N_{hi} - N_{lo}}\sum_{N_{lo}}^{N_{hi}}\hat{\Theta}_{N_s} \tag{2.5}$$

2.4 Correlation filters

In this part of the thesis, the main goal is to improve the learning phase of the correlation filter models. In this section, we present an overview of the MOSSE method [29]. In the literature, this method is considered as one of the reference methods that has adopted a unique linear correlation filter formulation.

2.4.1 MOSSE tracker

During visual tracking, the area of interest may be a specified object, an interesting point, or the entire image itself is called **sample**. The set N of training samples is designated by x_i where $x_i = \{x_1, x_2, ..., x_N\}$, each x_i represents a rectangular region with a width of M_1 and a height of M_2. Generally, each sample x_i is a function $x_i = \{0, ..., M_1\} \times \{0, ..., M_2\} \rightarrow \mathbf{R}$. The MOSSE tracker aims to find a filter that maintains the relation $x_i \circledast h \approx y_i$, and it is given as a cost function presented in equation 2.6.

$$h_{opt} = \underset{h}{\arg\min} \sum_{i=1}^{N} ||h \circledast x_i - y_i||^2 \tag{2.6}$$

where $\{y_i\}_i^N$ denotes the desired response, which represents the intensities of the region of interest. Typically, y_i is defined by a sampled Gaussian function with a narrow peak centered on the target object, as shown in figure 2.6. The index $\circledast$ indicates the correlation operation that results from the cyclically shifting of the original patch's features x_i and the previously learned model h. H_{opt} is the new learned filter model. We note that the correlation and convolution given by equations 2.7 and 2.8 are recalled to solve this linear regression problem, especially in the frequency domain.

$$c[n] = \sum a[m]b[n-m] = \mathscr{F}^{-1}\{A \odot B\} \tag{2.7}$$

$$c[n] = \sum a[m]b[n+m] = \mathscr{F}^{-1}\{A^* \odot B\} \tag{2.8}$$

Where $a[.]$ and $b[.]$ present the unidimensional discrete signals, $a[n-m]$ indicates the circular version of $a[n]$ with the delay m to the right. The symbol $*$ denotes the complex conjugate. The $\odot$ symbol indicates a elementwise multiplication, while $\mathscr{F}$ denotes the discrete Fourier transform (DFT). All capital letters designate frequency domain signals, while lowercase letters represent spacial domain signals. The extension of the theorem to the two-dimensional case is simple because the DFT operation has a separable two-dimensional definition. It is assumed that any derivation for the one-dimensional signal can be extended to two-dimensional signals if the operations can be split into two dimensions [38].

Using these properties, the correlation problem given by equation 2.6 is written as follows:

$$H_{opt} = \underset{H}{\arg\min} \sum_{i=1}^{N} ||H \odot X_i - Y_i||^2 \tag{2.9}$$

For simplifying the optimization of each element of H_{opt} separately, we can write:

$$\mathscr{L} = \sum_{i=1}^{N} H^* \odot H \odot X_i^* \odot X_i + Y_i^* \odot Y_i - H^* \odot X_i \odot Y_i^* - H \odot X_i^* \odot Y_i \tag{2.10}$$

Where H and H^* are assumed to be independent. The derivative of this function with respect to H^* for all its elements, we obtain:

$$\frac{\partial \mathscr{L}}{\partial H^*} = \sum_{i=1}^{N} H \odot X_i^* \odot X_i - X_i \odot Y_i^* \tag{2.11}$$

If $\frac{\partial \mathscr{L}}{\partial H^*}$ is equal to zero, the derivative of equation 2.10 with respect to H^* gives:

$$H = \frac{\sum_{i=1}^{N} X_i \odot Y_i^*}{\sum_{i=1}^{N} X_i^* \odot X_i} \tag{2.12}$$

The obtained model H maintains the minimization of the cost function in the frequency domain. As noted in section (2.2.3), the Fourier transform is the appropriate tool to perform convolution (elementwise multiplication) because the Fast Fourier Transform (FFT) has a complexity of $\mathscr{O}(Mlog(M))$, and this requires a signal of a complexity of $\mathscr{O}(M^2log(M))$. Also, this tool simplifies the operations of the matrix $M^2 \times M^2$ (inversion and multiplication) since it is practical as a solution of the linear least-squares method. During the tracking process, the estimation of the object's position is made at each input frame to locate the object in the next frame. At the instant $(t+1)$, the correlation response map f_{t+1} is computed by multiplying element by element the obtained filter model H_t at instant t with the sample Z extracted from the object patch at the instant $(t+1)$, as:

$$f_{t+1} = \mathscr{F}^{-1}\{H_t^* \odot Z_{t+1}\} \tag{2.13}$$

This response map is important for determining the new object's location, which depends on the coordinates of the pixel and that has the highest value. Indeed, these coordinates are then considered as the shift of the center between the consecutive frames at the instances t and $t+1$.

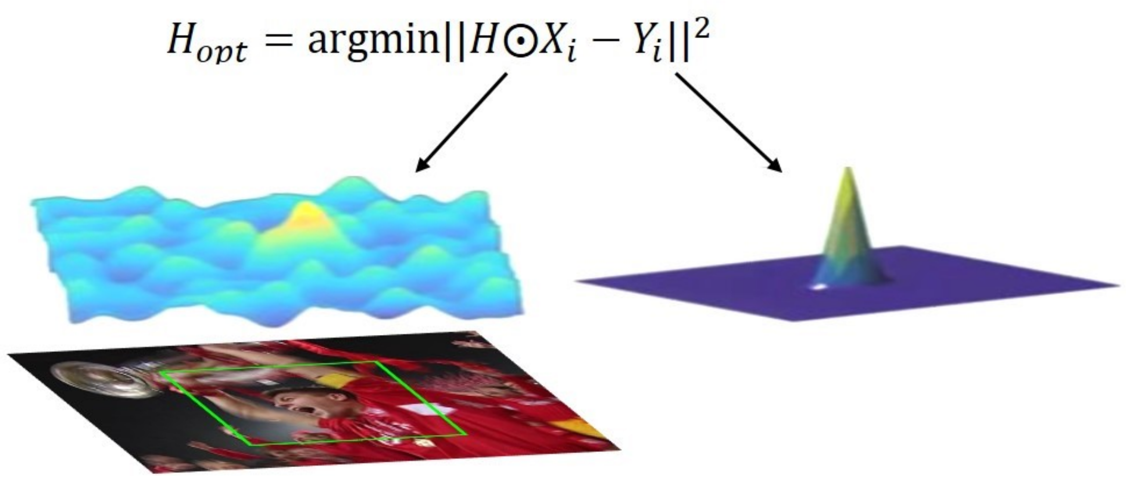


Figure 2.6: The region of the image corresponding to the patch's features x_i in MOSSE (left, green zone). Each pixel in the region is given a desired score y_i (right) [5].

Since the location of the tracked object is located in each frame, the efficiency of the filter model obtained from equation 2.13 must be protected by updating it. This main step is performed using a learning rate of γ. The role of this parameter lies in learning the numerator and denominator of the model H at the instant $t+1$ with a conserved percentage of the preceding filter model H_t, as follows:

$$\begin{aligned} A_{t+1} &= (1-\gamma)A_t + \gamma(X_{t+1} \odot Y^*_{t+1}), \\ B_{t+1} &= (1-\gamma)B_t + \gamma(X_{t+1} \odot X^*_{t+1}) \end{aligned} \tag{2.14}$$

So, we can write

$$H_{t+1} = \frac{A_{t+1}}{B_{t+1}} \tag{2.15}$$

where A_{t+1} and B_{t+1} denote the numerator and de-numerator of H_{t+1}. X_{t+1} and Y_{t+1} refer to the samples, and the desired response map respectively, at the instance t [29].

2.5 Image features

In the computer vision field, a feature extraction is an important tool. It is a special representation of the area of interest in the input image. The extraction of features is considered as a transformation into another image, obviously, with other values. The concatenation of these features in a vector gives the so-called **the feature map**. It consists of the sampled entities in a grid map of R^{abc}. Where a and b denote the number of pixels in the rows and columns and c indicates the depth of that map. In the literature, several types of features proved its effectiveness for the visual object tracking. In this thesis, we have chosen to adopt the CNN [79] and HOG [10, 80] for feature extraction phase.

2.5.1 Convolutional neural networks

In recent years, we have witnessed the emergence of deep learning-based methods, including CNN-based methods [11]. They have shown that CNN's features are very useful for capturing the semantic and detailed features of target object compared to other useful features on a wide range of visual recognition tasks [80]. Similar to the ordinary Neural Networks (NN) architecture, a CNN is a sequence of layers, where their type is specified in advance, but whose parameters are formed using a large dataset [7].

2.5.1.1 Architecture

There are several pre-formed architectures such as Le-Net [81], Alex-Net [51], ZF-Net [82], Google-Net [83], Res-Net [84] and VGG-Net [9]. These architectures are inspired by the architecture of neural networks, which is composed of neurons that have weights and biases that can be learned. Each neuron is fully connected to all neurons of the previous layer, and each one produces a scalar product (multiplication element by element) followed by a non-linear activation operation. Each set of these neurons builds a layer. [7, 85]. The neural network architecture receives a single input vector and transforms it through a series of hidden layers. The last layer is the output of the network and represents the classification scores.

The CNN architectures mentioned above take advantage of the fact that the input consists of images, which can process them more judiciously. Indeed, the difference between CNNs and neural network architecture is based on the shape of the layers, which are arranged in three dimensions height, width and depth in CNN's architectures. Note that the word depth here refers to the third dimension of each layer, and not to the depth of the entire architecture. Figure 2.7 shows that the convolutional neural network organizes its neurons in three-dimension layers (width, height, depth). Each layer transforms the 3-D input volume into a 3-D output volume of neurons. The most common known layers are summarized as follows [6]:

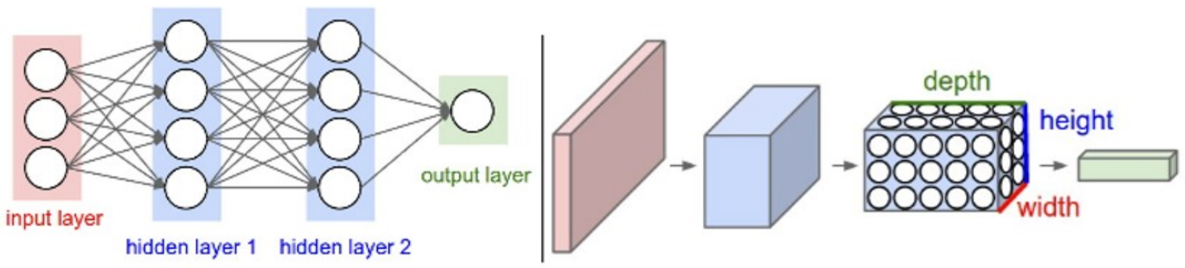


Figure 2.7: Left: a network of neurons with 3 layers. Right: A convolutional neural network organizes its neurons in three dimensions (width, height, depth), visualized for each layer. The red input layer presents the input image. The width and height correspond to the dimensions of the image and the three-channel red, green and blue [6].

- **Input layer:** It contains the pixel values of an RGB image.
- **Convolutional layer:** This layer calculates the output of neurons connected to local regions, using a scalar product between their weights and this region. A prefixed number of convolution filters of fairly small size, such as 3×3, are multiplied by the input respectively. The output is a volume of size $M \times N \times D$.
- **RELU layer:** In this layer, an activation function $max(0,x)$, with a threshold of zero, is applied to the resulting volume of a convolutional layer in which the size of the output is unchanged.
- **Pooling Layer:** A subsampling operation is performed on the width and height of the input volume, which reduces its output size. As shown in figure 2.8, the neighborhood of each pixel in the input image is limited by the maximum of that area. The values in this

area are replaced by the corresponding maximum value. The subsampling factor is usually set to 2.

- **Fully connected layer:** This layer is usually considered as the last layer where its components correspond to the probability that the input image belongs to a predefined class. These values are obtained with the function $softmax$, defined as:

$$f(x_j) = e^{x_j} / \sum_j e^{x_j} \tag{2.16}$$

The output is a volume of size $[1 \times 1 \times D]$, where D is the number of classes.

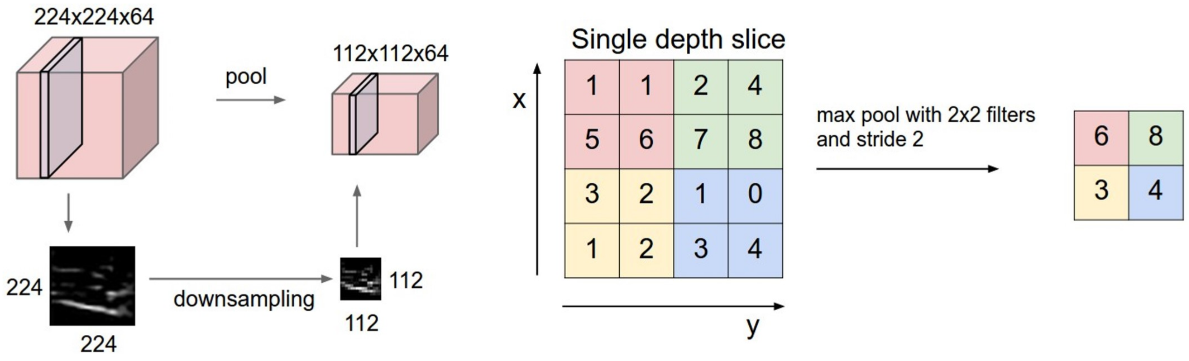


Figure 2.8: Illustration of max-pooling operation of an input image of size 224×224. The input image is to the left, the output image is to the right [7].

2.5.1.2 Local connectivity

As noted above, CNN is intended to process large inputs as images and each neuron is connected to a local region of the input volume. The extent of this connectivity is defined using a hyper-parameter called the neuron receptor field [6]. The depth of this connectivity always depends on the depth of the input and the connections are local over the width and height. In this context, figure 2.9 illustrates an interesting example. For an input image with the size of $32 \times 32 \times 3$ and if the receptor field (filter size) is equal to 5, each neuron of the convolution layer will have weights corresponding to a local region $[5 \times 5 \times 3]$ of the input volume. This means 75 weights and 1 bias. The extent of connectivity along the depth axis should be 3 since the depth of the input is 3.

2.5.1.3 Space planning

To know how the neurons are arranged at the exit of the layers, we have to discuss the three parameters that control the arrangement of the output volume [6].

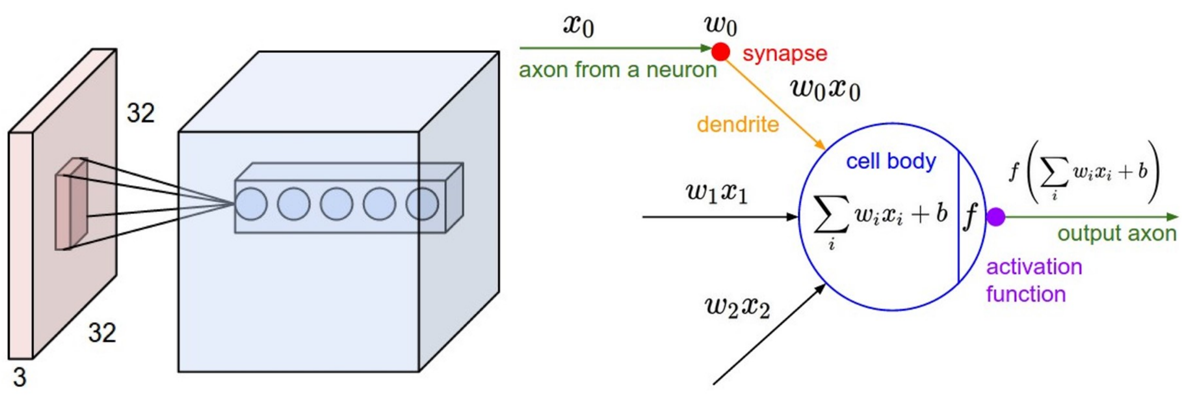


Figure 2.9: An example of convolutional layers [8].

- **The depth of the output volume:** It corresponds to the number of used filters.
- **Stride:** When the stride is equal to 1, we shift the filters one pixel at a time. When the stride is 2 (or rarely 3 or more, although this is rare in practice), we drag them 2 pixels at a time. This will produce a smaller output volume.
- **Zero-padding:** This parameter refers to fill the input volume with zeros around its border.

The nice property of this parameter is that it allows controlling the spatial size of the output volumes. The spatial size of the output volume is calculated based on the size of the input volume (W), the size of the receiver field (F), the stride (S) and the amount of zero (P). In other words, the convolution layer accepts a volume of size $W_1 \times H_1 \times D_1$ and uses the parameters: number of filters (K), the spatial extent (F), the stride (S) and the padding (P) to produces a volume of size $W_2 \times H_2 \times D_2$, where:

$$W_2 = (W_1 - F + 2P)/S + 1 \tag{2.17}$$

$$H_2 = (H_1 - F + 2P)/S + 1 \tag{2.18}$$

$$D_2 = K \tag{2.19}$$

With parameter sharing, it introduces weights of $F \times F \times D_1$ per filter, for a total of $F \times F \times D_1 \times K$ and B bias. In the output volume, the d^{th} slice (of size $W_2 \times H_2$) results from the execution of a valid convolution of the d^{th} filter on the input volume with a stride S, then offset by d^{th} bias. Figure 2.10 completely clarifies the convolutional layer spatial arrangement, where the input volume is blue, the used filters are is in red, and the output volume is in green. Each output element is calculated by multiplying element-wise the input with the filter, assembling it, and then shifting the result through the bias. Note that the convolution operation is adopted as an element-by-element product between the local region of the input and the filter. Backward propagation for a convolution operation (for both data and weights) is also a convolution operation.

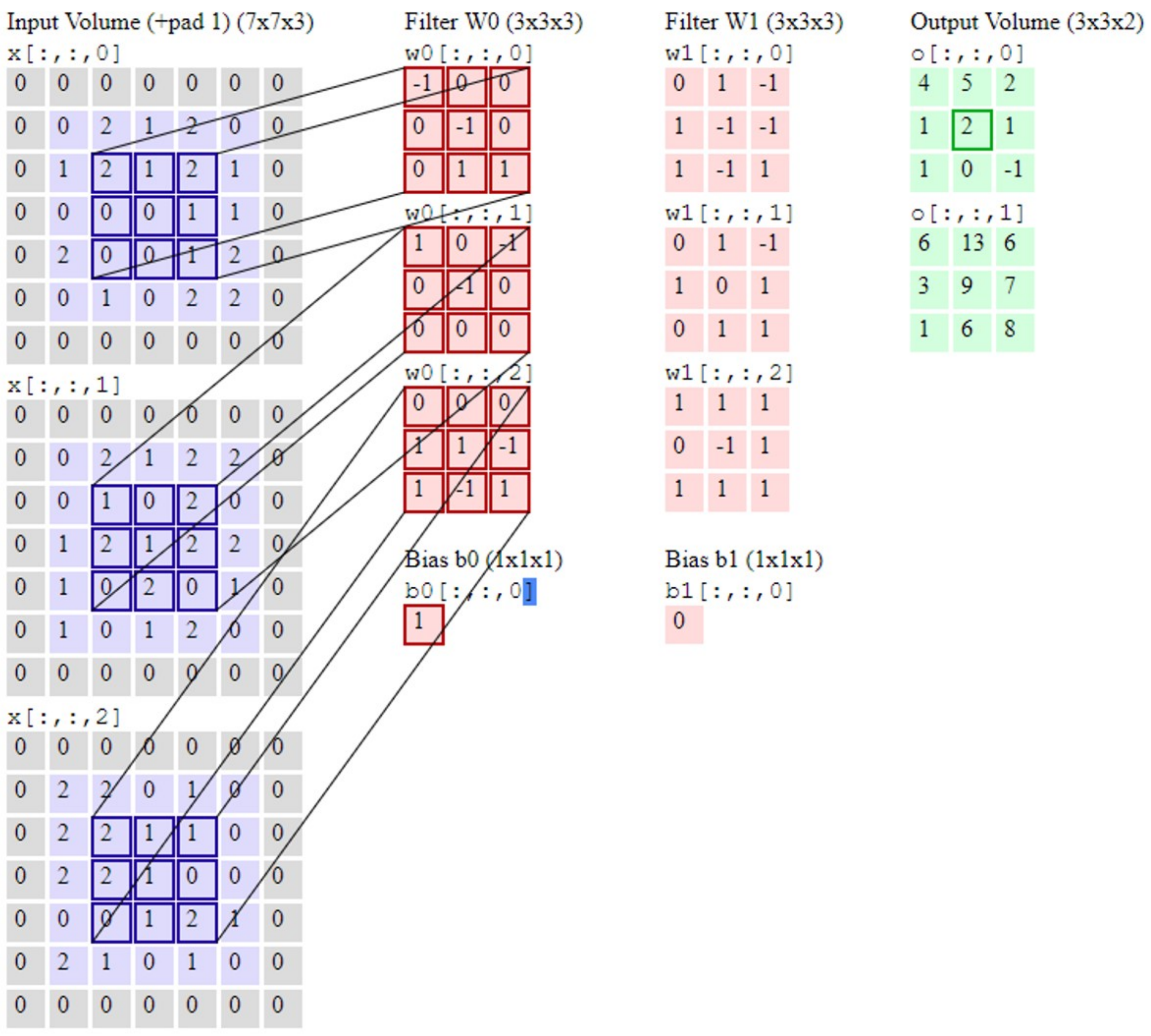


Figure 2.10: The input volume is $W_1 = 5$, $H_1 = 5$, $D_1 = 3$ and the convolutional layer parameters are $K = 2$, $F = 3$, $S = 2$, $P = 1$.Therefore, the size of the output volume has a size $(5-3+2)/2+1 = 3$. Also, note that a fill of $P = 1$ is applied to the input volume, making the outside edge of the input volume zero [6].

Regarding the pooling layer, it operates independently on each slice of the input using the MAX operation. In general, it is useful to insert a pooling layer between the convolution layers to reduce the number of parameters and calculations in the network. The pooling layer accepts a volume of size $W_1 \times H_1 \times D_1$ and only requires the parameters F, S to produce a volume of size $W_2 \times H_2 \times D_2$ where:

$$W_2 = (W_1 - F)/S + 1 \tag{2.20}$$

$$H_2 = (H_1 - F)/S + 1 \tag{2.21}$$

$$D_2 = D_1 \tag{2.22}$$

In addition to maximum pooling, other pooling functions can be used such as the average pooling or even the $L2$ standard pooling. Average pooling has often been used, but it has recently fallen out of favor compared to maximum pooling operation, which has proven its efficiency. In the fully connected layer, the neurons have a complete connection to all the neurons of the previous layer, as in neural networks.

In our work, we chose the VGG-19 [9] architecture shown in figure 2.11 for the following reasons:

- It worked well for both image classification and object detection compared to other reference architectures [12, 86].
- Available for Matlab users.
- The ability to modify this architecture to reduce the number of sharing parameters without degrading the performance. Most of these parameters are in the fully connected layers, and it has been found that these layers can be removed.
- Reference trackers have recently shown that the use of a convolution layer output as a feature map gives a significant performance gains [12].

During the visual tracking process, the input image is introduced into a pre-formed template for VGG-19. The output after different convolutional layers will have a different amount of function channels, which usually decreases the resolution depending on the number of function channels [7].

2.5.2 Histogram of oriented gradients

In the context of image recognition, several features had shown its ability to represent the area of interest. In which, the edge is the most represented as a feature. It can be obtained by using the dominant gradient amplitude on the entire image. In [41], they use the gradient of histogram (HOG) for pedestrian detection. They noted that the HOG features are highly efficient for describing the pedestrian movements where they give a high vertical position approximation, and they proved that this descriptor is particularly suitable for human detection [87]. The advantage of this technique is that effectively describes the appearance and shapes of the entire image using gradient intensity distribution. This latter is first obtained by dividing the image into small contiguous regions called cells, of a size typically equal to $8\times$ 8, then collecting of gradient directions of each cell to construct the histogram. To improve the detection accuracy with respect to the variation of the illumination and the shading, a local contrast normalization can be applied by calculating the intensities in a larger area, called a block, as shown in figure 2.12.

The extraction of HOG features is summarized below [10, 87].

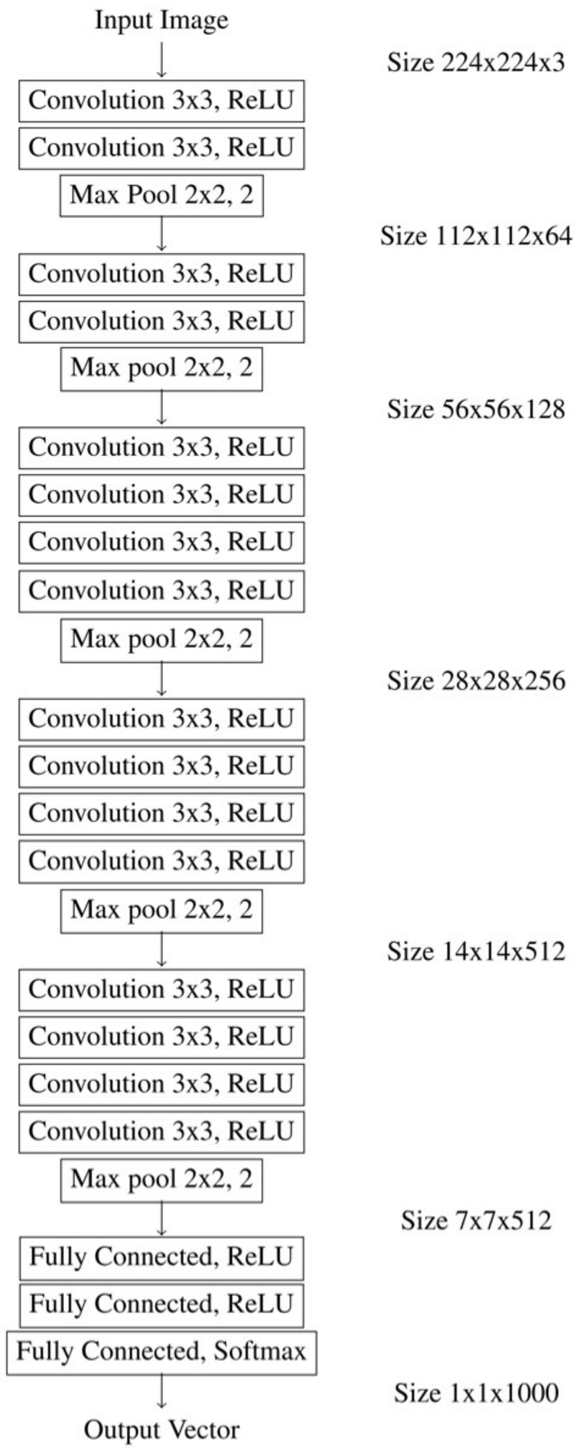


Figure 2.11: Illustration of the VGG-19 architecture. On the left are the types of operations performed. Right, the intermediate data sizes. A $224 \times 224 \times 3$ size is used as RGB input image. The output is a vector of length 1000 [9].

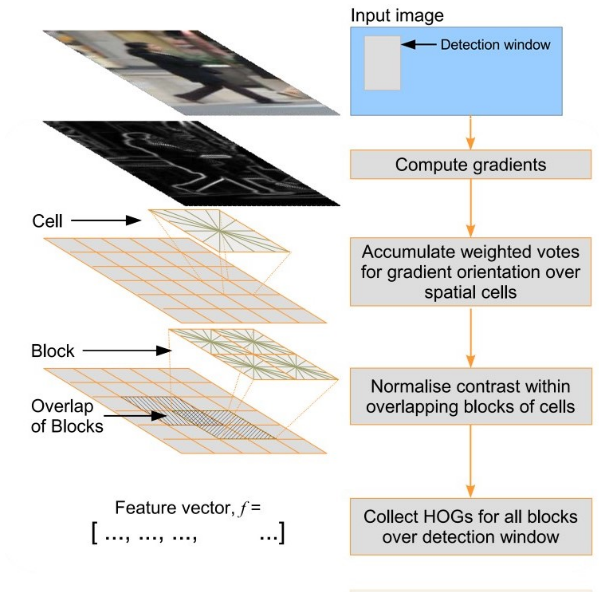

Input image
Detection window
Compute gradients
Accumulate weighted votes for gradient orientation over spatial cells
Normalise contrast within overlapping blocks of cells
Collect HOGs for all blocks over detection window
Cell
Block
Overlap of Blocks
Feature vector, f =
[..., ..., ..., ...]

Figure 2.12: An overview of HOG feature extraction [10].

- **Gradient calculation** This step is performed by applying the 1-D discrete derivative mask in the horizontal and vertical directions on each centered point, which has the following form:

$$[-1,0,1],[-1,0,1]^T$$

The magnitude and orientation at each pixel $I(x;y)$ is calculated by:

$$G_{mag}(x,y) = \sqrt{G_x^2(x,y) + G_y^2(x,y)} \tag{2.23}$$

$$\theta(x,y) = arctan(G_y(x,y)/G_x(x,y)) + \pi/2 \tag{2.24}$$

where $G_x(x,y)$ and $G_y(x,y)$ are the gradient values at each pixel in the horizontal and vertical directions, respectively. For color images, the channel with the largest magnitude gives the dominant magnitude and orientation of the pixel. It should be noted that the value $\pi/2$ is needed because the $arctan$ operator gives a range between $-\pi/2$ and $\pi/2$, but for an unsigned orientation scheme that gives better performance, it is between 0 and π.

- **Orientation Binning:** In this procedure, the histograms for each cell are created. The cells are rectangular or radial pixel regions and the boxes in the histogram are regularly extended from 0° to 180° (or from 0° to 360° in the case of a signed orientation). Each pixel of the cell generates a weighted vote in one of the 9 boxes of the histogram to which its orientation belongs. As regards the weight of the votes, it can be either the magnitude of the gradient itself or a function of the magnitude, for example, the square root or the square of the magnitude of the gradient. Generally, the gradient amplitude is used.

- **Bloc normalization:** There are three different ways to normalize blocks. Let v be the non-standard feature vector that collects all the cell histograms of a given block, $||v||_k$ denotes its $k-norm$ for $k = 1,2$ and ϵ a constant. Then, normalization schemes have the following forms:

$$\hat{v} = \frac{v}{\sqrt{||v||_2^2 + \epsilon^2}} \tag{2.25}$$

$$\hat{v} = \frac{v}{||v||_1 + \epsilon} \tag{2.26}$$

$$\hat{v} = \sqrt{\frac{v}{||v||_1 + \epsilon}} \tag{2.27}$$

Also, an L2-norm followed by clipping (limiting the maximum values of v to 0.2) then normalized by a Laplacian normalization [42]. All normalization schemes offer better performance than non-standard cases. The final HOG feature descriptor is then the vector containing the elements of normalized cell histograms from all block regions.

2.6 Conclusion

In this chapter, the visual object tracking state-of-the-art is presented based on the recent advances in the literature. We presented the categorization of visual tracking methods and the critical cases related to the advantages and disadvantages of all methods. Besides, we presented the most commonly used databases as well as its built-in challenge attributes and the different evaluation terms used to evaluate such a method in these databases. We also detail the formulation of the reference method based on the correlation filter, since the purpose of this part is to manage the attributes mentioned and to improve the modeling of the correlation filter, to maintain object localization and scale estimation tasks.

CHAPTER 3

PROPOSED METHODS FOR VISUAL OBJECT TRACKING

3.1 Introduction

This chapter introduces two visual object tracking methods. The proposed methods address the problems previously discussed methods such as fast motion, illumination variation, occlusion, scale variation, ...etc.

The proposed methods are decomposed into two tasks: the translation and scale estimations of the target object. Concerning the first method, it adopts the CNN features to adaptively learn three correlation filter models in the spatial domain via the PSO algorithm, which helps to reliably estimate the location of the target object in each frame. We propose a condition based on the HSV color base to enrich the feature extraction phase. Regarding the scale estimation task, HOG features are adopted to exhaustively search for the optimal scale. The second proposed method adopts the HOG features in both tasks. Furthermore, we adopt a re-detection module basing on the SVM classifier to improve the quality of tracking. We evaluate both methods on three reference databases OTB-50 [35], VOT 2016 [3] and UAV20L [2].

3.2 Patch processing

3.2.1 HSV-Energy condition

This section presents an innovative idea to solve the problem of illumination variation that most reference trackers can not handle it. The technique consists of using the components' energy of the HSV color space. The use of the energy concept has been widely exploited in various applications such as wireless sensor networks [88], image reconstruction [89] and many others. For each input RGB frame, the energy consumption of each HSV's component is used to classify each RGB frame into two classes: low light and high light. The first class includes the frames that present a low energy consumption and low light changes, while the second class includes the frames with high-energy's consumption and significant changes in lighting. As shown in figure 3.1, the HSV color base is then used to calculate the energy consumption of each input frame. The energy is generally ranked by equation 3.1 based on two thresholds, TH_1 and TH_2, where they are empirically selected [11]. This phase must be done for RGB input images, while it is ignored for grayscale images.

$$\begin{array}{l} if \quad (E_2-(E_1+E_3)>TH_1 \quad \cap \quad E_3<TH_2 \quad then \quad HSV(frame) \\ \quad else \qquad RGB(frame) \end{array} \tag{3.1}$$

With E_k represents the energy of the k^{th} component. The percentage of energy consumption of each HSV component is defined by:

$$\begin{array}{c} E_k = 100 \times \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(B_{kij})^2}{E_T} \qquad , \qquad k = 1,2,3 \\ E_T = \sum_{i=1}^{m}\sum_{j=1}^{n}((B_{1ij})^2+(B_{2ij})^2+(B_{3ij})^2) \end{array} \tag{3.2}$$

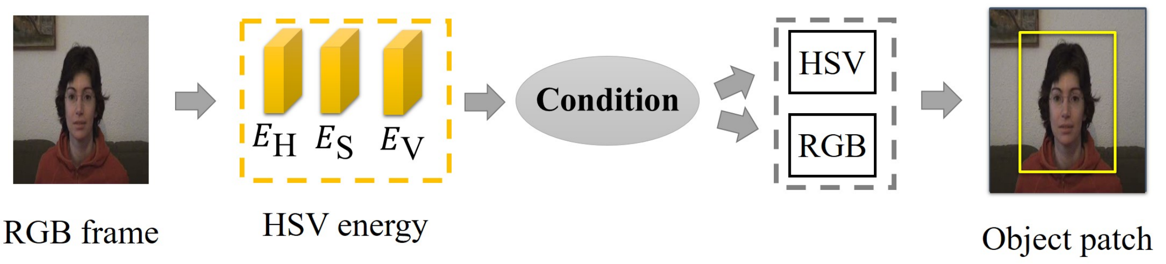


Figure 3.1: HSV-Energy condition.

Where B_k denotes the k^{th} component of the HSV frame and E_T refers to the total energy. Due to the effectiveness of the HSV color database [90], energy-intensive images are transformed into HSV color databases, while low energy ones remain in RGB base.

To illustrate the effectiveness of this idea, figure 3.2 shows the efficiency of the energy condition of HSV. It can be seen that without the use of the energy condition, the tracker can not recover the target object shown in the images 34^{th} and 35^{th} of the sequence *Singer 2*. On the other hand, it works robustly, in this case, using the energy condition, as it effectively manages the variation of energy in each input frame. In this context, according to the table 3.1, it is visible that the illumination varies between the 30^{th} and 40^{th} images, the energy's percentage of the brightness varying from 15.90 % to 26.15%. Concerning the 35^{th} and 36^{th} frames, the energy of the HSV components (H) and (V) are at the maximum value. In addition, the absolute difference between the energy of the component (H) and that of the component (S) is minimal compared to that of the other images, which means that for a maximum energy component (V), we have a favorable color separation in the component (H) with an appropriate color intensity for the component (S) [11].

Table 3.1: Illustration of the energy percentage of the three HSV components from the frame 30 to the 40^{th} of the sequence *Singer 2*.

