Coastline Tracking for UAVs Using Event-Triggered Image-Based Visual Servoing Nonlinear Model Predictive Control

Mario Sinani

Supervisor: Professor
K.J. Kyriakopoulos

School of Mechanical Engineering
National Technical University of Athens

This dissertation is submitted for the degree of
Diploma in Mechanical Engineering
October 2021

Acknowledgements

First, I would like to thank Professor Konstantinos Kyriakopoulos for providing me with the opportunity to investigate such an interesting topic through my diploma thesis at the Control Systems Laboratory at NTUA. His lectures on robotics and control during my studies sparked my interest to pursue further studies and research in these topics. His advice and guidance regarding my work and future studies has been immensely helpful.
Furthermore, I would like to thank Dr. George Karras for his valuable support during my thesis. His insightful directions and spherical approach contributed significantly towards the resolve of both theoretical and implementation issues.
I also want to acknowledge Sotiris Aspragkathos of the Control Systems Laboratory and thank him for his support with implementation and technical issues.
Finally, I would like to thank my family, Frederika, Albert and Klaudio, as well as my friends for the continuous support and encouragement throughout the course of my studies.

Abstract

Although Model Predictive Control (MPC) has prominent advantages pertaining to nonlinear systems under state and actuator constraints, the high-resource consumption of the algorithm due to the constant requirement of state feedback and the high computational cost of the optimization at each time step limits the spectrum of possible applications. This drawback of MPC is amplified when combined with Image-Based Visual Servoing and quadrotors due to the high computational cost of the Visual Tracking Algorithm (VTA) and the limited battery life of the vehicles. Hence, applications such as coastline surveillance using Unmanned Aerial Vehicles (UAV) pose challenges related to the low autonomy of the vehicles and the disturbances due to the VTA noise from the vision system. In this thesis, a Event-Triggered IBVS-NMPC scheme for coastline tracking using a quadrotor is proposed which aims at updating the optimal control signal sequence as sparsely as possible. Additionally, a fast C++ code is developed and implemented in realistic simulations, in which the efficacy and performance of the control scheme is demonstrated.

Περίληψη

Παρόλο που ο προβλεπτικός έλεγχος παρουσιάζει σημαντικα πλεονεκτήματα σε περιπτώσεις μη-γραμμικών συστημάτων υπο περιορισμούς κατάστασης και εισόδου, η υψηλή κατανάλωση πόρων του αλγορίθμου λόγω της συνεχής ανάγκης ανανέωσης της κατάστασης και του υψηλού υπολογιστικού κόστους που επιφέρει η επίλυση του προβλήματος του βέλτιστου ελέγχου σε κάθε χρονική στιμή, περιορίζει το φάσμα των πιθανών εφαρμογών του. Το μειονέκτημα αυτό γίνεται περισσότερο εμφανές σε εφαρμογές οπτικής ανατροφοδότησης με quadrotors λόγω του υψηλού υπολογιστικού κόστους και των χαμηλής διάρκειας μπαταριών. Επομένως οι εφαρμογές παρακολούθησης ακτογραμμής δυσχαιρένονται λόγω του θορύβου απο το σύστημα όρασης και της χαμηλής αυτονομίας του ιπτάμενου οχήματος. Σκοπός της εργασίας είναι ο σχεδιασμός ενός προβλεπτικού νόμου ελέγχου βασισμένου σε διεγέρσεις απο συμβάντα μέσω οπτικής ανατροφοδότησης που θα λύνει το πρόβλημα του βέλτιστου ελέγχου μόνο όταν είναι απαραίτητο. Επιπλέον παρουσιάζουμε έναν κώδικα σε C++ που εφαρμόστηκε σε ρεαλιστικό περιβάλλον προσομοίωσης για να αποδειχθεί η αποτελεσματικότητα του νόμου ελέγχου.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

The tracking of coastlines by Unmanned Aerial Vehicles (UAV) under an EventTriggered-Image Based Visual Servoing-Model Predictive Control (ET-IBVS-MPC) policy constitutes the main topic of this thesis.

The coastline tracking problem consists of a UAV equipped with on-board sensors and a stereo vision camera, which allow it to perceive the surrounding environment that it navigates in, traversing a coastline in order to precisely track it. The coastline detection is achieved on-line through a trained Convolutional Neural Network (CNN) running on a GPU. Once the coastline is detected, an irregular rectangular region of interest (ROI) is formed which encompasses the coastline in the image frame. The four corners of the ROI are used in the Image-Based Visual Servoing scheme as features whose desired positions are required to be tracked as the UAV moves along the coastline. In our studied case, our UAV is a quadrotor employing autopilot software to handle the low-level control of the vehicle. Furthermore, we have assumed that accurate localization throughout the tracking task is achieved. Once the coastline has been detected and the IBVS features are defined, the ET-MPC framework is triggered and the Optimal Control Problem (OCP) is solved based on the current state of the system in order to generate the control signal sequence over the receding horizon in the frame of the camera. This output is sent to the autopilot which generates the required torques and thrusts, which are transformed into voltages for each of the four individual rotors, in order to position the UAV to the desired state.

The part of the computed control signal sequence which is applied is defined by the event-triggering rule and depends on the error between the current and the predicted nominal state of the system. The main idea is that the control sequence provided by the controller is applied to the system in an open-loop fashion between actuator updates. Once the triggering

condition is violated, feedback from the Visual Tracking Algorithm (VTA) is obtained and the OCP is solved again in order to calculate the subsequent control signal sequence over the receding horizon. Flight in a real-time complex environment, such as the one in coastal areas, demands high computational cost in order to accurately and robustly execute the Visual Tracking Algorithm (VTA) and calculate the solution of the OCP at each time step. It is thus imperative to dictate an increase in the intervals between two triggers in order to achieve success in coastline surveillance missions. which in many cases occur in remote areas from UAVs with battery limitations.
Ultimately, the aim of the thesis is to propose an Event-Triggered NMPC algorithm that can significantly reduce energy consumption while guaranteeing feasibility from a variety of challenging initial configurations, stability and converge, and satisfying performance characteristics.

1.2 Motivation

Rapid advancements in sensor technologies, communication systems and computation have enabled the use of UAVs in a broad range of applications such as mapping, surveillance, environmental surveying and search and rescue missions. However, battery limitations can make long missions particularly challenging and operation in uncertain or dynamic environments requires resilience to disturbances and parameter uncertainty. As previously mentioned, despite the effectiveness and robustness of MPC schemes in tracking problems, the resource demands make such schemes difficult to apply. This issue becomes more apparent in long inspection and surveillance missions in complex environments that require both accurate VTA and a strong 8 autonomy rate. Furthermore, it can also deteriorate the accuracy and performance of the scheme by dictating increased sampling periods. An available solution entails recharging trips for the UAV from the location where the battery is compromised back to the Ground Control Station (GCS). However, this makes missions time-consuming and can be forbidden in challenging environments. Therefore, an alternative approach to reduce resource consumption becomes imperative.
Event-based MPC policies have been proven effective in causing significant reductions in energy usage while concurrently maintaining the stability and performance of the system. Hence, the question of how often do we need to trigger the MPC algorithm arises. This thesis aims to provide an answer to this question by generating an algorithm that can satisfy the aforementioned stability and performance characteristics, and trigger the VTA as well as the solution of the OCP as sparsely as possible.

1.3 Literature Review

1.3.1 Image-Based Visual Servoing

Accurate visual data are crucial for robots to safely navigate in unknown, dynamic environments. Vision systems allow robots to obtain three-dimensional geometric and qualitative image data of their surrounding environment that allow them to operate in it and execute tasks [1]. Some of the first studies for robots used in manipulation tasks for objects based on visual data were presented both in theory and experimentally in [2], [3], and [4]. In order to compensate for error between the real and the desired state of the system, the author in [5], employed a visual feedback loop in order to adjust the position of robotic manipulators and therefore increase the accuracy of the task to the level of the resolution for both the visual and mechanical system. This closed-loop feedback policy came to become known as visual servoing. Visual servoing consists of three main types: (i) Position-Based Visual Servoing (PBVS), (ii) Image-Based Visual Servoing (IBVS) and (iii) 2-1/2 Visual Servoing. Despite the potential visibility constraints, convergence and stability problems, which are investigated extensively in [6], IBVS does not suffer from the potential calibration and robustness problems of PBVS that are presented in [7]. Furthermore, methods for global path planning of the image features in accordance with the movement of the camera for robust IBVS, as the ones presented in [8] and [9], allow for Field Of View (FOV) constraints to be satisfied. A comprehensive study on visual servoing and its characteristics are presented in [10] and [11] by Chaumette and Hutchinson.

The aforementioned attributes of IBVS make it more appropriate for stabilization and tracking problems for UAVs as it was investigated by Kanellakis and Nikolakopoulos in [12]. A wide variety of IBVS algorithms pertaining to the projection of the image features and the control design type, for the stabilization problem of a quadrotor was successfully implemented experimentally in [13] and demonstrated exceptional performance. IBVS for UAVs using a single camera is proposed in [14] for the detection and navigation in an initially known environment in order to achieve a safe landing. A novel two-camera scheme with an onboard and a ground-located camera is used to estimate the pose of the quadrotor in [15]. The authors ensure autonomy for the quadrotor by implementing a feedback linearizing controller as well as a backstepping control scheme in experimental flights. Other control designs have been studied in the literature. For instance, in [16] and [17], the authors employed a PID control law for the tracking of angular momentum and thrust. Similar to [16] and [17], the authors in [18] utilize a low-level feedback controller without considering input constraints in the control scheme. However, they proposed a control scheme for a small lightweight quadrotor using only a single camera and an Inertial Measurement Unit (IMU) to achieve

a robust state estimation and, plan and execute dynamic trajectories in an obstacle-filled environment. Contrary to [18], high-level minimum-time trajectory planning which accounts for state uncertainty, collision probability, and visibility constraints was presented in [19] and [20]. In the [21] paper, a dynamic IBVS approach for the under-actuation problem of UAVs was presented. The aforementioned approach designed an IBVS controller considering the dynamics of the UAV as well as a combination of the inertial data of the robot orientation and the visual data from the vision system. Lastly, appropriate perspective image moments are employed in order to guarantee satisfying trajectories.

Model Predictive Control has been successfully applied in combination with visual servoing in a multitude of studies and experiments. In [22], the authors proposed an MPC-IBVS control scheme for a 6 DOF mechanical system in which great robustness to modeling uncertainty was achieved by incorporating a terminal cost into the OCP of the MPC problem. Furthermore, the scheme performed well against camera calibration error and image measurement noise. Similar to [22], the authors in [23] and [24] developed an MPC-IBVS control scheme that accounted for mechanical and FOV constraints while in parallel reduced the computational time of the constrained optimization problem by employing the concept of differential flatness. In the Visual Servoing via Nonlinear Predictive Control chapter of Chesi's and Hashimoto's book on Visual Servoing via Advanced Numerical Methods [25], the advantages and drawbacks of both global and local approaches are discussed. It is shown that the approximated local model lacks efficiency in comparison to the global one for challenging configurations but without the need for three-dimensional visual information. On the contrary, if such data is available, the global method is superior for any initial pose and trajectory to achieve. The authors expanded their work in [26], where input constraints were also considered and the choice between a local and a global model for the image prediction is made depending on the constraints that are imposed on the NMPC scheme. The under-actuation issue found in 6 DOF quadrotors under IBVS schemes is solved in [27] by employing a linearized MPC scheme to relax the issue while in parallel guaranteeing stability, actuator saturation, and FOV constraints. An IBVS-NMPC strategy was also proposed in [28] for a fixed-wing UAV, in which tracking of the desired trajectory was achieved while the UAV performed visual data-based-obstacle-avoidance maneuvers in an unknown environment under visibility and actuator constraints.

1.3.2 Coastline Surveillance

Coastline detection has been a well-studied topic in the existing literature. However, little research has into surveillance and tracking been conducted. On the contrary, there has been a multitude of papers on tracking and surveillance of curvilinear structures such as roads using both mobile and aerial robots. In [29], a method for outdoor guidance of an autonomous land vehicle (ALV) based on road following and color data clustering was proposed. Road-based guidance was also investigated for UAVs in [30], in which the authors proposed a road tracking scheme employing an RGB camera and a hyperspectral sensor. In [31], the authors proposed described a vision-based detection and control strategy for road following using a small autonomous aircraft. By employing a computer vision system to detect the road, the relative yaw and lateral displacements between the aircraft and the road were calculated and used to stabilize the aircraft. The success of the road-following scheme was verified for large distances as well as under a multitude of lighting and wind conditions.
Likewise in [32], a fixed-wing UAV was used for searching and mapping of structures such as rivers and coastlines. Onboard sensors, vision or near-infrared, were used to detect and track the structure. In order to generate the turn rate signals to track the curve of the structure, the control law was computed in a moving reference frame based on the dynamic shape alteration of the curve as observed by the onboard sensors. This online path planning strategy and predictive control scheme for the tracking of the curve is investigated in [33]. Near-infrared sensors were also used in [34] to detect coastlines or riverbanks using a low-flying UAV. The authors warped the image into the control coordinate space, in which the path planning problem is transformed into an image processing one. The followed path is controlled by determining and sending the desired roll angles to the autopilot of the UAV which leads to the UAV moving in a circular trajectory in order for the UAV to intercept the coastline on a tangent. In [35], a framework for UAVs employing computer vision strategies to track the boundary of various regions such as coastlines and roads is presented and verified through automated, in-field flights over a 1km coastline. A similar approach to [35] to detect and identify the coastline based on classical computer vision techniques was employed in [36] and an automatic flight controller was used to track the desired path.