Frame	E_H	E_S	E_V
30	12.90	71.89	15.20
31	11.75	73.01	15.22
32	12.15	72.35	15.49
33	11.86	72.27	15.86
34	10.72	73.36	15.90
35	41.92	34.33	23.73
36	44.81	29.03	26.15
37	12.87	70.82	16.29
38	14.77	68.86	16.35
39	14.12	69.71	16.16
40	15.02	68.08	16.88

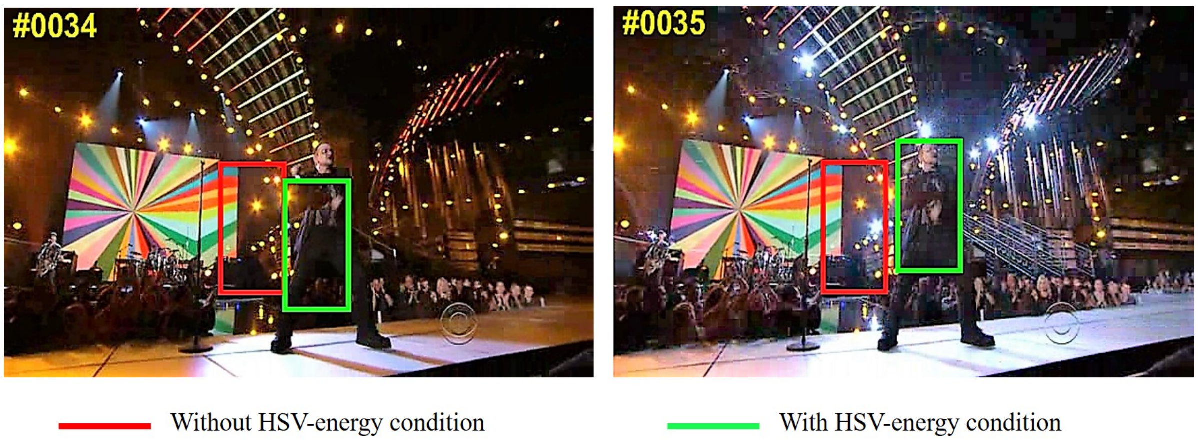


Figure 3.2: The results obtained from the 34^{th} and 35^{th} frames of the *Singer 2* sequence. It is clear that the energy condition maintains the long-term tracking of the target's appearance and improves the handling of unexpected background illumination changes [11].

3.2.2 Object patch extraction

The object patch extraction is one of the most important phases, which necessarily affects the efficiency of extracting contextual information. Similar to [12], our object patch extraction phase consists of expanding the patch selection, as shown in figure 3.3. A local background region is defined as a border that surrounds the foreground region. Specifically, we expand the bounding box around the current location of the target, by a scale factor τ, so that the size of the extended region is given by:

$$A_{pat}(t) = \tau^2 A_{obj}(t) = \tau^2 w_{obj} h_{obj} \tag{3.3}$$

where w_{obj} and h_{obj} denote the width and height respectively of the bounding box, and $A_{pat}(t)$ and $A_{obj}(t)$ are the fields of the foreground region and the object patch respectively.

3.3 Correlation filters

The correlation filters can encode effectively the appearance of the target object [29, 68], the correlation filter models w are learned by solving the following minimization problem:

$$w^* = \underset{w}{\arg\min} \sum_{m,n} ||w.x_{m,n} - y(m,n)||^2 + \lambda ||w||_2^2 \tag{3.4}$$

Where w^* refers to the learned correlation filter model. The feature vector x is of size $M \times N \times D$, where M, N and D indicate the width, height and number of channels respectively . λ indicates the regularization parameter ($\lambda \geq 0$). $w.x_{m,n} = \sum_{d=1}^{D} w_{m,n,d}^T . x_{m,n,d}$. In addition, the size of the correlation filter model is $M \times N$ [91]. Each shifted sample of $x_{m,n}(m,n) = \{0,1,...,M-1\} \times \{0,1,...,N-1\}$ is regressed to the gaussian function label $y(m,n)$.

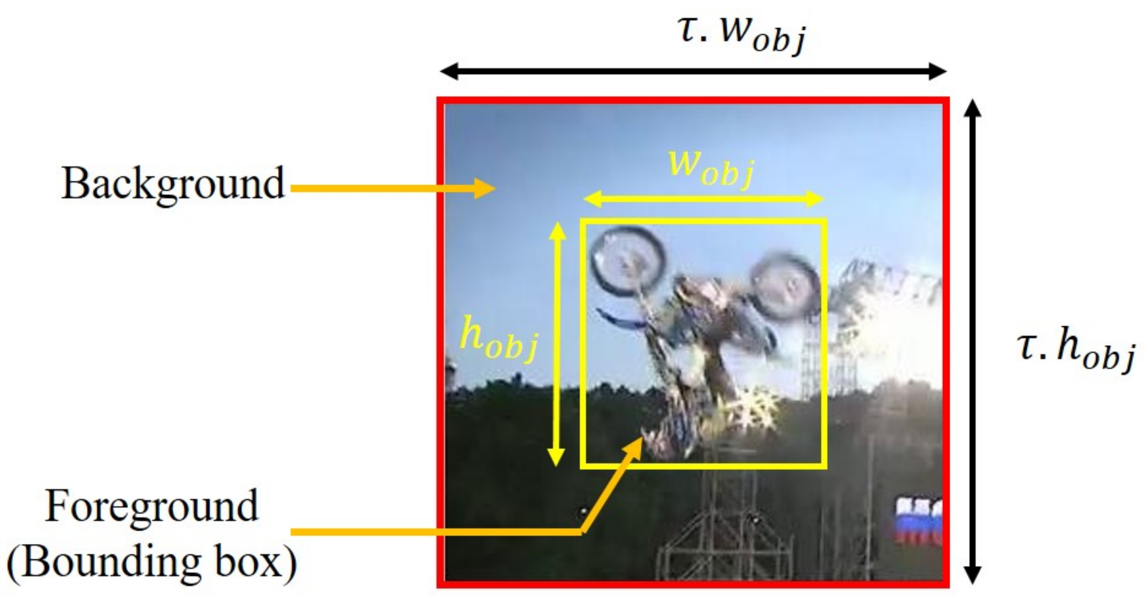


Figure 3.3: Object patch extraction.

$$y(m,n) = e^{-\frac{(m-\frac{M}{2})^2+(n-\frac{N}{2})^2}{2\sigma^2}} \tag{3.5}$$

Where σ is the standard deviation.

It is known that the large discontinuity between the opposite edges of a non-periodic image will result in a noisy representation in the frequency domain, especially in the Fourier domain, because the Fourier transform is periodic and does not respect the limits of the image [13]. The inverse Fast Fourier Transform (IFFT) is used to calculate the l^{th} correlation response map f_l, i.e.

$$f_l = \mathscr{F}^{-1}(\sum_{d=1}^{D} W^d \bigodot \overline{Z}^d) \quad , \quad l = 1,2,...,3 \tag{3.6}$$

During the tracking process, a multichannel vector of Z^l was used to compute f_l. The capital letters present the corresponding Fourier transform signals, the $\mathscr{F}^{-1}$ operator indicates the inverse FFT and the bar signifies complex conjugation. The $\bigodot$ operator is the Hadamard product (product per element). [1] Therefore, the new target location $(\widehat{x_p}, \widehat{y_p})$ is obtained by detecting the

[1]The Hadamard product of two matrices is defined by:

$$A \bigodot B = \sum_{i=1}^{m}\sum_{j=1}^{n} A(i,j).B(i,j)$$

Exemple:

$$if A = \begin{vmatrix} 1 & 2 \\ 3 & 1 \end{vmatrix} \quad and \quad B = \begin{vmatrix} 1 & 1 \\ 2 & 4 \end{vmatrix} \quad then \quad A \bigodot B = \begin{vmatrix} 1 & 2 \\ 6 & 4 \end{vmatrix}$$

maximum value of the response map f.

3.3.1 Estimation of the location of the object

In order to estimate the location of the object, we weight each response map with a weighting parameter γ_l. The final confidence score is obtained by accumulating these weighted maps. The location of the new target (x_p^*, y_p^*) is estimated using equation 3.7.

$$\underset{m,n}{\arg\max} \sum_{l=1}^{3} \gamma_l f_l(m,n) \tag{3.7}$$

Where the values of γ_l are defined experimentally at 0.019, 0.4 and 1.

3.3.2 Update models

During the tracking process, it is obvious that an important change of the object appearance is observed between two consecutive images, which can make the tracker drifts [92]. To solve this problem, it is necessary to update the correlation filter model obtained via equation 3.4 with a learning rate η as presenting in equation 3.8:

$$\begin{aligned} \widehat{x}^t &= (1-\eta)x^{t-1} + \eta x^t \\ \widehat{W}^t &= (1-\eta)W^{t-1} + \eta W^t \end{aligned} \tag{3.8}$$

3.3.3 Scale estimation

To quickly estimate the target scale variation, we use the features of the HOG [41] to form the scale feature pyramid with a fixed size $P \times Q \times S$, where P and Q are the height and width and S is the condidate scales. The scale estimation task is detailed as follows:

First, we build a pyramid of features around the new estimated position of the target using the HOG [41], by considering the scale factor α and K indicate the number of candidate scales S, where $S = \{\alpha^n | n = -\frac{K-1}{2}, -\frac{K-3}{2}, ..., \frac{K-1}{2}\}$. For each $s \in S$, the features of the patch J_s (of size $sP \times sQ$) are extracted on the estimated position $(\widehat{x_p}, \widehat{y_p})$ [93]. Then all feature channels are resized with the size $P \times Q$ again. In addition, we reduce the impact of boundary discontinuities by weighting each feature channel by a cosine window. Second, we form the R_s model in the frequency domain using Fast Fourier Transform (FFT) using the equation 3.9. Third, we weight R_s by a three-dimensional gaussian function to construct the desired correlation response map $\widehat{f_s}$. The optimal $\widehat{S}$ scale of the target is obtained using the equation 3.10. Finally, the scale model R_s is updated using the equation 3.8.

$$W^d = \frac{Y \odot \overline{X}^d}{\sum_{i=1}^{D} X^i \odot \overline{X}^i + \lambda} \tag{3.9}$$

Where λ is a regularization parameter ($\lambda > 0$). The Fast Fourier Transform (FFT) is adopted in the equation 3.4 to exploit the fact that the autocorrelation of a signal x in the Fourier domain represents the power spectrum [13, 25].

$$\widehat{s} = \arg\max_{s}(max(\widehat{f}_1), max(\widehat{f}_2), ..., max(\widehat{f}_K)) \tag{3.10}$$

We update the model R_s image by image using the equation 3.8. A predefined threshold T_a is used to update only R_s if $max(f) > T_a$.

3.4 CNN-based method for Visual object tracking

In this section, we present the important parts of our proposed visual tracking method. Our goal is to develop a robust method, suitable for handling the important appearance changes of the object, based on two main tasks. First, we estimated the object translation using three models of correlation filter learned with hierarchical convolution features. In addition, the multi-level correlation response maps are summed to produce the final confidence output. We propose to learn these models of correlation filters by the PSO algorithm. We adopt the energy of the HSV components to manage the illumination variations since it is effective at maintaining an efficient color separation. Second, we adopt the HOG features [41] for the scale estimation task. Figure 3.4 illustrates an overview of the proposed method steps.

3.4.1 Hierarchical features

In order to track the impression of the target, we extract the features of the object's patch from three layers of the CNN architecture, motivating of its ability of describing the target's patch through the layers in a suitable way [12], since the deeper architecture are more sensitive of important appearance changes . We use the VGG-Net model (visual geometry group) [9] to extract the multichannel features from the conv3-4, conv4-4, and conv5-4 layers respectively. The choice of using VGG-Net is done, because it is very adopted in many computer science applications, versus the other architectures. Since the CNN architecture acquires image of size 244×244, each input patch of size $M \times N$ is resized by the bilinear interpolation given by equation 3.11. Then, we resize each feature map to a fixed size $\frac{M}{4} \times \frac{N}{4}$ in order to alleviate the resolutions difference between layers conv5-4, conv4-4 and conv3-4.

$$x_i = \sum_k \alpha_{ik} h_k \tag{3.11}$$

Where x_i denotes the upsampled feature vector for the location i^{th} and h_k denotes the k^{th} feature map, while α_{ik} represents a weight interpolation which depends on the position of the vectors i and k of the neighboring features [12].

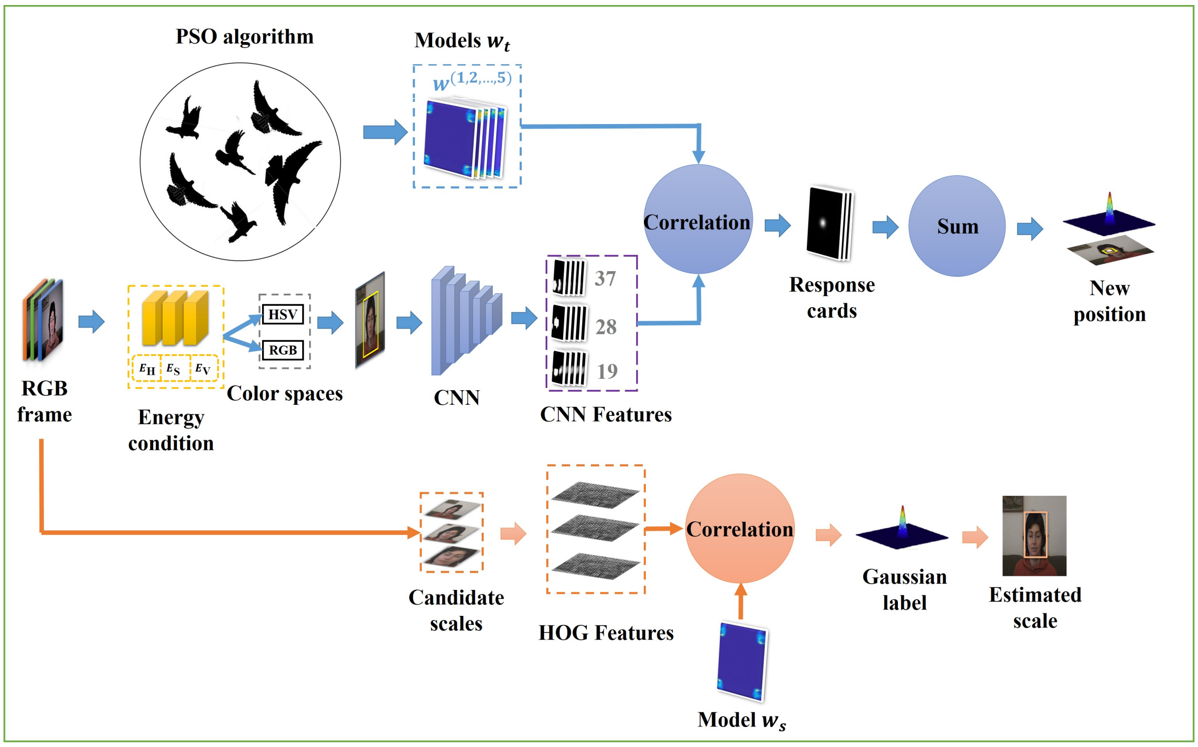


Figure 3.4: Flowchart of the CNN-based tracking method.

3.4.2 PSO algorithm

As with all other optimization problems, visual tracking is interpreted as a similarity function between the desired solution and the candidate solution [94]. In this context, many meta-heuristic algorithms are used to solve visual tracking problems [95]. Inspired by the social behavior of birds flock, Particle Swarm Optimization (PSO) is the most useful in this topic because of its ability to solve a non-linear, multimodal, and large-scale optimization algorithm. Also, the time computation remains low. In this section, we give a detailed description of the implementation of PSO for visual tracking.

As previously discussed that the Fourier transform will result in a noisy representation reflected of the large discontinuity between the opposite edges of a non-periodic image because it does not respect the limits of the image [13]. We propose to use the PSO algorithm to optimize correlation filter models in the spatial domain. The main idea is to optimize the equation 3.4, in order to reach the appropriate correlation, filter models. The basic steps of PSO are summarized as a pseudo-code presented in algorithm 1. More details on the implementation are presented below.

Algorithm 1 Pseudo-code of the PSO algorithm [96]

```
Input: $Problem_{size}$, CNN features, correlation filter model at the instant $t-1$.
Output: $W^*$
    for $i$ = 1 to $n$
        $p_i \leftarrow rand(Problem_{size}, n)$;
        $v_i \leftarrow rand(Problem_{size}, n)$;
        Initialize $W_i$(n,Problem$_{size}$,$CNN'sfeatures$);
        Generate $W_i$ according to the equation 3.4
        $x_i^{best} = x_i$;
        if $cost(p_i) \leq cost(p_g^{best})$ then
            $p_g^{best} \longleftarrow p_i$
            $W_g^{best} \longleftarrow W_i$
        end
    end
    while(iteration < 200)
        Generate new solution according to the equations 3.12 to 3.14;
        if $cost(p_i) \leq cost(p_i^{best})$ then
            $p_i^{best} \longleftarrow p_i$
            $W_i^{best} \longleftarrow W_i$
            if $cost(p_i^{best}) \leq cost(p_g^{best})$ then
                $p_g^{best} \longleftarrow p_i^{best}$
            end
        end
    end
    return $W_g^{best}$;
```

$$v_i(t+1) = v_i(t) + Q_1 + Q_2 \qquad , \qquad i = 1,2,...,n \tag{3.12}$$

$$Q_1 = c_1 \times rand \times (p_i^{best} - p_i(t)) \tag{3.13}$$

$$Q_2 = c_2 \times rand \times (p_g^{best} - p_i(t)) \tag{3.14}$$

where $v_i(t+1)$ is the new velocity for the i^{th} particle, n is the size of the population, c_1 and c_2 are the weights of the best position and best overall position respectively, $p_i(t)$ and $W_i(t)$ are the optimal position and correlation filter model at instant t. p_g^{best} and W_g^{best} denote the best position and the best model. The position and model of the filter of each particle are updated using equations 3.15 and 3.16 [96].

$$p_i(t+1) = p_i(t) + v_i(t) \tag{3.15}$$

$$W_i(t+1) = W_i(t) + v_i(t) \tag{3.16}$$

Table 3.2: The parameters of the PSO algorithm.

Parameter	Value
c_1	1.3
c_2	0.9
n	20

3.4.2.1 Configuration

The PSO parameters are summarized in table 3.2 [11].

3.4.2.2 Optimization procedure

As shown in algorithm 1, each particle is initialized by a set of candidate models, with an identical size to that of the corresponding features. We use 10 candidate models of correlation filters for each particle. Noting that this initialization is performed using previous correlation filter models in the spatial domain. Once the algorithm is initialized, we evaluate equation 3.4 to initialize the three best models. By resorting to 200 iterations, the dynamics of the PSO algorithm can suitably meet three appropriate correlation filter models, resulting in an accurate target location estimation, as shown in figure 3.4. This behavior could lead to efficient optimization of the correlation filter models as a solution of equation 3.4 in which the Fourier domain could not. For example, figure 3.5 shows that the proposed method is effective in handling changes in the appearance of objects, when the cases of fast motion, deformation, and occlusion occur in the frames 170^{th}, 240^{th} and 250^{th}, respectively of the challenging sequence *CarScal*.

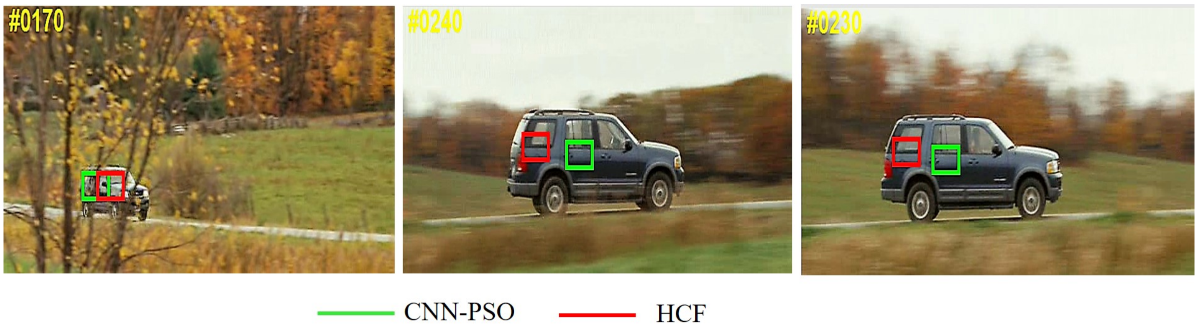


Figure 3.5: Comparison of our proposed CNN-PSO approach with the HCF tracker [12] in the challenging sequence of *CarScal*.

3.4.3 Results and discussion

The proposed method includes three main steps, as shown in figure 3.4. The first step relies on learning adaptively three correlation filter models in the spatial domain using convolution features via the PSO algorithm, which is effective in presenting good convergence properties. The

CNN features are then enriched using the HSV energy condition (Hue, Saturation, and Value). This condition has a substantial advantage in dealing with illumination variations and allows for suitable switching between the RGB and HSV color bases. The third step is to use HOG features to exhaustively search for the optimal scale of the target object.

3.4.3.1 Configuration

We implemented our tracking system in Matlab with an Intel (R) Core (TM) i7-6700K 4.00GHz CPU and with 32GB of RAM. We used MatConvNet toolbox [97] for feature generation. The regularization parameter λ is set to 10^{-4} in the equation 3.4 and the learning rate η to 0.01 in equation 3.8. For the object position estimation task, the size of the search window is set to 1.8. We set $T_a = 0.5$ to update R_s.

3.4.3.2 Databases

- **OTB-50**

  We evaluated our method by using two metrics, the distance precision (DP), which presents the average value of the frames where the estimated positions are within the threshold of 20 pixels [4, 75], and the overlap success (OS), which represents the average value of frames where the overlap between the estimated bounding box and the given one (ground-truth) exceeds a given threshold $t_o \in [0,1]$. The result of OP is presented in $t_o = 0.5$ [4].

  The proposed method is evaluated on a large reference database with 50 image sequences [4]. To validate the results, we compare our tracker with 9 reference trackers, including KCF [13], DLT [98], HCF [12], MEEM [14], TGPR [99], Struck [15], SCM [100], TLD [16] and LSHT [40].

  It's clear that our tracker works effectively compared to the reference trackers. Figure 3.6 shows that the HCF tracker achieves the best distance precision of 89.1 % and the best overlap success rate of 74 %. The efficiency of our method is demonstrated with improvement gain of 1.9 % in DP and 2.5 % in OS. Among the existing methods, it is obvious that our tracker outperforms the others in terms of precision and overlap.

  The superiority of the proposed tracker can be justified by the following points: First, the integration of the HOG features in the scale estimation has effectively solved this challenging case. Indeed, figures 3.7 and 3.8 indicate that our tracker robustly handles the case of scale variation versus reference trackers. Second, the correlation filter models are learned using a PSO algorithm in the spatial domain. Then, the resulted models are updated in the frequency domain. As a result, the proposed method can recover the target in the challenging cases of fast motion, occlusion. Figure 3.9 shows the worth of the proposed method against three state-of-the-art methods. The PSO algorithm improves the accuracy in cases of fast motion and occlusion through its effective optimization of correlation filter

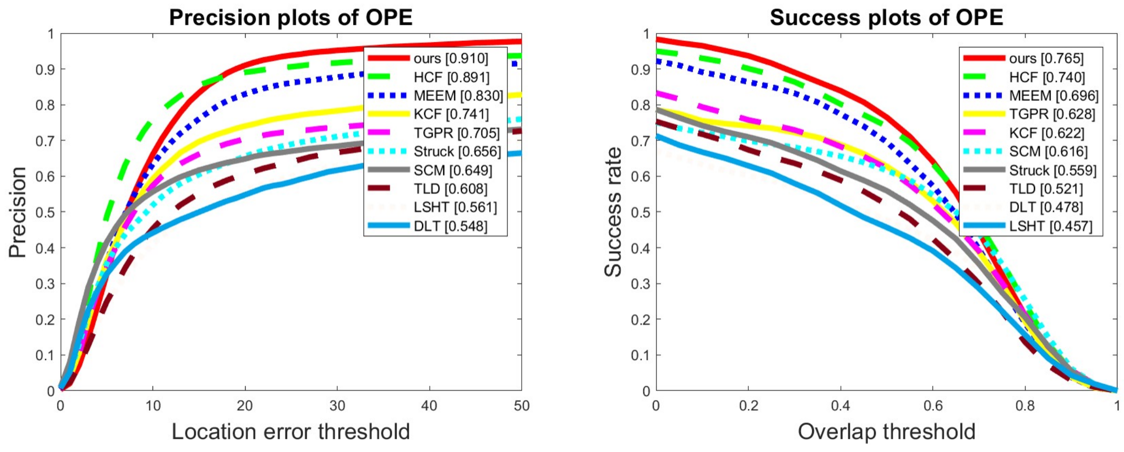


Figure 3.6: Comparison with nine reference trackers using distance precision and overlap success on the OTB-50 database [4].

models. Third, the switching between the RGB and HSV color bases allows for an accurate treatment of illumination variation case, as shown in figure 3.8. Compared to the three reference trackers, it is clear that our tracker (in red) follows the object robustly in case of illumination variation.

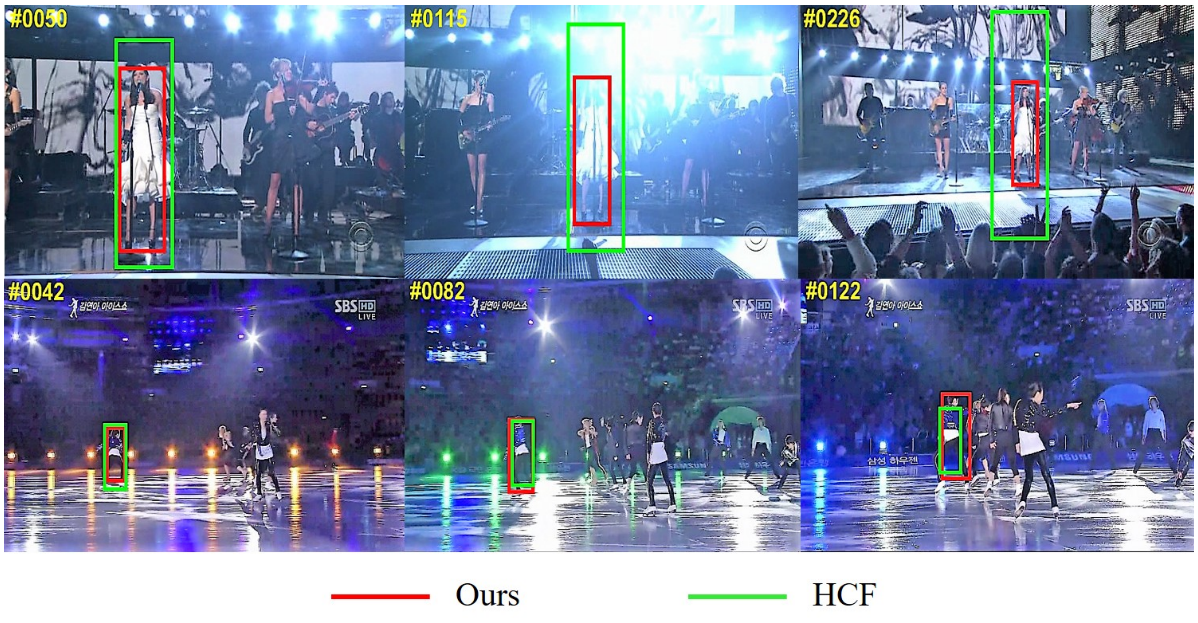


Figure 3.7: Illustration of handling of scale changes on sequences *Singer1* and *Skating1*.

- **VOT 2016**

  Figure 3.10 presents the comparison results of our proposed tracker with five highest-ranked trackers in the VOT 2016 dataset [3], in terms of expected average overlap (EAO).

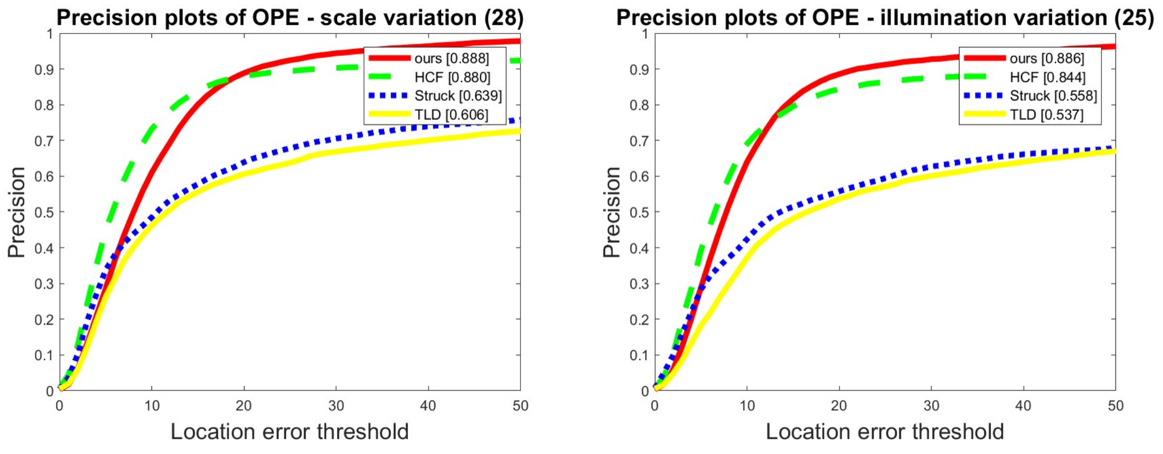


Figure 3.8: Precision distance plots of scale variation and illumination variation challenging cases.

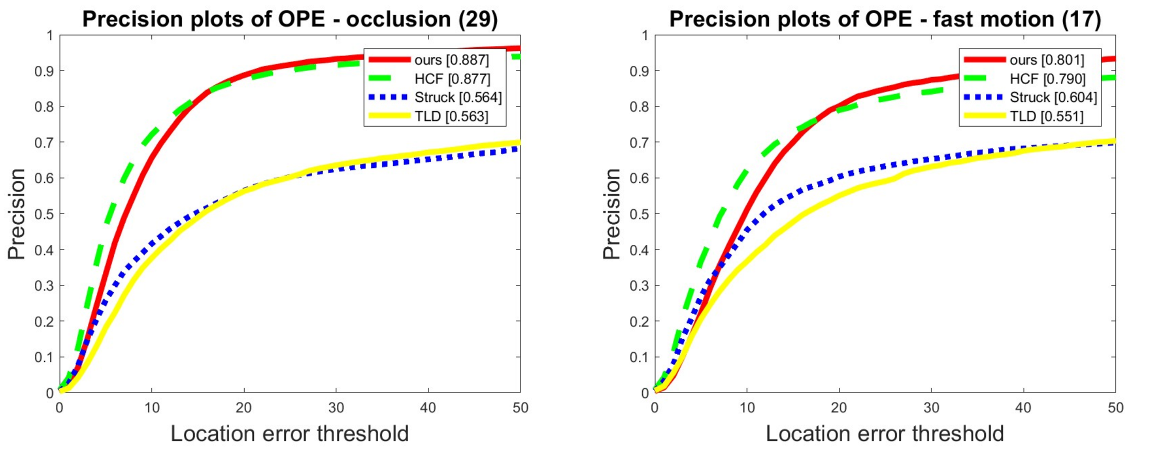


Figure 3.9: Precision distance plots of two tracking challenges: Occlusion and fast motion.

In addition, table 3.3 lists a detailed comparison of our method with the reference methods in VOT 2016 dataset including: CCCT [101], EBT [102], MDNET_N [103], STAPLE [104] and DNT [105]. We can see that our tracker exceeds the STAPLE tracker [104] by a significant gain. Indeed, our tracker demonstrates its effectiveness by obtaining the best EAO score of 0.31. Also, It gets the third-highest overlap score of 0.50 and the best result in failures metric with 15.09 %.

- **UAV20L**
  Finally, we evaluate our method on the UAV20L database [2] which contains 20 fully annotated image sequences ranging in length from 1.177 to 5.527 images. As shown in figure 3.11, the proposed tracker outperforms the other state-of-the-art trackers including PTAV [26], SRDCF [68], MEEM [14], Struck [15], DSST [5] , DCF [29], KCF [13], and

Table 3.3: Comparison results with the reference trackers in the VOT 2016 database.

Tracker	Overlap	Failures	EAO
ours	0.50	15.09	0.31
STAPLE	0.54	23.89	0.30
EBT	0.45	15.19	0.29
DNT	0.51	19.54	0.28
MDNET_N	0.54	21.08	0.26
CCCT	0.44	29.32	0.22

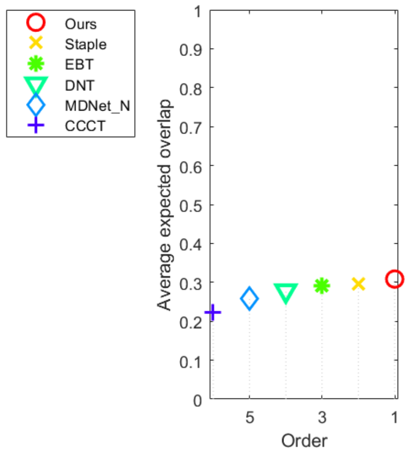


Figure 3.10: Expected average overlap results (EAO) with the top state-of-the-art trackers of the VOT 2016 database.

TLD [16]. We can see that our tracker outperforms the other trackers in both distance precision (DP) and overlap success rate (OS) with a gain of improvement of 2.3 % and 0.5 % respectively over the nearest competitor tracker, PTAV [26].

In addition, figure 3.12 demonstrates the high capabilities of our tracker, who manages long-term tracking even in difficult cases of fast motion, scale variation, illumination variation, and occlusion as opposed to other trackers.

3.5 HOG-based method for Visual object tracking

As shown in figure 3.13, we decompose the tracking process into two tasks: translation estimation and scale estimation. We first deduce the target position from the correlation response map f of the filter R_t in the frequency domain by using equations 3.6 to 3.9. To estimate the suitable target's scale, we adopt the feature pyramid strategy basing on HOG features to learn another correlation filter R_s by using the equations 3.9 to 3.10. Similar to the first proposed method, we reduce the impact of boundary discontinuities by weighting each feature channel by a cosine window. Besides, we propose a re-detection module built using the SVM classifier.

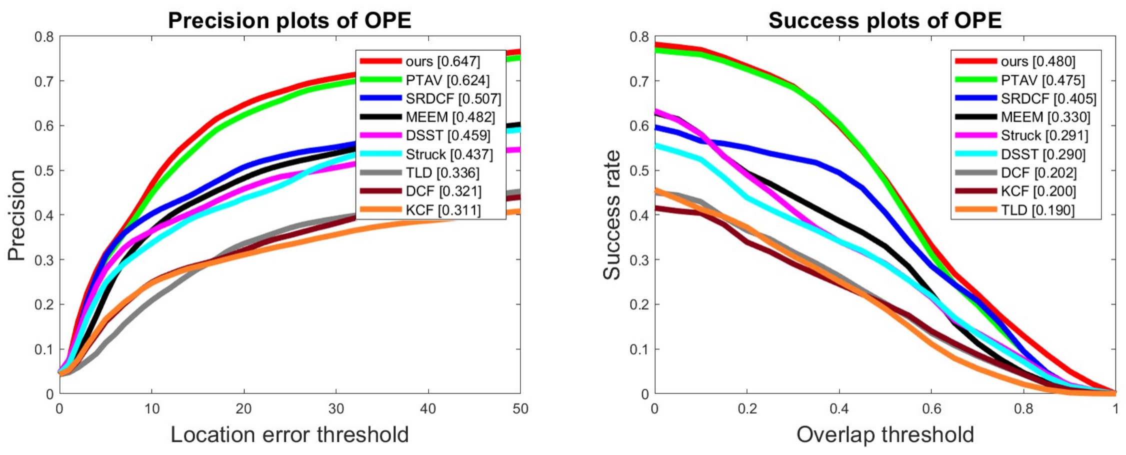


Figure 3.11: Comparison results with eight reference tracker on the UAV20L database [2] using distance precision (DP) and overlap success (OS) metrics.