1.3.3 Model Predictive Control

Model Predictive Control is an optimization-based control scheme, which guarantees state and input constraint satisfaction at every time step of a receding finite horizon for a dynamical system. MPC has been extensively employed in the field of control engineering, both in academic research as well as the industry, as it provides the possibility of formulating the problem in both the continuous and discrete-time domains while dealing with non-linearities and satisfying a set of hard and soft constraints in a multivariable framework. It also provides the ability to conduct online optimization which is crucial for real-time applications in which disturbances are present and the model of the nominal system used for the optimization scheme is an only approximation of the real system. Thus the MPC scheme encompasses robustness due to the state update at each time step. Furthermore, the overall design formulation of the scheme enables simple tuning of the model parameters which directly affect the performance characteristics of the system. However, Model Predictive Control is resource-intensive due to the constant requirement of state feedback and the high computational cost of the optimization at each time step. The literature on MPC is extensive with important mentions being [37], [38], [39], [40], [41], [42]. Nonlinear MPC is also extensively studied in [43], [44].

1.3.4 Event-Based Control

Hence there is a necessity to reduce the high computational cost of NMPC schemes in order to exploit its benefits in real-life robotic applications in which battery life, memory, and CPU capability are limited.

In recent decades, a new approach of MPC has presented research interest, as it entails computing the control law as infrequent as possible. More specifically, the decision for the computation of the new control law depends on a specific condition of the system of the state. Hence there is less frequent sampling as well as triggering of the optimization that produces the control law while crucial system properties such as feasibility, stability, and convergence are guaranteed. This leads to a sparse sampling and update of the control signal while crucial system properties such as feasibility, stability, and convergence are guaranteed, and subsequently to a significant reduction of system resource usage.

One of the most prevalent classes of this approach is the time-triggered control scheme which involves constant periodic sampling. The sampling rate is determined empirically and is usually as frequent as possible in order to ensure stability and performance characteristics in the worst-case scenario. Self and event-based comprise the other main classes of control policies in which an aperiodic sampling and control signal update takes place based on

the condition of the state of the system. In papers [45], [46], [47], first-order systems under the time and event-triggered policies are examined. The policy of the control scheme is to sample and compute the control law when the measurements become larger than certain limits. By comparing their closed-loop variance and sampling rate with results from equidistant sampling, it is proved that the event-based scheme results in improved performance. The concept of event-based feedback was also investigated under the name of interrupt-based feedback in the paper [48]. In paper [49], a heuristic event-based PID controller is investigated for a double-tank process application. The results showed similar control performance with the regular PID policy while significantly reducing CPU usage. Similarly, in paper [50] the author examines a triggering policy based on the norm of the measurement error, which is the difference between the current and the last updated state of the system, which ensures considerable performance and asymptotic stability of the system while reducing the number of the execution of the optimization. Specifically, whenever this norm surpasses a limit, which is a function of the current state, the control signal is updated by solving the optimization problem. In the [51], [52] papers, the authors are establishing a system theory for event-based control systems by investigating such a policy for perturbed linear systems in stabilization and tracking applications. Ultimate boundedness is used in order to examine the stability of the system in a practical fashion while the control signal is updated whenever the measurement error becomes significantly large.

Along with event-driven control, in paper [52] the authors introduce a self-trigger control framework, in which the sampling and control signal update instance are determined on the last available instance. A comparison between the reactive and proactive natures of the event and self-triggered policies respectively is presented alongside a comparison between state and output feedback on event-based schemes. Considerable research in event-triggered control for nonlinear state feedback systems was done in the paper [53], in which asymptotic stability is ensured as long as the continuous system is stabilizable. Additionally, input-to-state stability to measurement errors is employed to prove that task periods are bounded away from zero. The authors also proved thought simulations that the proposed event-based strategy produced much longer average task periods, due to the adoption of a flexible condition for the Lyapunov function, which requires only a sequence of the Lyapunov function to be monotone decreasing instead of it being monotone decreasing throughout the entire time as it was assumed in previous works such as [52], [54] and [55]. Event control frameworks for continuous-time systems are investigated in the papers [56], [57]. Event-based schemes have also been formulated in the discrete-time domain as investigated in papers [58], [59], [60]. In the paper [61], the authors have formulated an event-based control scheme for an intricate class of continuous-time hybrid dynamical systems, called integral continuous-time hybrid

automata (icHA), for which no assumptions for any prior information about the timing and order of the events are made. Considerable research in event-based control has been made in the context of network and multi-agent systems. More specifically, Miskowicz in [62] showed that event-based policies performed better than periodic ones in samples per time in wireless sensor networks, thus proving event-based triggering as an effective tool in reducing energy consumption. This concept was further studied by Mazo and Cao in [63] over wireless sensor and actuator networks, in which decentralized nonlinear event-control was applied without the need to synchronize the measurement updates or employ a central broadcasting node. Similarly, in [64], Mazo and Tabuada showcased a decentralized event-based application of centralized nonlinear controllers over wireless sensor and actuator networks. Research in event-based control for networked control systems was continued by Wang and Lemmon in [65] and [66] and expanded in cooperative control for multi-agent systems by Dimarogonas, Johansson, Frazzoli and Seyboth in [67], [68], [69] and [70].

1.3.5 Event-Based Model Predictive Control

Event-based policies have also found success in combination with model predictive control frameworks. As a continuation of their previous work in [61], the authors in [71] expanded their research into integral continuous-time hybrid automata (icHA) systems by creating a corresponding event-driven model for them that was used to formulate the optimal control problem. It was shown that the model was characterized by event and time-asymptotic convergence and proved to be effective in the verification of hybrid systems. In [72], Grunze and Muller followed an event-based approach for global optimal control of nonlinear systems in order to achieve a stabilizing feedback control law, in which the state measurements were available only when certain events of the state surpassing specific thresholds occurred. Event-based model predictive control schemes were also investigated for wireless network systems. In [73], an efficient trade-off between energy usage of sensors and performance decay was achieved while in [74] the event-based control policy dealt effectively with feedback delays and information loss. Similarly, such a distributed model approach was used in [75] to regulate the voltage magnitude of every distributed generator in an islanded microgrid network system to obtain a good trade-off between the control performance and communication and computation burdens. Marruedo, Alamo and Camacho in [76] presented a Robust Nonlinear Model Predictive Control scheme, in which the terminal region and state constraints are calculated to prove closed-loop robust feasibility given that the uncertainties are below a certain threshold. Furthermore, it is proved that the input-to-state stability, as well as the convergence of the closed-loop plant, is ensured despite the suboptimality of the optimal control problem solution of the NMPC framework. A comprehensive event-based

Robust Nonlinear Model Predictive Control framework for both continuous and discrete-time systems was presented in [77]. The presented approach examined the feasibility and input-to-state stability of uncertain systems with additive disturbances and provided sufficient triggering conditions for the event-based scheme. Research on both linear and nonlinear discrete-time systems is presented in [78]. A similar event-based NMPC approach to [77], [78] for discrete-time systems was followed in [79], in which the authors presented two triggering conditions depending on whether or not the state of the system was within the terminal region. Discrete-time LTI under stochastic disturbances was investigated under Robust event-triggered MPC was presented in [80]. A robust tube MPC approach was employed to ensure constraint satisfaction and asymptotic stability while the triggering conditions were designed in order to attain a specific average frequency between the controller and actuator given specific probability distributions for the additive disturbances. Robust tube event-based MPC was also implemented in [81] to guarantee recursive feasibility and input-to-state stability in unicycle robot applications under additive uncertainty. Event-based MPC has also found application in Unmanned Aerial Vehicles (UAV) and especially in multi-agent scenarios. In [82], the authors proposed a distributed ET-MPC policy was employed for formation control of UAV swarms, in which a collision avoidance strategy for a no-fly zone is incorporated into the cost function of the local OCP problem of each agent. A major drawback of Networked Control Systems (NCS) in UAV applications is the issue of information loss and time delays which deteriorate the individual performance of each agent. Even though delay compensation algorithms can mitigate these effects, they considerably increase the resource usage of the network and may lead to congestion. The authors of [83] proposed an event-based hierarchical MPC policy for network control of UAVs which omits the calculation of less important control signals and accounts for network delays in the event generator in order to utilize the network in a more efficient manner. An event-based scheme for networked control was also presented in [84] for VTOL-UAVs (Vertical Take-off and Landing-Unmanned Aerial Vehicles), in which each an event generator based on the error between the current and last broadcasted state of each agent dictated when to provide state feedback to the network. The scheme significantly reduced the network transmission rate while maintaining satisfying performance for the altitude and formation control of the UAVs.

1.4 Thesis Contribution

In this thesis, we propose an event-triggered policy for Image-Based Visual Servoing-Nonlinear Model Predictive Control for a quadrotor UAV in a coastline tracking mission under additive noise disturbance. Our approach includes:

1. The computation of the terminal region under the state and input constraints in order to achieve robust feasibility of the closed-loop system given a specific bound on the additive noise.

2. Sufficient conditions to ensure input-to-state (ISS) stability, under state and input constraints, in relation to the additive disturbance in order to ensure that the state progresses towards an ultimately bounded compact set.

3. A triggering condition for the events which is based on the error between the current and the nominal prediction of the state.

4. An efficient, fast ET-IBVS-NMPC code developed in C++ and integrated into the Robot Operating System (ROS) framework for coastline tracking for a UAV.

We wish to declare that to the best of our knowledge, this is the first time that an Event-Based NMPC scheme has been designed and implemented in conjunction with Image-Based Visual Servoing in a coastline tracking scenario using an UAV.

1.5 Thesis Outline

The rest of the thesis is structured as follows. Chapter 2 contains a description of Image-Based Visual Servo Control (IBVS) in which the design and stability of the control scheme are discussed, and the problem formulation for the IBVS for coastline tracking of Unmanned Aerial Vehicles (UAV) is presented. In Chapter 3, we present the feasibility and input-to-state (ISS) stability analysis of the Nonlinear Model Predictive Control (NMPC) scheme under the event-based policy as well as the event triggering condition. In Chapter 4, we demonstrate the performance of the proposed Event-Triggered-Image Based Visual Servoing-Model Predictive Control (ET-IBVS-MPC) scheme in a coastline tracking scenario. Chapter 5 accommodates the conclusion of the thesis and includes the challenges in the design and implementation of the control scheme. Furthermore, recommendations and proposals for further research are given. Lastly, Appendix A contains proofs for the calculation of the various Lipschitz constants that were mentioned throughout the feasibility and stability analysis of the ET-IBVS-MPC scheme.

Chapter 2

Image Based Visual Servo Control

2.1 Introduction to Visual Servo Control

Visual Servoing refers to a control approach that is based on computer vision, image processing and control theory, and utilizes computer vision data in the closed loop to control the movement of a robot. There are various configurations for the placement of the camera sensor relative to the position of the robot, with the main ones being the eye-in-hand and eye-to-hand configurations. In the eye-in-hand case the camera is mounted on the end-effector of a robotic manipulator and records the position of the target relatively to the robot, while in the eye-to-hand case the camera is statically fixed and observes the target as well as the robot. The visual servoing controller aims to minimize an positional error between a number of features and their desired position, which is defined by:

$$e(t) = s(m(t), a) - s^* \tag{2.1}$$

The vector $m(t)$ is a set of image measurements which are utilized to calculate a vector of a k number of features, with a being a set of intrinsic parameters than encompasses important information about the system. The vector s^* represents the desired positions of the features. The construction of the s vector defines the type of the visual servoing approach, with the image-based visual servo control IBVS and position-based visual servo control PBVS being the two main schemes. In the case of the IBVS, the feature vector is computed based on visual data that are can be immediately extracted from the camera sensor, while in the PBVS case, s comprises of a set of 3-D parameters that are calculated based on image measurements. In our case we are considering a stationary desired pose of the features in the frame of the camera sensor. Therefore the vector s solely depends on the movement of the camera. Hence, a velocity controller is an ideal approach to minimize the aforementioned error. To achieve

this, the expression between the camera velocity and the time derivative of the feature vector is defined by:

$$\dot{s} = L_s V_c \tag{2.2}$$

The vector $V_c = (T, \Omega) = (T_x, T_y, T_z, \Omega_x, \Omega_y, \Omega_z)$ represents the camera velocity, with T and Ω being the instantaneous linear and angular velocoties of the origin of the camera frame. Also, the matrix $L_s \in \mathbb{R}^{k\times 6}$ represents the interaction matrix or feature Jacobian and k the number of features. Combining expressions 2.1 and 2.2, we can obtain the relationship between the camera velocity and the feature error time derivative:

$$\dot{e} = L_e V_c \tag{2.3}$$

Where $L_e = L_s$. In order to ensure an exponential decrease of the error, as defined by:

$$\dot{e} = -\lambda e \tag{2.4}$$

we can express the velocity input of the robot as follows:

$$V_c = -\lambda L_e^+ e \tag{2.5}$$

The matrix L_e^+ is chosen to be the Moore-Penrose pseudo-inverse of the interaction matrix L_e, and is $L_e^+ = (L_e^T L_e)^{-1} L_e^T$ when the interaction matrix is of full rank 6. However, in real visual servoing applications, it is impossible to determine the exact values of L_e and L_e^+. Thus, an estimation $\widehat{L_e}$ of the interaction matrix and its pseudo-inverse should be used. Substituting the $\widehat{L_e^+}$ matrix notation in the control law expression 2.5, we can obtain the interaction matrix L_s for our system:

$$V_c = -\lambda \widehat{L_e^+} e \tag{2.6}$$

2.2 Quadrotor Dynamics

Quadrotor dynamics have been extensively studied in the existing literature. Thrust and torque generated by each propeller motor drives the movement of the quadrotor and thus the roll, pitch and yaw of the UAV as well as their rates. Let C_i and C_q be the inertial and fixed-body coordinate frames respectively. Without loss of generality, we choose the inertial frame as the North-East-Down (NED) coordinate frame which is commonly used in UAV applications. As presented by the authors in [85], Using the Newton-Euler equations of motion for a six degrees-of-freedom (DOF) for a rigid body, we can derive the following model that describes the dynamics of the quadrotor.