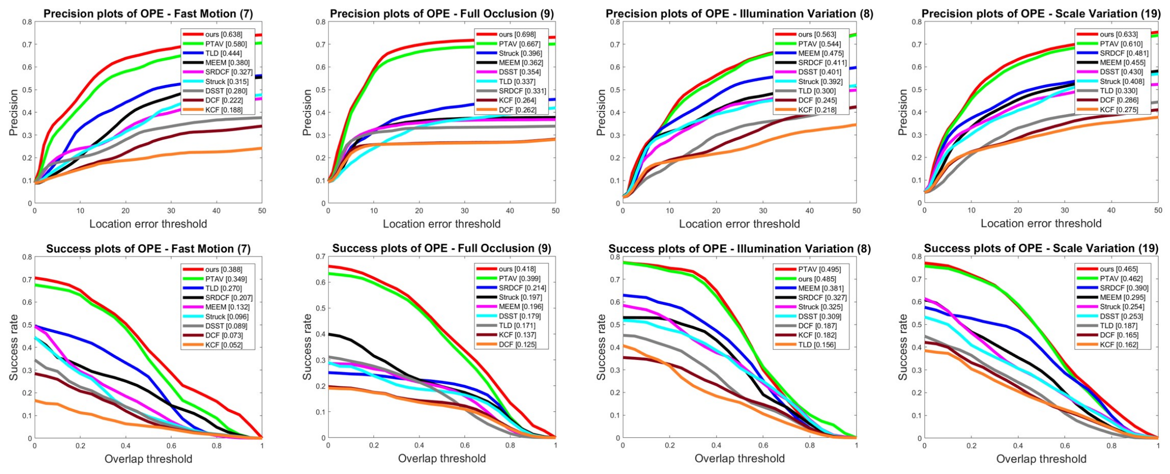


Figure 3.12: Comparison results with eight reference trackers using distance precision and overlap success metrics on difficult cases of fast motion, scale variation, illumination variation, and occlusion on the UAV20L database [2].

It is used to recover the tracker from drifting. We activate the re-detection module when the correlation response is less than a given threshold T_r. Note that we only adopt the detection result when it is very confident. In other words, when the correlation response is greater than a given threshold T_a, we only adopt the detection result.

3.5.1 Histogram of oriented gradient features

The features extraction procedure is as follows:

- To improve the contrast of the images, we use the Gamma transformation. It performs a conversion of the gray level of the images. The transformation function is given as:

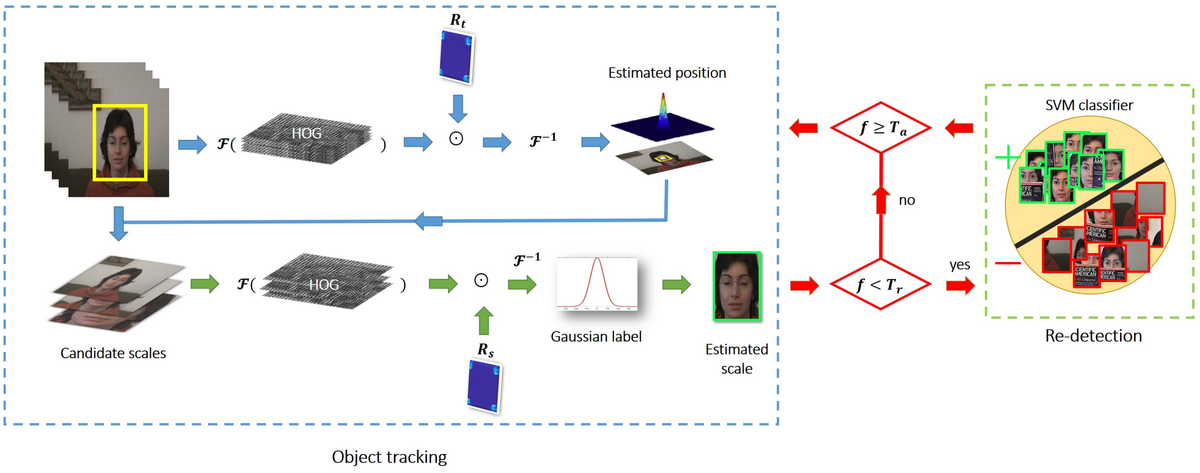


Figure 3.13: Flowchart of the proposed HOG-based method.

$$x(n_1,n_2) = x^{\epsilon}(n_1,n_2) \tag{3.17}$$

where $x(n_1,n_2)$ is the input pixel. We set the parameter ϵ to 0.5.

- The gradient can be calculated using the following equations:

$$y_i(n_1,n_2) = h_i(n_1,n_2) \times x(n_1,n_2) \qquad , \qquad i = 1,2 \tag{3.18}$$

Where $h_i(n_1,n_2)$ is the derivative mask.

$$h_1(n_1,n_2) = [-1 \quad 0 \quad 1]^T \qquad , \qquad h_1(n_1,n_2) = [-1 \quad 0 \quad 1] \tag{3.19}$$

where h_1 et h_2 present the discrete derivative masks in the vertical and horizontal directions. The overall gradient and orientation image are calculated as follows:

$$y_m(n_1,n_2) = \sqrt{y_1^2(n_1,n_2) + y_2^2(n_1,n_2)} \tag{3.20}$$

$$y_\theta(n_1,n_2) = \arctan(y_1(n_1,n_2)/y_2(n_1,n_2)) + \pi/2 \tag{3.21}$$

- The target patch is partitioned into rectangular cells. Each cell is associated with an edge orientation histogram. For each cell, the histogram is calculated as follows:

$$H(\tau) = \sum_{i=1}^{N} y_m(n_1,n_2)\delta.[y_\theta^{'}(n_1,n_2) - \tau] \tag{3.22}$$

Where δ is the delta function of Kronecker and $y'_{\theta}(n_1,n_2)$ is a quantized orientation, computed from $y_{\theta}(n_1,n_2)$ and N is the total number of pixels in each cell [41]. Then, the set of magnitude sums of the gradient is represented as an N-orientation histogram:

$$H^{all} = \{H(1), H(2), ..., H(N)\} \tag{3.23}$$

In our implementation, we concatenate HOG features in 31 bins [106], to provide precision for important lighting changes. As a result, a vector with 63 channels is used to form the model R_s.

3.5.2 Online re-detector

A robust tracking algorithm requires a detection module to recover the target from potential challenging cases of significant occlusion or fast motion. For each feature vector z, we calculate its confidence index in the form $C = max(f(z))$. We activate the detector only if the confidence index C is below the predefined re-detection threshold T_r. The main purpose of using T_r is to reduce the computational load by avoiding searching window detection in each image [25, 107]. For more efficiency, we use an online SVM classifier as a detector.

Similar to [14], we gradually train the SVM classifier by drawing learning samples around the estimated position and then assigning binary labels to these samples based on their overlap rates. The learning set is defined as $\{(v_i, c_i)|i = 1,2,...,N\}$ with N samples for an input image, where v_i is the feature vector generated from the i^{th} sample and $c_i \in \{+1, -1\}$ is the label of the class.

The objective function for solving the hyperplane h of the SVM detector is given as follows:

$$\begin{gathered} \underset{h}{min} \frac{1}{2}||h||^2 + \frac{1}{N}\sum_i l(h;(v_i,c_i)) \\ l(h;(v_i,c_i)) = max\{0, 1 - c\langle h, v\rangle\} \end{gathered} \tag{3.24}$$

With $\langle h, v\rangle$ indicates the element-wise product between h and v.

3.5.3 Results and discussion

3.5.3.1 Configuration

The regularization parameter λ in equation 3.4 is set to 10^{-4}. The learning rate η in equation 3.8 is set to 0.01. The number of candidate scales s is 25, and the scale factor α is set to 1.049. To learn the SVM detector, we construct samples densely using a window centered on the estimated location. We assign these samples with positive labels when their overlap ratios with the target selection frame are greater than 0.5, and we assign the others as negative labels. To activate the formed function detector, we set the threshold T_r

to 0.2. The parameter T_a is set to 0.45. We implement our tracker in Matlab with an Intel (R) Core (TM) i7-6700K 4.00 GHz processor with 32 GB of RAM.

- **OTB-50**

  We evaluate our method using distance precision (DP) and overlap success (OS) metrics [4]. We compare the proposed method with 8 reference methods, including KCF [13], DLT [98], MEEM [14], TGPR [99], Struck [15], SCM [100], TLD [16] and LSHT [40]. Figure 3.14 shows the comparison results on the OTB-50 database. The MEEM tracker obtains the second-best result with a distance precision of 83.0% and an overlap success of 69.6%. The proposed method proves its efficiency with an improvement gain of 1.1 % at the distance precision and 1.8 % at the overlap success.

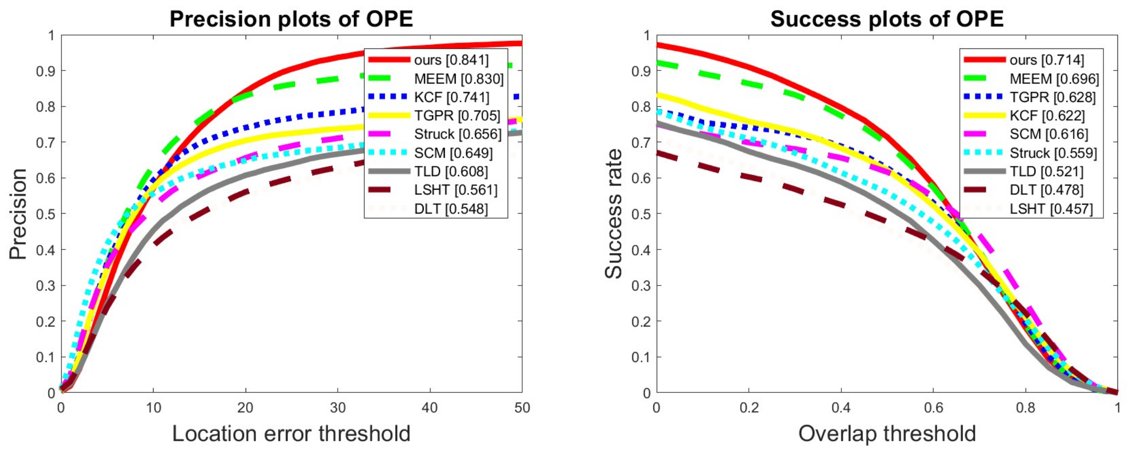


Figure 3.14: Comparison with eight reference followers using distance precision and overlap success on the OTB-50 database [4].

In addition, we compare our method with three reference trackers (KCF [13], MEEM [14] and TLD [16]) on four challenging sequences illustrated in the figure 3.15. The KCF tracker is based on a correlation filter learned using HOG features and is therefore similar to our follower. The KCF tracker works well in handling large deformations and fast motion due to the robust representation of HOG features. However, it drifts when target objects suffer from heavy occlusions and consequently cannot re-detect targets in these cases. In addition, the KCF tracker is unable to handle the background clutter, as only HOG features are less effective at distinguishing targets from the cluttered background. The TLD tracker works well in case of rotation (*David*). However, it does not correctly follow the target in case of large deformation and fast motion (*Tiger2*). The MEEM tracker is based on an expert selected to restore the current tracked object by using minimal entropy criteria to correct unwanted model updates. This explains its ability to handle heavy occlusion (*Lemming*, *Bolt*), but it does not work well when deformation is important (*David*, *Tiger2*). Overall,

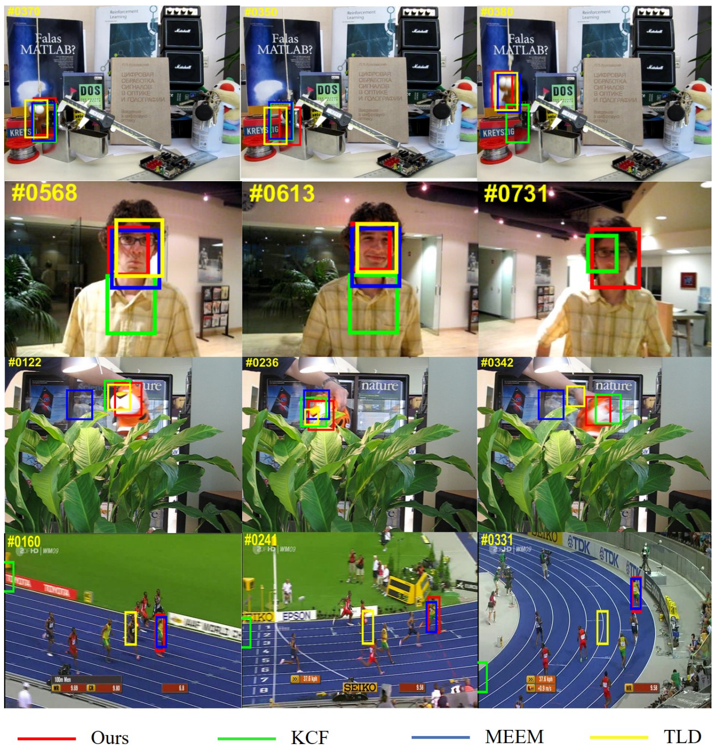


Figure 3.15: The comparaison results of our method with the KCF [13], MEEM [14], Struck [15] and TLD [16] methods on four complex sequences (from top to bottom are *lemming*, *David*, *Tiger2*, *Bolt*), respectively.

our proposed method allows estimating efficiently both the scale and the position of the target object on these challenging sequences. Its efficiency is attributed to three reasons. First, the model R_t is learned from robust features rather than just HOG features or a simple brightness intensity and to estimate the translation of the target object. The proposed features are less sensitive to background lighting and clutter (*David*), and to the occlusion (*Lemming*, *Tiger2*). Second, the model R_s is conservatively updated therefore the errors of scale estimation are not accumulated to affect subsequent frames. Consequently, our method mitigates effectively the problem of scale variation. Thirdly, the proposed detector effectively re-detects the target objects in case of tracking failure, for example with a heavy occlusion (*Lemming*).

- **VOT 2016**

  Figure 3.16 illustrates the comparison results with three reference trackers on the VOT 2016 dataset [3] in terms of expected average overlap (EAO). Our proposed tracker exceeds the best tracker EBT [102]. Table 3.4 presents the detailed results of our method with three reference methods in VOT 2016 including EBT [102], SRBT [74] and DNT [105]. The proposed tracker demonstrates its effectiveness by obtaining

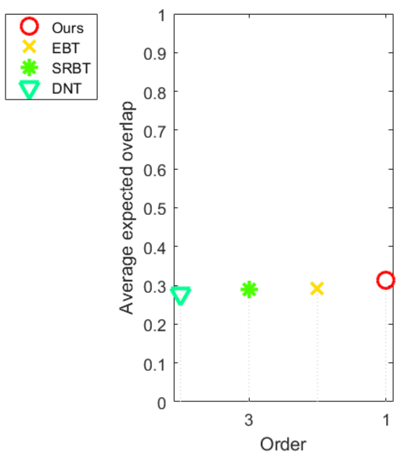


Figure 3.16: Expected average overlap result (EAO) with three reference trackers evaluated on the VOT 2016 database.

the best EAO score of 0.32. In addition, it provides a considerable result of 0.5 in the overlap metric and the second-best result of 17.60% in failure metric.

Table 3.4: Comparison in terms of EAO, overlaps, and failures, with the three state-of-the-art trackers of the VOT 2016 database.

Tracker	Overlap	Failures	EAO
Ours	0.50	17.60	0.32
EBT	0.45	15.19	0.29
SRBT	0.48	21.32	0.29
DNT	0.51	19.54	0.28

- **UAV20L**

Finally, we evaluate our method on the UAV20L database [2]. As shown in figure 3.17, the proposed tracker outperforms the other trackers, including PTAV [26], SRDCF [68], MEEM [14], Struck [15], DSST [5] , DCF [29], KCF [13], and TLD [16]. It is clear that our tracker surpasses the other trackers on distance precision (DP) and overlap success (OS) metrics with an improved gain of 1.1 % and 0.9 % respectively against the competitor tracker PTAV [26]. To give more credence, figure 3.18 clearly demonstrates the high capabilities of our tracker, which manages the long-term tracking in difficult cases of fast motion, scale variation, and occlusion, compared to the aforementioned reference trackers.

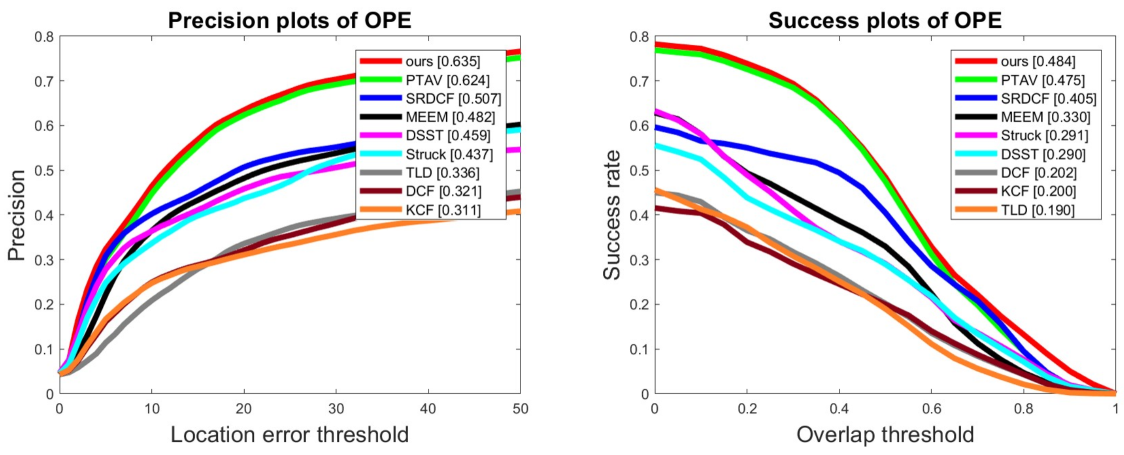


Figure 3.17: Comparison results with eight reference trackers on the UAV20L database [2] using distance precision (DP) and overlap success (OS) metrics.

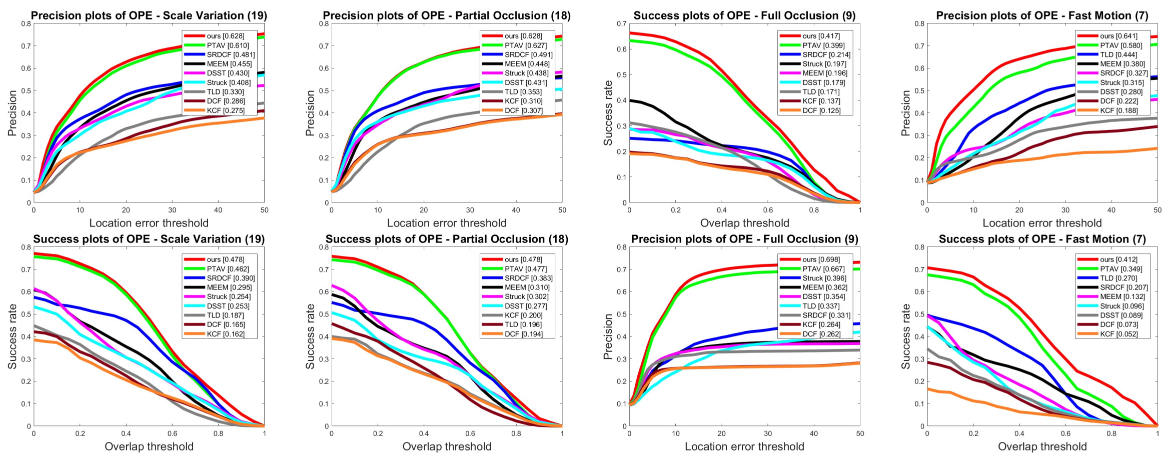


Figure 3.18: Comparison results with eight reference followers using distance precision and overlap success metrics for the challenging cases of fast motion, scale variation and occlusion on the UAV20L database [2].

3.6 Conclusion

In this chapter, we proposed two methods for visual object tracking. These methods based on image preprocessing, feature extraction, to perform two tasks of translation and scale estimations. Our preprocessing easily extract the patch from the target object, which helps the system to retain the important information about the object (foreground and background). At the feature extraction phase, the CNN and HOG features are used in our work. We have used the correlation filters in the tasks of the translation and scale estimation to maintain the object tracking. We finished with a comparison with the state-of-the-art trackers in visual tracking, which prove the efficiency of our method on three most well-known databases.

Part II

Visual Servoing

CHAPTER 4

STATE-OF-THE-ART

4.1 Introduction

In the mid-nineties, the first focus on the concept of visual servoing was done by Hutchinson and his collaborators [108]. They provide a tutorial introduction to visual servo control. Later, a group of researchers including Chaumette, Corke, Mahony, and others have led many advances and remain the visual servoing active in this field to date. In this chapter, we present an introduction to unmanned aerial vehicles and the different visual servoing approaches applied to this type of vehicle. We discuss the definition of visual servoing and its classes. Then, we introduce the three basic approaches of visual servoing, by focusing on the strengths and weaknesses of each approach. In addition, we present the needed basics to understand the concept of visual servoing, which includes the digital image and the projection model. Finally, a proof of stability is presented for the visual servoing based on the image.

4.2 Unmanned aerial vehicles

The UAVs are small unmanned aircrafts. They can be operated remotely by a human or be autonomous; autonomous vehicles are controlled by an on-board computer that can be pre-programmed to perform a specific task or a wide range of tasks. This section first presents a summary of UAVs in a historical way, followed by an introduction of the applications and classification of these types of vehicles. Then we discuss the vision control works applied on quadrotor UAVs.

4.2.1 History

Since the beginning, primitive man was fascinated by the idea of flying. In the 19th century, fantasy became a fact. Scientists based on the principle of creating an airplane done with a density lower than that of the air, to develop the first aircraft. However, it was not until the 20th century, where the most important improvements in air navigation were made. In 1903, the brothers Wright made the first aircraft with a density greater than that of air and propelled by engines. The first drones were manufactured by Lawrence and Sperry in 1916. They named it the **aviation torpedo** (Figure 4.1). They were able to fly it for a distance of 30 miles [109]. Later, the first experimental helicopter was built for the US Army Air Service. Following this model and with many electronic improvements, many scientists have focused their research on autonomous control of air vehicles to improve their control and make them autonomous in flight. Over time, the emergence of a specific variant, such as quadcopters able to fly independently, has aroused great interest from users and researchers until the date [17].

Figure 4.1: Lawrence and Sperry drone [17].

4.2.1.1 Applications

Drones can be implemented in different applications [17, 110]:

- **Research:** Research institutes use drones equipped with the appropriate sensors to observe some environmental phenomena such as pollution.
- **Military:** It is generally accepted that the use of drones is limited to dangerous military roles such as ammunition.
- **Environmental situations:** Such as forest fire detection, pollution monitoring, and control, sampling and analysis for fisheries forecasting and protection.
- **Search and Rescue:** The drones can search for survivors after natural disasters such as earthquakes and hurricanes or survivors of wrecks and plane crashes.
- **Agriculture:** Agricultural monitoring and spraying.

Some other applications, such as ambulances or satellite drones, are still in development. These applications require the largest drones to carry out their activities. Nevertheless, they face problems of battery life and high energy consumption. These potential problems led the appearance of several works to solve these issues, for example, the optimization of the batteries for longer flight time, the optimization of the weight of the UAV, the command autonomous flight, mapping and positioning of the UAV.

4.2.1.2 Classification of drones

In the literature, it exist many ways to classify drones, either according to their action, their aerodynamic configuration, their size, and their payload, or depending on their level of autonomy [17]. UAVs can be categorized according to their maximum altitude and endurance as shown in figure 4.2:

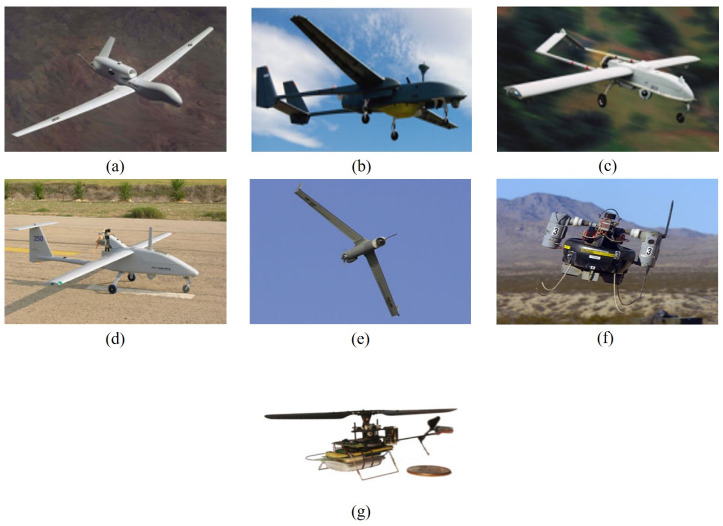


Figure 4.2: Classe of drones (a) UAV HALE, (b) UAV MALE, (c) UAV TUAV, (d) Short Range UAV, (e) Mini UAV, (f) Micro UAV, (g) Nano UAV [17].

- **High-Altitude Long Endurance (HALE):** They can fly over 15.000m with an endurance of more than 24 hours. They are mainly used for long-range surveillance missions.
- **Medium-Altitude Long-Endurance (MALE):** They can fly between 5.000 and 15.000 m altitude for up to 24 hours. These drones are used for surveillance missions.
- **Tactical Unmanned Air Vehicle (TUAV):** They are smaller and operate with simpler systems than HALE and MALE. They can fly in an altitude between 100 and 300 km.
- **Short Range UAV:** They are mainly used in civil applications such as power line inspection, crop spraying, traffic monitoring, homeland security, for a range of 100 km.
- **Mini UAV (MUAV):** They weigh about 20 kg and their range is about 30 km.
- **Micro UAV (MAV):** Their maximum wingspan is 150 mm. They are mainly used indoors, where they must fly slowly and remain to hover.
- **Nano UAV (NAV):** They have a small size of about 10 mm. Their main use is for swarm applications.

In addition, the aerodynamic configuration plays an important role in the following classification:

- **Fixed-wing drones** This class requires a runway to take off and land. They can fly long and at high-speed cruising. They are mainly used in scientific applications such as weather reconnaissance and environmental monitoring.

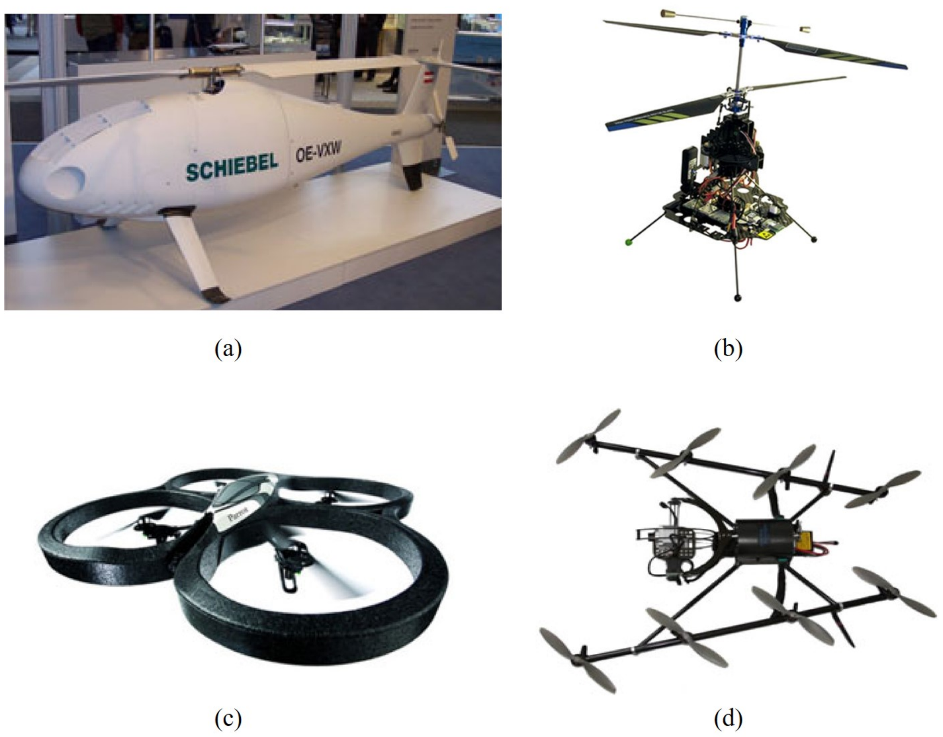


Figure 4.3: (a) Monorotor drone, (b) Coaxial rotary-wing drone, (c) Quadrotor drone, (d) Multirotor drone [17].

- **Rotary Wing Drones** They can take off, land vertically and even fly over and fly with great maneuverability. As illustrated in figure 4.3, this type of UAV is classified as follows:
  (a) **Monorotor:** They have the main rotor at the top and another rotor at the rear for stability, as in the configuration of the helicopter.
  (b) **Coaxial:** They have two rotors rotating in opposite directions mounted on the same tree.
  (c) **Quadrotor:** They have four rotors mounted in a cross.
  (d) **Multirotor:** UAV with six or eight rotors. They are agile and fly even in case of engine failure, due to the redundancy and the number of rotors.
- **Swing-wing drones** This class is inspired by birds and flying insects (Figure 4.4.a). These drones have small wings and have extremely low payload and endurance. On the other hand, they consume little energy and can make vertical take-off and landing.
- **Airship Drones** They can look like balloons or airships, they lift by their helium-filled body, as shown in figure 4.4.b. They are very light and have a large size. They can fly long and slow.

Figure 4.4: (a) Swinging wing drones, (b) Airship drones [17].

4.2.2 Visual servoing of drones

The visual servo control design for quadrotor UAVs has been a daunting task due to the under-actuation property of quadrotors. The design of the visual servoing controller usually consists of two control loops: the outer loop (vision-based loop), which creates control of the translation and rotational speeds basing on visual measurements. The internal loop controls the quadrotor to follow the desired reference signals. Different visual servoing methods combined with other control techniques for quadrotor stabilization have been proposed, including the adaptive robust control [111], the sliding mode control [112], the control by fuzzy logic [113], etc. One of the methods is the PID control for which its effectiveness has been proven in [114]. In this paper, the authors examined PID control, integral backstepping control, full adaptive backstepping, and fuzzy logic control. They found that PID control was the most effective. Also, Hamel et al [115] propose an image-based control strategy for the stabilization of an autonomous helicopter on a marked landing area. Another similar application was studied in [116], where a real-time vision-based landing algorithm was developed for a stand-alone helicopter, and in [117], where vision-based control has been implemented for autonomous road surveillance. Over the past five years, several developments in quadrotor visual servoing have been discussed in [118, 119]. Zehra et al [118] proposed an image-based controller of a quadrotor unmanned aerial vehicle. The authors did not address the negative effects of the under-actuation property, such as the handling of the image error due to tilt and loss of field of view. The authors in [119] suggested several modifications to the classical visual servo scheme that overcame the under-actuation property, such as the introduction of adaptive gains of the visual servo controller. However, this work did not provide a comparison of the results with the classical approach and only considered the case where the observed target was static. In this thesis, we adopt the ideas of [118, 119] to adapt the movement of the target by assuming that the quadrotor must follow only the position of the target.

4.3 Visual servoing

Visual servoing (VS) is an approach that uses visual feedback signals provided by the image sensor, which is often its a camera, in order to control the movement of a robot for reaching the desired position relative to a target object. The camera provides images containing a visual feature, which are then used to guide the robot robustly by the user with respect to these features.

4.3.1 VS classes

In the literature, visual servoing systems can be classified into two groups depending on the location of the camera as sown in figure 4.5 [18, 120].

– **Eye-to-hand** This configuration consists of placing the cameras in the workspace directed towards the robot. It is useful if many robots are operating in a confined space, to provide high precision position and speed measurements to robots, where the provided image is independent of the robot movement.

– **Eye-in-hand** As the name indicates, the camera is mounted on the end effector of the robot, where its movement is then linked to the movement of the robot.

Both configurations are valid. In the case of a drone, the second configuration is the most appropriate, in which a single camera can be used for a larger space because it is more versatile and less expensive.

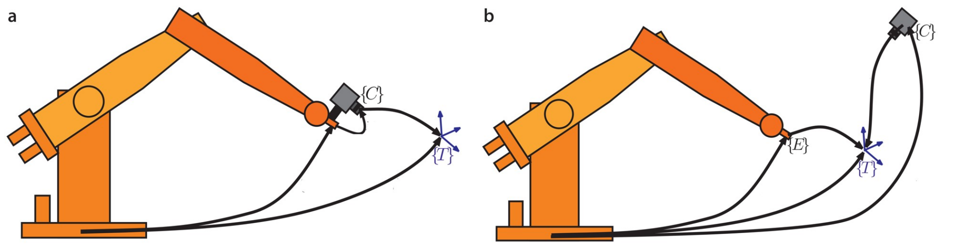


Figure 4.5: Visual servo configurations: (a) Eye in hand, (b) Eye to hand [18].

4.3.2 VS approaches

Vision-based control methods are generally categorized into three approaches [120, 121]:

– Image-Based Visual Servoing (IBVS),

– Position-Based Visual Servoing (PBVS),

– Hybrid Based Visual Servoing (HVS).

These approaches based on designing a velocity controller that gives the robot controller with suitable control signals to perform the desired application. VS's approaches differ in terms of designing the image features used in the control law. The IBVS is also called 2-D visual servoing because the exploited features are extracted from the projection of the given 3D-point in the image plane of the camera, which resulted in image coordinates in the unit of pixels. It aims to keep the target object in the camera's FOV by minimizing the error of the detected image features and the desired one, as shown in figure 4.6. The difference between these two signals allows the control law to calculate the convenient velocity speed. This approach is considered as computationally efficient because it does not provide an explicit calculation of the relative pose between the camera and a target [18, 122]. However, in position-based visual servoing (PBVS) [108], the geometric model of the target is used, together with the visual features extracted from the image, to estimate the relative pose, which is then compared to the desired position. This approach aims to estimate the suitable velocity signals corresponding to the desired 3-D target's position. The estimation of the relative pose requires a priori knowledge of the object shape, where the dimensions are required. Indeed, the regulation of the position is one of the fundamental problems of control and there are many studies on how to implement various architectures or control mechanisms. This aspect is one of the advantages of the PBVS because it performs the regulation of the relative position by separating the control problem from the error computation. The other advantage is that PBVS relies on global stability. However, the position-based approach requires a position reconstruction, which requires more information about the object, and even a precisely calibrated camera. In addition, the visual features can easily leave the field of view (FOV) because the control law does not necessarily take into account the image information of the camera. [18, 120]. Image-based visual servoing requires only a few points of interest of the image, the desired position on the image and an estimated depth of these points. Its main disadvantage is to fall into the local minima problem, which results in an unrealizable image because the region of stability is not defined robustly [123].

In order to reduce the inconvenience of IBVS, two types of work have been proposed: The first type focuses on adopting soft conventional features decoupled from each other, for example, points, lines or spheres. The second type focused on combining the advantageous features of IBVS and PBVS into hybrid methods, where the control problem was decoupled as a function of translation and rotation elements evaluated separately [124]. However, this approach was limited because of its sensitivity to noise, as well as the need to use at least 8 points to reconstruct the homography. The basis of this work is back to work Chaumette et al [125]. Another technique consists of decoupling the movements around the different axes by separating the movement on the Z-axis for a robot with 6 degrees of freedom (DOF) because this axis is less sensitive to the movement [126]. They use two

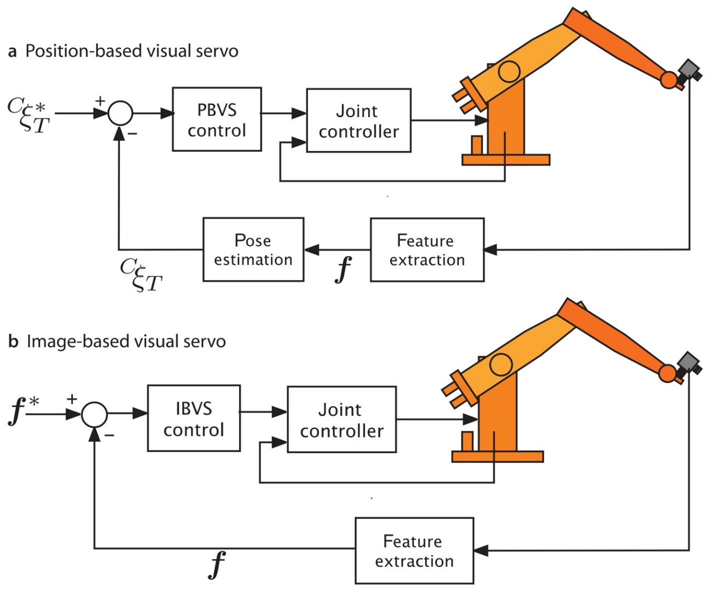


Figure 4.6: The two basic approaches of visual servoing [18].

neural network image features to improve the conditioning of the Jacobian image for the X and Y rotation axes. In particular, for the displacement of UAVs such as the case of quadrotors, the implementation of the IBVS approach made an additional problem due to the under-actuation fact of these systems. This fact led researchers to develop techniques to suppress roll/pitch coupling with image error. There are four main methods in the literature [121, 127–129].

- **Methods uses spherical projection**: They treat all the points of the image as if they were on a sphere unit, then turning them to keep them pointing down [115].
- **Methods uses a spring-based virtual approach**: These methods use the classical IBVS on the yaw and altitude parameters and add a compensation term related to roll and pitch angles. However, it increases the image error since the pitch and roll do not result in other command entries [121].
- **Methods uses the homography projection**: The homography matrix integrates the information of the transformation between two images. The two selected images are the desired view of the target and the current view which is then used for the control [130].
- **Methods uses a virtual camera approach**: They mathematically rotate the image to find what the image points would look like if the camera was pointed down [127].

4.3.3 Basic notions

As discussed in the previous section, there are two basic approaches of visual servoing: image-based (IBVS) and position-based (PBVS) visual servoing. The main idea is to minimize an error $e(t)$ defined as [125]:

$$e(t) = s(m(t), a) - s^* \tag{4.1}$$

where m(t) is a set of image measurements (e.g., the image coordinates of interest points or the image coordinates of the object's center). These image measurements are used to compute a vector of k visual features, s(m(t), a), in which a is a set of parameters that represent potential additional knowledge about the system (e.g., coarse camera intrinsic parameters or 3-D models of objects) [120]. In the IBVS, s represents a vector of visual features of the image and s^* represents the desired values of the features such as points, the distribution of colors the linens, the orientations, etc. These features are directly extracted from the image information provided by the camera. In PBVS, s and s^* are the actual and desired 3-D parameters that must be estimated from the image measurements.

4.3.4 IBVS control law

In this thesis, we adopt the IBVS approach. Once the vector s is defined, a velocity controller must be designed to minimize equation 4.1. To do this, it is necessary to establish the relation between the movement of the camera and the movement of the image features.

4.3.4.1 Static target

Let consider the case of a fixed desired pose and a motionless target, i.e., s^* is constant, and the changes of s depend only on the camera motion [120].

$$\dot{s} = L_s.v_c \tag{4.2}$$

where v_c presents the spatial velocity of the camera and L_s is the interaction matrix or Jacobian matrix, which links the motion of the features s to that of the camera. Deriving the equation 4.1 and substituting in 4.2 we obtain:

$$\dot{e} = L_e.v_c \tag{4.3}$$

where the interaction matrix L_e is related to the error e. We consider a decreases exponentially of the error i.e.,

$$\dot{e} = -\lambda.e \tag{4.4}$$

We substitute the equation 4.4 in 4.3, the resulting control law will be:

$$v_c = -\lambda . L_e^{-1} . e \tag{4.5}$$

L_e will not always be an invertible matrix, as it has been shown that its dimensions depend on the number of features k [18, 125]. Even if $L_e \in R^{6\times 6}$, its determinant is not always equal to zero. Thus, it is preferable to adopt the Moore-Penrose pseudo-inverse matrix of L_e:

$$L_e^+ = (L_e^T . L_e)^{-1} L_e^T \tag{4.6}$$

In the literature, it is difficult to know the value of L_e, so an approximation is necessary. The procedure for calculating the interaction matrix is only an estimation because there is no way to know exactly the value of the depth (third component) of each point. For this reason, the value of the third component must be estimated from other sources. There are several ways for constructing the estimate $\hat{L}_e^+$ to be used in the control law. Some of the proposed approaches propose that if the current depth Z of each point is available and $L_e = L_x$ is known, they consider $\hat{L}_e^+ = L_e^+$. Practically, at each iteration of the control scheme, this parameter must be estimated. The more practical solution is proposed in [124], they estimate only the desired depth of all points and choose $\hat{L}_e^+ = L_{e^*}^+$ where, $L_{e^*}^+$ is constant, which means that the 3-D parameters will not be estimated.

4.3.4.2 Dynamic Target

In the case of moving object, the equation 4.3 is modified to consider the generally unknown target motion:

$$\dot{e} = L_e v_c + \frac{\partial e}{\partial t} \tag{4.7}$$

By considering an exponential decay of the error given by equation 4.4, the control law becomes as follows:

$$v_c = -\lambda L_e^+ e - L_e^+ \frac{\widehat{\partial e}}{\partial t} \tag{4.8}$$

where $\frac{\widehat{\partial e}}{\partial t}$ is an estimation of the compensation term $\frac{\partial e}{\partial t}$, which can be obtained using the error and the velocity components from the previous time step.
From the equation 4.7, we get:

$$\frac{\widehat{\partial e}}{\partial t} = (e(t) - e(t-\triangle t))/\triangle t - L_e v_c(t-\triangle t) \tag{4.9}$$

We note that the use of larger control gain λ may omit the compensation term. However, it can make the quadrotor system unstable.

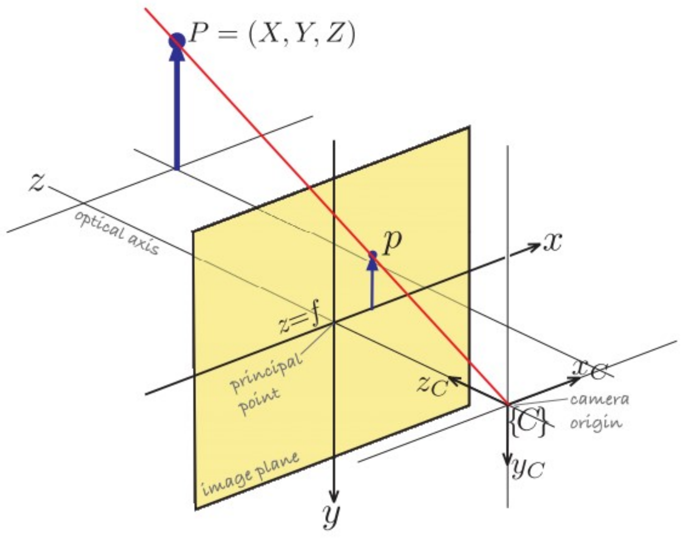


Figure 4.7: The central projection model [18].

4.3.5 Fundamentals of vision

In this section, we discuss the fundamentals of computer vision that underpin the work being addressed.

4.3.5.1 Perspective transformation

There are several ways to design the model of digital images. Among them, the commonly used is the **Pinhole** model or the central perspective model shown in figure 4.7, which allows the camera's physical model to be correctly approximated and thus to give an appropriate relationship between the 3-D and 2-D coordinates of the object. The pinhole model aims to extract the 2-D coordinates of the object using a perspective projection of the 3-D coordinates [131]. This model is based on rays that converge on the origin C of the camera frame called **projection center**.

Using similar triangles, we can show that a point with real coordinates $P = (X,Y,Z)$ is projected on the plane of the image $p = (x, y)$, considering that the non-inverted image is projected on the image plane located at $z = f$, and the path of light perceived by the camera is a line passing through p and C from P. The intersection of this line with the projection plane is the projection of the point P. The equation 4.10 links the coordinates of a point in space in R^3 with the coordinates of its projection in the image plane [18, 131].

$$x = f.\frac{X}{Z} \quad , \quad y = f.\frac{Y}{Z} \tag{4.10}$$

Preserving homogenization of coordinates on the projective space regardless of the distance

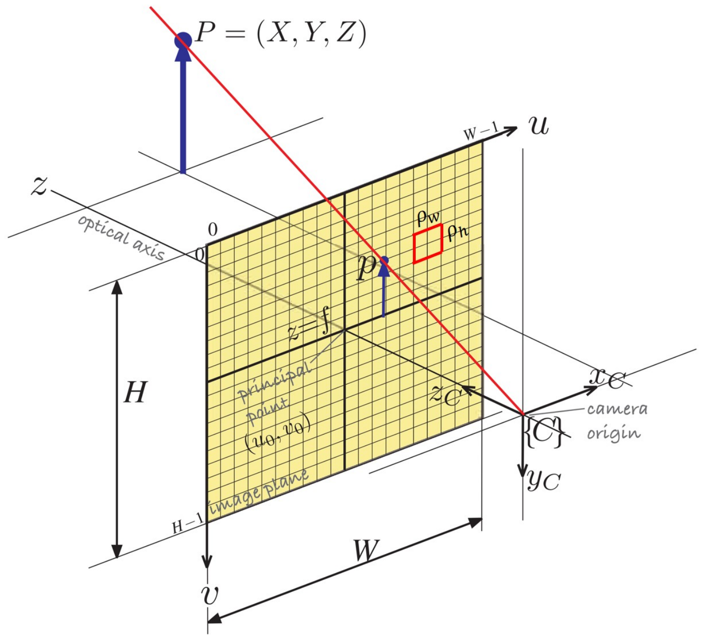


Figure 4.8: Illustration of the central projection model, the image plane and the discrete pixels [18].

between P and C requires the use of homogeneous coordinates to rewrite the perspective projection in linear form as [18]:

$$p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} {}^cX \\ {}^cY \\ {}^cZ \\ 1 \end{bmatrix} \tag{4.11}$$

where the homogeneous coordinates relative to the camera frame C are: ${}^cP = ({}^cX, {}^cY, {}^cZ, 1)^T$ and $p = (x, y, 1)^T$.

4.3.5.2 Image plane

In a digital camera, the image plane is a $W \times H$ grid of light-sensitive elements, called **photosites**, that correspond to components called pixels, as shown in figure 4.8. The coordinates of the pixels are presented by a vector of non-negative integers (u, v), defining the position of each pixel on this grid. The origin is usually at the top left of the image plane [18]. The transformation for passing the point coordinates p to the pixel coordinates as:

$$u = \frac{x}{\rho_w} + u_0 \quad , \quad v = \frac{y}{\rho_h} + v_0 \tag{4.12}$$

where ρ_w and ρ_h are respectively the width and height of each pixel, and (u_0, v_0) is the main point.
This relationship is written as [131]:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\rho_w} & 0 & u_0 \\ 0 & \frac{1}{\rho_h} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.13}$$

From the equations 4.11 to 4.13, the projection relation of a point P, of coordinates ${}^cP = ({}^cX, {}^cY, {}^cZ)^T$ in the image plane is given by [131]:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\rho_w} & 0 & u_0 \\ 0 & \frac{1}{\rho_h} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} {}^cX \\ {}^cY \\ {}^cZ \\ 1 \end{bmatrix} \tag{4.14}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{\rho_w} & 0 & 0 & 0 \\ 0 & \frac{f}{\rho_h} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} {}^cX \\ {}^cY \\ {}^cZ \\ 1 \end{bmatrix} \tag{4.15}$$

4.3.5.3 Camera's model and image plane's dynamics

The considered system in this thesis is presented in figure 4.9, which based on three essential frames (inertial frame I, camera frame C and target frame O). The camera moves with the speed v_c as the object moves with the speed v_o. The camera model is considered as described in the previous section (perspective projection model). We can write:

$$r_c + X = r_o \tag{4.16}$$

Where: r_c presents the camera's position with respect to the inertial frame, r_o presents the object's position with respect to the inertial frame and X presents the distance between the object and the camera in space. We take the derivative on both sides to obtain:

$$v_c + \dot{X} = v_o \tag{4.17}$$

With: v_o and v_c are expressed in the inertial frame I. The measure of X gives:

$$\dot{X} = \overset{\circ}{X} + \omega_c \times X \tag{4.18}$$

where $\overset{\circ}{X}$ is the derivative of X corresponding to the camera frame and ω_c shows the rotation speed of the camera relative to the inertial frame I. We obtain as follows:

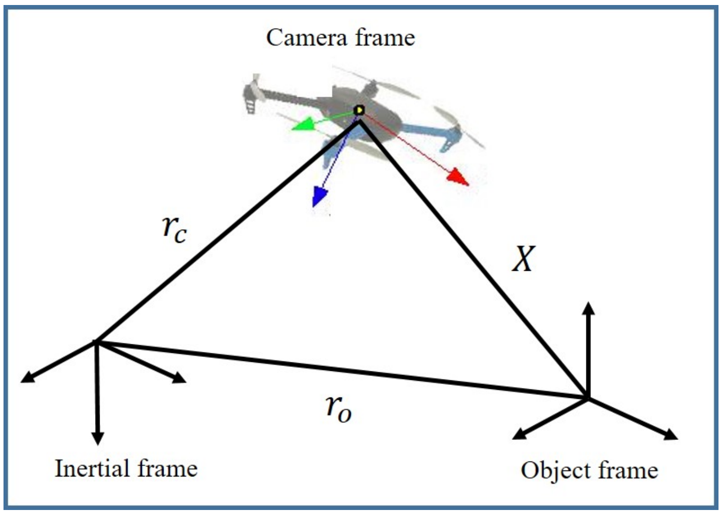


Figure 4.9: The relation between frames.

$$\overset{\circ}{X} = -v_c - \omega_c \times X + v_o \tag{4.19}$$

The speed coordinates of a target point are measured in the camera frame as follows:

$$\begin{bmatrix} {}^c\dot{X} \\ {}^c\dot{Y} \\ {}^c\dot{Z} \end{bmatrix} = -\begin{bmatrix} v_{cx} \\ v_{cy} \\ v_{cz} \end{bmatrix} + \begin{bmatrix} v_{ox} \\ v_{oy} \\ v_{oz} \end{bmatrix} + \begin{bmatrix} \omega_z {}^cY - \omega_y {}^cZ \\ \omega_x {}^cZ - \omega_z {}^cX \\ \omega_y {}^cX - \omega_x {}^cY \end{bmatrix} \tag{4.20}$$

with : $v_c = [v_{c_x}, v_{c_y}, v_{c_z}, \omega_{c_x}, \omega_{c_y}, \omega_{c_z}]^T$.
Deriving the equation 4.10, we obtain:

$$\begin{aligned} \dot{x} &= f\frac{\dot{X}_c}{{}^cZ} - f\frac{X_c {}^c\dot{Z}}{{}^cZ^2} \\ \dot{y} &= f\frac{\dot{Y}_c}{{}^cZ} - f\frac{Y_c {}^c\dot{Z}}{{}^cZ^2} \end{aligned} \tag{4.21}$$

Using the equation 4.20 and the equation 4.21 we immediately obtain:

$$\begin{aligned} \dot{x} &= -\frac{v_{cx} - v_{ox}}{{}^cZ} + x\frac{v_{cz} - v_{oz}}{{}^cZ} + xy\omega_{cx} - (1+x^2)\omega_{cy} + y\omega_{cz} \\ \dot{y} &= -\frac{v_{cy} - v_{oy}}{{}^cZ} + y\frac{v_{cz} - v_{oz}}{{}^cZ} - xy\omega_{cy} + (1+y^2)\omega_{cx} - x\omega_{cz} \end{aligned} \tag{4.22}$$

with: $s = [x \quad y]^T$. Finally, the relation between the temporal variation of the image features $\dot{s}$ and the spatial velocity of the camera v_c is expressed through equation 4.23.

$$\dot{s} = L_e v_{co} \tag{4.23}$$

where:

$$L_e = \begin{bmatrix} -\frac{1}{Z_c} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z_c} & \frac{y}{Z} & (1+x^2) & -xy & -x \end{bmatrix} \tag{4.24}$$

and:

$$v_{co} = \begin{bmatrix} v_{cx} - v_{ox} \\ v_{cy} - v_{oy} \\ v_{cz} - v_{oz} \\ \omega_{cx} \\ \omega_{cy} \\ \omega_{cz} \end{bmatrix} \tag{4.25}$$

4.3.6 Stability analysis

4.3.6.1 Global asymptotic stability

To ensure the stability of the system, an analysis is done. We used the Lyapunov analysis [120]. We consider the candidate Lyapunov function defined by the standard squared error:

$$\mathscr{L} = \frac{1}{2}||e(t)||^2 \tag{4.26}$$

whose derivative is given by:

$$\begin{aligned} \dot{\mathscr{L}} &= e^T \dot{e} \\ \dot{\mathscr{L}} &= -\lambda e^T L_e \hat{L}_e^+ e \end{aligned} \tag{4.27}$$

Overall asymptotic stability is guaranteed when the condition given by equation 4.28 is fulfilled so that the product matrix is positive definite.

$$L_e \hat{L}_e^+ > 0 \tag{4.28}$$

4.3.6.2 Local asymptotic stability

It is deduced from [120] that local stability can be ensured with a new error:

$$e' = \hat{L}_e^+ e \tag{4.29}$$

whose derivative is given by:

$$\begin{aligned} \dot{e}' &= \hat{L}_e^+ \dot{e} + \dot{\hat{L}}_e^+ e \\ \dot{e}' &= (\hat{L}_e^+ L_e + O) V_c \end{aligned} \tag{4.30}$$

Where O is a matrix $\in R^{6\times6}$ that will be zero, when $e = 0$ and close to zero in the neighborhood of the reference. Using the equation 4.5, we obtain:

$$\dot{e}' = -\lambda(\hat{L}_e^+ L_e + O)e' \tag{4.31}$$

which guarantees a local stability in the neighborhood of $e = 0$ if $\hat{L}_e^+ L_e > 0$ is positive definite. This study is used in the next chapter to verify local stability when the drone is hovering at the reference point.

4.3.6.3 Stability proof for dynamic image

In IBVS, four image points are controlled. In the case of a dynamic image, this relation cannot be held because $\dot{s}^* \neq 0$. Taking the time derivative of equation 4.1 gives:

$$\dot{e}(t) = \dot{s}(t) - \dot{s}^* \tag{4.32}$$

thus:

$$-\lambda e = L_e v_c - \dot{s}^* \tag{4.33}$$

This is rearranged to obtain the desired control law, with the entry v_c:

$$v_c = \hat{L}_e^+(-\lambda e + \dot{s}^*) \tag{4.34}$$

Here, the Moore-Penrose pseudo-inverse given in equation 4.6 is used. Based on Theorem 4.18 of Khalil's manual [132], where it is described in the following [133]:
Lemma 1. Let $D \subset \mathscr{R}^n$ a domain that contains the origin and $V : [0,\infty) \times D \longrightarrow \mathscr{R}$ is a continuous and differentiable function, such as:

$$\alpha_1(||x||) \leq V(t,x) \leq \alpha_2(||x||) \tag{4.35}$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t,x) \leq -W_3(x), \forall ||x|| \geq \mu > 0 \tag{4.36}$$

$\forall t \geq 0$ and $\forall x \in D$, where α_1 and α_2 are class functions K and $W_3(x)$ is a continuous positive definite function. Taking $r > 0$ such as $B_r \subset D$ and suppose that:

$$\mu < \alpha_2^{-1}(\alpha_1(r)) \tag{4.37}$$

So, there is a function β of class KL. The initial state $x(t_0)$ that satisfies $||x(t_0)|| \leq \alpha_2^{-1}(\alpha_1(r))$, there is $T \geq 0$ (Depends on $x(t_0)$ and μ) such as the solution satisfied :

$$||x(t)|| \leq \beta(||x(t_0)||, t - t_0), \forall t_0 \leq t \leq t_0 + T \tag{4.38}$$

$$||x(t)|| \leq \alpha_1^{-1}(\alpha_2(\mu)), \forall t \geq t_0 + T \tag{4.39}$$

Moreover, if $D = \mathscr{R}^n$ and α_1 belong to the class K_∞, the two previous equations are valid for any initial state $x(t_0)$, without restriction as to the size of μ.
In the lemma above, a class function K is a continuous function $\alpha : [0,a) \longrightarrow [0,\infty)$ which

strictly increases and has $\alpha(0) = 0$. It is furthermore $K1$ if $a = \infty$ and $\alpha(r) \longrightarrow \infty$ as $r \longrightarrow \infty$. A class function KL is a continuous function $\beta : [0,a) \times [0,\infty) \longrightarrow [0,\infty)$ if for each fixed s, correspondence $\beta(r,s)$ is of class K compared to r and, for each fixed r, the correspondence $\beta(r,s)$ decreases compared to s and $\beta(r,s) \longrightarrow 0$ as $s \longrightarrow \infty$. We consider the candidate Lyapunov function defined by equation 4.26. Then by appling equations 4.27 and 4.32 lead to:

$$\dot{\mathscr{L}} = e^T(\dot{s}(t) - \dot{s}^*) \tag{4.40}$$

By replacing in equations 4.2 and 4.34:

$$\begin{aligned} \dot{\mathscr{L}} &= e^T[L_e\hat{L}_e^+(-\lambda e + \dot{s}^*) - \dot{s}^*] \\ \dot{\mathscr{L}} &= -\lambda e^T L_e \hat{L}_e^+ + e^T L_e \hat{L}_e^+ \dot{s}^* - e^T \dot{s}^* \end{aligned} \tag{4.41}$$

We pose:

$\Lambda = (1 - L_e\hat{L}_e^+)$

$$\dot{\mathscr{L}} = -\lambda e^T L_e \hat{L}_e^+ - e^T \Lambda \dot{s}^* \tag{4.42}$$

We observe that:

$$e^T \Upsilon \le \frac{1}{2} e^T e + \frac{1}{2} \Upsilon^T \Upsilon \tag{4.43}$$

By taking $\Upsilon = \Lambda \dot{s}^*$, we obtain:

$$e^T \Lambda \dot{s}^* \le \frac{1}{2} e^T e + \frac{1}{2} \dot{s}^{*T} \Lambda^T \Lambda \dot{s}^* \tag{4.44}$$

thus:

$$\dot{\mathscr{L}} \le -\lambda e^T L_e \hat{L}_e^+ e + \frac{1}{2} e^T e + \frac{1}{2} \dot{s}^{*T} \Lambda^T \Lambda \dot{s}^* \tag{4.45}$$

Tacking $\Gamma = (\lambda L_e \hat{L}_e^+ - \frac{1}{2})$, in this case, there exists a quadratic form of a real, non-symmetrical, square matrix. Due to the quadratic form $e^T \Gamma e$, only the symmetrical part of Γ will affect the result:

$$\Gamma_s = \frac{\Gamma + \Gamma^T}{2} \tag{4.46}$$

As this part is symmetrical and square, its eigenvalues will be real. We further assume that Γ_s is set positive. This allows you to write:

$$\dot{\mathscr{L}} \le -\lambda_{min}(\Gamma_s) ||e||^2 + \frac{1}{2} \dot{s}^{*T} \Lambda^T \Lambda \dot{s}^* \tag{4.47}$$

Similarly, it is easily proven that $\Lambda^T \Lambda$ is semi-definite positive.
Thus : $\dot{s}^{*T} \Lambda^T \Lambda \dot{s}^* \leq \lambda_{max}(\Lambda^T \Lambda)||\dot{s}^*||^2$

$$\dot{\mathscr{L}} = -\lambda_{min}(\Gamma_s)||e||^2 + \lambda_{max}(\Lambda^T \Lambda)||\dot{s}^*||^2 \tag{4.48}$$

This is an equation of the form by $by^2 - ax^2 = (\sqrt{b}y + \sqrt{a}x)(\sqrt{b}y - \sqrt{a}x)$, thus :

$$(\sqrt{b}\sqrt{||\dot{s}^*||^2} + \sqrt{a}\sqrt{||e||^2})(\sqrt{b}\sqrt{||\dot{s}^*||^2} - \sqrt{a}\sqrt{||e||^2}) \tag{4.49}$$

The first term is always positive. The result will be negative if the second term is negative. As a result, the requirement is that:

$$\begin{gathered} \sqrt{b}\sqrt{||\dot{s}^*||^2} - \sqrt{a}\sqrt{||e||^2} \leq 0 \\ ||e||^2 \geq \frac{b}{a}||\dot{s}^*||^2 \end{gathered} \tag{4.50}$$

So, if the error is greater than $\sqrt{\frac{\lambda_{max}(\Lambda^T \Lambda)}{\lambda_{min}(\Gamma_s)}}||\dot{s}^*||$, the derivative of the Lyapunov function will be negative. As it is not possible to determine when the error is equal to zero as long as the desired image points move. The error will converge to zero once the points have stopped moving.
Returning to the theorem 4.18 [132]. We are looking for the functions of class K defined continuous, $[0,a) \longrightarrow [0,\infty)$, strictly growing, and having $\alpha(0) = 0$. The function is K_∞ and even have $a = \infty$ and $\alpha(r) \longrightarrow \infty$ for $r \longrightarrow \infty$.
By selecting $\alpha_1(e) = \frac{1}{4}||e||^2$ and $\alpha_2(e) = ||e||^2$, with: $\alpha_1^{-1}(e) = 2\sqrt{e}$ and $\alpha_2^{-1}(e) = \sqrt{e}$, these functions are continuous, strictly increasing, $\alpha_1(0) = \alpha_2(0) = 0$. Moreover, they go to infinity as e goes to infinity. So these functions are K_∞, and satisfy equation 4.35.

$$\frac{1}{4}||e||^2 \leq \frac{1}{2}||e||^2 \leq ||e||^2 \tag{4.51}$$

W_3 must be a continuous positive definite function. In this case, choose $W_3(e) = \lim\limits_{c \to 0^+} ce^2$, which means that for reasonable values of e and $\dot{s}^*$, the function $W_3(e) \approx 0$. Because $\dot{V}(e) \leq 0$ is true for:

$$e \geq \sqrt{\frac{\lambda_{max}(\Lambda^T \Lambda)}{\lambda_{min}(\Gamma_s)}}||\dot{s}^*|| \quad , \quad W_3(e) \approx 0$$

In equation 4.36, μ can be set to $\mu = (\sqrt{\frac{\lambda_{max}(\Lambda^T \Lambda)}{\lambda_{min}(\Gamma_s)}}||\dot{s}^*|| + 0.1)$, since $\dot{V}$ must always be negative for $\mu \geq ||e||$, and it is possible to have $\dot{V} = 0$ when $\mu = \sqrt{\frac{\lambda_{max}(\Lambda^T \Lambda)}{\lambda_{min}(\Gamma_s)}}||\dot{s}^*||$.
Indeed, $D = \mathscr{R}^n$ and $\alpha_1(e)$ is K_∞, so the equations below are worth for all $x(t_0)$, without restriction on μ. We thus satisfied the conditions of the theorem 4.18 [132] and there exists a time T such that:

$$||e|| \leq \beta(e(t), t) \quad , \quad \forall t_0 \leq t \leq T \tag{4.52}$$

$$||e|| \leq 2\mu \quad , \quad \forall t \geq T \tag{4.53}$$

Thus:

$$||e|| \leq 2\sqrt{\frac{\lambda_{max}(\Lambda^T \Lambda)}{\lambda_{min}(\Gamma_s)}} ||\dot{s}^*||_{max} + 0.2 \quad , \quad \forall t \geq T \tag{4.54}$$

Where β is an unspecified function of KL class. Note that the t_0 dependency has been removed because our system is standalone, thus, the dynamics do not change over time. The observed error will be limited by the movement's speed of displacement of the desired image, scaled by the eigenvalues of the matrices Γ_s and $\Lambda^T \Lambda$ which means that the camera can instantly accelerate to the desired value.

4.4 Conclusion

In this chapter, we have given an overview of the drones. Then we presented a detailed review of the visual servoing approaches applied to them. After discussing the definition of visual servoing and its three approaches, we have described the basic concepts of visual servoing in terms of the digital image plane and the projection model and even the stability proof of the IBVS approach.

CHAPTER 5

IMAGE-BASED CONTROL LAW DESIGN FOR QUADROTOR CONTROL

5.1 Introduction

This chapter presents the modeling of a drone and the design of the control laws for a visual tracking application. We adopt the basic concept of the IBVS approach to allow the quadrotor to autonomously tracking a target object by keeping it centered on the image plane. The points that we are interested in are described as follows:

- The dynamic model of the system and the design of a proportional-derivative controller that stabilizes the quadrotor. Indeed, we propose to optimize the performances of this controller by the bat algorithm (BA), where its utility is to minimize the error by optimizing the gains of the PD controller.
- The image-based controller design takes into account both the dynamics of the quadrotor and the movement of the target;
- Detect and maintain accurately the target in the camera's FOV.

This chapter is organized as follows. Section 5.2 describes the quadrotor's dynamics as well as the considered model. Then, we present the control scheme based on the BA and the verification of its validity in section 5.3. Section 5.4 presents the simulation of the control scheme based on the IBVS approach.

5.2 Modeling and dynamics of the quadrotor

5.2.1 Operation

A quadrotor is a helicopter with four rotors mounted symmetrically around its center. Its movement results from the increase or decrease of rotational speeds of all rotors. The quadrotor design is illustrated in figure 5.1.a. Two motors mounted on the same arm rotate in a different direction from the two other motors mounted on the second arm, which cancels the aerodynamic effects and gyro moments in the hovering of [134, 135]. Before deriving the quadrotor model, it is important to explain the mechanism that the quadrotor uses in its motion shown in figure 5.1.b.

- **Lateral motion:** This movement is achieved by decreasing the rotational speeds of the rotors and by increasing the rotational speeds of the opposite rotors to the desired direction of movement.
- **Longitudinal movement** The longitudinal movement is obtained by a rotational moment of the rotors at different speeds, in the same way, that a lateral movement [136].

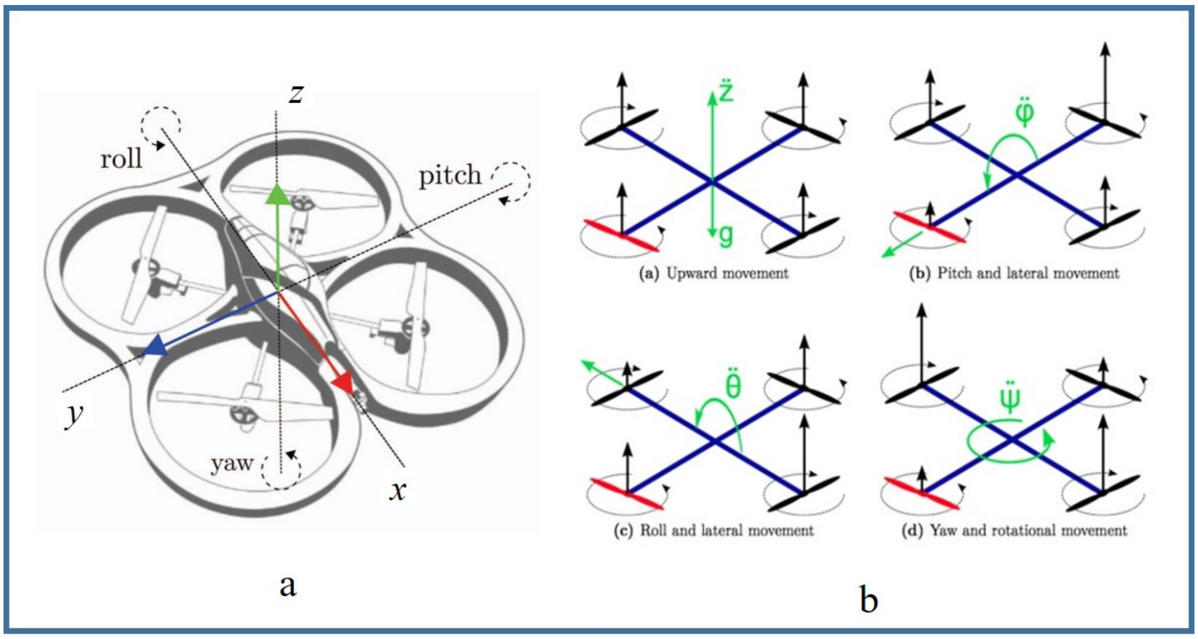

z
roll
pitch
y
yaw
x
a
z̈
g
(a) Upward movement
φ̈
(b) Pitch and lateral movement
θ̈
(c) Roll and lateral movement
ψ̈
(d) Yaw and rotational movement
b

Figure 5.1: (a) Quadrotor design, (b) Quadrotor basic movements [19].

- **Altitude motion:** This movement is obtained by modifying the speed of all the rotors of the same quantity, which raises or decreases the quadrotor with respect to the reference frame [137, 138].
- **Roll motion** The roll motion is a rotation obtained by simultaneously reversing the speed of the left and right rotors. This rotation (ϕ) is produced around the axis x which causes the quadrotor to move in the direction of the axis y.
- **Pitch motion:** This movement is a rotation obtained by simultaneously reversing the speed of the front and rear rotors. The pitching motion (θ) is the rotational motion around the y axis that causes the quadrotor to move in the direction of the x axis.
- **Yaw motion:** The yaw movement (ψ) is the rotational movement around the z axis, obtained by simultaneously reversing the speed of the front and rear rotors and the left-right torque.

5.2.2 Mathematical model of quadrotor

The system modeling is an essential step in the field of robotics, especially to verify the suitable behavior of drones to realize a real application. In this section, we present the model of the quadrotor **Parrot AR Drone 2.0** (described in detail in the next chapter).

5.2.2.1 Model assumptions

Due to the non-linearity and the strong coupling of the system dynamics, it can be difficult to develop the right model. For that, we present hypotheses simplifying the modeling task [139, 140]:

- The structure is rigid and symmetrical.
- The center of gravity of the quadrotor coincides with the origin of its body.
- The propellers are rigid.
- The thrust and the drag are proportional to the square of the speed of the propeller.

5.2.2.2 Frames and notations

We first define the frames that they will be used. The modeling of quadrotor consists of establishing a relationship between its body frame and the inertial frame as a function of its positions (x, y, z) and Euler angles (ϕ, θ, ψ). Figure 5.2 shows the reference frames used to describe the movement of the drone in space. The inertial frame is attached to a specific location at the ground. Concerned, which is the body frame B, it coincides with the center of gravity of the vehicle.

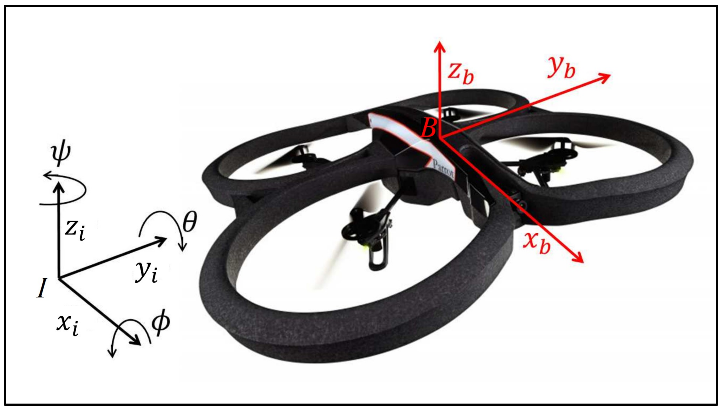


Figure 5.2: The reference frames.

- **Rotation matrix** R
  To describe the orientation of the quadrotor in space, two intermediate coordinate systems must be defined; the vehicle-1 frame v_1 and the vehicle-2 frame v_2 with the two previously defined frames [141, 142].

$$R_B^I = R_{v_2}^I . R_{v_1}^{v_2} . R_B^{v_1} \tag{5.1}$$

$$R_B^I = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \tag{5.2}$$

$$R_B^I = \begin{bmatrix} c\theta c\psi & c\psi s\theta s\phi - s\psi c\theta & c\psi s\theta c\phi + s\psi s\phi \\ c\theta s\psi & s\psi s\theta s\phi + c\psi c\theta & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{5.3}$$

where $R_{v_1}^I$ presents the transformation matrix of the reference v_1 to the frame I, $R_{v_1}^{v_2}$ presents the transformation matrix of the frame v_1 to the frame v_2 and $R_B^{v_2}$ presents the transformation matrix of the frame B to the frame v_2. With $c(.)$ And $s(.)$ Present the mathematical abbreviations for $cos(.)$ and $sin(.)$ functions respectively.

- **Velocities**

Using equation 5.3, the linear velocities between the two frames I and B are expressed as:

$$\begin{bmatrix} v_x^I \\ v_y^I \\ v_z^I \end{bmatrix} = R_B^I . \begin{bmatrix} v_x^B \\ v_y^B \\ v_z^B \end{bmatrix} \tag{5.4}$$

Concerning the angular velocities, the transformation between the rotational speeds in the fixed frame I ($\dot{\zeta} = [\dot{\theta}, \dot{\phi}, \dot{\psi}]^T$) and those in the body frame B ($\dot{\omega} = [\dot{p}, \dot{q}, \dot{r}]^T$) is represented in equation 5.5 as follows [142]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{p} \\ 0 \\ 0 \end{bmatrix} + R_B^{v_1 -1} . \begin{bmatrix} 0 \\ \dot{q} \\ 0 \end{bmatrix} + (R_{v_1}^{v_2} . R_B^{v_1})^{-1} . \begin{bmatrix} 0 \\ 0 \\ \dot{r} \end{bmatrix} \tag{5.5}$$

Thus, we find:

$$\dot{\zeta} = \begin{bmatrix} 1 & 0 & -s\phi \\ 0 & c\theta & s\theta c\phi \\ 0 & -s\theta & c\theta c\phi \end{bmatrix} . \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{5.6}$$

In the case of small rotations:

$$\begin{cases} c\theta = c\phi = c\psi = 1 \\ s\theta = s\phi = s\psi = 0 \end{cases} \tag{5.7}$$

The ω and ζ vectors are similar.