$$ {}^{i}\dot{p}_q = {}^{i}T_q \tag{2.7} $$

$$ m_q {}^{i}\dot{T}_q = {}^{i}R_q F_q \tag{2.8} $$

$$ J_q \dot{\Omega}_q = \tau_q \tag{2.9} $$

where ${}^{i}p_q = [{}^{i}x_q, {}^{i}y_q, {}^{i}z_q]$ is the position and ${}^{i}T_q = [{}^{i}T_{xq}, {}^{i}T_{yq}, {}^{i}T_{zq}]$ the velocity vectors of the quadrotor with relation to the inertial coordinate frame. The mass as well as the inertia matrix of the quadrotor are denoted as m_q and J_q respectively, while its angular velocity w.r.t the fixed-body frame is denoted as $\Omega_q = [\Omega_{xq}, \Omega_{yq}, \Omega_{zq}]$. The matrix ${}^{i}R_q$ is used for the transformation from the fixed-body frame to the inertial frame. The forces and torques acting on the quadrotor are defined by the following expressions:

$$ F_q = F_g + F_p + F_d \tag{2.10} $$

$$ \tau_q = \tau_p + \tau_d \tag{2.11} $$

where F_g is the gravitational force vector, F_p is thrust vector from the quadrotor's propellers and F_d is the drag force vector. As far as the torques are concerned, we denote as τ_p the torque vector from the propellers and τ_d as the drag moments vector. In our case, an autopilot is used for the low-level control of the quadrotor. Specifically, PID controllers following an inner and outer loop control architecture are used to handle the low-level control. The inner loop receives throttle, roll, pitch and yaw reference values to achieve attitude control while the outer loop receives velocity reference values to realize translation control. This architecture enables an IBVS approach, as the outer loop is able to receive velocity inputs w.r.t to the fixed-body coordinate frame. These velocities are generated by the IBVS algorithm after they have been transformed from the image frame, denoted here as C_c, to the fixed-body coordinate frame of the quadrotor using the rotation and translation matrices ${}^{q}R_c$ and ${}^{q}t_c$. A schematic of the image and body-fixed coordinate frames are presented in Figure 2.1.

The output of the low-level controller are the thrust and torque values that are used as voltage input to the motors of the propellers. Furthermore, this autopilot low-level control architecture is able to handle the under-actuation problem of the quadrotor. The generated camera velocities from the IBVS scheme consist of six control inputs, three translational and three angular. However, this is not a suitable approach for the control of the under-actuated dynamics of the UAV. Therefore, the autopilot system is able to handle this under-actuation

issue by accepting velocity reference values solely for the three translational velocities and the yaw rate.

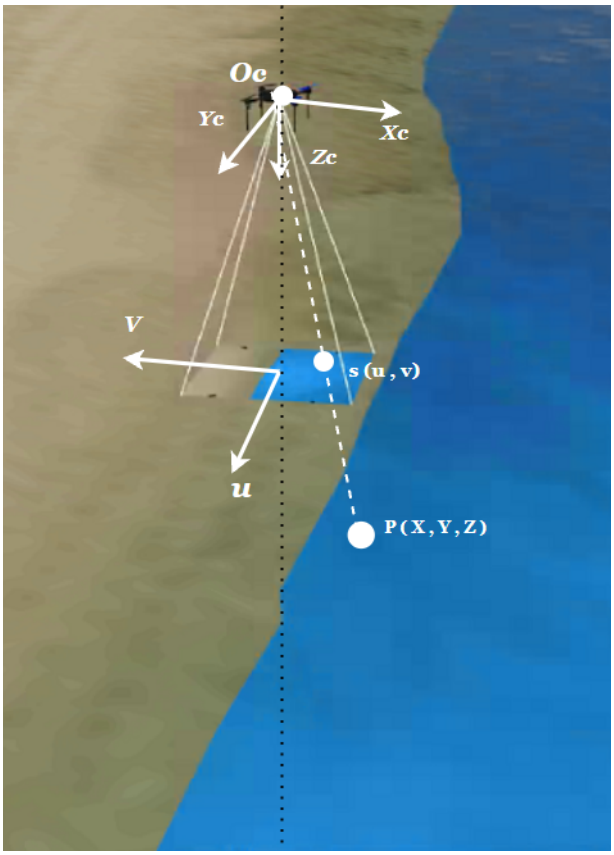

Figure 2.1 The image coordinate frame w.r.t the body-fixed quadrotor coordinate frame

2.3 Basic Components of IBVS

Here we consider the IBVS scheme where the image measurements m consist of the pixel coordinates of the points and the a parameter set includes the camera characteristics that transform the pixel coordinates to image-plane features. Therefore, the feature vector s is defined as the set of the coordinates in the image plane. More specifically, a 3-D point X with $X = (X, Y, Z)$ camera frame coordinates is projected in the image frame as a 2-D point with coordinates $s = (x, \gamma)$ which are defined by:

$$\begin{aligned} x &= \frac{X}{Z} = \frac{u - c_u}{f} \\ \gamma &= \frac{Y}{Z} = \frac{v - c_v}{f} \end{aligned} \tag{2.12}$$

A schematic representation of the camera coordinate frame in a typical IBVS scheme is presented in Figure 2.2. The image measurement set is defined as $m = (u, v)$ and provides the pixel coordinates of each point. Also, $a = (c_u, c_v, f, a)$ includes the camera intrinsic parameters , with c_u and c_v being the principal point coordinates and f the focal length of the camera. Therefore, the feature set $s = s(x, \gamma)$ includes the points in the image-plane. Taking the time derivative of the image plane coordinates of 2.12, we get:

$$\begin{aligned} \dot{x} &= \frac{\dot{X}}{Z} - X\frac{\dot{Z}}{Z^2} = \frac{\dot{X} - x\dot{Z}}{Z} \\ \dot{\gamma} &= \frac{\dot{Y}}{Z} - Y\frac{\dot{Z}}{Z^2} = \frac{\dot{Y} - \gamma\dot{Z}}{Z} \end{aligned} \tag{2.13}$$

We can connect the velocity of the 3-D points in the camera frame to the velocity of the camera through the $\dot{\bar{X}} = -T - \Omega \times \bar{X}$ expression:

$$\begin{aligned} \dot{X} &= -T_x + \Omega_z Y \\ \dot{Y} &= -T_\gamma - \Omega_z X \\ \dot{Z} &= -T_z \end{aligned} \tag{2.14}$$

Combining equations 2.13 and 2.14, we can express the time derivatives of the image plane coordinates as:

$$\begin{aligned}\dot{x} &= -\frac{T_x}{Z} + \frac{xT_z}{Z} + x\gamma\Omega_x - (1+x^2)\Omega_y + \gamma\Omega_z \\ \dot{\gamma} &= -\frac{T_\gamma}{Z} + \frac{\gamma T_z}{Z} + (1+\gamma^2)\Omega_x + x\gamma\Omega_y - x\Omega_z\end{aligned} \tag{2.15}$$

Therefore, the interaction matrix L_s is defined by:

$$L_s = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & x\gamma & -(1+x^2) & \gamma \\ 0 & \frac{-1}{Z} & \frac{\gamma}{Z} & (1+\gamma^2) & x\gamma & -x \end{bmatrix} \tag{2.16}$$

where Z is the depth of the point relative to the camera frame. In case where the exact value of the depth Z is unknown, an approximation must be estimated in order to compute the interaction matrix. Subsequently, an approximation of the interaction matrix $\widehat{L_x}$ should also be used in this case.
Taking into account the under-actuation nature of quadrotor, the interaction matrix L_s can be revised as:

$$L_s = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & \gamma \\ 0 & \frac{-1}{Z} & \frac{\gamma}{Z} & -x \end{bmatrix} \tag{2.17}$$

In general, in order to be able to control a 6 DOF robot, at least 3 points are necessary. In our case, four points are used to define the region of interest ROI that encompasses the coastline. Thus, by stacking 4 interaction $L_{s,i}$ matrices, one for each feature $s_i = (x_i, \gamma_i)$ with depth Z_i for $i = (1,2,3,4)$, we obtain:

$$L_s = \begin{bmatrix} L_{s1} \\ L_{s2} \\ L_{s3} \\ L_{s4} \end{bmatrix} \tag{2.18}$$

In our case the depth Z is equal to the altitude of the UAV, which is known since it can be measured in each iteration of the control scheme. Hence, we can assume that $L_e = L_x$ and subsequently $\widehat{L_e^+} = L_e^+$. In expression $V_c = (T, \Omega) = {}^cV_c = ({}^cT_c, {}^c\Omega_c)$ we have used the

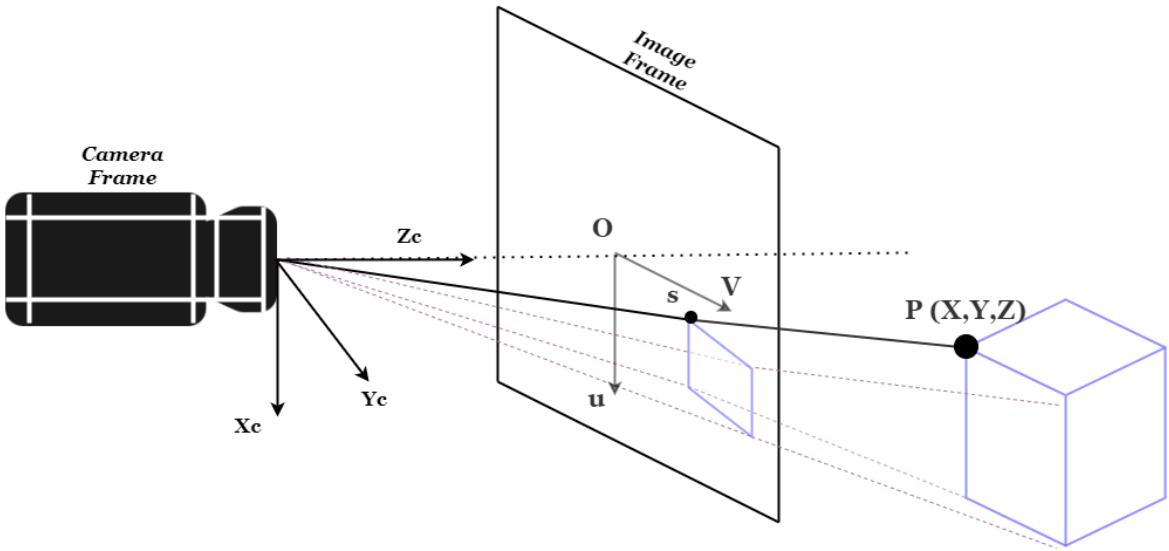

Figure 2.2 The coordinate frame of a camera system in an IBVS scheme

IBVS scheme to define the velocity screw of the camera with respect to the camera frame. However, the UAV frame does not coincide with the camera frame. Thus, the velocity screw of the camera needs to be expressed with respect to the UAV frame as ${}^{q}V_c = ({}^{q}T_c, {}^{q}\Omega_c)$. The translation ${}^{q}t_c$ and rotation ${}^{q}R_c$ matrices between the two aforementioned frames are used for the transform.

2.4 Stability

In order to examine the stability of the closed loop visual servo system, we utilize Lyapunov analysis. More specifically, we can use the candidate Lyapunov function defined by the squared norm of the feature error:

$$L = \frac{1}{2}\|e(t)^2\| \tag{2.19}$$

Thus, the time derivative of the Lyapunov function can be obtained:

$$\dot{L} = e^T \dot{e} = -\lambda e^T L_e \widehat{L_e^+} e \tag{2.20}$$

Thus, we can guarantee global asymptotic stability if the following condition is guaranteed:

$$L_e \widehat{L_e^+} > 0 \tag{2.21}$$

Since the number of features is equal to the number of the camera degrees of freedom and the matrices L_e and L_e^+ are of full rank 4, then the Lyapunov global asymptotic stability condition is ensured since the depth Z can be measured and therefore no approximation for the pseudo-inverse of the feature Jacobian is needed. Note that for IBVS only local asymptotic stability around a small neighbourhood of the desired position is ensured, even with perfect knowledge of the interaction matrix. Moreover, the four selected features, which are the corners of the the Region of Interest (ROI) that encompasses the detected coastline, ensure the non-singularity of the Interaction matrix as these four points can never be collinear.