- **Aerodynamic effects**

To design an accurate and realistic model, the aerodynamic effects must be included. There are two types of aerodynamic effects [143], forces and moments (Figure 5.3).

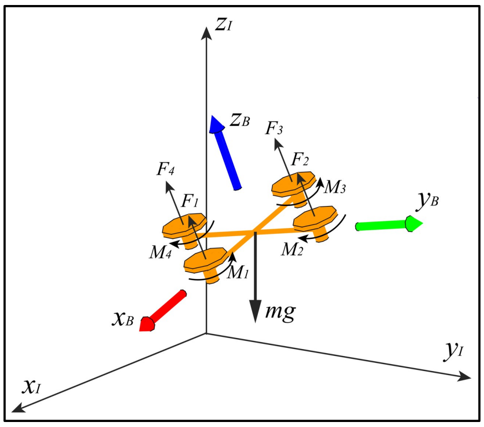


Figure 5.3: Forces and moments acting on the quadrotor [20].

* **Forces**

  · **The quadrotor's weight**: This force results from the impact of the gravity g on the weight of quadrotor m.

$$P = m.g \tag{5.8}$$

  · **Lift Force**: As given by equation 5.9, this force results from the sum of the forces F_i produced by the rotation of the drone's engines. Each force F_i is proportional to the square of a corresponding rotational speed w_i, whose direction is perpendicular to the propellers' plane.

$$\begin{gathered} F_i = k.w_i^2 \\ T = \sum_{i=1}^{4} F_i \end{gathered} \tag{5.9}$$

Where k presents the lift coefficient.

  · **Drag force**: There are two drag forces acting on the system:

1- A force acts on the propellers. It is proportional to the square of the rotational speed of each rotor w_i

$$D_r = b_r.w_i^2 \qquad , \qquad i = 1,...,4 \tag{5.10}$$

Where b_r is the coefficient of the drag.

2- A force resulted from the movement of the body along the axes (x, y, z), given by:

$$D_m = b_m.v \tag{5.11}$$

where v presents the linear velocity and b_m is the translation drag coefficient.

∗ **Moments**

There are two types of moments acting on the body of the quadrotor [142, 143]:

· **Moments that produce rotations around the axes** (x, y, z): The existence of a difference between the forces of push and/or drag causes the creation of these moments.

$$\begin{aligned} M_x &= l(f_4 - f_2) = l.k(w_4^2 - w_2^2) \\ M_y &= l(f_3 - f_1) = l.k(w_3^2 - w_1^2) \\ M_z &= D_{r2} + D_{r4} - D_{r1} - D_{r3} = b_r(w_4^2 + w_2^2 - w_3^2 - w_1^2) \end{aligned} \tag{5.12}$$

Where l is the distance between the center of mass of the quadrotor and the rotor's axis.

The values of w_i are extracted as follows:

$$\begin{aligned} w_1 &= \sqrt{\tfrac{T}{4k} - \tfrac{M_y}{2kl} - \tfrac{M_z}{4b_r}}, \\ w_2 &= \sqrt{\tfrac{T}{4k} - \tfrac{M_x}{2kl} + \tfrac{M_z}{4b_r}}, \\ w_3 &= \sqrt{\tfrac{T}{4k} + \tfrac{M_y}{2kl} - \tfrac{M_z}{4b_r}}, \\ w_4 &= \sqrt{\tfrac{T}{4k} + \tfrac{M_x}{2kl} + \tfrac{M_z}{4b_r}} \end{aligned}$$

· **Gyro moments** due of the following movements::

· Quadrotor movement.

$$Gyr_1 = \dot{\zeta} \wedge J\dot{\zeta} \tag{5.14}$$

with: J is the system's inertia, $\wedge$ is the vector product and $\dot{\zeta}$ is the angular velocity in the fixed frame I.

· The rotation of the propellers.

$$Gyr_2 = \sum_{i=1}^{4} \dot{\zeta} \wedge J_r[0 \quad 0 \quad (-1)^{i+1} w_i]^T \tag{5.15}$$

5.2.2.3 Mathematical model

The quadrotor's model can be divided into two subsystems; rotation subsystem (roll ϕ, pitch θ and yaw ψ) and translation (altitude z and positions x and y). The rotation subsystem is fully actuated while the translation subsystem is under-actuated [143–145].

1- **Translational motion equations**: We use Newton's second law, where the forces applied to the system are expressed in the fixed inertial frame I as:

$$m\ddot{\Theta} = F_g + F_t + F_d \tag{5.16}$$

with:
Θ: the quadrotor position vector $[x,y,z]^T$,
m: the total mass of the quadrotor,
F_d: drag force.

$$F_d = \begin{bmatrix} -a_x & 0 & 0 \\ 0 & -a_y & 0 \\ 0 & 0 & -a_z \end{bmatrix} \dot{\Theta} \tag{5.17}$$

Where a_x, a_y and a_z are the coefficients of the drag force.
F_t: total force generated by the rotors,

$$F_t = R_B^I \times \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{5.18}$$

F_g : force of gravity,

$$F_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{5.19}$$

Therefore, we get:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -mg. \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + T \begin{bmatrix} c\psi s\theta c\phi + s\phi s\psi \\ s\psi s\theta c\phi - c\psi s\phi \\ c\phi c\theta \end{bmatrix} - \begin{bmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & a_z \end{bmatrix} . \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{5.20}$$

We then obtain:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \frac{T}{m}[c\psi s\theta c\phi + s\phi s\psi] - \frac{a_x}{m}\dot{x} \\ \frac{T}{m}[s\psi s\theta c\phi - c\psi s\phi] - \frac{a_y}{m}\dot{y} \\ \frac{T}{m}[c\phi c\theta] - \frac{a_z}{m}\dot{z} - g \end{bmatrix} \tag{5.21}$$

2- **Rotational equations**: Idem, we use Newton's second law to develop equations of the rotational motion.

We have :

$$J\ddot{\zeta} = -Gyr_1 + M_f - Gyr_2 \tag{5.22}$$

with:

$$M_f = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} \tag{5.23}$$

$$J = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{5.24}$$

I_x, I_y and I_z present the coefficients of aerodynamic friction. Therefore, we get:

$$\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = - \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \wedge \left(\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \right) + \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} J_r \Omega_r \dot{\phi} \\ -J_r \Omega_r \dot{\theta} \\ 0 \end{bmatrix} \tag{5.25}$$

with:

$$\Omega_r = w_1 + w_3 - w_4 - w_2 \tag{5.26}$$

Finally, the rotation equations are given by:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{(I_y - I_z)}{I_x} \dot{\theta} \dot{\psi} \\ \frac{(I_z - I_x)}{I_y} \dot{\phi} \dot{\psi} \\ \frac{(I_x - I_y)}{I_z} \dot{\theta} \dot{\phi} \end{bmatrix} + \begin{bmatrix} \frac{-J_r \dot{\theta} \Omega_r}{I_x} \\ \frac{J_r \dot{\phi} \Omega_r}{I_y} \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{M_x}{I_x} \\ \frac{M_y}{I_y} \\ \frac{M_z}{I_z} \end{bmatrix} \tag{5.27}$$

The complete dynamic model of the quadrotor is as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{T}{m}(c\psi s\theta c\phi + s\phi s\psi) - \frac{a_x}{m}\dot{x} \\ \frac{T}{m}(s\psi s\theta c\phi - c\psi s\phi) - \frac{a_y}{m}\dot{y} \\ \frac{T}{m}(c\phi c\theta) - \frac{a_z}{m}\dot{z} - g \\ \frac{(I_y - I_z)}{I_x} \dot{\theta} \dot{\psi} - \frac{J_r \dot{\theta} \Omega_r}{I_x} + \frac{M_x}{I_x} \\ \frac{(I_z - I_x)}{I_y} \dot{\phi} \dot{\psi} + \frac{J_r \dot{\phi} \Omega_r}{I_y} + \frac{M_y}{I_y} \\ \frac{(I_x - I_y)}{I_z} \dot{\theta} \dot{\phi} + \frac{M_z}{I_z} \end{bmatrix} \tag{5.28}$$

3- **The state representation of the system**: In our case, we consider X as the state vector of the model:

$$X = [\phi \quad \dot{\phi} \quad \theta \quad \dot{\theta} \quad \psi \quad \dot{\psi} \quad x \quad \dot{x} \quad y \quad \dot{y} \quad z \quad \dot{z}]^T \tag{5.29}$$

Avec:

$$X = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8 \quad x_9 \quad x_{10} \quad x_{11} \quad x_{12}]^T \tag{5.30}$$

We obtain :

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{(I_y - I_z)}{I_x} x_4 x_6 - \frac{J_r x_4 \Omega_r}{I_x} + \frac{M_x}{I_x} \\ x_4 \\ \frac{(I_z - I_x)}{I_y} x_6 x_2 + \frac{J_r x_2 \Omega_r}{I_y} + \frac{M_y}{I_y} \\ x_6 \\ \frac{(I_x - I_y)}{I_z} x_2 x_4 + \frac{M_z}{I_z} \\ x_8 \\ \frac{T}{m}(cx_5 sx_3 cx_1 + sx_5 sx_1) - \frac{a_x}{m} x_8 \\ x_{10} \\ \frac{T}{m}(sx_5 sx_3 cx_1 - cx_5 sx_1) - \frac{a_y}{m} x_{10} \\ x_{12} \\ \frac{T}{m}(cx_3 cx_1) - \frac{a_z}{m} x_{12} - g \end{bmatrix} \tag{5.31}$$

Since the dynamics of the quadrotor is based on the creation of the rotations, it is inevitable to release the U commands of the quadrotor depending on these rotations, i.e.

$$\begin{bmatrix} T \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} k & k & k & k \\ 0 & -lk & 0 & lk \\ -lk & 0 & lk & 0 \\ -b_r & b_r & -b_r & b_r \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} \tag{5.32}$$

By putting:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} T \\ M_x \\ M_y \\ M_z \end{bmatrix} \tag{5.33}$$

The system state representation can be written as follows:

$$\dot{X} = f(X, U) \tag{5.34}$$

with :

$$f(X,U)=\begin{bmatrix} x_2 \\ \frac{(I_y-I_z)}{I_x}x_4x_6-\frac{J_rx_4\Omega_r}{I_x}+\frac{U_2}{I_x} \\ x_4 \\ \frac{(I_z-I_x)}{I_y}x_6x_2+\frac{J_rx_2\Omega_r}{I_y}+\frac{U_3}{I_y} \\ x_6 \\ \frac{(I_x-I_y)}{I_z}x_2x_4+\frac{U_4}{I_z} \\ x_8 \\ \frac{U_1}{m}(cx_5sx_3cx_1+sx_5sx_1)-\frac{a_x}{m}x_8 \\ x_{10} \\ \frac{U_1}{m}(sx_5sx_3cx_1-cx_5sx_1)-\frac{a_y}{m}x_{10} \\ x_{12} \\ \frac{U_1}{m}(cx_3cx_1)-\frac{a_z}{m}x_{12}-g \end{bmatrix} \tag{5.35}$$

The parameters of the model are shown in table 5.1 [140]:

Table 5.1: The parameters of the model

Parameter	*Description*	*Value*
k	Coefficient of lift	$8.048\times10^{-6}N.S^2$
b_r	Coefficient of drag	$2.43\times10^{-7}Nm.s^2$
I_x	Moment of inertia along the axis x	$0.002237568kg.m^2$
I_y	Moment of inertia along the axis y	$0.002985236kg.m^2$
I_z	Moment of inertia along the axis z	$0.00480374kg.m^2$
a_x,a_y,a_z	coefficients of the drag force	$0.25Nm.s^2$
l	Distance between the center of gravity and each propeller	$0.1785m$
m	Mass	$0.450kg$
g	Gravity	$9.81m/s^2$
J_r	Rotor inertia	$2.029585\times10^5kg.m^2$

5.3 Control law design

The aspect of servoing is the essential and often necessary aspect in the robotic field when it is expected that the system behaves in the desired manner depending on the inputs. In other words, the looped system can maintain a signal of input (set of points for example) by correcting the produced error. More precisely, the system must base on a designed control law that corrects the measured error between the input and the output. In this section, we present a control system based on the hybridization between the 2-D visual servoing approach, and the optimization of the PD's gains using the bat algorithm (BA), where the goal is to stabilize the mathematical model previously developed in the course of performing properly the task of visual object tracking.

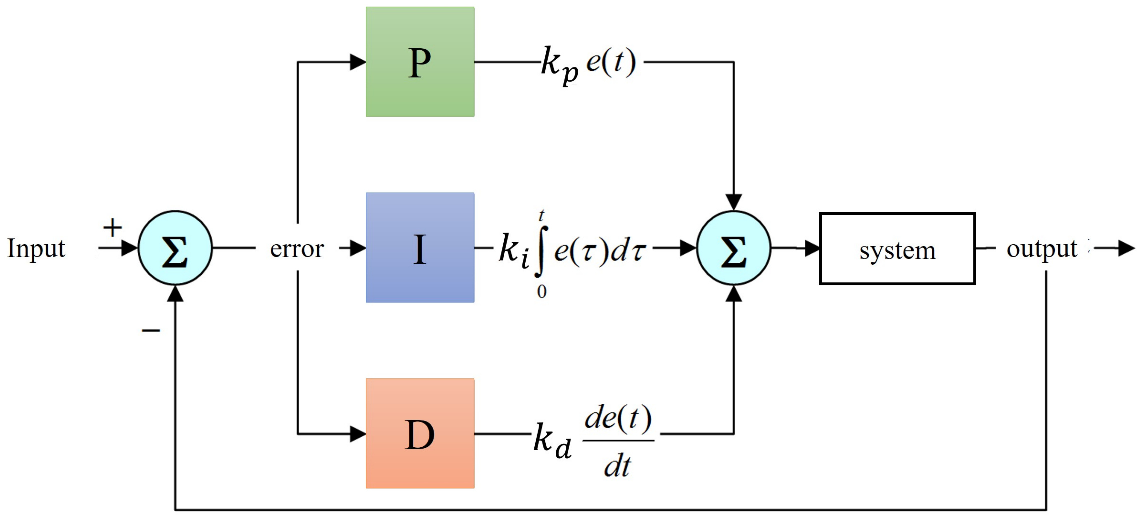


Figure 5.4: PID controller architecture.

5.3.1 The PID controller

As its name suggests, the principle of the Proportional (P) Integral (I) Derivative (D) controller resides in three essential actions:

* **Proportional action**: The error is multiplied by a gain k_p.
* **Integral action**: The error is integrated over an interval t, then multiplied by a gain k_i.
* **Derivative action**: The error is derived over an interval t, then multiplied by a gain k_d.

The combination of these three actions differs into three architectures: series, parallel or mixed. In our work, we use the parallel architecture shown in figure 5.4.

5.3.1.1 Altitude and orientation controllers

Since the quadrotor is an under-actuated system, four PID controllers are developed to control the altitude z, the roll ϕ, the pitch θ and the yaw ψ motions. The PID controller has been simplified to become a PD controller since no steady-state error has been observed (Figure. 5.5), where the responsible vector for controlling the quadrotor to the desired inputs is given as follows:

$$[U_z \quad U_\phi \quad U_\theta \quad U_\psi]^T = [U_1 \quad U_2 \quad U_3 \quad U_4]^T$$

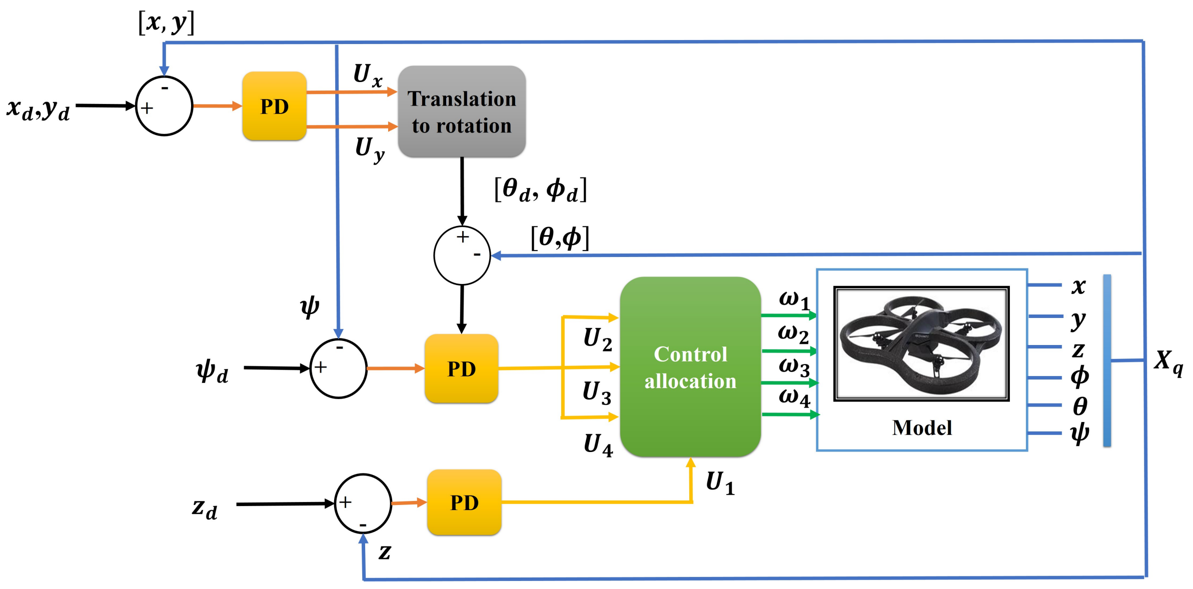


Figure 5.5: System architecture.

With:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} k_{p_1}(z_d - z) + k_{d_1}(\dot{z_d} - \dot{z}) \\ k_{p_2}(\phi_d - \phi) + k_{d_2}(\dot{\phi_d} - \dot{\phi}) \\ k_{p_3}(\theta_d - \theta) + k_{d_3}(\dot{\theta_d} - \dot{\theta}) \\ k_{p_4}(\psi_d - \psi) + k_{d_4}(\dot{\psi_d} - \dot{\psi}) \end{bmatrix} \tag{5.37}$$

The symbols U_x and U_y present two commands, called virtual, used to generate the desired angles ϕ_d and θ_d.

$$\begin{aligned} U_x &= k_p(x_d - x) + k_d(\dot{x_d} - \dot{x}) \\ U_y &= k_p(y_d - y) + k_d(\dot{y_d} - \dot{y}) \end{aligned} \tag{5.38}$$

$$\text{where: } \begin{cases} k_p\text{: Proportional gain,} \\ k_d\text{: Derived gain,} \\ z_d\text{: desired altitude,} \\ \phi_d\text{: desired roll,} \\ \theta_d\text{: desired pitch,} \\ \psi_d\text{: Desired yaw.} \end{cases}$$

5.3.1.2 Position controller

As shown in figure 5.5, it is clear that the quadrotor's positions along the axis x and y are not directly controlled by the commands $U_1, .., U_4$. The roll and pitch commands will be used as an intermediates to calculate the desired accelerations $\ddot{x}$ and $\ddot{y}$.

By giving :

$$\begin{aligned} U_x &= \ddot{x}_d = \tfrac{U_1}{m}(c\psi s\theta_d c\phi_d + s\psi s\phi_d) - \tfrac{a_x}{m}\dot{x} \\ U_y &= \ddot{y}_d = \tfrac{U_1}{m}(s\psi s\theta_d c\phi_d - c\psi s\phi_d) - \tfrac{a_y}{m}\dot{y} \end{aligned} \tag{5.39}$$

We can use the small angle hypothesis ($sin\theta_d = \theta_d$; $sin\phi_d = \phi_d$ et $cos\theta_d = cos\phi_d = 1$), since during the operation of the quadrotor, small change (between -20° to 20°) for roll and pitch angles occur, which results in neglected small gyroscopic effects. We can rewrite equation 5.39 as:

$$\begin{aligned} U_x &= \tfrac{U_1}{m}(c\psi\theta_d + s\psi\phi_d) \\ U_y &= \tfrac{U_1}{m}(s\psi\theta_d - c\psi\phi_d) \end{aligned} \tag{5.40}$$

which can be inverted to get the desired roll and pitch angles ϕ_d and θ_d:

$$\begin{aligned} \phi_d &= \tfrac{m}{U_1}(U_x s\psi - U_y c\psi) \\ \theta_d &= \tfrac{m}{U_1}(U_x c\psi + U_y s\psi) \end{aligned} \tag{5.41}$$

5.3.1.3 Bat algorithm (BA)

The bat algorithm (BA) is one of the efficient metaheuristics [146]. It simulates the echolocation characteristics of bats with different emission rates and pulse intensities. Since its first implementation, BA has proven to perform better than other well-known algorithms in solving a wide range of global optimization issues. Also, studies have shown that this algorithm has become an active area of research due to the appearance of several BA extensions. In this section, we focus on the use of bat algorithm in our work.

Table 5.2: BA parameters

Parameter	Value
Population size	25
Pulsation rate r	0.55
Loudness A	0.25
λ	0.9
ω	0.99

5.3.2 Procedure

The basics of bat algorithm can be summarized in the following idealized three rules [146].

* All bats use echolocation to detect distance, and they can also distinguish between food/prey and obstacles;

* For the prey searching, each bat flies randomly with a velocity v_i to the position x_i with a fixed frequency and loudness. According to the proximity of the target, they adjust automatically the wavelength (or frequency) of their emitted pulses.

Algorithm 1 Pseudo-code of bat algorithm

```
Input: $Problem_{size}$, PD's gains of the previous step.
Output: $PD^*$
    for $i$ = 1 to $n$ do
        $x_i \leftarrow rand(Problem_{size}, n)$;
        $v_i \leftarrow rand(Problem_{size}, n)$;
        Define $q_i$ at $x_i$;
        Initialize the pulse rates $r_i^0$ and the loudness $A_i^0$;
        Generate $PD_i(n, Problem_{size})$;
    end
    while (iteration number < 150)
        Generate new solution according to Eq.s (5.42) to (5.45);
        if $rand > r_i$ do
            Select a solution among the best solutions;
            Generate a local solution around the selected;
            best solution according to Eq.(5.46);
        end
        Generate new solution by flying randomly;
        if $(rand > A)\&(f(x_i) < f(x^*))\&(f(PD_i) < f(PD^*))$ do
            accept the new solutions;
            Generate a local solution around the selected;
            increase $r_i$ and decrease $A_i$ using Eq.(5.47);
        end
        Extract the current best $x^*, PD^*$ from the best bat;
    end
    return $PD^*$;
```

* It is assumed that the loudness A varies from a large (positive) value A_0 to a minimum value A_{min}.

5.3.2.1 Gain based scheduling via BA algorithm

The position $x_i^t - 1$ and the velocity $v_i^t - 1$ of the i^{th} bat are defined in a d-dimensional search space, and they are updated subsequently in each iteration. The new solutions x_i^t, PD_i^t and velocities v_i^t at time step t are calculated using the following equations [147].

$$q_i = q_{min} + (q_{max} - q_{min})\beta \tag{5.42}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*)q_i \tag{5.43}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{5.44}$$

$$PD_i^t = PD_i^{t-1} + v_i^t \tag{5.45}$$

$\beta \in [0,1]$ is a random vector drawn from a uniform distribution. $i = 1,..,n$, x^* is the current global best solution among all the bats at the current iteration. q_i is the frequency value of the i^{th} bat

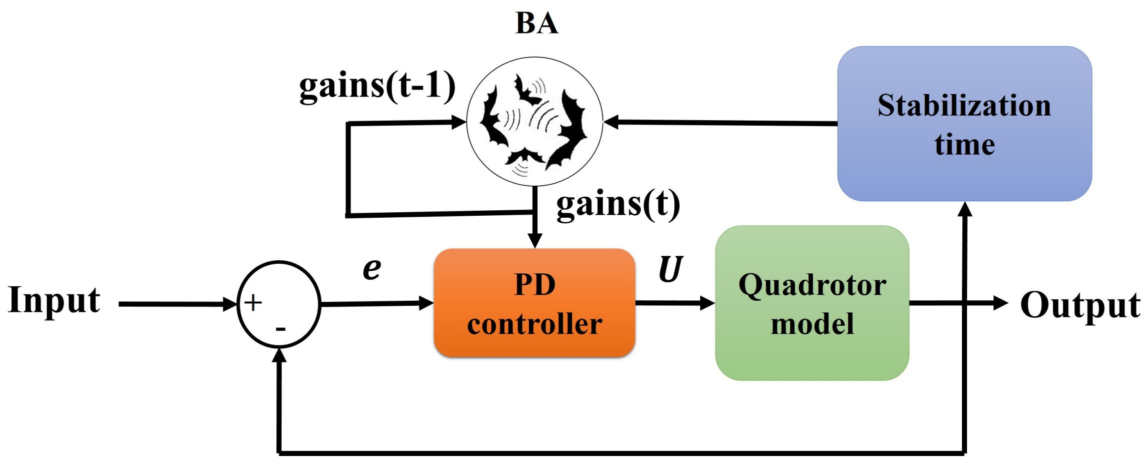


Figure 5.6: Setting the PD controller using the BA.

and PD_i^t present the candidate PD's gains generated for each bat in time step t. For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated using random walk i.e.,

$$\begin{aligned} x_{new} &= x_{old} + \varepsilon A^t \\ PD_{new} &= PD_{old} + \varepsilon A^t \end{aligned} \tag{5.46}$$

where $\varepsilon \in [0,1]$ is a random value, x_{old}, PD_{old} are the solutions in the current optimization solution set, and A^t is the average loudness of all the bats at time step t. In addition, the loudness A_i and the pulse rate r_i are updated if the solution is found, i.e.,

$$\begin{aligned} A_i^{t+1} &= \omega A_i^t \\ r_i^{t+1} &= r_i^0[1 - e^{-\lambda t}] \end{aligned} \tag{5.47}$$

where r_i^0 is the initial pulse rate, ω is the loudness attenuation coefficient and λ is the pulse's increasing coefficient. For any $\omega > 0$, and $\lambda < 1$ and for $t \to \infty$, we have:

$$A_i^t \to 0 \quad \text{and} \quad r_i^t \to r_i^0 \tag{5.48}$$

The basic steps of BA can be summarized as the pseudo-code shown in Algorithm 1 [146].

5.3.2.2 Tests and simulation

In the context of managing the PD controller's limit to stabilize the system, the chosen parameters for all the PD's controllers are adjusted using the bat algorithm, where the objective function is established to minimize the stabilization time as much as possible. The BA parameters are shown in table 5.2.

The developed PD controller based on gain scheduling is shown in figure 5.6.

* **Altitude controller:** For the altitude controller, the BA was used to select control's gains for the PD controller with the desired altitude z_d of $3m$. The gains generated by the BA were $k_p = 4.658$ and $k_d = 2.981$. Table 5.3 shows the performance obtained from the simulations, where the gains obtained give a stabilization time of $S_t = 1.97s$ and an overflow of $D = 1.36\%$.

* **Rotation controllers:** Similarly, the gains of rotation controllers presented in table 5.3. The system responses and the control inputs are shown in figures 5.7

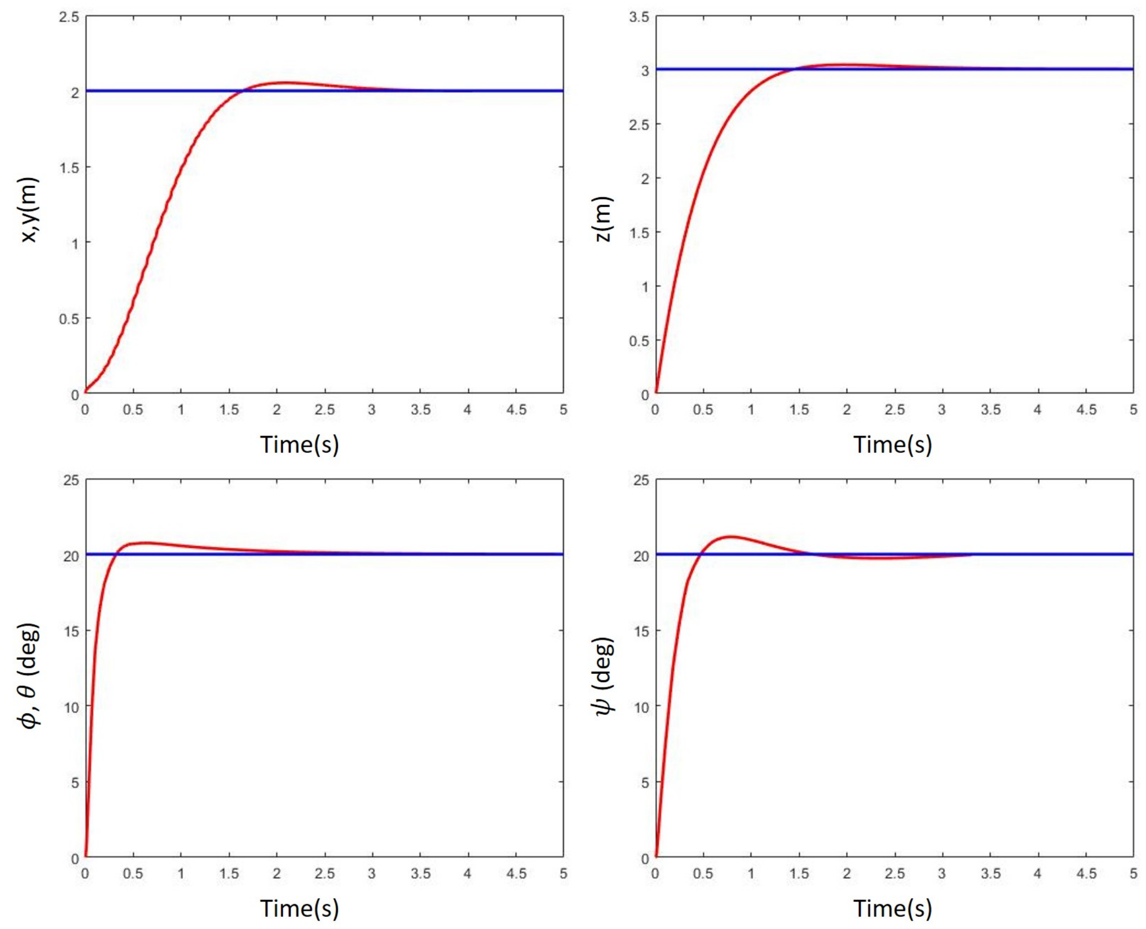


Figure 5.7: System responses

Table 5.3: Simulation results

Parameter	Desired value	k_p	k_d	$S_t(s)$	$D(\%)$
Altitude (z)	3(m)	4.658	2.981	1.970	1.36
Position (x et y)	2(m)	9.545	5.102	1.678	1.67
Roll/Pitch (ϕ,θ)	20°	8.557	1.235	1.3036	3.72
Yaw (ψ)	20°	32.652	5.213	1.2814	5.70

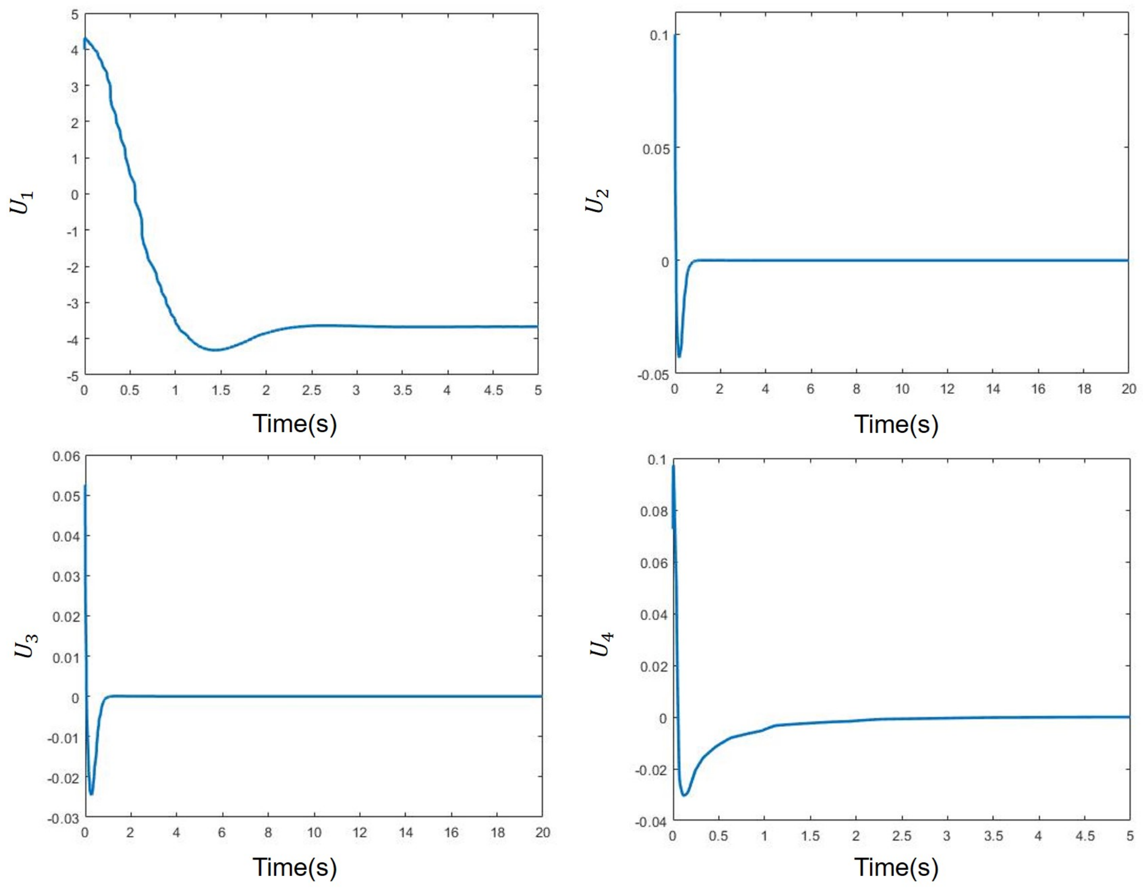


Figure 5.8: Control signals

and 5.8 respectively. It is apparent that the pitch ϕ and the roll θ control signals seem equivalent due to the symmetry of the quadrotor. On the other hand, this fact has been verified and demonstrated for the positions x and y which are equivalent for the same reason.

∗ **PD based on gain scheduling:** As mentioned briefly, the quadrotor is an under-actuated system, which implies that it is controllable through four control inputs and six outputs. Two main choices are possible. The first is to control angular velocities and total thrust. The second choice is to control the velocities in the Cartesian directions, as well as the yaw angle. In this subsection, we design a self-adjusting PD controller for trajectory tracking. We maintain the trajectory tracking capability optimally by setting four PD controllers for the ψ angle and the positions $\{x, y, z\}$ along the desired path given by:

$$\begin{aligned} x_d &= 0.5.cos(t), \\ y_d &= 0.5.sin(t), \\ z_d &= 1 + \tfrac{t}{10} \\ \psi_d &= 5^\circ \end{aligned} \tag{5.49}$$

with: $-90^\circ < \theta, \phi < 90^\circ$.

During the tracking process, the BA operates as follows. First, it generates a "random population of bats" where each bat aims to find its prey, which in our case will be a vector of the gains of the PD controller. The gains on the iteration $t-1$ are used to initialize the BA at iteration t, as shown in figure 5.6. We use the basic control law presented by equation 5.37 and the gains obtained at iteration $t-1$ to minimize the time of establishment by the following objective function:

$$T_s = -\frac{ln(0.02)}{\xi w_n} \tag{5.50}$$

Where ξ and w_n are the damping and the natural pulsation calculated for each system response.

By applying 150 iterations, the dynamics of the BA can effectively give the appropriate gains of the PD controllers that guarantee the stability of the system. We show in the figure. 5.9 the performance of the system to follow the desired trajectory in the 3-D plane. In addition, figure 5.10 shows the system responses.