2.5 IBVS for NMPC of Quadrotors - Problem Formulation

In equation **??** we have expressed the time derivative of the feature vector in the continuous time frame. However, in our case we have to express the system in the discrete-time frame. By applying the Newton-Euler method, the discrete-time expression of the system at time step $k+1$ with $k = \{1, \ldots, N\}$, for a sampling time dt, is:

$$\begin{aligned} \frac{s_{k+1} - s_k}{dt} &= L_k V_k \\ s_{k+1} &= s_k + L_k V_k dt \\ s_{k+1} &= f(s_k, V_k) \end{aligned} \tag{2.22}$$

where s_k is the state of the nominal system at time step k:

$$s_k = \begin{bmatrix} s_k^1 \\ s_k^2 \\ s_k^3 \\ s_k^4 \end{bmatrix} = \begin{bmatrix} x_k^1 \\ \gamma_k^1 \\ x_k^2 \\ \gamma_k^2 \\ x_k^3 \\ \gamma_k^3 \\ x_k^4 \\ \gamma_k^4 \end{bmatrix} \tag{2.23}$$

Also, we use the notations L_k and V_k for the interaction matrix and camera velocity at sampling time k, respectively. More specifically, at time step k the camera velocity vector is:

$$V_k = \begin{bmatrix} T_{x,k} \\ T_{y,k} \\ T_{z,k} \\ \Omega_{z,k} \end{bmatrix} \tag{2.24}$$

The discrete-time system 2.22 can be expressed in a more general form as:

$$s_{k+1} = f(s_k, V_k) + \xi_k \tag{2.25}$$

However, in real systems, uncertainties and disturbances need to be included in the model that describes the system in order to ensure accuracy. In our case, we are assuming that the

IBVS system is affected by an external disturbance. Therefore, an ξ vector representing these disturbances can be expressed at each time step as:

$$\xi_k = \begin{bmatrix} \xi_k^1 \\ \xi_k^2 \\ \xi_k^3 \\ \xi_k^4 \end{bmatrix} = \begin{bmatrix} \xi_{x,k}^1 \\ \xi_{\gamma,k}^1 \\ \xi_{x,k}^2 \\ \xi_{\gamma,k}^2 \\ \xi_{x,k}^3 \\ \xi_{\gamma,k}^3 \\ \xi_{x,k}^4 \\ \xi_{\gamma,k}^4 \end{bmatrix} \tag{2.26}$$

These disturbances are considered to belong to the set Ξ_{set} where:

$$\Xi = \{\xi_k \in \Xi \subseteq \mathbb{R}^8, \quad \|\xi_k\|_s \leq \bar{\xi} = \xi_{max} \quad \forall \xi_k \in \Xi_{set}\} \tag{2.27}$$

where $\|\xi_k\|_s$ is the s-norm of the disturbances.
Similarly, the control inputs of the camera are bounded with V_k forming a velocity constraint set V_{set}:

$$V_b = \{V_k \in V \subseteq \mathbb{R}^4, \quad \|V_k\| \leq \bar{V_k} = V_{max} \quad \forall \quad V_k \in V_{set}\} \tag{2.28}$$

$$\|V_k\| \leq \bar{V_k} = V_{max} \iff \begin{cases} \|T_{x,k}\| \leq \bar{T_x} = T_{x,max} \quad \forall \quad T_{x,k} \in V_{set} \\ \|T_{y,k}\| \leq \bar{T_y} = T_{y,max} \quad \forall \quad T_{y,k} \in V_{set} \\ \|T_{z,k}\| \leq \bar{T_z} = T_{z,max} \quad \forall \quad T_{z,k} \in V_{set} \\ \|\Omega_{z,k}\| \leq \bar{\Omega_z} = \Omega_{z,max} \quad \forall \quad \Omega_{z,k} \in V_{set} \end{cases}$$

Lastly, the system is also subject state constraints due to the limits of the camera field of view. Hence, the state constraint set S_{set} is defined as:

$$S = \{s_k \in S \subseteq \mathbb{R}^8, \quad \underline{s} \leq s_k \leq \bar{s} \quad \forall \quad s_k \in S_{set}\} \tag{2.29}$$

$$s_{min} = \underline{s} \leq s_k \leq \bar{s} = s_{max} \iff \begin{cases} x_{min} = \underline{x} \leq x_k \leq \bar{x} = x_{max} \quad \forall \quad x_k \in S_{set} \\ \gamma_{min} = \underline{\gamma} \leq \gamma_k \leq \bar{\gamma} = \gamma_{max} \quad \forall \quad \gamma_k \in S_{set} \end{cases}$$

where the upper and lower bounds for the features in the camera frame are defined by 2.12 for the maximum and minimum pixel values of u_{max}, u_{min} and v_{max}, v_{min} that define the limits of the camera field of view. The classic IBVS control scheme is presented schematically in Figure 2.3.

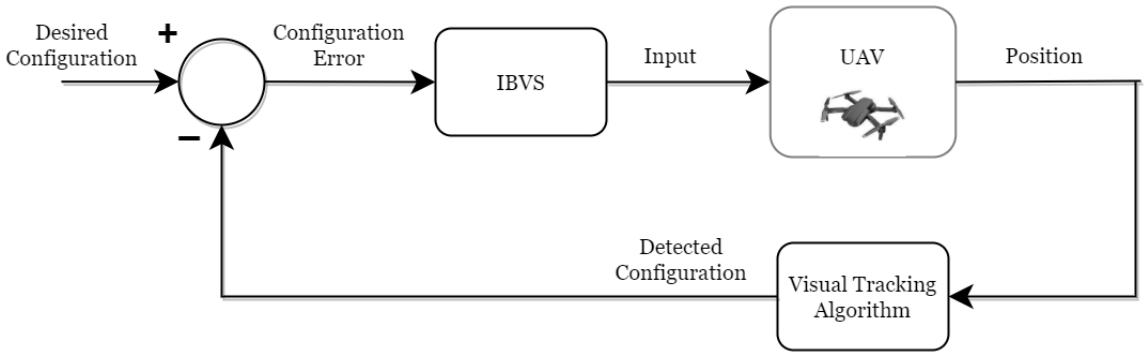

Figure 2.3 Image-Based Visual Servoing Scheme

We have now formulated the model of our system 2.25 and defined the input 2.28 and state constraints 2.29 that it subjects to. The nominal model along with the aforementioned constraints will be used to formulate the NMPC problem in Chapter 3.

Chapter 3

Event-Triggered Nonlinear Model Predictive Control

3.1 ET-NMPC-IBVS Problem Formulation

The aim of the discrete-time NMPC controller is to drive the system 2.25 to a desired state $s_{des} \in S_{set}$ by solving an Optimal Control Problem over a receding finite horizon of size N. The OCP is solved on-line and is based on the measurement of the state at timestep k. The solution provides an N-sized control sequence $V_F(k) = [V(k|k), V(k+1|k), \ldots, V(k+N-1|k)]$ that minimizes a specific cost function J_N, while satisfying the set state and control constraints defined in 2.28 and 2.29. The desired state vector for each feature is defined as:

$$s_{des} = \begin{bmatrix} s_{des,1} \\ s_{des,2} \\ s_{des,3} \\ s_{des,4} \end{bmatrix} \tag{3.1}$$

where

$$s_{des,i} = \begin{bmatrix} x_{des,i} \\ \gamma_{des,i} \end{bmatrix} = \begin{bmatrix} \frac{u_{des,i} - c_u}{f} \\ \frac{v_{des,i} - c_v}{f} \end{bmatrix} \tag{3.2}$$

In order to track the coastline, the velocity of the desired v image feature coordinates of the ROI are described by a linear increase of rate α as described in 3.3, which is a constant representing the velocity of the desired features on the v direction of the image coordinate frame. This approach ensures that the camera will traverse along the coastline with a velocity

that depends on the value of the α constant. Hence, the following expression for the desired state of the features can be derived:

$$\dot{v}_{des,i} = \alpha \tag{3.3}$$

$$\frac{v_{des,k+1} - v_{des,k}}{dt} = \alpha$$

$$v_{des,k+1} = v_{des,k} + \alpha dt \tag{3.4}$$

On the contrary, the desired u_{des} image feature coordinates are considered to be constant. The nominal system 2.22 is used for the formulation of the IBVS-NMPC problem. The notation $\hat{s}(k+i|k) = f(\hat{s}(k+i-1|k), V(k+i-1|k)$ is used to express the predicted state of the system at time step $k+i$ based on the measurement of the real state $s_k = \hat{s}(k|k)$ at time step k and by applying the part of the control sequence from time step k until $k+i-1$. On the other hand, the notation s_k is used to denote the real state of the system at timestep k that is affected by additive disturbance ξ, namely the VTA noise. The IBVS-NMPC scheme for the quadrotor UAV is presented in Figure 3.1

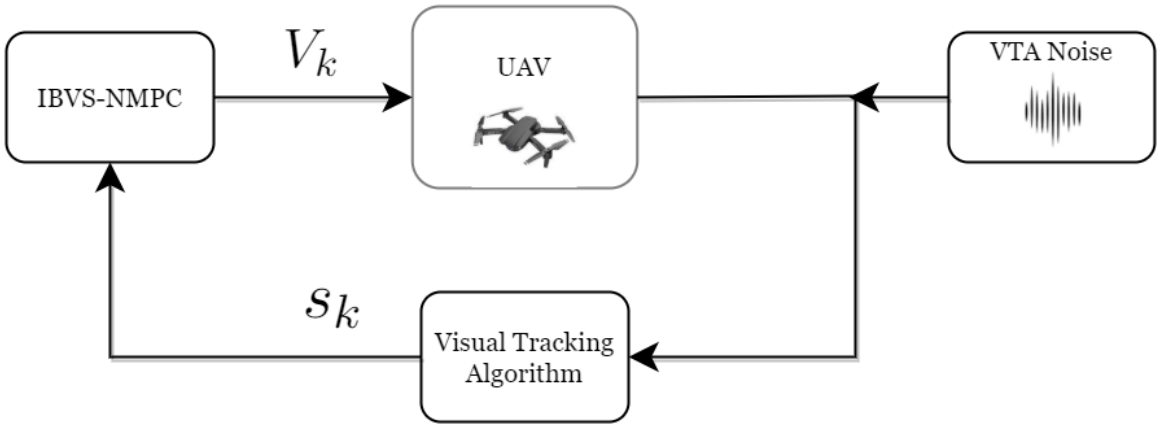

Figure 3.1 IBVS-NMPC scheme using a quadrotor UAV

The disturbance term uncertainty term of the real system can cause discrepancies between the predicted state, given from the nominal model 2.22 subject to a specific sequence of inputs, and the actual state given from 2.25 for the same sequence of inputs. In order to account for this mismatch the error e is introduced in the analysis. The error e is defined as the norm of the difference between the predicted and the real evolution of the state. In the sequel, the double subscript notation will be reserved for the error, too. The NMPC apporach is presented in Figure 3.2

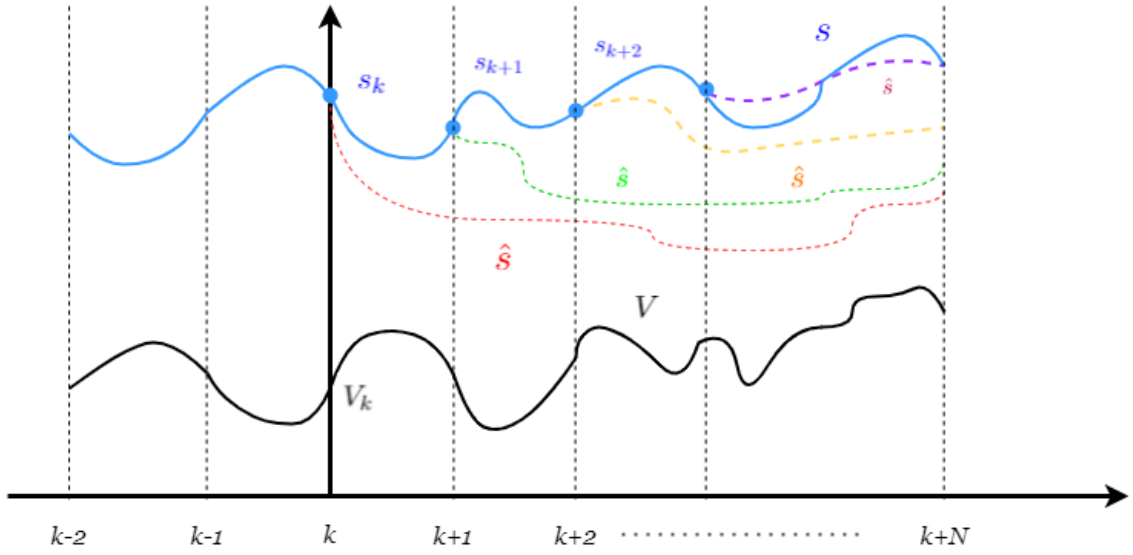

Figure 3.2 Nonlinear Model Predictive Control: The colored dotted trajectories represent the predicted trajectories $\hat{s}_k$ based on the state at each time step. The blue trajectory depicts the real state of the system. The black trajectory represents the control input.

The open-loop Optimal Control Problem (OCP) of the discrete-time IBVS-NMPC is given by:

$$\begin{aligned}
&\min_{V_F(k)} J_N(s_k, V_F(k)) = \min_{V_F(k)} \sum_{i=0}^{N-1} F(\hat{s}(k+i|k), V(k+i|k)) + E(\hat{s}(k+N|k)) \\
&\text{subject to:} \\
&\hat{s}(k+i|k) \in S_{set} \quad \forall \ i = \{1, \ldots, N-1\} \\
&V(k+i|k) \in V_{set} \quad \forall \ i = \{1, \ldots, N-1\} \\
&\hat{s}(k+N|k) \in E_f
\end{aligned} \tag{3.5}$$

$$\min_{V_F(k)} J_N(s_k, V_F(k)) = \min_{V_F(k)} \sum_{i=0}^{N-1} F(\hat{s}(k+i|k), V(k+i|k)) + E(\hat{s}(k+N|k)) = \tag{3.6}$$

$$\min_{V_F(k)} \sum_{i=0}^{N-1} \hat{s}(k+i|k)^T Q \, \hat{s}(k+i|k) + V(k+i|k))^T R \, V(k+i|k)) + \hat{s}(k+N|k)^T P \, \hat{s}(k+N|k)$$

where F and E are the stage and terminal costs respectively defined as the following quadratic functions:

$$F(\hat{s},V) = \hat{s}^T Q\,\hat{s} + V^T R\,V \tag{3.7}$$

$$E(\hat{s}) = \hat{s}^T P\,\hat{s} \tag{3.8}$$

where Q, R and P are the following diagonal matrices:

$$Q = \ diag(q_1,\ldots,q_8) \tag{3.9}$$

$$R = \ diag(r_1,\ldots,r_4) \tag{3.10}$$

$$P = \ diag(p_1,\ldots,p_8) \tag{3.11}$$

The discrete-time NMPC algorithm is presented in Algorithm 1.

Algorithm 1 NMPC

1: Time step equals to k
2: **while** $x_k \in S_{set}$ **do**
3: Update the state measurement of system s_k
4: Solve the OCP and generate the open-loop input sequence $V_k^* = [V_k,\ldots,V_{k+N}]$
5: Apply the first control input V_k to the system
6: Update the time-step: $k \leftarrow k+1$
7: **end while**
8: **go to** 3

The additive disturbance from the noise of the VTA algorithm that affects the system causes a difference between the predicted and the real state of the system. We denote this disparity as e, which expresses the error between the predicted and real state of the system at time step $k+i$ given the input sequence from $k-1$ until $k+j-1$. The error $e(k+i|k-1)$ is defined by:

$$e(k+i|k-1) = \|s_{k+i} - \hat{s}(k+i|k-1)\| \tag{3.12}$$

It is trivial to prove that, at time step $k-1$ when the NMPC is triggered, $e(k-1|k-1) = 0$ because $s_{k-1} = \hat{s}(k-1|k-1)$. As discussed previously, the main idea behind the event-based policy is the triggering of the optimisation algorithm whenever the norm of the error exceeds a certain threshold. While the error norm remains below this bound, the control input sequence calculated based on the real state of the system at time step k is applied to the quadrotor in an open-loop fashion. The ET-NMPC strategy is presented in Figure 3.3

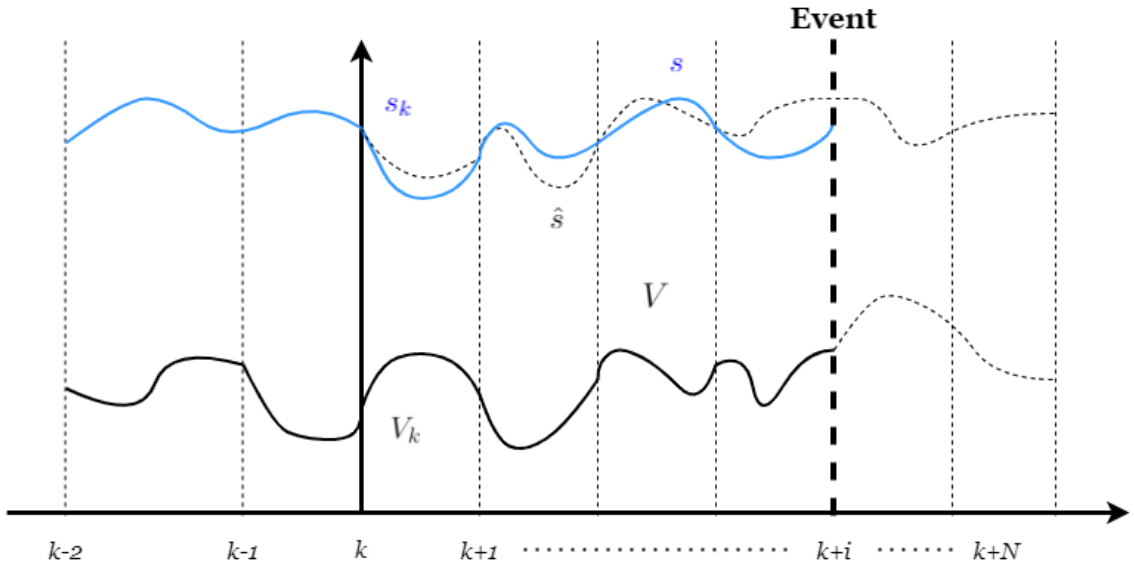

Figure 3.3 Event-Triggered NMPC: The dotted trajectory depicts the predicted trajectory $\hat{s}_k$ based on the state at each time step k. The blue trajectory depicts the real state of the system. The black trajectory represents the control input. Only the control input sequence dictated by the event occurring at time step $k+i$ is applied.

In order to prove the feasibility and input-to-state stability of the scheme, we consider the following Assumptions and Lemmas.

Assumption 1 The nominal model is such that the origin is a steady state and $f(s,V)$ is locally Lipschitz in s in the domain $S_{set} \times V_{set}$, where there is a $0 < L_f < \infty$ such that for all $s_1, s_2 \in S_{set}$ for all $V \in V_{set}$,

$$\|f(x_1,V) - f(x_2,V)\|_s = L_f \, \|s_1 - s_2\|_s \tag{3.13}$$

where the L_f constant denotes the Lipschitz constant in some s norm, i.e. $s = 1, 2, \ldots \infty$.

In order to consider the mismatch effect between the evolution of the predicted and the real state, we have to set a bound for the disturbance ξ that causes the error.

Lemma 1 Consider the nominal system 2.25, which for Assumption 1 is satisfied and the disturbances affecting the system belong to the Ξ_{set} from 2.27 and are bounded by $\xi \leq \bar{\xi}$. Then the error between the predicted and real state of the system at time step $k+i$, given an input sequence, is bounded by:

$$e(k+i|k-1) = \|s_{k+i} - \hat{s}(k+i|k-1)\| \leq \frac{L_f^i - 1}{L_f - 1}\bar{\xi} \tag{3.14}$$

where L_f is the Lipschitz constant of the system in the s norm.

Proof

We can employ the triangle inequality and the Lipschitz constant L_f of the system 2.22 to derive that:

$$\begin{aligned}
\|s_k - \hat{s}(k|k-1)\| &= \xi_k \leq \bar{\xi} \\
\|s_{k+1} - \hat{s}(k+1|k-1)\| &\leq L_f \, \|s_{k+1} - \hat{s}(k+1|k-1)\| + \xi_{k+1} \\
&= (L_f + 1)\, \bar{\xi} \\
&\vdots \\
\|s_{k+i} - \hat{s}(k+i|k-1)\| &\leq L_f \, \|s_{k+i-1} - \hat{s}(k+i-1|k-1)\| + \xi_{k+i-1} \\
&\leq \sum_{j=0}^{i-1} L_f^j \, \bar{\xi} \\
&= \frac{L_f^i - 1}{L_f - 1}\bar{\xi}
\end{aligned}$$

Lemma 2 Based on Assumption 1, the nominal system 2.22 subject to the input 2.28 and FOV constraints 2.29 is Lipschitz continuous in S_{set} with a Lipschitz constant $0 < L_f < \infty$ that is defined by:

$$L_f = [2max\{4(1 + \frac{\overline{T_z}}{Z}dt)^2, 4(\overline{\Omega_z}dt)^2\}]^{\frac{1}{2}} \tag{3.15}$$

Proof Appendix A.

The following assumptions are stated:

Lemma 3 The stage cost $F(s,V)$ is Lipschitz continuous in $S_{set} \times V_{set}$ with a Lipschitz constant in the s-norm L_F for all $s_1, s_2 \in S_{set}$ and a Lipschitz constant in the s-norm L_{F_V} for all $V_1, V_2 \in V_{set}$. The Lipschitz constant L_F is defined by:

$$\begin{aligned}
&\|F(s_1,V) - F(s_2,V)\| = \|s_1^T Q s_1 - s_2^T Q s_2\| = \\
&\|s_1^T Q s_1 - s_1^T Q s_2 + s_1^T Q s_2 - s_2^T Q s_2\| = \\
&\|s_1^T Q(s_1 - s_2) + (s_1^T - s_2^T) Q s_2\| = \\
&\|s_1^T Q(s_1 - s_2) + s_2^T Q(s_1 - s_2)\| = \\
&\|(s_1^T + s_2^T) Q(s_1 - s_2)\| \iff \\
&\|F(s_1,V) - F(s_2,V)\| \leq (\|s_1\| + \|s_2\|)\sigma_{max}(Q)\|s_1 - s_2\|
\end{aligned}$$

It is true that $\forall s \in S_{set} : \|s\| = (\|x_1\|^2 + \|\gamma_1\|^2 + \cdots + \|x_4\|^2 + \|\gamma_4\|^2)^{\frac{1}{2}}$. Thus, we can derive that:

$$\begin{aligned}
\|s\| &= (\sum_{i=1}^{i=n} \|x_i\|^2 + \|\gamma_i\|^2)^{\frac{1}{2}} \\
\|s\| &\leq (n(\|\bar{x}_i\|^2 + \|\bar{\gamma}_i\|^2))^{\frac{1}{2}}
\end{aligned}$$

Therefore the Lipschitz constant is equal to $L_F = 2(n(\|\bar{x}_i\|^2 + \|\bar{\gamma}_i\|^2))^{\frac{1}{2}}\sigma_{max}(Q)$

Lemma 4 The stage cost $F(s,V)$ is such that $F(0,0) = 0$ and there are some positive constants $a > 0$ and $\sigma \geq 1$ such that $\forall s \in S_{set}$ and $\forall\, V \in V_{set}$ it is true that $F(s,V) \geq \underline{F}(s,V) = a\|(s,V)\|^{\sigma}$, where $\underline{F}(s,V)$ is a K_∞ - function. More specifically, it is true that:

$$F(s,V) \geq min(q_1, \ldots, q_8, r_1, \ldots, r_4)\|s\|^2 \tag{3.16}$$

Proof

$$\begin{aligned} F(s,V) &= s^T Qs + V^T RV \\ &= s^T diag(q_1,\ldots,q_8)s + V^T diag(r_1,\ldots,r_4)V \\ &\geq [s,V]^T diag(q_1,\ldots,q_8,r_q,\ldots,r_4)[s,V] \\ &\geq min\{q_1,\ldots,q_8,r_q,\ldots,r_4\}\|[s,V]\|^2 \iff \\ F(s,V) &\geq min\{q_1,\ldots,q_8,r_q,\ldots,r_4\}\|s\|^2 \end{aligned}$$

The terminal region E_f in 3.5 is calculated as a subset of an admissible positively invariant set E_n for the system described in 2.22. Therefore, we can make the following assumptions for the terminal region:

Assumption 2 For the system described by 2.22, there is an admissible positively invariant set $E_n \subset S_{set}$ such that $E_f \subset E_n$, where $E_n = \{s \in S_{set} : \|s\| \leq \varepsilon_0\}$ with ε_0 being a positive parameter.

Assumption 3 There is a local controller $V_k = h(s_k) \in V_{set}$ for the set E_n, which stabilizes the system, and an associated Lyapunov function E such that:

$$E(f(s_k,h(s_k))) - E(s_k) \leq -F(s_k,h(s_k)) \tag{3.17}$$

$\forall\, s \in E_n$

Assumption 4 The terminal region associated Lyapunov function $E(s)$ is Lipschitz continuous in E_n with a Lipschitz constant L_E in the s-norm that is given by:

$$\|E(s_1) - E(s_2)\|_s \leq L_E \|s_1 - s_2\|_s \tag{3.18}$$

where $L_E = 2\varepsilon_0 \sigma_{max}(P) \quad \forall\, s \in E_n$

Proof

$$
\begin{aligned}
&\|E(s_1) - E(s_2)\| = \|s_1^T P s_1 - s_2^T P s_2\| = \\
&\|s_1^T P s_1 - s_1^T P s_2 + s_1^T P s_2 - s_2^T P s_2\| = \\
&\|s_1^T P(s_1 - s_2) + (s_1^T - s_2^T) P s_2\| = \\
&\|s_1^T P(s_1 - s_2) + s_2^T P(s_1 - s_2)\| = \\
&\|(s_1^T + s_2^T) P(s_1 - s_2)\| \iff \\
&\|E(s_1) - E(s_2)\| \leq (\|s_1\| + \|s_2\|)\sigma_{max}(P)\|s_1 - s_2\|
\end{aligned}
$$

It is true that from Assumption 2: $\forall s \in E_n : \|s\| = (\|x_1\|^2 + \|\gamma_1\|^2 + \cdots + \|x_4\|^2 + \|\gamma_4\|^2)^{\frac{1}{2}} \leq \varepsilon_0$. Therefore the Lipschitz constant is equal to $L_E = 2\varepsilon_0\sigma_{max}(P)$

For the local stabilizing controller h_k we can make the following assumptions:

Assumption 5 There is a local stabilizing controller $h(s) \in V_{set}$ for which there exists a $L_h > 0$, $\forall s \in E_n$ such that $\|h(s)\| \leq L_h\|s\|$. Thus, it is also true that for $L_{f_h} > 0$, $\forall s \in E_n$ we have $\|f(s,h(s))\| \leq L_{f_h}\|s\|$.

Assumption 6 For the set E_n, the associated Lyapunov function is bounded as described by: $E(s) = s^T P s \leq \alpha_\varepsilon$ with $\alpha_\varepsilon = max\{p_1, \ldots, p_8\}\varepsilon_0^2 > 0$. Also, assuming that $E_n = \{s \in S_{set(N-1)} : h(s) \in V_{set}\}$ and for taking a positive parameter $\alpha_{\varepsilon f}$ for which $\alpha_{\varepsilon f} \in (0, \alpha_\varepsilon)$, we can assume that the termianl set $E_f = \{s \in \mathbb{R}^8 : E(s) \leq \alpha_\varepsilon\}$ is such that $\forall s \in E_n$, $f(s,h(s)) \in E_f$.