From figures. 5.8 to 5.10, it is very clear that the PD control based on the BA provides the stability that gives suitable tracking results.

5.4 Visual servoing based on the image

Our work in this section is divided into two parts: The first part deals with the object tracking task where the object is prefixed to the desired position. Indeed, we aim to validate that the IBVS approach is suitable to control the developed model of the drone. In the second part, we demonstrate that the same law is useful to control the drone to follow a dynamic object. The simulations have been implemented in MATLAB/Simulink.

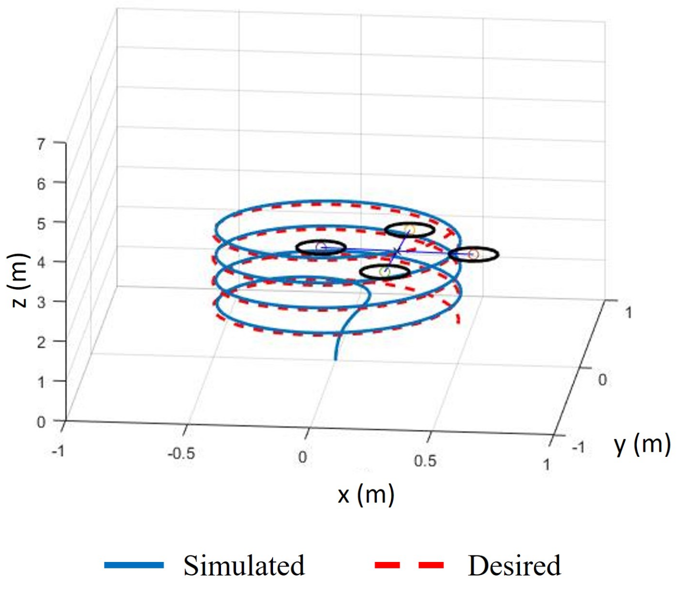


Figure 5.9: The desired and simulated trajectories.

5.4.1 IBVS control law design

The relationship between the dynamics of the point in the image plane and the speed of the camera is the key to designing the control law (described in chapter 4). To follow an object with precision it is inevitable to specify at each moment the object by a bounding box centered on it. For this purpose, it is commonly considered that the target object will be the four corners of the object for the following reasons [133]:

- The squares are frequently visible because they have an identifying feature that allows the detection of orientation (a corner with a specific color).
- Visual object trackers use squares (bounding box) to locate the tracked object.
- Delete the limit for the IBVS implementation. Three points are insufficient to represent practically the object.

We adopt that the object feature vector s and the vector of the desired features s^* are presented by:

$$s = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \\ v_1 & v_2 & v_3 & v_4 \end{bmatrix} \tag{5.51}$$

and

$$s^* = \begin{bmatrix} u_1^* & u_2^* & u_3^* & u_4^* \\ v_1^* & v_2^* & v_3^* & v_4^* \end{bmatrix} \tag{5.52}$$

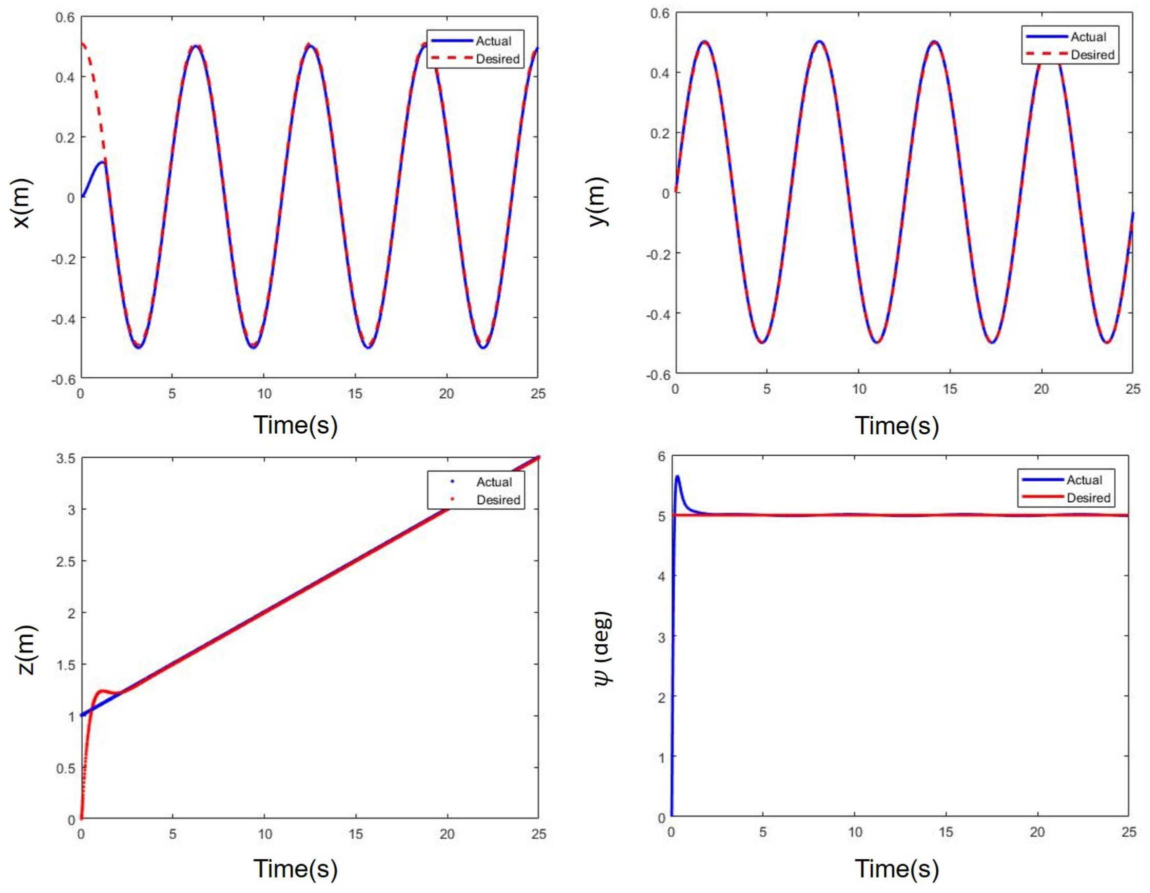


Figure 5.10: System responses.

The error is defined as:

$$e = s(t) - s^* \tag{5.53}$$

As detailed in chapter 4, we adopt the relation 4.23 of which $L_e \in R^{8\times6}$. Figure 5.11 shows the architecture of the proposed control method. This method calculates the feature velocity expressed in the camera coordinate system to drive the drone to the desired position. As noted in the quadrotor modeling section, we can provide a reference signal for the translations along the axis x, y, z and the ψ rotation.

The control architecture consists of three main subsystems: the quadrotor controllers, the visual sensor, and the image-based controller. The inputs of the quadrotor controllers are: the reference velocity vector v_q^*, which is obtained by multiplying the desired velocity vector of the camera v_c^* by the homogeneous transformation matrix T_q^c, and the quadrotor state vector $X_q = \{x, y, z, \phi, \theta, \psi\}$. This vector is returned to the visual sensor model as a homogeneous transformation matrix 4×4, using the transformation matrix T given by equation 5.54. Similarly, the homogeneous transformation matrix of the quadrotor T_q^I is determined from the output vector X_q. Then, T_q^I is converted to a homogeneous camera transform matrix T_c^I using the matrix T_c^q.

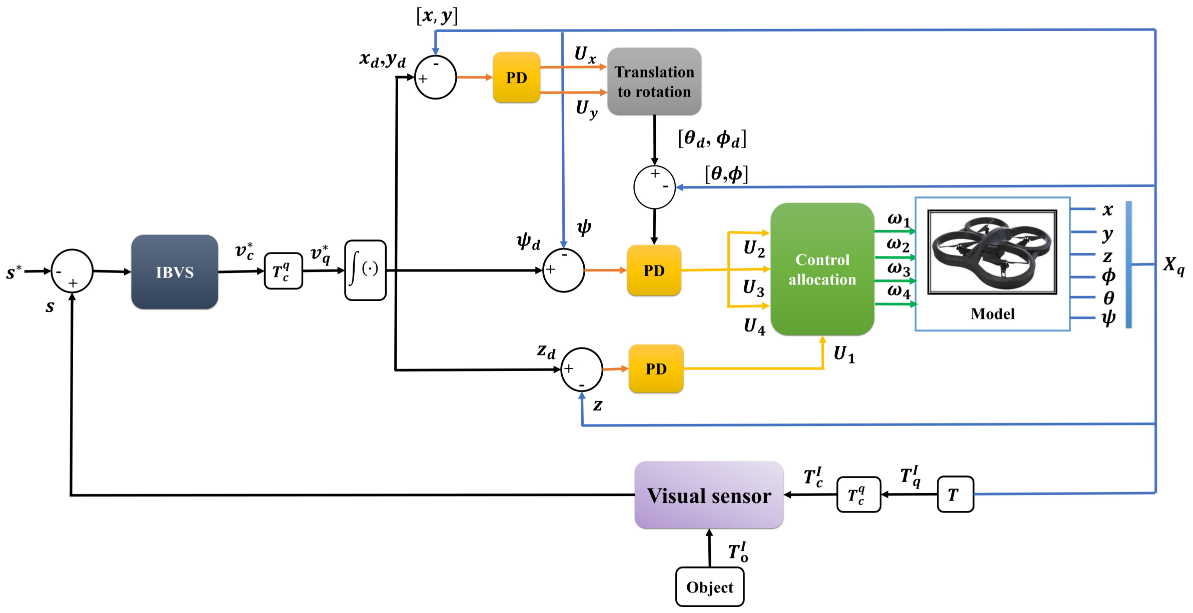


Figure 5.11: System architecture.

$$T = \begin{bmatrix} R_B^I & \mathcal{V} \\ \mathcal{O} & 1 \end{bmatrix} \tag{5.54}$$

With: $\mathcal{V} = [x, y, z]^T$, $\mathcal{O} = [0, 0, 0]$ and R_B^I is expressed by equation 5.3.

Concerned the visual sensor model, as shown in figure 5.12, it is decomposed from a camera model and an image processing block. The camera provides the image to the image processing block to construct the feature vector s. The image processing block requires the models of camera C and object O [148]. We define the target object as four 3-D points of coordinates $P = (X, Y, Z)$ [149]. Each point is projected as a 2-D point with the coordinates $p = (x, y)$, i.e.

$$\begin{aligned} x &= f\frac{X}{Z} = p_u - c_u \\ y &= f\frac{Y}{Z} = p_v - c_v \end{aligned} \tag{5.55}$$

Where p_u and p_v represent the coordinates of the image point in pixels. The set of parameters (c_u, c_v, f) present the intrinsic parameters of the camera. (c_u, c_v) and (f) are the coordinates of the main point and the focal length, respectively.

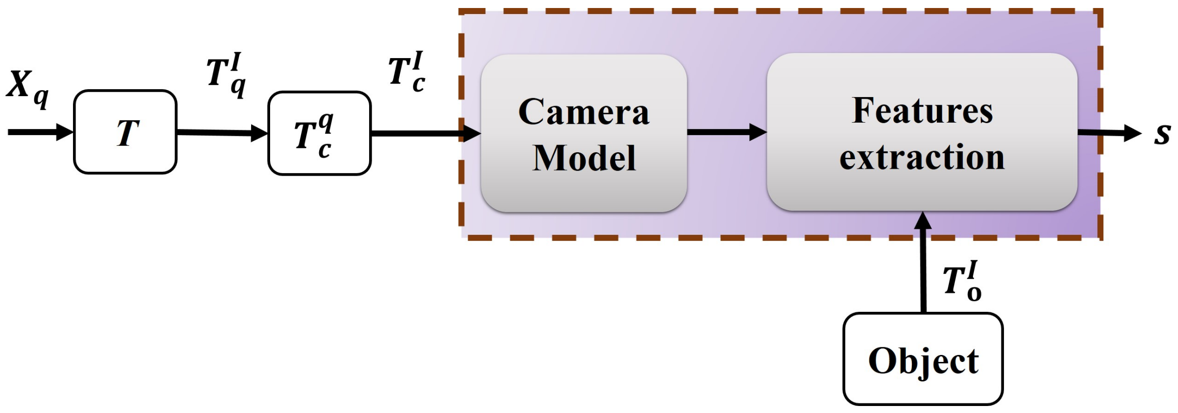


Figure 5.12: Visual sensor model.

5.4.1.1 Simulations and results

In this subsection, we aim to check the theoretical work presented above (quadrotor control for trajectory tracking) using the IBVS controller. We show the IBVS controller response when the quadrotor immediately reached the desired setpoints and to ensure that it satisfies the proof of stability presented in chapter 4. Also, we present the advantages of using the bat algorithm presented previously to control the quadrotor, using the classic IBVS approach in terms of guaranteeing the stability and accuracy of the vision controller to achieve the desired optimal positions.

The simulation was performed using a visual servo toolbox in MATLAB [149]. The quadrotor has an eye-in-hand configuration, where the camera is considered fixed in the center of the quadrotor. The camera is supposed to be fully controllable (6 DOF). The figure 5.13 shows the quadrotor frame F_q, the camera frame F_c and the object frame F_o. It is assumed that the object is described by a square of length 0.25 m and it is located at the position $X_p = (0,0,0)$. The simulations run for 25 seconds with time steps of 0.05 second. The homogeneous transformation matrix of the quadrotor T_q^I is determined from the output vector X_q. Then, T_q^I is converted to a homogeneous camera transform matrix T_c^I using the matrix T_c^q.

- **Static object:** We start with the case where the target is immobile, at the position $(0,0)$. The camera must follow two trajectories, where the object must be maintained in the FOV. Table 5.4 presents the suggested trajectories. The proportional gain is set to $\lambda = 2$. This value is chosen to stabilize both image errors and the Cartesian errors at a time not exceeding 4 second. The initial and the desired images for the target are given in figure 5.14 (a).

  We start with the first trajectory, where the initial image is taken at the height 2 m. The rotation angles are initialized by $\{\frac{\pi}{2}, 0, -\frac{\pi}{2}\}$. The desired image is taken

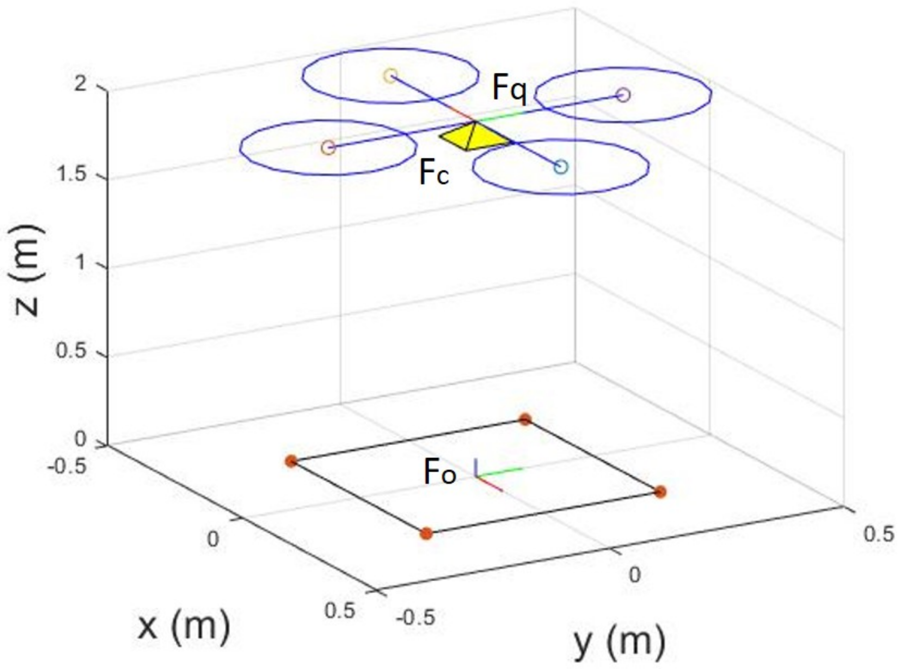


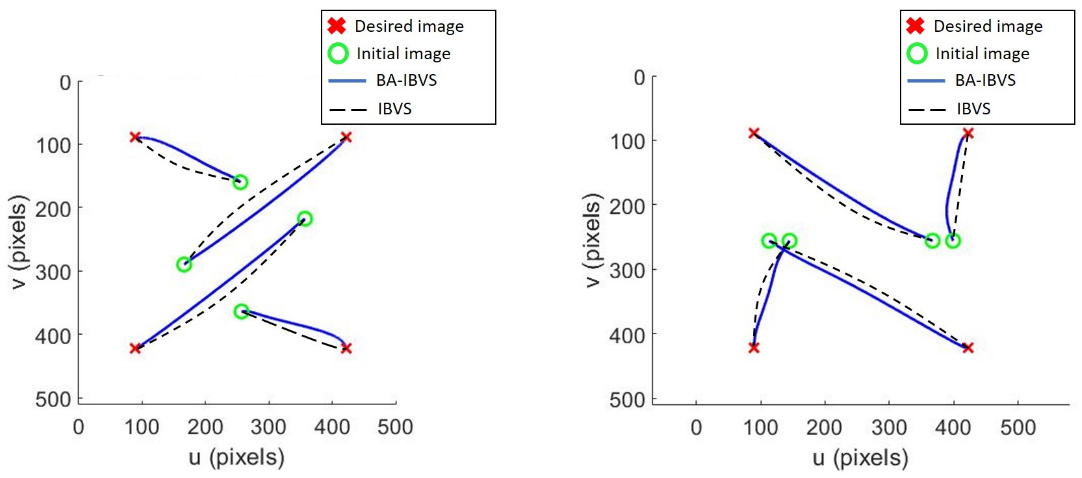
Figure 5.13: Illustration of frames: quadrotor, camera and object.

directly in front of the target at the position $(0, 1.5)$ m with pitch and roll of $0°$ and yaw of $90°$. Concerned the second trajectory, it is desirable that the camera reaches the position $(0, -1.5, 0)$ and the orientation $(0, 0, \frac{\pi}{2})$, where the initial image is taken directly in front of the target at position $(2.1521, -2.0901)$ with pitch and roll of $0°$ and yaw of $-90°$. The results are shown in figures 5.14 and 5.15.

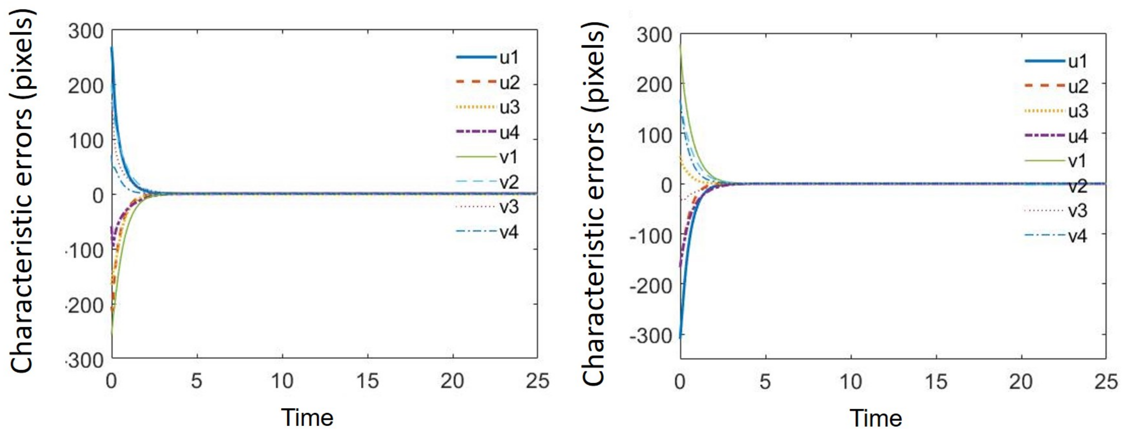
Figure 5.14.a shows that the IBVS-BA controller is more robust than the IBVS in terms of reaching the desired position. This is evident in the absence of overshoots in the exponential convergence of errors presented in figure 5.15.c. During the movement from the first position to the end position, the camera path is shown in figure 5.15 (a-b). It can be seen that the quadrotor maintains a stable pitch and roll angles and reaches the desired yaw angle. At the same time, it provides a favorable arrival to the desired position. Figure 5.14.b shows the quadrotor velocities during its movement. v_x, v_y, v_z represent the linear velocities of the

Table 5.4: Trajectories of the camera.

	Trajectory 1	*Trajectory 2*
Initial condition	$\{0, 0, 2, \frac{\pi}{2}, 0, -\frac{\pi}{2}\}$	$\{2.1521, -2.0901, 0, 0, 0, -\frac{\pi}{2}\}$
Final condition	$\{0, 1.5, 0, 0, 0, \frac{\pi}{2}\}$	$\{0, -1.5, 0, 0, 0, \frac{\pi}{2}\}$



(a)



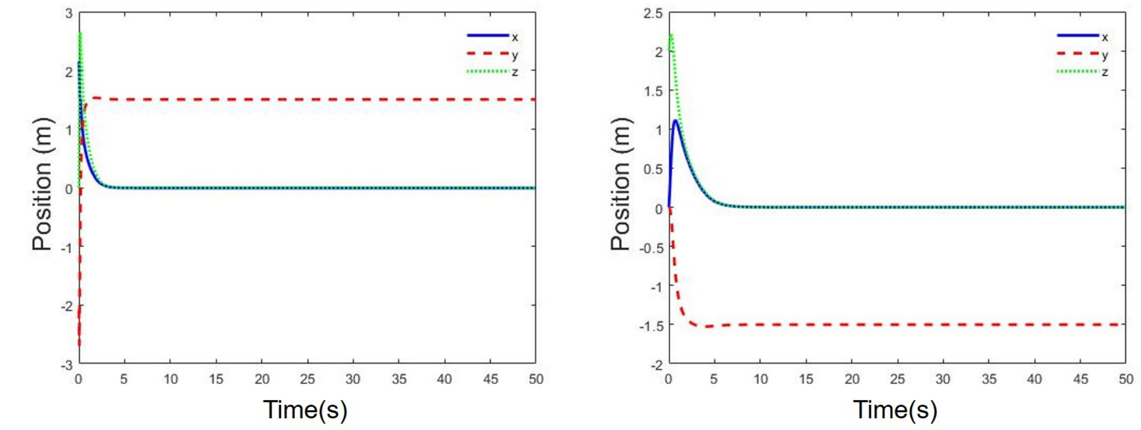
(b)

Figure 5.14: (a) Trajectories of the feature points and (b) Errors of the feature points for trajectory 1 (Left) and trajectory 2 (Right).

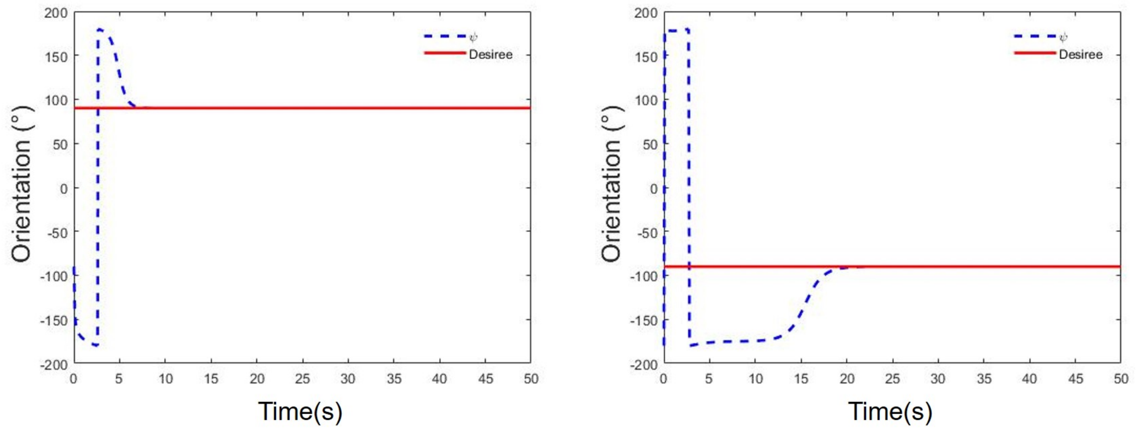
quadrotor and w_z represents the angular velocity around the axis z. It is clear that all speeds reach zero at the desired position.

To prouve that the IBVS control scheme is asymptotically stable for both trajectories, we plot in the figure 5.16 the value of the candidate function $\mathscr{L} = \frac{1}{2}||e(t)||$ and its derivative $\dot{\mathscr{L}} = -\lambda e^T J J^+ e$. This guarantees the overall asymptotic stability of the control law.

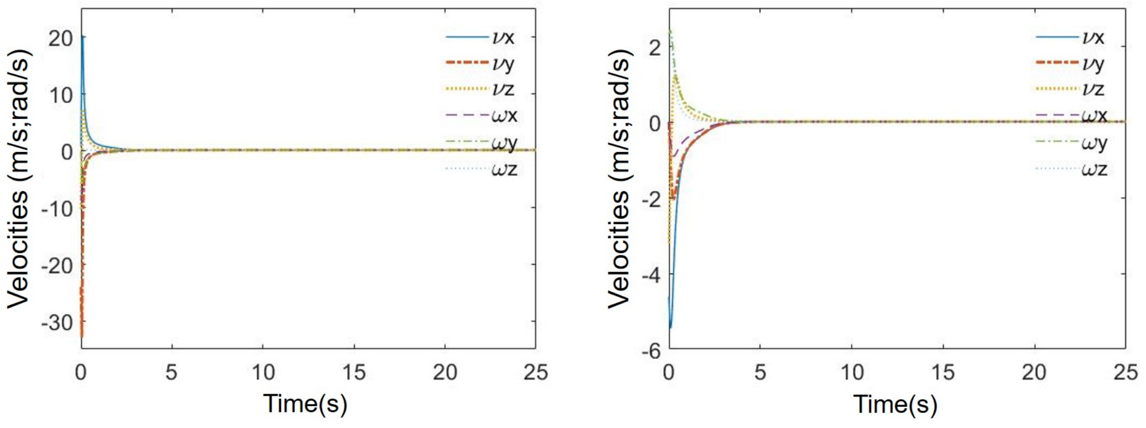
- **Dynamic object:** We consider the case where the target starts from the position $(0,0)$ following the trajectory:



(a)



(b)



(c)

Figure 5.15: (a) Positions, (b) orientation and (c) velocities of the camera for trajectory 1 (Left) and trajectory 2 (Right).

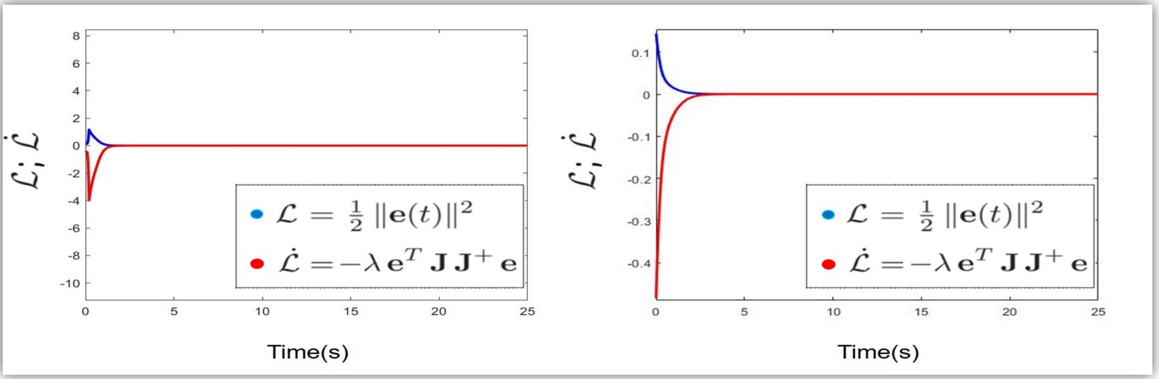


Figure 5.16: The values of the candidate function of Lyapunov and its derivative.

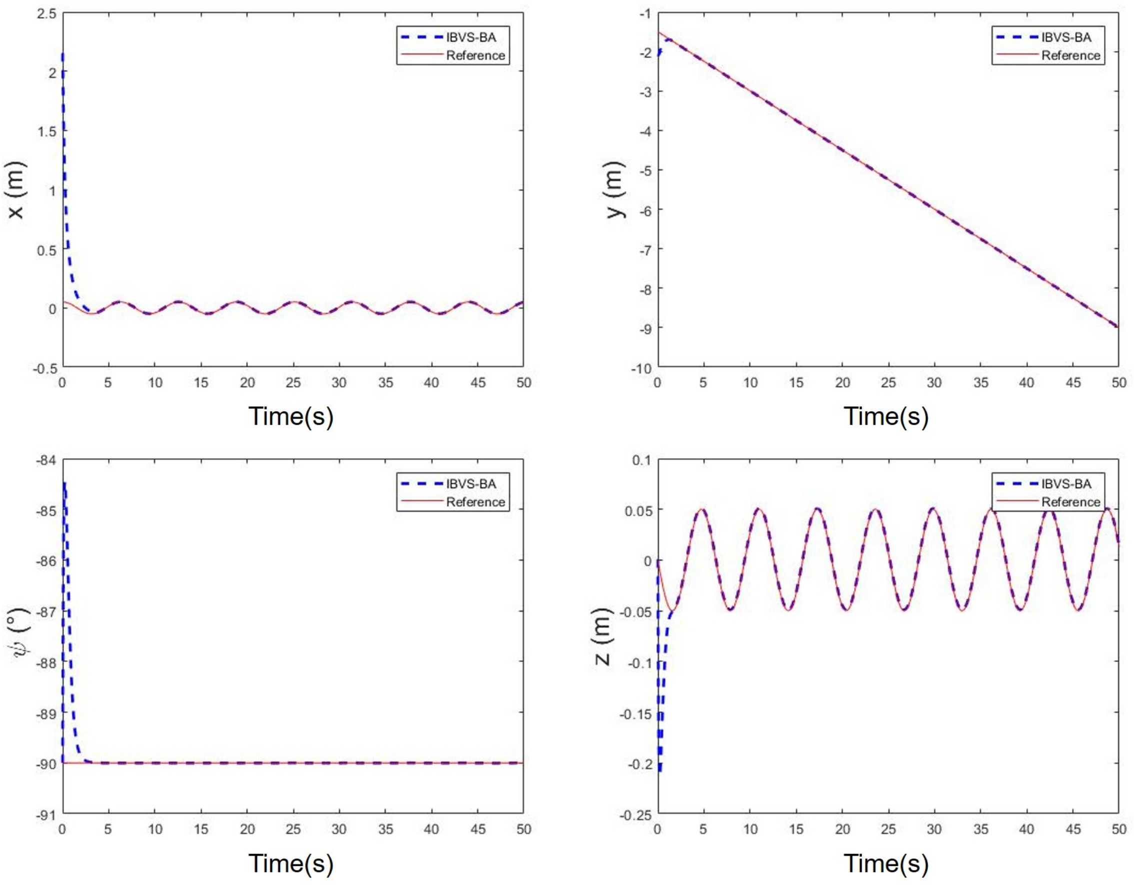


Figure 5.17: The desired and the actual trajectories.

$$
\begin{aligned}
x_o &= 0.05 * cos(t),\\
y_o &= -1 - t/10, \qquad (5.56)\\
z_o &= 0.05 * sin(t)
\end{aligned}
$$

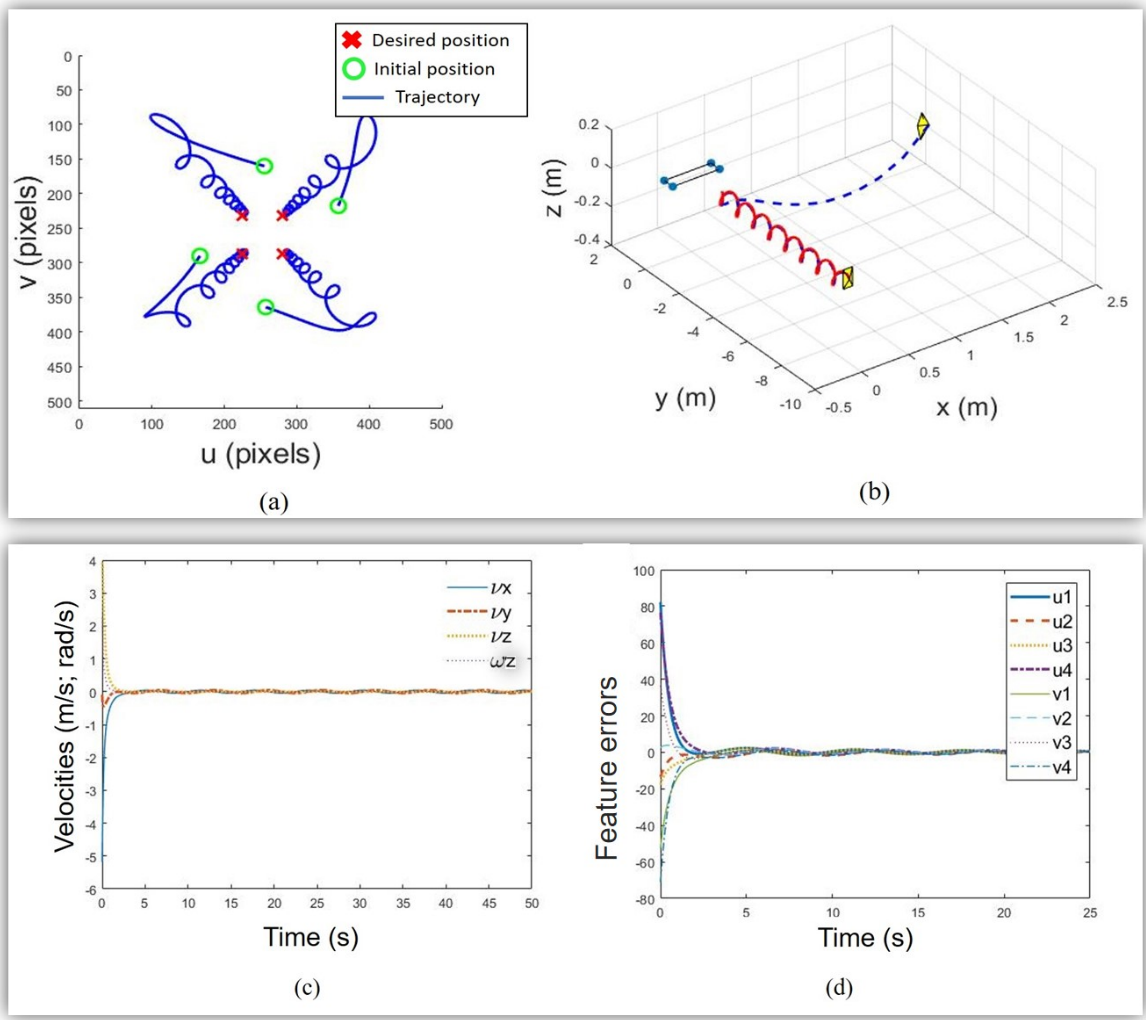


Figure 5.18: (a) Desired and actual trajectory in the image plane, (b) desired and actual trajectory in 3-D plane, (c) velocities of the camera and (d) errors of the feature points.

The camera starts from the position $\{2.1521, -2.0901, 0, 0, 0, -\frac{\pi}{2}\}$, and it must make the first two rotations of 5° around the axis z in positive then in the negative directions, and then track the object during the rest of the simulation.

As it is shown in figures 5.17 and 5.18, the camera suitably follows the target, which resulted in the quick convergence to the desired position. Figure 5.19 shows that the IBVS control scheme is asymptotically stable.