3.2 Feasibility

Firstly, we are interested in proving that feasibility initially at time step $k-1$ when the first event is triggered, implies feasibility afterwards at any time step m when the next event is triggered. To achieve the aforementioned goal, we state the following definitions about feasibility:

Definition 1: Feasible Set The feasible set S_F is defined as the set of initial states s for which the NMPC problem with horizon N is feasible, i.e.

$$S_F = \{s \mid \exists\, [V_0, \ldots, V_{N-1}] \text{ such that the set state and input constraints are satisfied.}\} \tag{3.19}$$

Definition 2: Recursive Feasibility The NMPC problem is called recursively feasible, if for all feasible initial states feasibility is guaranteed at every state along the closed-loop trajectory.

Definition 3: Invariant Set A set S_F is positively invariant for the system $s_{k+1} = f(s_k)$, if:

$$s_k \in S_F \quad \Rightarrow \quad f(s_k) \in S_F\,, \quad \forall k \in N. \tag{3.20}$$

For our EB-NMPC case, we can assume that at time step $k-1$ an event is triggered and the OCP of the NMPC is solved, thus producing an optimal control sequence of the form $V_F^*(k-1) \triangleq [V_F^*(k-1|k-1), \ldots, V_F^*(k+N-2|k-1)]$, for which the $*$ notation is used. Using Definition 1, let S_F be the set containing all the state vectors for which a feasible control sequence of the form exists that satisfies the constraints of the OCP 3.5. Then, any control sequence $\overline{V_F}(k+m)$ at a subsequent time step $k+m$ with $m = 0, \ldots, N-1$ is described by:

$$\overline{V_F}(k+m) = \overline{V}(k+i|k+m) = \begin{cases} V^*(k+i|k-1) & \text{for } i = m, \ldots, N-2 \\ h(\hat{x}(k+i|k+m)) & \text{for } i = N-1, \ldots, N+m-1 \end{cases} \tag{3.21}$$

where $V^*(k+i|k-1)$ for $i=m,\ldots,N-2$ is part of the optimal control sequence computer at time step $k-1$.

We can derive from the feasibility of the optimal control sequence $V_F^*(k-1)$ computer at the initial triggering instance $k-1$ and Assumption 5, that the feasible control sequence computed at all subsequent time steps $m=\{0,\ldots,N-1\}$ also belongs to the V_{set}, i.e. $\overline{V}(k+i|k+m)\in V_{set}$.

In order to prove recursive feasibility, it is necessary to prove that $\hat{s}(k+N-1|k+m)\in E_f$ for all time steps $m=\{0,\ldots,N-1\}$ when an event can be triggered. Using the error expression from 3.12 and its bound 3.15 from Lemma 1, we can derive:

$$\begin{aligned}
\|\hat{s}(k+N-1|k)-\hat{s}(k+N-1|k-1)\| &\leq L_f^{N-1}\, e(k|k-1)\\
\|\hat{s}(k+N-1|k+1)-\hat{s}(k+N-1|k-1)\| &\leq L_f^{N-2}\, e(k+1|k-1)\\
&\vdots\\
\|\hat{s}(k+N-1|k+m)-\hat{s}(k+N-1|k-1)\| &\leq L_f^{(N-1)-m} e(k+m|k-1)
\end{aligned} \tag{3.22}$$

Using the Lipschitz constant L_E for the terminal cost from Lemma 4, we can derive that:

$$\begin{aligned}
&E(\hat{s}(k+N-1|k+m))-E(\hat{s}(k+N-1|k-1))\leq\\
&L_E\|\hat{s}(k+N-1|k+m)-\hat{s}(k+N-1|k-1)\|\leq\\
&L_E L_f^{(N-1)-m}\, e(k+m|k-1)
\end{aligned} \tag{3.23}$$

From Assumption 6 and considering that initial feasibility is guaranteed with $\hat{s}(k+N-1|k-1)\in E_f$, we can prove that:

$$V(\hat{s}(k+N-1|k+m))\leq \alpha_V + L_V L_f^{(N-1)-m}\, e(k+m|k-1) \tag{3.24}$$

Since we want to show that $\hat{s}(k+N-1|k+m)\in E_f$ it should hold that $V(\hat{s}(k+N-1|k+m))\leq\alpha$ and thus:

$$\begin{aligned}
&\alpha_V + L_V L_f^{(N-1)-m}\, e(k+m|k-1)\leq\alpha \Rightarrow\\
&e(k+m|k-1)\leq \frac{\alpha-\alpha_V}{L_V L_f^{(N-1)-m}}
\end{aligned} \tag{3.25}$$

which states that as long as the disturbances are bounded by 3.25 then S_F is a robust positively invariant set and thus there is a local stabilizing controller $h(s)$ which ensures that $\hat{s}(k+N-1|k+m)\in E_f$ for all $m=\{0,\ldots,N-1\}$.

3.3 Input-to-State (ISS) Stability

For the ISS Stability we can use the following Lemma, which was stated by Lazar, Heemels and Teer in their paper [86].

Lemma 4 Let $\mathbb{X}, \mathbb{E}, \mathbb{D} \subseteq \mathbb{R}^n$ and let $\alpha_1, \alpha_2, \alpha_3 \in \mathscr{K}_\infty$, $\sigma_1, \sigma_2 \in \mathscr{K}$, $\mathbb{X} \subseteq \mathbb{R}^n$ with $0 \in \text{int}(\mathbb{X})$. Let $J : \mathbb{X} \to \mathbb{R}_+$ be a function with $J(0) = 0$ and consider the following equations:

$$\alpha_1(\|x\|) \leq J(x) \leq \alpha_2(\|x\|) \tag{3.26}$$

$$J(\Psi(x,e,d)) - J(x) \leq -\alpha_3(\|x\|) + \sigma_1(\|e\|) + \sigma_2(\|d\|) \tag{3.27}$$

(i) Assume that $\mathbb{X}$ is a robustly positively invariant set for the perturbed system $x(k+1) = \Psi(x(k), e(k), d(k))$ for all $e(k) \in \mathbb{E}$ and $d(k) \in \mathbb{D}$ and (ii) assume that the two aforementioned inequalities hold for all $x \in \mathbb{X}$, $e \in \mathbb{E}$ and all $d \in \mathbb{D}$. Then, the system $x(k+1) = \Psi(x(k), e(k), d(k))$ is ISS in $\mathbb{X}$ for inputs in $\mathbb{E}$ and $\mathbb{D}$.

In order to prove the Input-to-State Stability of the system in the Event-Based NMPC framework, the cost function of the OCP problem 3.5 under the optimal control sequence V_F^* is chosen as a Lyapunov function. We denote the optimal cost at some time step $k+m$ as $J^*(k+m) = J(x_{k+m}, \overline{V_F}(k+m))$. We define the optimal cost difference between the initial triggering instant $k-1$ and the subsequent time steps $k+m$, with $m = \{0, \ldots, N-1\}$ as:

$$\Delta J_m^* = J_N^*(k+m) - J_N^*(k-1) \tag{3.28}$$

We can use the aforementioned difference, to derive the following lemma:

Lemma 5 For the system of 2.22 under the constraints stated in 3.5 that satisfies the Assumptions (2-6), the difference between the optimal cost at time step $k-1$ and all subsequent time steps $k+m$, with $m = \{0, \ldots, N-1\}$, is bounded as described by:

$$\Delta J_m^* \leq Lz_m e(k+m|k-1) - \sum_{i=0}^{m} \underline{F}(\|x_{k-1-i+m}\|) \tag{3.29}$$

where the Lz_m is defined by: $Lz_m = L_E L_f^{(N-1)-m} + L_F \frac{L_f^{(N-1)-m} - 1}{L_f - 1}$.

Proof In order to prove the boundedness of the optimal cost difference, we will employ the feasible cost at time step $k+m$ in a similar way to the one proposed in [87], which we denote as $\bar{J}_N(k+m) \triangleq J_N(s_{k+m}, \bar{V}_F(k+m))$. The feasible cost is the cost function under the feasible control sequence 3.21, and thus the difference between the feasible cost at time step $k+m$ and the optimal cost at time step $k-1$ is given by:

$$\Delta J_m = \bar{J}_N(k+m) - J_N^*(k-1) \tag{3.30}$$

By starting with $m=0$ we can derive that:

$$\begin{aligned}
&\Delta J_0 = \bar{J}_N(k) - J_N^*(k-1) = \\
&\sum_{i=0}^{N-1} \{ F(\bar{s}(k+i|k), \bar{V}(k+i|k)) - F(\hat{s}(k+i-1|k-1), V^*(k+i-1|k-1)) \} \\
&+ E(\bar{s}(k+N|k)) - E(\hat{s}(k+N-1|k-1)) = \\
&\sum_{i=0}^{N-2} \{ F(\bar{s}(k+i|k), \bar{V}(k+i|k)) - F(\hat{s}(k+i|k-1), V^*(k+i|k-1)) \} \\
&- F(s_{k-1}, V_{k-1}) + F(\bar{s}(k+N-1|k), h(\bar{s}(k+N-1|k))) \\
&+ E(\bar{s}(k+N|k)) - E(\hat{s}(k+N-1|k-1))
\end{aligned} \tag{3.31}$$

where $\bar{x}(k+i+1|k+m)$ is state of the system under the feasible control sequence from time step $k+m$ until time step $k+i$.

From the feasible control sequence of 3.21, it is evident that:

$$\overline{V}(k+i|k) = V^*(k+i|k-1) \quad \forall i = \{0, \dots, N-2\} \tag{3.32}$$

It is simple to derive:

$$\|\hat{s}(k+i|k) - \hat{s}(k+i|k-1)\| \leq L_f^i \, e(k|k-1) \tag{3.33}$$

Then using the Lipschitz property of the stage cost L_F and the expression from 3.33, it holds that:

$$F(\bar{s}(k+i|k), \bar{V}(k+i|k)) - F(\hat{s}(k+i|k-1), V^*(k+i|k)) \leq L_F L_f^i \, e(k|k-1) \tag{3.34}$$

Since we proved that initial feasibility implies feasibility afterwards, then $\hat{s}(k+N-1|k+m) \in E_f$ for all $m = \{0, \dots, N-1\}$. Thus for $m=0$ it holds that $\hat{s}(k+N-1|k) \in E_f$. Utilising

Assumption 3 we get:

$$E(\bar{s}(k+N|k)) - E(\hat{s}(k+N-1|k)) + F(\bar{s}(k+N-1|k), h(\bar{s}(k+N-1|k))) \leq 0 \tag{3.35}$$

Therefore, using the Lipschitz constant of the terminal cost and the expression from 3.23 we derive:

$$\Delta J_0 \leq Lz_0 \, e(k|k-1) - \underline{F}(\|x_{k-1}\|) \tag{3.36}$$

where $Lz_0 = L_E L_f^{N-1} + L_F \frac{L_f^{N-1}-1}{L_f-1}$.

In a similar fashion we can prove that for $m = 1$:

$$\begin{aligned}
&\Delta J_1 = \bar{J}_N(k+1) - J_N^*(k-1) = \\
&\sum_{i=0}^{N-1} \{ F(\bar{s}(k+i+1|k+1), \bar{V}(k+i+1|k+1)) - F(\hat{s}(k+i-1|k-1), V^*(k+i-1|k-1)) \} \\
&+ E(\bar{s}(k+N+1|k+1)) - E(\hat{s}(k+N-1|k-1)) = \\
\\
&\sum_{i=0}^{N-3} \{ F(\bar{s}(k+i+1|k+1), \bar{V}(k+i+1|k+1)) - F(\hat{s}(k+i+1|k-1), V^*(k+i+1|k-1)) \} \\
&- F(s_{k-1}, V_{k-1}) - F(s_k, V_k) \\
&+ F(\bar{s}(k+N-1|k+1), h(\bar{s}(k+N-1|k+1))) + E(\bar{s}(k+N+1|k+1)) \\
&+ F(\bar{s}(k+N|k+1), h(\bar{s}(k+N|k+1))) - E(\hat{s}(k+N-1|k-1))
\end{aligned} \tag{3.37}$$

Since $\bar{s}(k+N|k+1) \in E_f \subseteq E_n$ and $\bar{s}(k+N-1|k+1) \in E_n$, we can use Assumption 3 to derive:

$$E(\bar{s}(k+N+1|k+1)) - E(\bar{s}(k+N|k+1)) + F(\bar{s}(k+N|k+1), h(\bar{s}(k+N|k+1))) \leq 0 \tag{3.38}$$

$$E(\bar{s}(k+N|k+1)) - E(\bar{s}(k+N-1|k+1)) + F(\bar{s}(k+N-1|k+1), h(\bar{s}(k+N|k+1))) \leq 0 \tag{3.39}$$

Also, using the Lipschitz constant for the terminal cost as well as the expression 3.23 for $m = 1$:

$$E(\bar{s}(k+N-1|k+1)) - E(\bar{s}(k+N-1|k-1)) \leq L_E L_f^{N-2} e(k+1|k-1) \tag{3.40}$$

Hence, for $m = 1$ the bound for the difference is given by:

$$\Delta J_1 \leq Lz_1 \; e(k+1|k-1) - \underline{F}(\|x_{k-1}\|) - \underline{F}(\|x_k\|) \tag{3.41}$$

where $Lz_1 = L_E L_f^{N-2} + L_F \frac{L_f^{N-2}-1}{L_f-1}$

Therefore, using a similar approach we can derive that for all $m = \{0, \ldots, N-1\}$ it holds that:

$$\Delta J_m \leq Lz_m e(k+m|k-1) - \sum_{i=0}^{m} \underline{F}(\|x_{k-1-i+m}\|)$$

From the optimality of the solution it holds that for all m:

$$\begin{aligned} J_N^*(k+m) &\leq \bar{J}_N(k+m) && \Longleftrightarrow \\ J_N^*(k+m) - J_N^*(k-1) &\leq \bar{J}_N(k+m) - J_N^*(k-1) && \Longleftrightarrow \\ \Delta J_m^* \leq \Delta J_m &\leq Lz_m e(k+m|k-1) - \sum_{i=0}^{m} \underline{F}(\|x_{k-1-i+m}\|) && \end{aligned} \tag{3.42}$$

which concludes the proof.