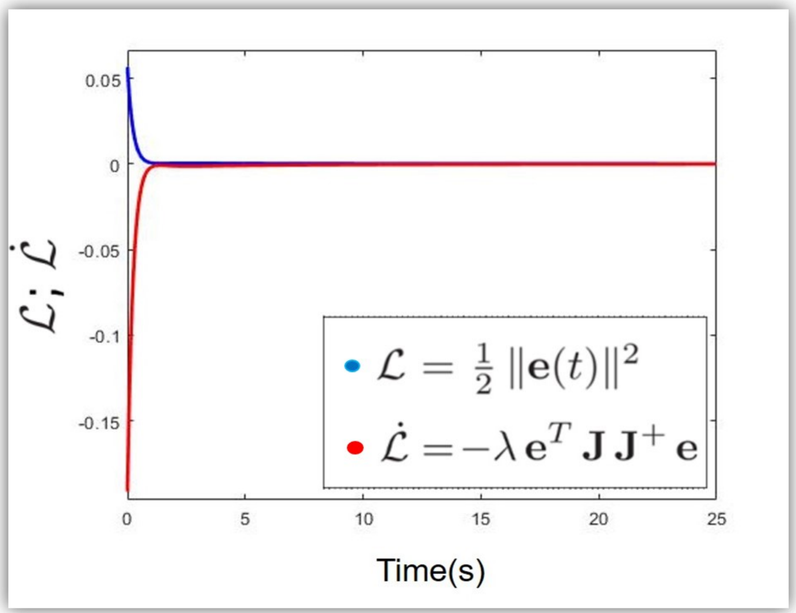


Figure 5.19: The values of the candidate function of Lyapunov and its derivative.

5.5 Conclusion

In this chapter, we have studied the problem of controlling a quadrotor for a static and dynamic target-tracking task. After presenting the used quadrotor model, we described the design of the controllers to control the position and the rotation of the drone. The proposed system has been validated for a tracking task of a static target and then a moving target.

CHAPTER 6

VALIDATION OF THE HOG-BASED OBJECT TRACKING METHOD

6.1 Introduction

In this chapter, we introduce a virtual application for a drone to perform object tracking tasks. The proposed architecture IBVS-HOG is based on the following components: the visual tracking method, and robotic operating system (ROS), whose purpose is to validate the use of the approach of the visual servoing presented in the previous chapter and the efficiency of the HOG-based visual tracking method proposed in chapter 3.

In this chapter, we briefly present the adopted drone and the robotic operating system (ROS), in section 2. Then, in section 3, we proceed to a detailed description of the proposed method. We present the evaluation of our method on two objects in two tests of robustness.

6.2 Hardware environment

6.2.1 Parrot AR Drone 2.0

The specific model used in this thesis is Parrot AR Drone 2.0, which was introduced in 2010 in France. It is a small helicopter with four propellers. It can be controlled with a device through a Wifi connection (Figure 6.1) [150]. Indeed, it can create itself a wireless network to which computers or smart-phones under iOS, Symbian or Android can be connected. Connecting a smart-phone allows the pilot to control the drone easily and it allows to play with it as a high-tech toy in the AR games, which is the original idea of Parrot society. The AR Drone parrot is relatively light, about 400 g, and has a flight endurance of $10-15$ min. Table 6.1 gives more technical information.

The drone can hold in position using an ultrasonic altimeter and a pressure sensor. In addition, it can be controlled to perform position maintenance based on visual algorithms using the bottom/frontal camera. The propellers are light enough to be stopped with your hands. The choice of this model was made because of its low price, about 300 €, its great programming capacity and its vast potential for development. The great interest in this model has created a huge community on the internet where the users can share their information and experiences. There are several code storage platforms such as **GitHub** [151] that allow users to share different programming **packages** and **libraries**. For a detailed description of this model regarding its equipment, capabilities, and performance, please see [1, 152].

6.2.2 Robot Operating System (ROS)

ROS is an operating system derived from the Unix operating system. It was created in 2007 by the Stanford Artificial Intelligence Laboratory as part of the Stanford

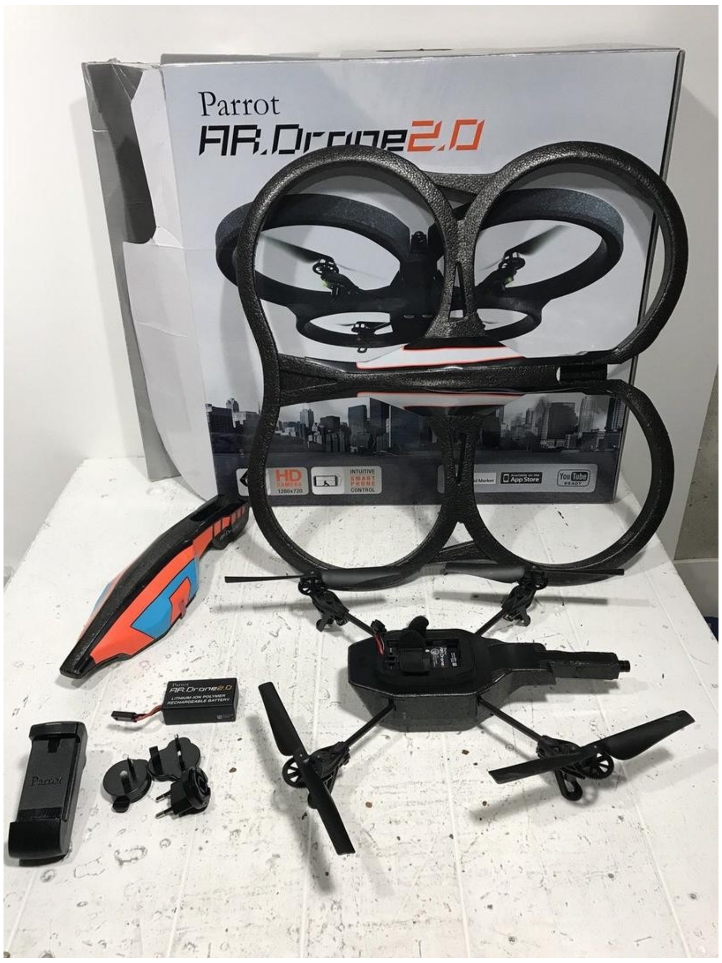


Figure 6.1: Parrot AR Drone 2.0.

Table 6.1: Technical characteristics of AR Drone 2.0 [1].

Weight	380 g without hull , 420 g with hull
Cameras	Front HD, 720p: Diagonal Viewing Area 92° Bottom: QVGA, 320× 240 Diagonal Viewing Area 64°
Payload Maximum speed Networking	$0-100$ g, with 250 g The drone could not take off $2-3$ m/s 802.11n WiFi
IT Resources	Processor $ARMCortex-A8$ clocked at 1 GHz Digital Video Signal Processor 800 MHz 256 Mo (1 $Gbit$) of RAM $DDR2$
Sensors	IMU, Ultrasonic Elevation Sensor (6 m) Pressure Sensor (+/− 10 Pa precision) 3 gyroscopes 3 accelerometers

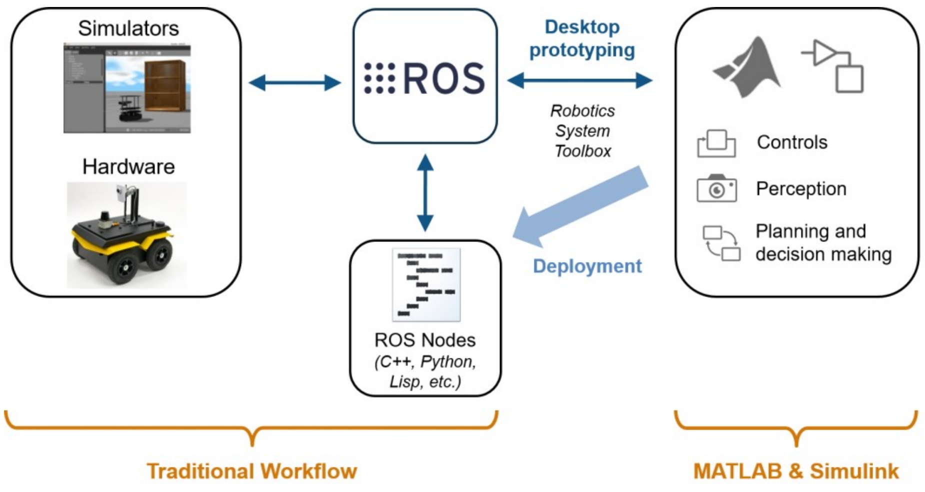


Figure 6.2: Flowchart of control under ROS [21].

AI robotics project (STAIR) under the BSD license (*Berkeley Software Distribution*) [153]. This license allows the freedom of commercial use and research in the context of the development of algorithms in robotics. Continuously and increasingly, the robots become more and more complex by the requirement of the integration of a large number of sensors and actuators [154].

Several research institutes have begun to develop platforms that support ROS to facilitate the rapid implementation of algorithms and to reduce the time required for the integration of new hardware, by adding new robots and sharing their hardware documentation, their packages, and their codes [155]. This effect has limited the utility of ROS to an open-source communication link that manages interactions between programs and libraries, as shown in figure 6.2.

6.2.2.1 ROS operation

The launch of ROS is equivalent to the launch of the **master** and the global parameters server, which allows to store and share the parameters of the robot in a memory from which each executable program can read and/or modify these parameters. In fact, the master is the core processor that manages a graphical architecture of different nodes, messages, and topics [154].

- **Nodes**: The main feature of ROS operation is represented in its nodes architecture. Each of these nodes is an executable $C++$ or Python program. Each node is created to publish and/or read messages of a topic independently of other nodes.
- **Messages**: The nodes communicate with each other by transmitting messages. A message is a strictly typed data structure. The standard primitive types (integer,

floating-point, boolean, etc.) are supported. The messages can be composed of other messages and arrays of other arbitrarily nested messages.

- **Topics**: A node sends a message by posting it in a given topic, which is simply a transport system, which controls how nodes can send and receive information between them. They can be multiple simultaneous publishers and subscribers for the same subject, and only one node can publish and/or subscribe to multiple topics.

6.2.2.2 Ardrone_autonomy

The Ardrone ROS package [156] is a driver developed at Simon Fraser University in Canada for the AR Drone 1.0 and 2.0. It is open-source which uses the Parrot SDK to provide an ROS node with a designed interface to control drones and collect all kinds of information. With this package, the users can communicate with AR Drone and extract data through topic. These topics are published once the AR Drone is connected to the computer. The most useful are:

- Navdata, which provides general information about the drone, such as battery level, acceleration or speed,
- Odometer data to locate the drone,
- IMU data for internal gyroscopes and accelerometers.

The package also allows sent commands, such as take-off and landing commands, as well as velocity commands to move the quadrotor.
The commands that can be sent to the drone via ardrone_autonomy, are sent by publishing messages in the topic presented in table 6.2.

Table 6.2: Topics and messages of AR Drone 2.0 [1].

Topic	Message	Characteristic
/ardrone/reset	std_msg/Empty	Reset the quadrotor
/ardrone/land	std_msg/Empty	Give the landing order
/ardrone/takeoff	std_msg/Empty	Give the take-off order
/cmd_vel	geometry_msg/Twist	Set the speed of the drone
/ardrone/front/image_raw	sensor_msgs/Image	Actual image recorded

For simulations, there are several open-source simulators with different characteristics on unmanned vehicles. The gazebo is a realistic open-source simulator compatible with ROS [157] that provides the necessary information to simulate a complete unmanned aerial system. For this purpose, we use the Gazebo simulator, combined with the ROS-ar2landing_neural package [158], which offers all the necessary tools to simulate and control the quadrotor.

We have created a ROS-Matlab node that acts as an IBVS controller: the program retrieves the visual information from Gazebo, as well as the positions and angles of the drone. Then, since these positions are known, the node can easily republish them as shown in figure 6.3.

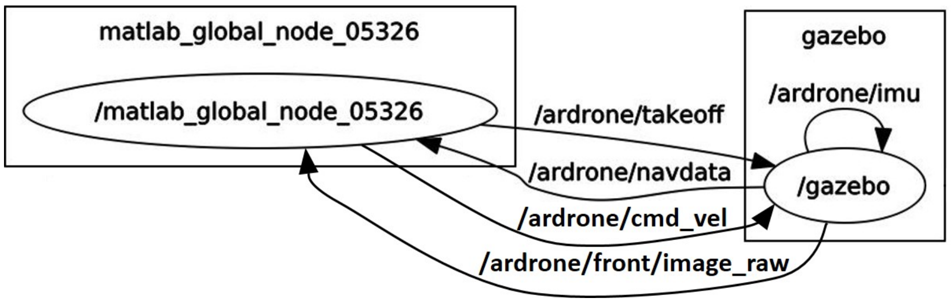


Figure 6.3: System architecture.

The controller produces control inputs (velocities) that move the quadrotor from its current pose to the desired pose. The controller receives the estimated visual features of the detection module (HOG) and sends the flight commands to the AR Drone vehicle.

6.3 ROS-based simulations

Through this section, we present our framework for visual tracking of 3D objects in motion by an unmanned aerial vehicle. This system can autonomously track a target object in 3-D motion, maintaining it with a fixed distance and centered it on the camera's FOV. The system is tested by the robotic operating system (ROS) in several scenarios using the drone model discussed previously, demonstrating the robustness of the system in the face of occlusion, fast motion, and unexpected scale variation. The results obtained indicate that the proposed system is ideally suited for object tracking. The hypotheses considered are the following [131] :

- The environment is unknown,
- The model of the object is previously unknown,
- The tracking must be in real-time.

The drone must be in front of the target and perform similar movements for a fixed altitude.

6.3.1 System architecture

As illustrated in figure 6.4, our visual servoing system is based on two essential modules.

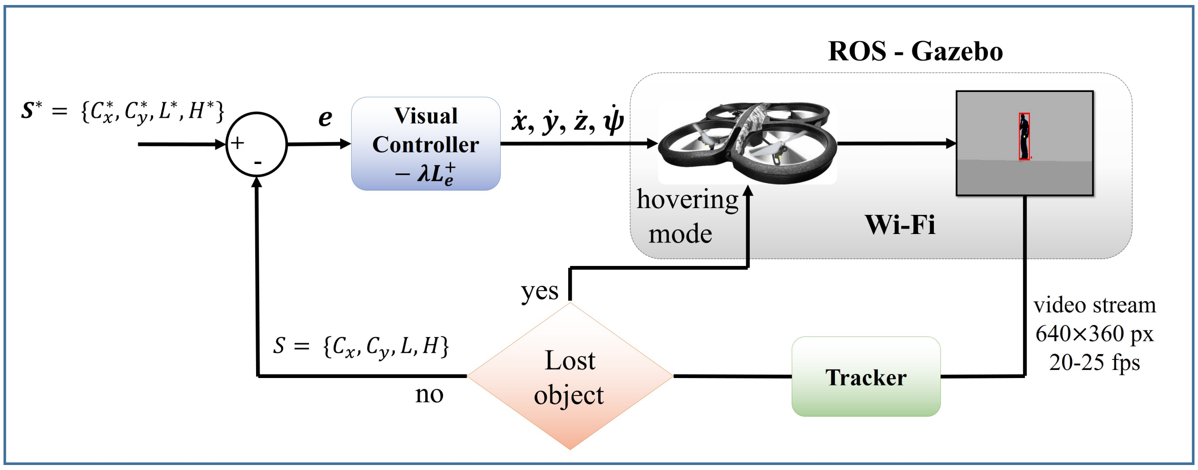

ROS - Gazebo
$S^* = \{C_x^*, C_y^*, L^*, H^*\}$
+
-
e
Visual Controller $-\lambda L_e^+$
$\dot{x}, \dot{y}, \dot{z}, \dot{\psi}$
hovering mode
Wi-Fi
yes
video stream 640×360 px 20-25 fps
$S = \{C_x, C_y, L, H\}$
Lost object
no
Tracker

Figure 6.4: Framework overview, the drone is controlled via a WiFi connection. The main components of the system are the drone driver (ardrone_autonomy package), the HOG-based tracker and the IBVS controller.

- **IBVS controller:** The controller based on two entries, which are the centroid of the bounding box and the size of the target, see figure 6.4. The references are the desired positions. The goal is to make the drone look at the target and roughly control its relative position to it.
- **HOG-based object tracker:** Our tracker proposed in chapter 3 can robustly track the object via the drone's video stream. The tracker returns the bounding box coordinates (position, height, and width) around the tracked object, see the objects shown in figure 6.5. The only constraint considered important for obtaining high repeatability during the tests is the following: the initialization of the tracker must be accurate since the correlation filters used to detect the object are sensitive to the background of the target.

During the experiments, the tracker generates a feature vector $\{C_x, C_y, L, H\}$. The error function compares the feature of the actual vector to the desired one $\{C_x^*, C_y^*, L^*, H^*\}$. The error is entered into the visual controller, which transmits the speed information to the flight controller. Meanwhile, a state estimation algorithm processes the internal sensor data to maintain the system's stability [159]. The simulation was done in Gazebo [160] for stabilization and tracking tasks, the object is considered as a 3-D model of a car or a man (Figure 6.5) [161].

6.3.2 Tracking strategy

To track an object, it is inevitable that the drone will follow it if it moves. Specifically, we want the object to be always in the center of the drone's camera as it moves.

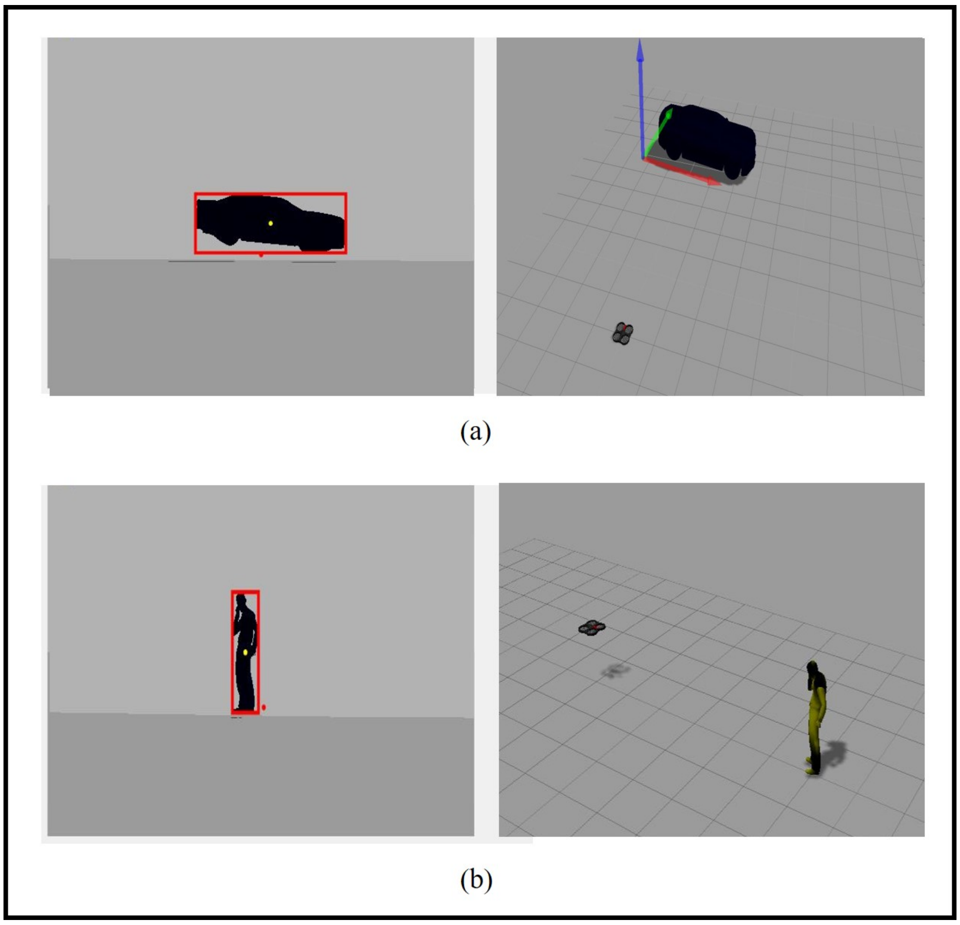
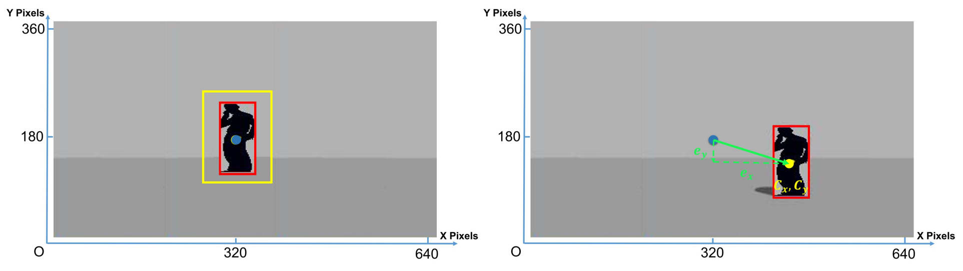


Figure 6.5: Tracked objects: (a) car and (b) man. Left: The tracker output. Right: Screenshot in the Gazebo simulator.

Whenever the object tracking is lost, the drone automatically recommended switching to an internal "hovering" mode.

6.3.2.1 Onboard cameras

The drone has two onboard cameras, one facing forward (front camera) and the other facing down (bottom camera). The front camera is our main tool for capturing live video streams, which allows the tracking methods to locate the object. Since this camera supports $720p$ video quality, we chose a resolution of 640×360 to reduce the computing time in real-time without losing too much useful information of the video stream.



(a) (b)

Figure 6.6: (a) Description of the desired and the actual bounding box coordiantes. (b) Distance error signals.

6.3.2.2 Distance error signal

We can see in figure 6.6 that the object is not always in the center of the camera. Therefore, it is reasonable to handle the error signals indicating the distance between the center of the object and that of the camera. The target object is given by the red bounding box, the center is defined by the pixel coordinates (C_x, C_y). In addition, the pixel point $(320, 180)$ is defined as the center of the camera (in blue). The error signal is defined by the center coordinate of the camera minus the center coordinate of the object, then, we normalize them in the range of -0.5 to 0.5 as follows:

$$\begin{aligned} e_x &= \frac{320 - c_x}{640}, \\ e_y &= \frac{180 - c_y}{360} \end{aligned} \tag{6.1}$$

6.3.2.3 Zone report

When the object appears smaller or larger depending on its distance from the observer. The bounding box size should be estimated in a suitable way to overlap the outline of the object. The proposed method addresses this issue during the tracking process. In this context, we aim to maintain a fixed distance between the target and the camera using the outline of the target and the estimated bounding box's coordinates. We calculate the ratio of these two areas to decide if the object is getting far to or closer to the observer using the equation 6.2. The idea is to estimate the distance depending on a computed ratio to give the drone the ability to maintain a fixed distance in front of the object. The ratio is computed using the desired area of the object (red) and the current area (yellow) shown in figure 6.6.a.

$$\hat{Z}_d = \sqrt{\frac{A_i}{A_a}} \tag{6.2}$$

With $\hat{Z}_d$, A_i and A_a present the ratio, the initial and the current areas of the object respectively.

When the drone moves in the desired direction, these errors decrease. Thus, the drone can follow the object. e_x expresses the horizontal errors leading to the horizontal movement of the drone, while e_y expresses the vertical errors leading to the vertical movement of the drone. The object center (C_x, C_y) should coincide with the center of the camera $(320, 180)$, which indicates the efficiency of the proposed tracking system. The repport $\hat{Z}_d$ is used to keep the distance between the drone and the object.

6.3.3 Simulation results

In this section, we aim to demonstrate that our system maintains the tracking object task efficiently. The system has been tested for stabilization and tracking tasks for 10 minutes and the z altitude set to 1 m. If $\hat{Z}_d = 1$, this means that the drone maintains the desired distance.

6.3.3.1 Stabilization task

The stabilization task is presented to validate the correct behavior of the system's hovering in front of the target. Figure 6.7 shows the trajectory of targets car and man in the camera's FOV. It is obvious that the two objects remain in the center of the image, with an average error of 0.6241 and 0.3040 pixels along the x and y axes for the object car and with an average error of 0.7944 and 0.5133 pixels along the x and y axes for the object man.

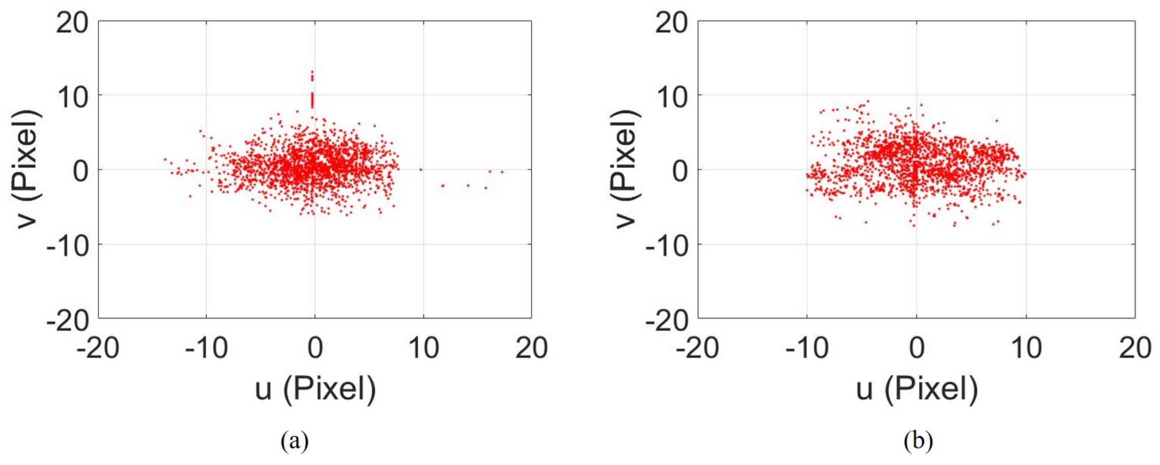


Figure 6.7: Trajectory of: (a) car, (b) man in the camera's FOV.

Figures 6.8 and 6.9 show the velocities $(\dot{x}, \dot{y}, \dot{z}, \dot{\psi})$ and its corresponding telemetry values. The AR drone is able to be stabilized in front of both objects.

The distance between the quadrotor and the object is important so that the quadrotor does not collide on the object. We estimate this distance using equation 6.2. The

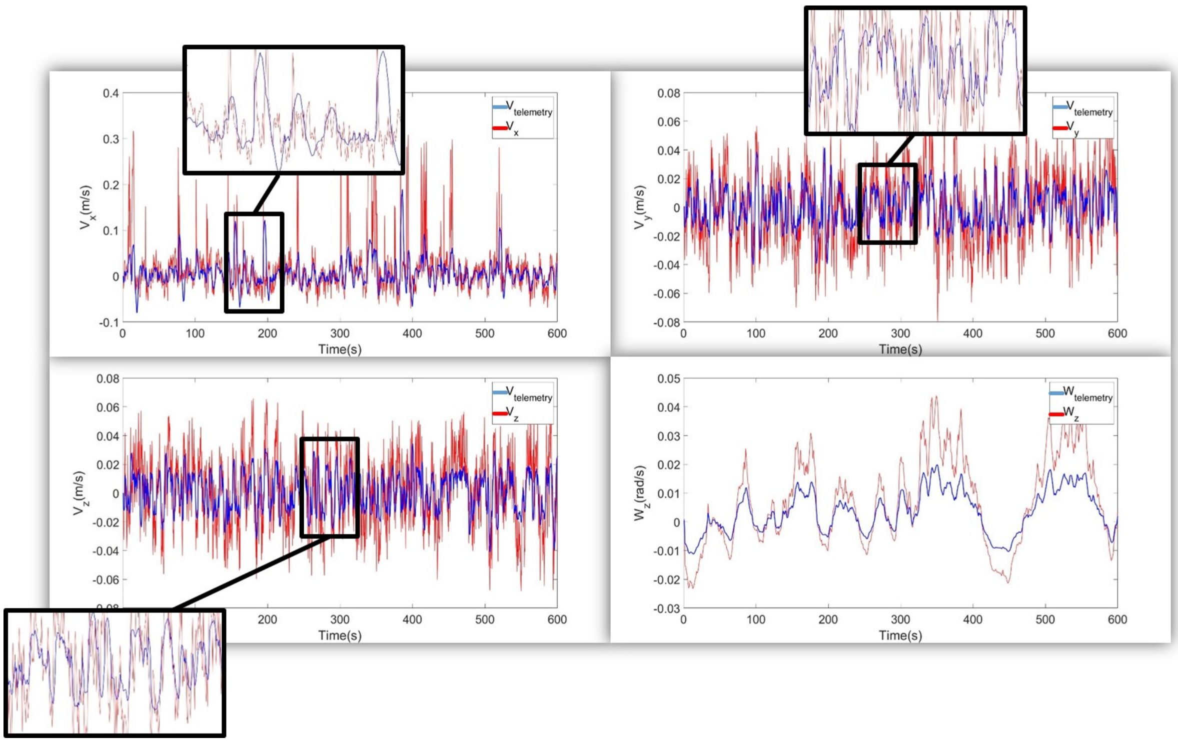


Figure 6.8: Commands values of $(\dot{x}, \dot{y}, \dot{z}, \dot{\psi})$ and corresponding values obtained by telemetry for the object man.

estimated distance is accept if $\hat{Z}_d$ is around the value 1. Which means that A_i and A_a are equal. Figure 6.10 shows the command $\hat{Z}_d$ generated for both objects (Car, Man). The command values converge to the desired value. All commands signals have an average close to zero because the drone tracks a static object over a distance that is conserved based on the estimated ratio $\hat{Z}_d$.

6.3.3.2 Pursuit task

This task is dedicated to the treatment of challenging cases: occlusion and object scale change. In this context, we decompose this task into two tests. Figure 6.11 shows that the first test deals with tracking the object car. The object moves quickly along the x axis with a speed of 1 m/s. During this movement, it is partially covered by a square of the same color. The second test deals with the tracking of the object man, which is fixed and does not move. The object man can be completely covered by a square. In this case, we change its position either by approaching it or by moving it away from the quadrotor, as illustrated in figure 6.12. The problem of scale variation is addressed in both tests.

- **Teste 1**

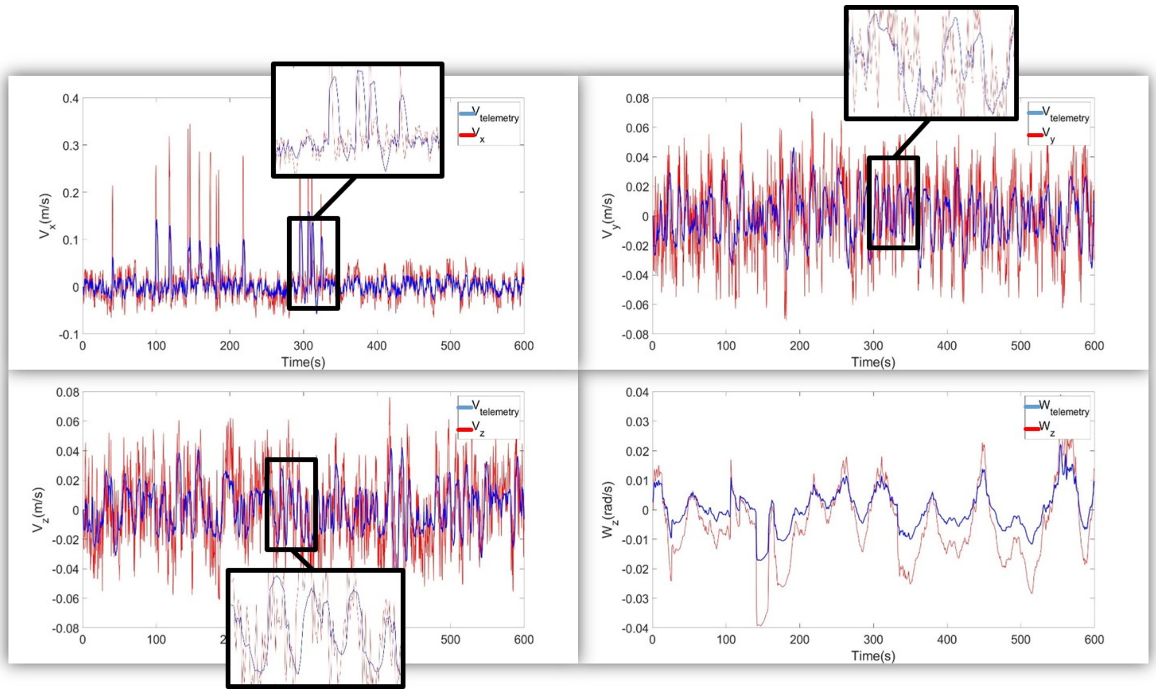


Figure 6.9: Commands values of $(\dot{x}, \dot{y}, \dot{z}, \dot{\psi})$ and corresponding values obtained by telemetry for the object car.

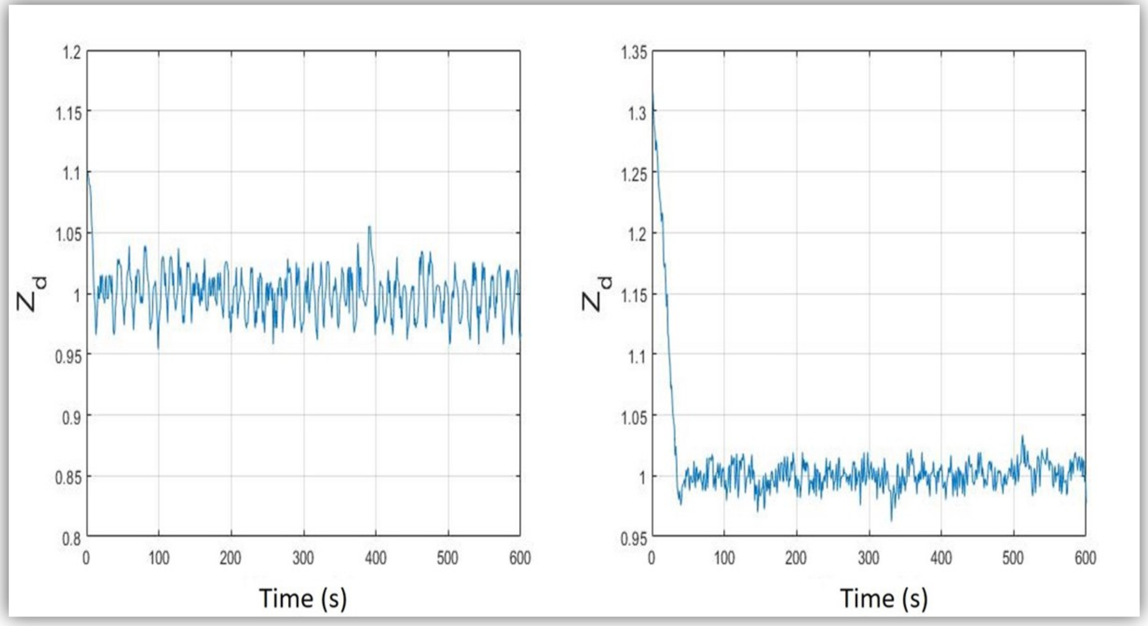


Figure 6.10: Command $\hat{Z}_d$ generated for the objects (a) man and (b) car.

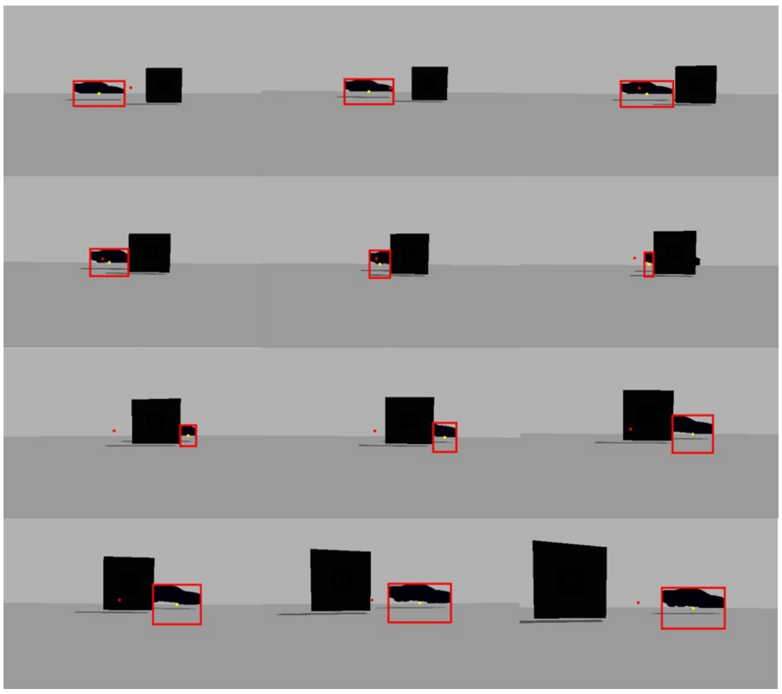


Figure 6.11: Example of handling partial occlusion and scale variation cases.