We have now proved the first inequality condition of 3.26 from Lemma 4 taking into account solely outer perturbations. Hence, it is evident that:

$$\sigma_1(\|e\|) = Lz_m e(k+m|k-1) \tag{3.43}$$

$$\alpha_3(\|s\|) = \sum_{i=0}^{m} \underline{F}(\|x_{k-1-i+m}\|) \tag{3.44}$$

In order to determine the lower and upper bounds of the Lyapunov function, as indicated in 3.42, we can use the optimality of the solution to derive that:

$$\underline{F}(\|x_k\|) \leq J_N^*(k) \leq \bar{J}_N(k) \leq (L_E + L_{F_V} L_h)\frac{L_{f_h}^N - 1}{L_{f_h} - 1}\|x_k\| + L_E L_{f_h}\|x_k\| \iff \tag{3.45}$$

$$\alpha_1(\|x\|) \leq J_N^*(k) \leq \alpha_2(\|x\|) \tag{3.46}$$

where $\alpha_1(\|x\|) = \underline{F}(\|x_k\|)$ and $\alpha_2(\|x\|) = (L_E + L_{F_V} L_h)\frac{L_{f_h}^N - 1}{L_{f_h} - 1}\|x_k\| + L_E L_{f_h}\|x_k\|$

3.4 Event-triggering Condition

Now that we have proved the feasibility and input-to-state stability of the event-based NMPC scheme, we are going to provide the triggering rule. By taking into account only outer perturbations, the expression 3.26 for the Input-to-State stability becomes :

$$J(s_{k+1}) - J(s_k) \leq -\alpha_3(\|s\|) + \sigma_1(\|e\|) \tag{3.47}$$

Therefore, the system remains ISS stable under the event-based NMPC scheme as long as:

$$\sigma_1(\|e\|) \leq \sigma\alpha_3(\|s\|) \tag{3.48}$$

where σ is a positive constant $0 < \sigma < 1$.

Therefore, combining 3.29 with 3.47 and 3.48 we can derive the triggering condition as:

$$Lz_m e(k+m|k-1) \leq \sigma \sum_{i=0}^{m} \underline{F}(\|x_{k-1-i+m}\|) \tag{3.49}$$

Additionally, the stability of the event-base scheme needs to be guaranteed for all subsequent $m = \{0,\ldots,N-1\}$. The optimal cost difference between all two subsequent time steps needs to be decreasing and thus it should hold that:

$$\Delta J^*_{m+1} \leq \Delta J^*_m \iff$$

$$Lz_{m+1} e(k+m+1|k-1) - Lz_m e(k+m|k-1) \leq \underline{F}(\|x_{k-1+m}\|) \tag{3.50}$$

Therefore, the OCP problem of the NMPC will be solved again if the triggering condition 3.49 is violated. The ET-IBVS-NMPC algorithm is presented schematically in Algorithm 2

Algorithm 2 Event-Triggered NMPC

1: Time step equals to k
2: **while** $x_k \in S_{set}$ **do**
3: Update the state measurement of system s_k
4: Solve the OCP for the open-loop input sequence $V_k^* = [V(k|k), \ldots, V(k+N-1|k)]$
5: Calculate the predicted state sequence $\hat{s}_k = [\hat{s}(k|k), \ldots, \hat{s}(k+N-1|k)]$
6: Set $k \leftarrow k_0$ ▷ $e(k|k_0) = \|s_k - \hat{s}(k|k_0)\| = 0$
7: **while** State error does not violate the event-triggering condition **do**
8: Apply the control input $V(k|k_0)$ to the system
9: Update the time-step: $k \leftarrow k+1$
10: Update the state measurement s_k
11: Calculate the state error $e(k|k_0)$
12: **end while**
13: **go to** 3
14: **end while**

The ET-IBVS-NMPC scheme is presented in Figure 3.4

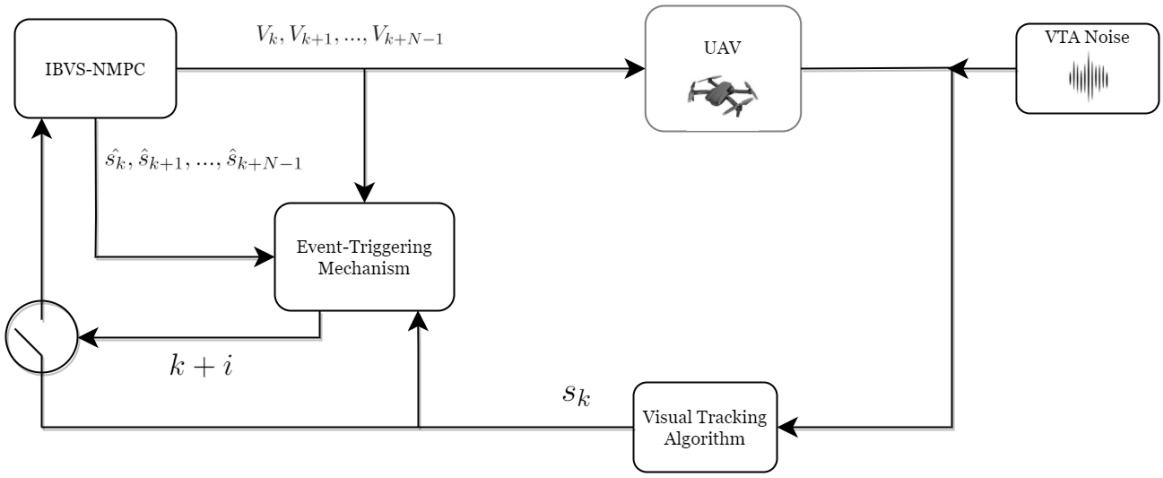

Figure 3.4 ET-IBVS-NMPC scheme using a quadrotor UAV

Chapter 4

Simulations

To verify the efficiency and performance of our proposed ET-IBVS-NMPC scheme, simulations in a realistic coastline environment were conducted. The Robot Operating System (ROS) in conjunction with the Gazebo simulator was employed to act as the Software In The Loop (SITL) on a computer running Ubuntu 20.04 LTS. The ArduPilot firmware was used as the autopilot simulated controller while the MAVROS ROS package was utilized to establish the communication between the Ardupilot and ROS under the MAVLink protocol.

4.1 Simulation Setup

The performance and efficiency of the ET-IBVS-NMPC scheme for the coastline tracking using a quadrotor UAV were demonstrated in simulations. Specifically, the Gazebo framework was employed in order to create a realistic version of an outdoor coastline environment. Gazebo is an open-source 3D robotics simulator supporting the development of terrestrial, aerial, space, and underwater robots for indoor and outdoor environments which integrates the ODE physics engine, OpenGL rendering, as well as support code for sensor simulation and actuator control. Gazebo is an open-source software developed by Open Source Robotics Foundation (OSRF) and maintained by the robotics community and has emerged as the de facto standard environment for prototyping and simulating robots. The software follows a modular approach which supports the following characteristics:

1. Gazebo integrates a multitude of physics engines to simulate dynamics and resolve the collision, reaction, and contact force among rigid bodies.

2. Gazebo includes a vast module library that facilitates the employment of robotic sensors such as GPS, cameras, lasers, IMUs, etc.

3. Gazebo supports a Graphical User Interface (GUI) that allows the visualization and exploration of various environments and robotic systems as well as the interaction between them using high-quality graphics.

4. Gazebo supports the creation of user-defined plugs using the Robot Operating System (ROS) in different programming languages such as C++ and Python.

More information on the Gazebo simulator can be found in [88].

The coastline environment used in our simulations features a model for the ocean-based on realistic oceanographic statistical descriptions which are computationally complex, realistically looking, and is characterized by high fidelity. The waves of the ocean are modeled based on Gerstner Waves [89] as well as the approach followed in [90], which are used often for ocean representation in computer graphics as shown in [91], depict the ocean surface using a trochoidal shape. In order to achieve higher visual realism, Gerstner waves are characterized by larger amplitude, edgy crests, and wider troughs thus exhibiting strong real world-resemblance. In addition, this wave model offers high flexibility within the Gazebo framework as its parameters (period, gain, direction, amplitude) can be easily configurable through a plugin integrated into Gazebo. The coastline representation in our simulation is complemented by the use of an island model to represent the shore. The quadrotor UAV model is based on the IRIS 3DR quadrotor [92] in which a downward-facing ZED Stereo-vision RGB-D camera that records data of the surrounding environment and detects the coastline is added.

4.2 Software Architecture

In our simulations, the coastline tracking approach was implemented through the Robot Operating System (ROS) middleware suite. ROS consists of a collection of software frameworks for robotic applications and is the de-facto standard for programming in the robotics community. ROS was originally developed in 2007 at the Artificial Intelligence Laboratory at Stanford University and has been managed by the Open Source Robotics Foundation (OSRF) since 2013. ROS follows a peer-to-peer approach with individual programs communicating over a specific Application Programming Interface (API), which allows them to communicate over a standard protocol. Each script containing functional code is called a node and is capable of sending and receiving information from and to any other node through publishing or subscribing to a topic. Topics are named buses that allow communication between nodes through messages containing information such as data, video, or images. This enables multiple nodes to communicate with each other, resulting in a streamlined path of information

exchange that can include various types of information such as sensor data and actuator commands. Programs following the ROS architecture include ROS nodes and a ROS master. The ROS master is a node in itself that handles the communication between the other nodes, which are executables running in order to achieve a desired function of the robotic system. Examples include nodes that can handle processes such as low-level device control, reading and processing of sensor data, actuator handling, hardware abstraction, and communication between processes. Given the open-source nature of ROS, there is an extensive collection of libraries, packages, and nodes that are written in a variety of programming languages, with the most popular being C++, Python, and Matlab. Lastly, since ROS follows a graph architecture of nodes, programs can be run in different computers communicating over a network in a distributed fashion, thus offering great flexibility in various situations such as experimental setups. More information on ROS can be found in [93]. In our simulation, the Melodic version of ROS 2 [94] running on Ubuntu 20.04 LTS [95] was used in order to develop the program architecture for the implementation. Our software cluster consists of the following programs :

4.2.1 Gazebo Environment

The Gazebo program that encompasses the simulation environment including the ocean, shore, IRIS 3DR quadrotor, and ZED stereo-vision camera. It also includes a node for the Graphical User Interface (GUI) that allows us to explore and manipulate the environment and its objects. The IRIS quadrotor is depicted inside the coastline environment of Gazebo in Figure 4.1

4.2.2 MAVROS Package

MAVROS is the package that provides communication drivers between ROS and the ArduPilot autopilot software under the MAVLink communication protocol [96][97]. The MAVLink communication protocol is a lightweight messaging protocol for communication with drones and between onboard drone components. It was released by Lorenz Meier in 2009 and is maintained by hundreders of contributors of the open-source robotics community. Regardless of the deployed platform, MAVLink utilized a binary format after it serializes and performs checksum verification on the messages and commands it handles. As a result of this binary serialization nature, the communication protocol is lightweight and therefore provides accuracy and reliability in experimental applications in which small information parcels can be transmitted bidirectionally between the Ground Control Station (GCS) and the UAV over wireless communication networks [98] [99]. Hence, it acts as the middleware communication

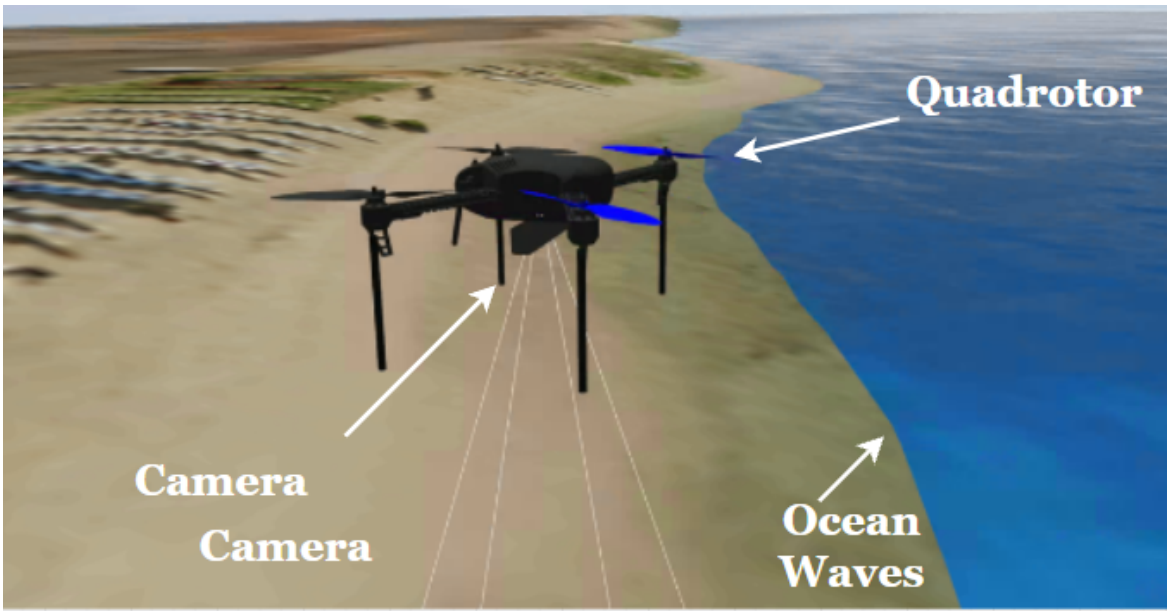

Figure 4.1 Iris quadrotor inside the Gazebo coastline environment

layer between the GCS and the ArduPilot of the UAV, or in our case between the SITL and the Ardupilot.