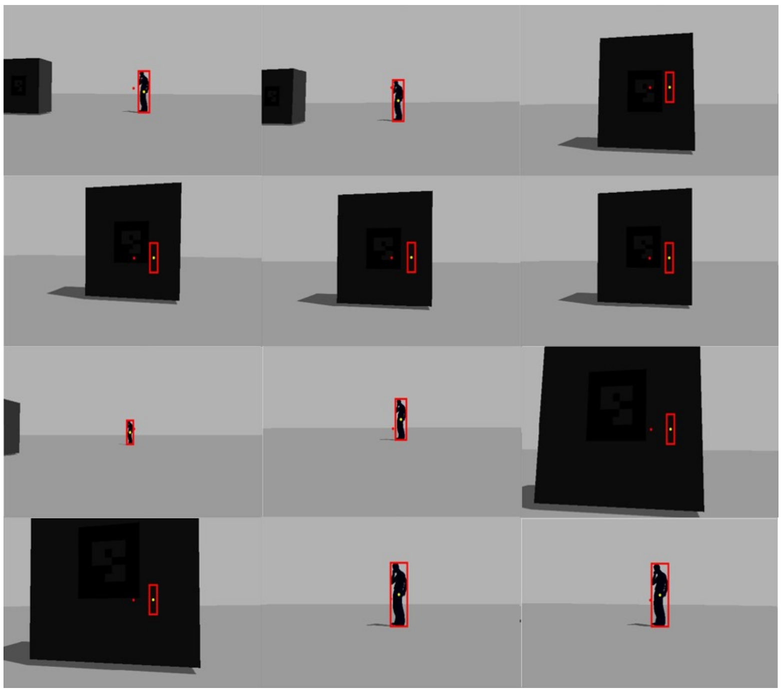
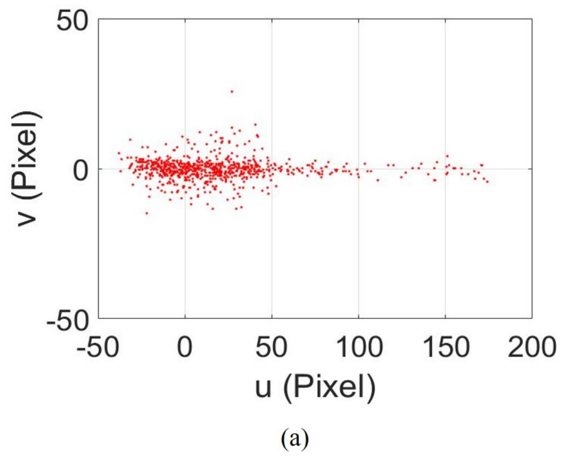


Figure 6.12: The HOG-based tracking method keeps the object in the camera's FOV during it's motion.

Figure 6.13.a shows the trajectory of the target in the image plane. The displacement of the car is mainly on the axis x, we naturally observe a larger position error along this axis. The average error is 9.522 pixels along the x axis and 1.696 pixels along the y axis. The fact that the average position is positive on the x axis presents the efficiency of the command to handle the acceleration of the car. The control signals generated by the quadrotor are shown in figure 6.14. The ranges of $t = [44, 85]s$ and $t = [183, 197]s$ correspond to times when the object was partially occluded. The ranges of V_x exceeds 0.1 m/s correspond to times when the object was considered in translation with a displacement speed of 1 m/s. Figure 6.11 shows that the car remains in the field of view of the camera, which allows the successful completion of the tracking task.



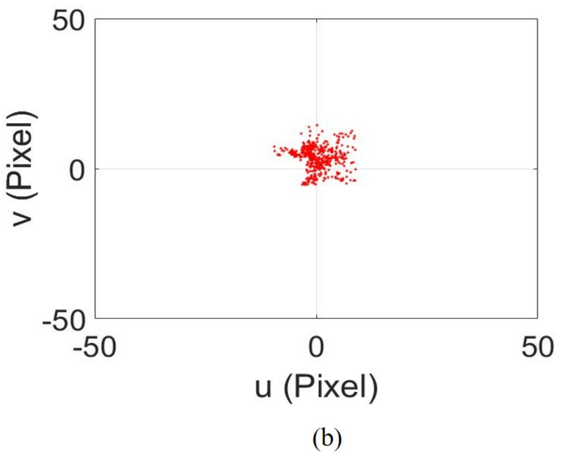


Figure 6.13: Trajectory of the center of (a) car, (b) man in pixellic coordinates. The origin corresponds to the desired position.

- **Teste 2**

  Figure 6.13.b shows the trajectory of the target in the image plane. The average error is 1.045 pixels on the x axis and 3.456 pixels on the y axis. The control signals generated are shown in figure 6.15. The ranges of $t = [85, 118]s$, $t = [177, 210]s$, $t = [266, 285]s$, $t = [348, 396]s$ and $t = [478, 513]s$ correspond to moments when the object was considered entirely occluded. Figure 6.13 shows that the object remains in the field of view of the camera which means that the HOG-based tracking method estimates the appropriate center of the object which helps the IBVS controller to give the appropriate command signals to the quadrotor to maintain the tracking task. While the command $\dot{x}$ has higher values, the other commands have an average close to zero because the drone follows an object in translation and must move forward to follow it. The command $\dot{y}$ is weak and it is used only during longitudinal movements of the drone. As shown in figure 6.16, it is obvious that the proposed tracking system was successful in keeping

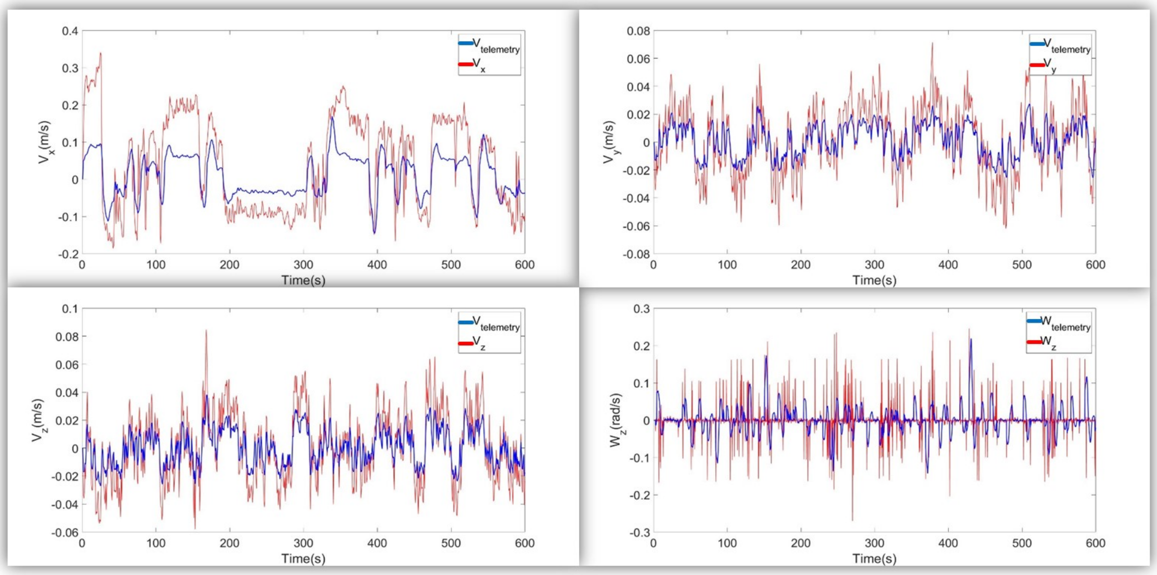


Figure 6.14: Commands values of $(\dot{x}, \dot{y}, \dot{z}, \dot{\psi})$ and corresponding values obtained by telemetry for the object car.

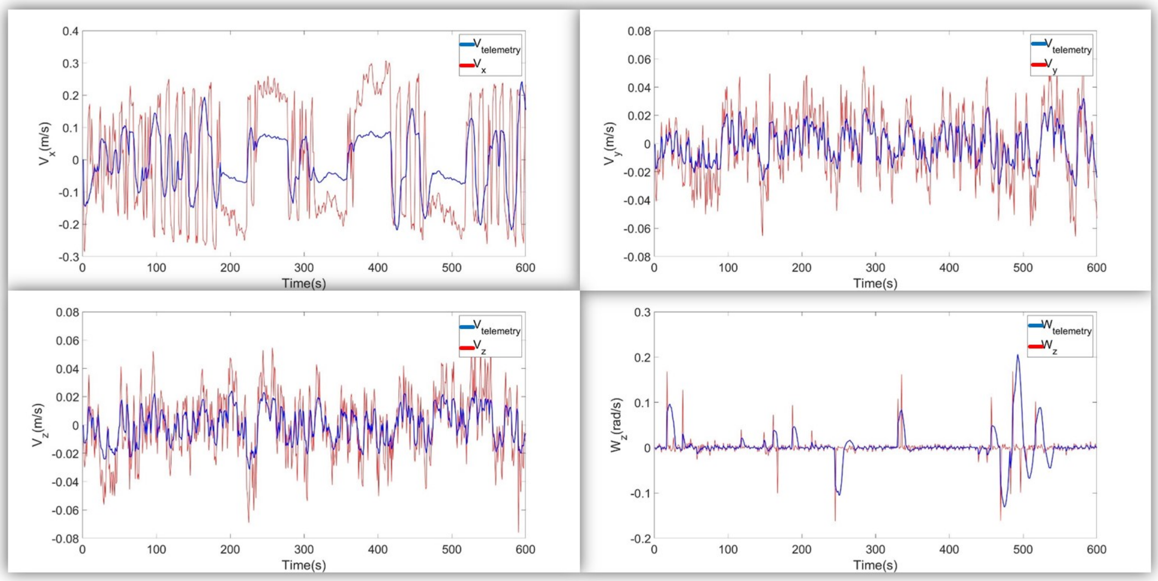


Figure 6.15: Commands values of $(\dot{x}, \dot{y}, \dot{z}, \dot{\psi})$ and corresponding values obtained by telemetry for the object man.

the values of $\hat{Z}_d$ close to 1 for both tests. This value means that the area of the initial tracking frame A_i matches the area of the current tracking frame A_a.

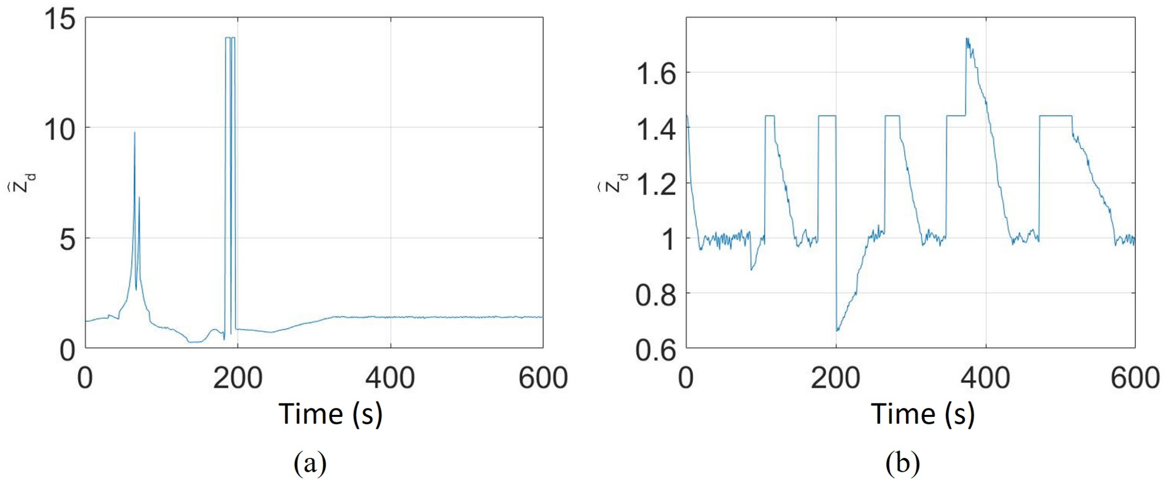


Figure 6.16: The command $\hat{Z}_d$ generated for the objects: (a) car and (b) man.

6.4 Conclusion

In this chapter, a visual object tracking application for a UAV drone is presented. The contribution of this is twofold: First, it has been demonstrated that the proposed HOG-based method can work reliably on the quadrotor's fixed camera. Second, our architecture has been able to track two difficult targets of different sizes and distances, showing the robustness of our system against occlusion, and scale variation, proving the efficiency and reliability of the IBVS-HOG based system.

CHAPTER 7

CONCLUSION

7.1 Conclusion

The need to implement innovative and reliable visual tracking methods, to explore the capabilities of advanced computer vision methods and their applicability in robotics. In this thesis, we address the problem of visual tracking of a single object, which aims primarily to estimate the position of a target object in each frame of a video. In particular, correlation filter-based tracking methods are adopted thanks to their outstanding performance, their computational efficiency, and the efficient updating of models. The main contributions of this thesis are summarized below:

- As explained in chapter 3, two methods had been proposed for visual object tracking. The first method proposed is mainly composed of three fundamental steps: spatial learning of correlation filter models using convolutional features, enrichment of CNN features using the HSV energy condition and the use of HOG features to exhaustively search for the optimal scale. The spatial learning of the correlation filter models is based on the PSO algorithm which has mainly improve the performances in the case of occlusion and fast motion. Besides, a substantial advantage has been proposed for illumination variations by switching between the RGB and HSV color bases. We evaluate our method on the three reference databases including OTB-50 [35], VOT 2016 [3], UAV20L [2].

  The results reveal the superiority of the proposed methods compared to the reference trackers in terms of short-term on OTB-50, in where we achieve the best distance precision of 91.0 % and the best overlap success rate of 76.5 %. On VOT 2016, our tracker demonstrates its effectiveness by obtaining the best EAO score of 0.31 and getting the third-highest overlap score of 0.50 and the best result in failures metric with 15.09 %. Regarding the long-term tracking capacity, our tracker proves its efficiency on the UAV20L dataset, where it achieves the best distance precision of 64.7 % and the best overlap success rate of 48.0 %.

  We proposed a second method for real-time application. This method adopts the HOG features to robustly estimate the location of the target object. In terms of scale estimation, the HOG features are adopted in both methods to exhaustively search for the optimal scale. The same three reference databases of OTB-50 [35], VOT 2016 [3], UAV20L [2] are used to evaluate the performance of proposed method.

  On the OTB-50 dataset, a comparison with eight reference trackers is realized. The MEEM tracker obtains the second-best result with a distance accuracy of 83.0% and an overlap success of 69.6%. The proposed method proves its efficiency with an improvement gain of 1.1 % at the distance precision and 1.8 % at the overlap success. The results reported on the VOT 2016 show that the proposed tracker demonstrates its effectiveness by obtaining the best EAO score of 0.32. In addition, it provides a considerable result of 0.5 in overlap metric and the second-best result of 17.60% in failure metric. Finally, we evaluate our method on the UAV20L database. Our tracker surpasses the other trackers on distance precision (DP) and overlap success (OS) metrics with an improved gain of 1.1 % and 0.9 % respectively against the competitor tracker.

- A visual object tracking architecture was designed, implemented and evaluated. This architecture has demonstrated that the proposed HOG-IVBS tracking framework can work reliably on the quadrotor's fixed camera. Our architecture has been able to follow two difficult targets in terms of occlusion, fast motion, and scale variation. In addition, it has been validated using the robotic operating system (ROS). The system has been successful in solving these problems.

7.2 Future works

Visual tracking systems become very important in the daily life of human beings, whether for protection or other purposes.

- Our first perspective is related to visual object tracking. Since CNN-based tracking is one of the best tracking methods. Despite, this method remains restrictive in real-time because of their computation time. In the future, we plan to solve this limitation by re-forming the CNN model and reducing the number of layers without affecting their performance. We aim also to validate the two proposed tracking methods, especially the ability of the proposed HSV-energy condition in a real application.
- The second perspective is related to the reliability of the image-based control system when the tracked object leaves the camera's FOV and takes a long time to reappears in several environments. In fact, we aim to alleviate the issue of the estimation of the interaction matrix by proposing to learn a CNN architecture over the input frames and the quadrotor poses.

BIBLIOGRAPHY

[1] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, “The navigation and control technology inside the ar. drone micro uav,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1477–1484, 2011.

[2] M. Mueller, N. Smith, and B. Ghanem, “A benchmark and simulator for uav tracking,” in *European conference on computer vision*. Springer, 2016, pp. 445–461.

[3] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin Zajc, T. Vojir, G. Häger, A. Lukežič, and G. Fernandez, “The visual object tracking vot2016 challenge results,” Springer, Oct 2016. [Online]. Available: http://www.springer.com/gp/book/9783319488806

[4] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[5] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, “Accurate scale estimation for robust visual tracking,” in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[6] A. Karpathy, F. Li, and J. Johnson, “Cs231n: Convolutional neural networks for visual recognition, 2016,” *URL http://cs231n. github. io*, 2017.

[7] J. Johnander, “Visual tracking with deformable continuous convolution operators,” 2017.

[8] K. Naidoo, O. Konan, and L. Schwartzkopff, “Hardware accelerated convolutional neural networks.”

[9] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[10] N. Dalal, “Finding people in images and videos,” Ph.D. dissertation, Institut National Polytechnique de Grenoble-INPG, 2006.

[11] D. E. Touil, N. Terki, and S. Medouakh, “Learning spatially correlation filters based on convolutional features via pso algorithm and two combined color spaces for visual tracking,” *Applied Intelligence*, vol. 48, no. 9, pp. 2837–2846, 2018.

[12] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, “Hierarchical convolutional features for visual tracking,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3074–3082.

[13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[14] J. Zhang, S. Ma, and S. Sclaroff, “Meem: robust tracking via multiple experts using entropy minimization,” in *European Conference on Computer Vision*. Springer, 2014, pp. 188–203.

[15] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[16] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-learning-detection,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[17] L. R. G. Carrillo, A. E. D. López, R. Lozano, and C. Pégard, *Quad rotorcraft control: vision-based hovering and navigation*. Springer Science & Business Media, 2012.

[18] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised*. Springer, 2017, vol. 118.

[19] R. Barták and A. Vykovskỳ, “Any object tracking and following by a flying drone,” in *Artificial Intelligence (MICAI), 2015 Fourteenth Mexican International Conference on*. IEEE, 2015, pp. 35–41.

[20] W. Sun, Y. Pan, J. Lim, E. A. Theodorou, and P. Tsiotras, “Min-max differential dynamic programming: Continuous and discrete time formulations,” *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 12, pp. 2568–2580, 2018.

[21] D. L. B. MathWorks, “Documentation (2018),” 2018.

[22] T. Liu, G. Wang, and Q. Yang, "Real-time part-based visual tracking via adaptive correlation filters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4902–4912.

[23] Y. Li, J. Zhu, and S. C. Hoi, "Reliable patch trackers: Robust visual tracking by exploiting reliable patches," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 353–361.

[24] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 472–488.

[25] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5388–5396.

[26] H. Fan and H. Ling, "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking," in *Proc. IEEE Int. Conf. Computer Vision, Venice, Italy*, 2017.

[27] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[28] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking." in *ICCV*, 2017, pp. 1144–1152.

[29] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2544–2550.

[30] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic visual servo control of robots: an adaptive image-based approach," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2. IEEE, 1985, pp. 662–668.

[31] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *ieee Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, 1992.

[32] A. Yger, "Traitement du signal et ondelettes," 2006.

[33] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.

[34] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[35] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt *et al.*, “The visual object tracking vot2013 challenge results,” in *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*. IEEE, 2013, pp. 98–111.

[36] X.-S. Yang and X. He, “Bat algorithm: literature review and applications,” *International Journal of Bio-Inspired Computation*, vol. 5, no. 3, pp. 141–149, 2013.

[37] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, “Visual tracking: An experimental survey,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 1, p. 1, 2013.

[38] E. G. GDU, “Good features to correlate for visual tracking,” Ph.D. dissertation, MIDDLE EAST TECHNICAL UNIVERSITY, 2017.

[39] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, “A survey of appearance models in visual object tracking,” *ACM transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, p. 58, 2013.

[40] S. He, Q. Yang, R. W. Lau, J. Wang, and M.-H. Yang, “Visual tracking via locality sensitive histograms,” in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2427–2434.

[41] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[42] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[43] S. Avidan, “Ensemble tracking,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 2, 2007.

[44] Q. Gan, Q. Guo, Z. Zhang, and K. Cho, “First step toward model-free, anonymous object tracking with recurrent neural networks,” *arXiv preprint arXiv:1511.06425*, 2015.

[45] Q. Bai, Z. Wu, S. Sclaroff, M. Betke, and C. Monnier, “Randomized ensemble tracking,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2040–2047.

[46] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time object tracking via online discriminative feature selection," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4664–4677, 2013.

[47] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 983–990.

[48] K. Zhang, L. Zhang, and M.-H. Yang, "Fast compressive tracking," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 1, pp. 1–1, 2014.

[49] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. Van Den Hengel, "Part-based visual tracking with online latent structural learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2363–2370.

[50] J. Ning, J. Yang, S. Jiang, L. Zhang, and M.-H. Yang, "Object tracking via dual linear structured svm and explicit feature map," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4266–4274.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[52] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[53] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1265–1274.

[54] J. Jin, A. Dundar, J. Bates, C. Farabet, and E. Culurciello, "Tracking with deep neural networks," in *Information Sciences and Systems (CISS), 2013 47th Annual Conference on*. IEEE, 2013, pp. 1–5.

[55] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *International Conference on Machine Learning*, 2015, pp. 597–606.

[56] X. Mei and H. Ling, "Robust visual tracking using 1 minimization," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1436–1443.

[57] C. Bao, Y. Wu, H. Ling, and H. Ji, “Real time robust l1 tracker using accelerated proximal gradient approach,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1830–1837.

[58] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan, “Robust object tracking with online multi-lifespan dictionary learning,” in *Proceedings of the IEEE International conference on computer vision*, 2013, pp. 665–672.

[59] Y. Wu, B. Shen, and H. Ling, “Visual tracking via online nonnegative matrix factorization,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 3, pp. 374–383, 2014.

[60] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, “Robust visual tracking via structured multi-task sparse learning,” *International journal of computer vision*, vol. 101, no. 2, pp. 367–383, 2013.

[61] L. Wang, W. Ouyang, X. Wang, and H. Lu, “Visual tracking with fully convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3119–3127.

[62] A. V. Oppenheim, *Discrete-time signal processing*. Pearson Education India, 1999.

[63] A. Mahalanobis, B. V. Kumar, and D. Casasent, “Minimum average correlation energy filters,” *Applied Optics*, vol. 26, no. 17, pp. 3633–3640, 1987.

[64] P. Réfrégier, “Optimal trade-off filters for noise robustness, sharpness of the correlation peak, and horner efficiency,” *Optics Letters*, vol. 16, no. 11, pp. 829–831, 1991.

[65] W. Zuo, X. Wu, L. Lin, L. Zhang, and M.-H. Yang, “Learning support correlation filters for visual tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[66] A. Bibi and B. Ghanem, “Multi-template scale-adaptive kernelized correlation filters,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 50–57.

[67] Y. Li and J. Zhu, “A scale adaptive kernel correlation filter tracker with feature integration,” in *European conference on computer vision*. Springer, 2014, pp. 254–265.

[68] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, “Learning spatially regularized correlation filters for visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4310–4318.

[69] ——, “Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1430–1438.

[70] R. Tao, E. Gavves, and A. W. Smeulders, “Siamese instance search for tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1420–1429.

[71] J. L. Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[72] Z. Cui, S. Xiao, J. Feng, and S. Yan, “Recurrently target-attending tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1449–1458.

[73] S. E. Kahou, V. Michalski, and R. Memisevic, “Ratm: recurrent attentive tracking model,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2015, pp. 1613–1622.

[74] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, “The visual object tracking vot2015 challenge results,” in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 1–23.

[75] Y. Wu, J. Lim, and M.-H. Yang, “Object tracking benchmark,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[76] R. Tokola and D. Bolme, “Ensembles of correlation filters for object detection,” in *2015 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2015, pp. 935–942.

[77] T. B. Dinh, Q. Yu, and G. Medioni, “Co-trained generative and discriminative trackers with cascade particle filter,” *Computer Vision and Image Understanding*, vol. 119, pp. 41–56, 2014.

[78] M. George, B. R. Jose, and J. Mathew, “Performance evaluation of kcf based trackers using vot dataset,” *Procedia Computer Science*, vol. 125, pp. 560–567, 2018.

[79] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[80] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[81] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[82] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[83] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[84] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[85] M. A. Nielsen, *Neural networks and deep learning*. Determination press USA, 2015, vol. 25.

[86] Ç. F. Özgenel and A. G. Sorguç, "Performance comparison of pretrained convolutional neural networks on crack detection in buildings," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 35. IAARC Publications, 2018, pp. 1–8.

[87] Y. Yun, "Analysis and classification of object poses-using visual/infrared images and feature fusion," 2011.

[88] W. Wei, Z. Sun, H. Song, H. Wang, and X. Fan, "Energy balance-based steerable arguments coverage method in wsns," *IEEE Access*, 2017.

[89] W. Wei, X.-L. Yang, B. Zhou, J. Feng, and P.-Y. Shen, "Combined energy minimization for image reconstruction from few views," *Mathematical Problems in Engineering*, vol. 2012, 2012.

[90] E. Dahlman, C. Oestges, A. C. Bovik, B. A. Fette, K. Jack, F. Dowla, S. Parkvall, J. Skold, C. DeCusatis, E. da Silva *et al.*, *Communications engineering desk reference*. Academic Press, 2009.

[91] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Adaptive correlation filters with long-term and short-term memory for object tracking," *arXiv preprint arXiv:1707.02309*, 2017.

[92] V. N. Boddeti, T. Kanade, and B. V. Kumar, “Correlation filters for object alignment,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013, pp. 2291–2298.

[93] Y. Li, Y. Zhang, Y. Xu, J. Wang, and Z. Miao, “Robust scale adaptive kernel correlation filter tracker with hierarchical convolutional features,” *IEEE Signal Processing Letters*, vol. 23, no. 8, pp. 1136–1140, 2016.

[94] C. Mollaret, F. Lerasle, I. Ferrané, and J. Pinquier, “A particle swarm optimization inspired tracker applied to visual tracking,” in *IEEE International Conference on Image Processing (ICIP 2014)*, 2014, pp. pp–426.

[95] J. Fourie, S. Mills, and R. Green, “Harmony filter: a robust visual tracking system using the improved harmony search algorithm,” *Image and Vision Computing*, vol. 28, no. 12, pp. 1702–1716, 2010.

[96] J. Brownlee, “Clever algorithms: Nature-inspired programming recipes, lulu. com,” 2012.

[97] A. Vedaldi and K. Lenc, “Matconvnet: Convolutional neural networks for matlab,” in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 689–692.

[98] N. Wang and D.-Y. Yeung, “Learning a deep compact image representation for visual tracking,” in *Advances in neural information processing systems*, 2013, pp. 809–817.

[99] J. Gao, H. Ling, W. Hu, and J. Xing, “Transfer learning based visual tracking with gaussian processes regression,” in *European Conference on Computer Vision*. Springer, 2014, pp. 188–203.

[100] W. Zhong, H. Lu, and M.-H. Yang, “Robust object tracking via sparse collaborative appearance model,” *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2356–2368, 2014.

[101] D. Chen, Z. Yuan, Y. Wu, G. Zhang, and N. Zheng, “Constructing adaptive complex cells for robust visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1113–1120.

[102] G. Zhu, F. Porikli, and H. Li, “Beyond local search: Tracking objects everywhere with instance-specific proposals,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 943–951.

[103] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[104] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1401–1409.

[105] Z. Chi, H. Li, H. Lu, and M.-H. Yang, "Dual deep network for visual tracking." *IEEE Trans. Image Processing*, vol. 26, no. 4, pp. 2005–2015, 2017.

[106] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[107] D. E. Touil, N. Terki, and S. Medouakh, "Hierarchical convolutional features for visual tracking via two combined color spaces with svm classifier," *Signal, Image and Video Processing*, vol. 13, no. 2, pp. 359–368, 2019.

[108] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.

[109] A. Azzam and X. Wang, "Quad rotor arial robot dynamic modeling and configuration stabilization," in *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, vol. 1. IEEE, 2010, pp. 438–444.

[110] Z. Sarris and S. Atlas, "Survey of uav applications in civil markets," in *IEEE Mediterranean Conference on Control and Automation*, 2001, p. 11.

[111] B. Hu, L. Lu, and S. Mishra, "Fast, safe and precise landing of a quadrotor on an oscillating platform," in *American Control Conference (ACC), 2015*. IEEE, 2015, pp. 3836–3841.

[112] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a vtol uav on a moving platform using image-based visual servoing," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 971–976.

[113] T. Yüksel, "An intelligent visual servo control system for quadrotors," *Transactions of the Institute of Measurement and Control*, p. 0142331217751599, 2018.

[114] S. A. Raza and J. Etele, "Autonomous position control analysis of quadrotor flight in urban wind gust conditions," in *AIAA Guidance, Navigation, and Control Conference*, 2016, p. 1385.

[115] T. Hamel and R. Mahony, "Visual servoing of an under-actuated dynamic rigid-body system: an image-based approach," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 187–198, 2002.

[116] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *Robotics and automation, 2002. Proceedings. ICRA'02. IEEE international conference on*, vol. 3. IEEE, 2002, pp. 2799–2804.

[117] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta, "Vision-based road-following using a small autonomous aircraft," in *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, vol. 5. IEEE, 2004, pp. 3006–3015.

[118] Z. Ceren and E. Altuğ, "Vision-based servo control of a quadrotor air vehicle," in *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation-(CIRA)*. IEEE, 2009, pp. 84–89.

[119] D. Lee, H. Lim, H. J. Kim, Y. Kim, and K. J. Seong, "Adaptive image-based visual servoing for an underactuated quadrotor system," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 4, pp. 1335–1353, 2012.

[120] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[121] R. Ozawa and F. Chaumette, "Dynamic visual servoing with image moments for a quadrotor using a virtual spring approach," in *IEEE Int. Conf. on Robotics and Automation, ICRA'11*, 2011, pp. 5670–5676.

[122] F. Chaumette and S. Hutchinson, "Visual servo control. ii. advanced approaches [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[123] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Quadrotor landing on an inclined platform of a moving ground vehicle," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2202–2207.

[124] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," in *Geometric reasoning for perception and action*. Springer, 1993, pp. 106–136.

[125] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 713–723, 2004.

[126] Y.-M. Zhao, W.-F. Xie, S. Liu, and T. Wang, "Neural network-based image moments for robotic visual servoing," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 2, pp. 239–256, 2015.

[127] G. Fink, H. Xie, A. F. Lynch, and M. Jagersand, "Experimental validation of dynamic visual servoing for a quadrotor using a virtual camera," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*. IEEE, 2015, pp. 1231–1240.

[128] H. Xie and A. F. Lynch, "Input saturated visual servoing for unmanned aerial vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 952–960, 2017.

[129] R. Mahony, P. Corke, and T. Hamel, "Dynamic image-based visual servo control using centroid and optic flow features," *Journal of Dynamic Systems, Measurement, and Control*, vol. 130, no. 1, p. 011005, 2008.

[130] H. De Plinval, P. Morin, P. Mouyon, and T. Hamel, "Visual servoing for underactuated vtol uavs: a linear, homography-based framework," *International Journal of Robust and Nonlinear Control*, vol. 24, no. 16, pp. 2285–2308, 2014.

[131] C. Teulière, "Approches déterministes et bayésiennes pour un suivi robuste: application à l'asservissement visuel d'un drone," Ph.D. dissertation, Université Rennes 1, 2010.

[132] H. K. Khalil and J. Grizzle, *Nonlinear systems*. Prentice hall Upper Saddle River, NJ, 2002, vol. 3.

[133] J. R. King, "Quadrotor visual servoing for automatic landing," Ph.D. dissertation, 2017.

[134] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Epfl, Tech. Rep., 2007.

[135] A. Prayitno, V. Indrawati, and G. Utomo, "Trajectory tracking of ar. drone quadrotor using fuzzy logic controller," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 12, no. 4, pp. 819–828, 2014.

[136] B. E. Demir, R. Bayir, and F. Duran, "Real-time trajectory tracking of an unmanned aerial vehicle using a self-tuning fuzzy proportional integral derivative controller," *International Journal of Micro Air Vehicles*, vol. 8, no. 4, pp. 252–268, 2016.

[137] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *None*, no. LSA-CONF-2005-003, 2005.

[138] I. Gonzalez, S. Salazar, H. Romero, R. Lozano, and J. Torres, "Attitude control of a quad-rotor using speed sensing in brushless dc motors," in *Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on*. IEEE, 2011, pp. 1–6.

[139] M. Moussid, A. Sayouti, and H. Medromi, "Dynamic modeling and control of a hexarotor using linear and nonlinear methods," *International Journal of Applied Information Systems*, vol. 9, no. 5, 2015.

[140] O. Doukhi, A. R. Fayjie, and D. J. Lee, "Intelligent controller design for quad-rotor stabilization in presence of parameter variations," *Journal of Advanced Transportation*, vol. 2017, 2017.

[141] R. Beard, "Quadrotor dynamics and control rev 0.1," 2008.

[142] Z. Bellahcene, M. Bouhamida, M. Benghanem, and A. Laidani, "La commande intégrale backstepping appliquée à un hélicoptère à quatre hélices," in *2nd International Conference on Maintenance, Management, Logistics and Electrical Engineering, ENSET Oran*, 2012, pp. 19–21.

[143] A. F. Robots, "Unmanned aerial vehicles and micro aerial vehicles," *Kenzo Nonami, Farid Kendoul, Satoshi Suzuki and other.- Springer Tokyo, New York*, 2010.

[144] Z. BELLAHCENE, "Synthèse de lois de commande robuste pour un hélicoptère à quatre hélices," Ph.D. dissertation, usto, 2013.

[145] A. Nagaty, S. Saeedi, C. Thibault, M. Seto, and H. Li, "Control and navigation framework for quadrotor helicopters," *Journal of intelligent & robotic systems*, vol. 70, no. 1-4, pp. 1–12, 2013.

[146] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, 2010, pp. 65–74.

[147] M.-L. Gao, J. Shen, L.-J. Yin, W. Liu, G.-F. Zou, H.-T. Li, and G.-X. Fu, “A novel visual tracking method using bat algorithm,” *Neurocomputing*, vol. 177, pp. 612–619, 2016.

[148] A. D. Wu, E. N. Johnson, and A. A. Proctor, “Vision-aided inertial navigation for flight control,” *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 9, pp. 348–360, 2005.

[149] E. Cervera, “Visual servoing toolbox,” *Jaume I University, Castello*, 2003.

[150] J. Pestana Puerta, “Vision-based autonomous navigation of multirotor micro aerial vehicles,” Ph.D. dissertation, Industriales, 2017.

[151] [Online]. Available: http://github.com/

[152] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, “Ar-drone as a platform for robotic research and education,” in *International conference on research and education in robotics*. Springer, 2011, pp. 172–186.

[153] J. M. O’Kane, “A gentle introduction to ros,” 2014.

[154] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[155] P. Corke, “Integrating ros and matlab [ros topics],” *IEEE Robotics & Automation Magazine*, vol. 22, no. 2, pp. 18–20, 2015.

[156] M. Monajjemi *et al.*, “ardrone autonomy: A ros driver for ardrone 1.0 & 2.0,” 2012.

[157] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, “Comprehensive simulation of quadrotor uavs using ros and gazebo,” in *International conference on simulation, modeling, and programming for autonomous robots*. Springer, 2012, pp. 400–411.

[158] U. Ananthakrishnan, N. Akshay, G. Manikutty, and R. R. Bhavani, “Control of quadrotors using neural networks for precise landing maneuvers,” in *Artificial Intelligence and Evolutionary Computations in Engineering Systems*. Springer, 2017, pp. 103–113.

[159] A. Chakrabarty, R. Morris, X. Bouyssounouse, and R. Hunt, “Autonomous indoor object tracking with the parrot ar. drone,” in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*. IEEE, 2016, pp. 25–30.

[160] N. P. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator.” in *IROS*, vol. 4. Citeseer, 2004, pp. 2149–2154.

[161] Google, “3d warehouse,” 2014. [online]. available: http://sketchup. google.com/3dwarehouse. [Online]. Available: Google,\T1\textquotedblleft3dwarehouse,\T1\textquotedblright2014. [Online].Available:http://sketchup.google.com/3dwarehouse/