4.2.3 ET-IBVS-NMPC Algorithm

This node contains the ET-IBVS-NMPC algorithm which is used for the coastline tracking task. The module was developed in C++ in order to generate fast efficient code which is crucial for this type of application. The numerical computations were done using the Eigen Libary [100] while the Optimal Control Problem of the NMPC was solved using the NLopt Library. NLopt is an open-source library of various routines for nonlinear optimization [101]. For our study, the BOBYQA algorithm was employed which is a derivative-free algorithm for constrained nonlinear optimization originally developed by M.J.D. Powell [102] and converted to C++.

4.2.4 Coastline Detection CNN

This module is utilized for image extract from the camera topic as well as for the detection of the coastline. The coastline detection is achieved online through a trained Convolutional Neural Network (CNN) running on a GPU. Once the coastline is detected, the Region of Interest (ROS) is created through the four points that were detected from the CNN. The four corners of the ROI are used in the IBVS problem formulation to track the coastline. This

node has been developed using Python, OpenCV [103], and the Keras framework [104] by members of the Control System Laboratory (CSL) at the National Technical University of Athens (NTUA). The detected coastline is presented in Figure 4.2

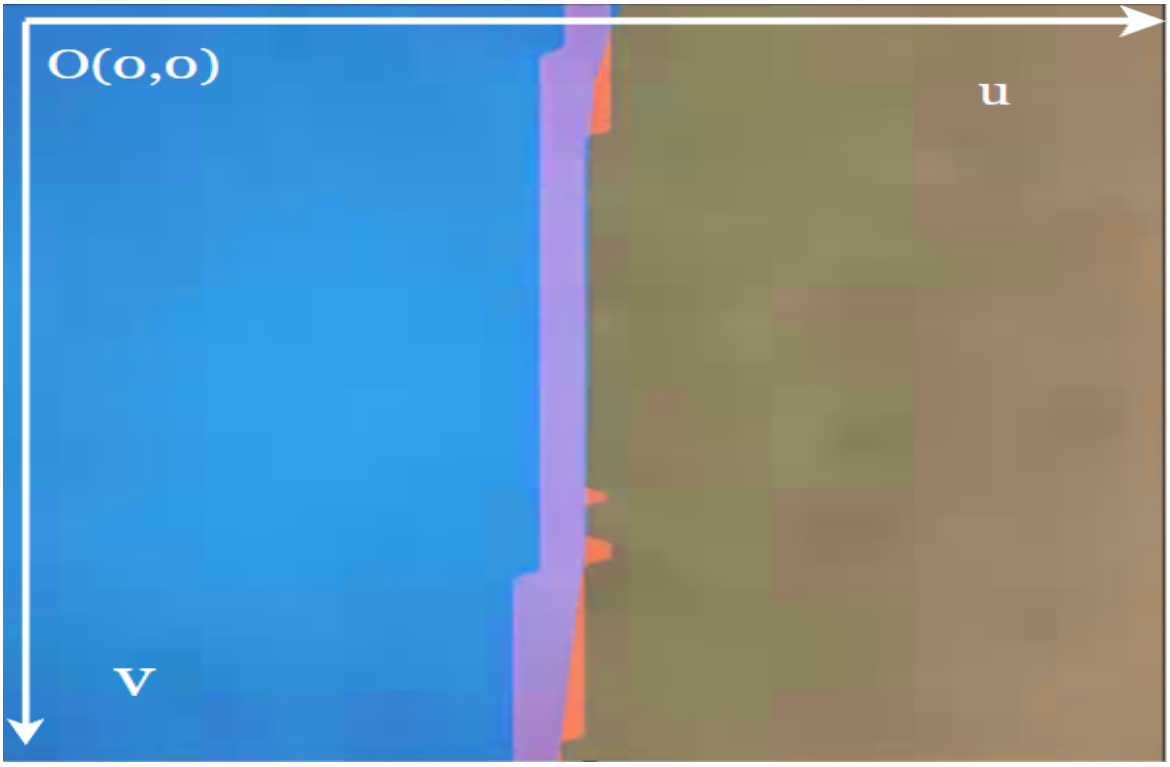

Figure 4.2 Coastline Detection using a CNN

4.2.5 ArduPilot Flight Controller

The flight controller used in our simulations utilized the ArduPilot autopilot software. ArduPilot is an open-source node-based autopilot that enables direct communication with ROS through the MAVROS package and benefits from interfaces that enable the use of a broad variety of sensors, computational units and communication systems. The system offers high flexibility as it allows the integration of both low and high-level controllers, and supports user-defined software packages. Another characteristic of this software suite is the provision of hardware safety guarantees which can reduce the accident occurrence in experimental scenarios. More information on Ardupilot can be seen in [105]. In our simulations, the ArduPilot ensures the low-level control that handles the achievement of the four desired velocities that are generated from the ET-IBVS-NMPC scheme, which include the three translational velocities in the x,y, z-axis as well as the yaw rate, in the North-East-Down (NED) body coordinate frame of the quadrotor.

4.3 Simulation Results

In our study, two simulation scenarios were investigated in order to prove the efficiency and performance of the ET-NMPC by comparing it to the classic NMPC scheme. Specifically, the algorithm was tested in simulations, in which the quadrotor started from challenging initial configuration w.r.t the desired pose of the camera starting from positions with relatively low altitudes (below 10 meters). In both scenarios the coastline was initially within the field of view of the camera and never violated any of the imposed FOV constraints. For comparison purposes, each scenario was investigated under both the NMPC and the ET-NMPC with the quadrotor approximately the same initial configuration and altitude.

The Field Of View constraints, due to the 720x480 resolution of the ZED Stereo Vision Camera are described in *pixels* below. They are based on the coordinate frame depicted in 4.2 whose center is located on the top left corner.

$$\begin{bmatrix} \underline{u} = 0 \\ \underline{v} = 0 \end{bmatrix} \leq \begin{bmatrix} u(t) \\ v(t) \end{bmatrix} \leq \begin{bmatrix} \bar{u} = +720 \\ \bar{v} = +480 \end{bmatrix} \tag{4.1}$$

Furthermore, the focal length of the camera is equal to $f = 252.07$ and the principal point has the following coordinates $(c_u, c_v) = (360, 240)$.

The input constraints describing the camera translation and angular velocity bounds are described in m/sec and rad/sec respectively below:

$$\begin{bmatrix} \underline{T}_x = & -0.5 \\ \underline{T}_y = & -3 \\ \underline{T}_z = & -0.1 \\ \underline{\Omega}_z = & -1 \end{bmatrix} \leq \begin{bmatrix} T_x(t) \\ T_y(t) \\ T_z(t) \\ \Omega_z(t) \end{bmatrix} \leq \begin{bmatrix} \bar{T}_x = & +0.5 \\ \bar{T}_y = & +3 \\ \bar{T}_z = & +0.1 \\ \bar{\Omega}_z = & +1 \end{bmatrix} \tag{4.2}$$

In order to ensure convergence and stability of the algorithm a relatively large prediction horizon is advised. However, an increase to the size of the prediction horizon consequently causes an increase of the computational time of the OCP problem at each iteration. Furthermore, as seen in the Lz_m constant of the triggering condition, a very large horizon can cause an increase of events even in cases in which the error between the predicted and real state of the system is small, and can thus lead to higher energy consumption and less autonomy. Therefore, a trade-off between performance and computational efficiency has to occur, in which a large enough prediction horizon which ensures both efficiency and does not harm the efficiency of the scheme should be chosen. For these reasons, the size of the receding

prediction horizon is chosen equal to $N = 6$. The sampling time in all simulation scenarios is chosen equal to $dt = 0.1$ seconds. The parameters for the waves of the ocean based on the Gerstner Wave model are presented in the following Table. In both cases the motion of the waves is only considered in the direction vertical to the coastline.

Gazebo Wave Parameters				
	Wave Period	Wave Scale	Wave Angle	Wave Gain
Simulation 1	2.5	1.4	0.4	1.6
Simulation 2	2.5	2	0.4	2.2

In the following simulations scenarios, the results demonstrating the efficacy and performance of the ET-NMPC are presented. Comparisons are made to the classic NMPC approach. Video recordings of the simulation flights can be seen in [106].

4.3.1 Simulation 1

For simulation 1, the initial configuration, denoted as $p_{i,1}$, is shown in Figure 4.3 and described below:

$$p_{i,1} = [u_1, v_1, u_2, v_2, u_3, v_3, u_4, v_4] = [21, 370, 682, 310, 684, 332, 23, 393] \tag{4.3}$$

The coastline detection is shown on the window on the top left corner of the image. The α constant for the IBVS-NMPC and ET-IBVS-NMPC cases is presented below:

$$\begin{aligned} \dot{v}_{des}(t) &= \alpha_{IBVS-NMPC} = 30 \quad \text{pixels/sec} \\ \dot{v}_{des}(t) &= \alpha_{ET-IBVS-NMPC} = 50 \quad \text{pixels/sec} \end{aligned} \tag{4.4}$$

The desired configuration $p_{i,des}$ is presented below. The desired v coordinate values presented below are the values for the time step $k = 0$, the start of the prediction horizon, and progress in the horizon based on the $v_{des,k+1} = v_{des,k} + \alpha dt$ expression.

$$\begin{aligned} p_{i,des} =& [u_{1,des}, v_{1,des}, u_{2,des}, v_{2,des}, u_{3,des}, v_{3,des}, u_{4,des}, v_{4,des}] \\ =& [340, 480, 340, 0, 380, 0, 380, 480] \end{aligned} \tag{4.5}$$

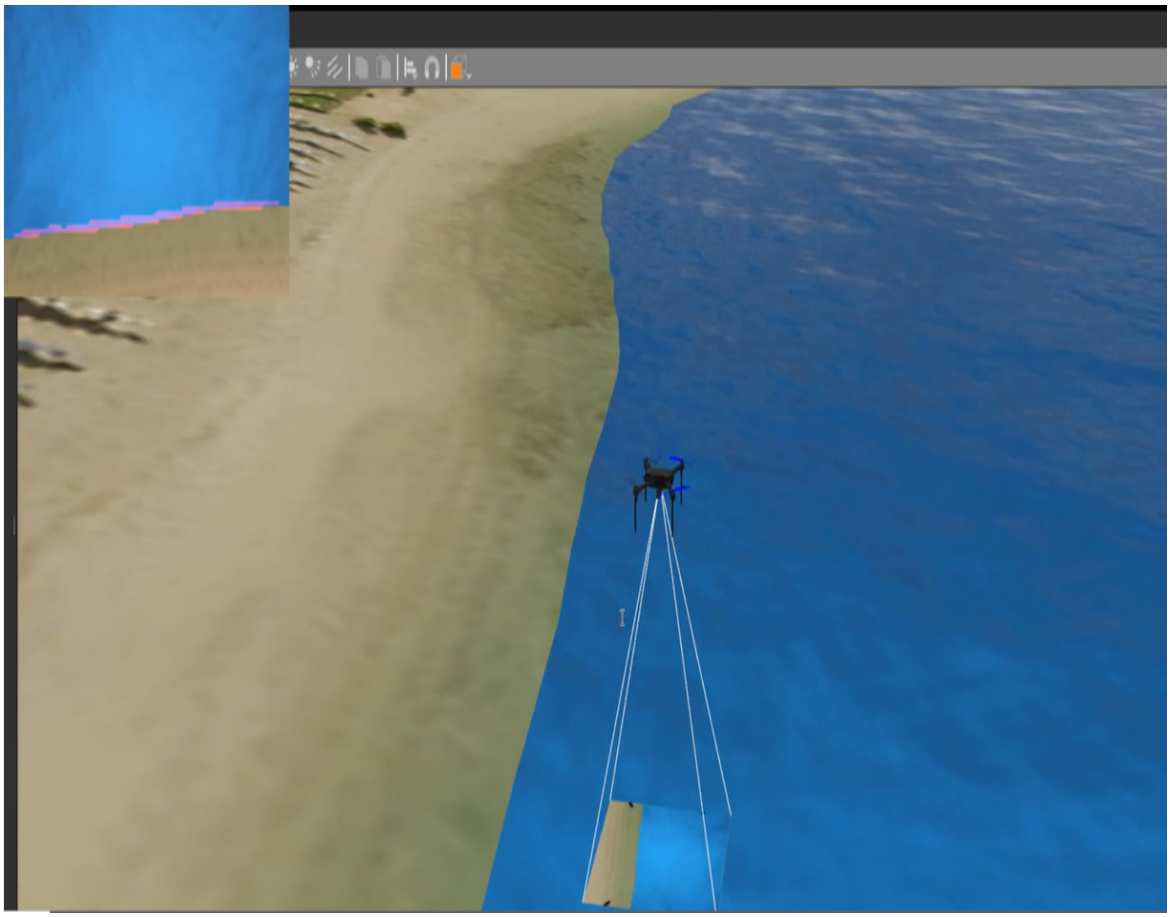

Figure 4.3 Simulation 1: Quadrotor Initial Configuration

In Figures 4.4, 4.5, 4.6 and 4.7, the image error evolution for feature 1,2,3 and 4 respectively is depicted under both the NMPC and ET-NMPC schemes.

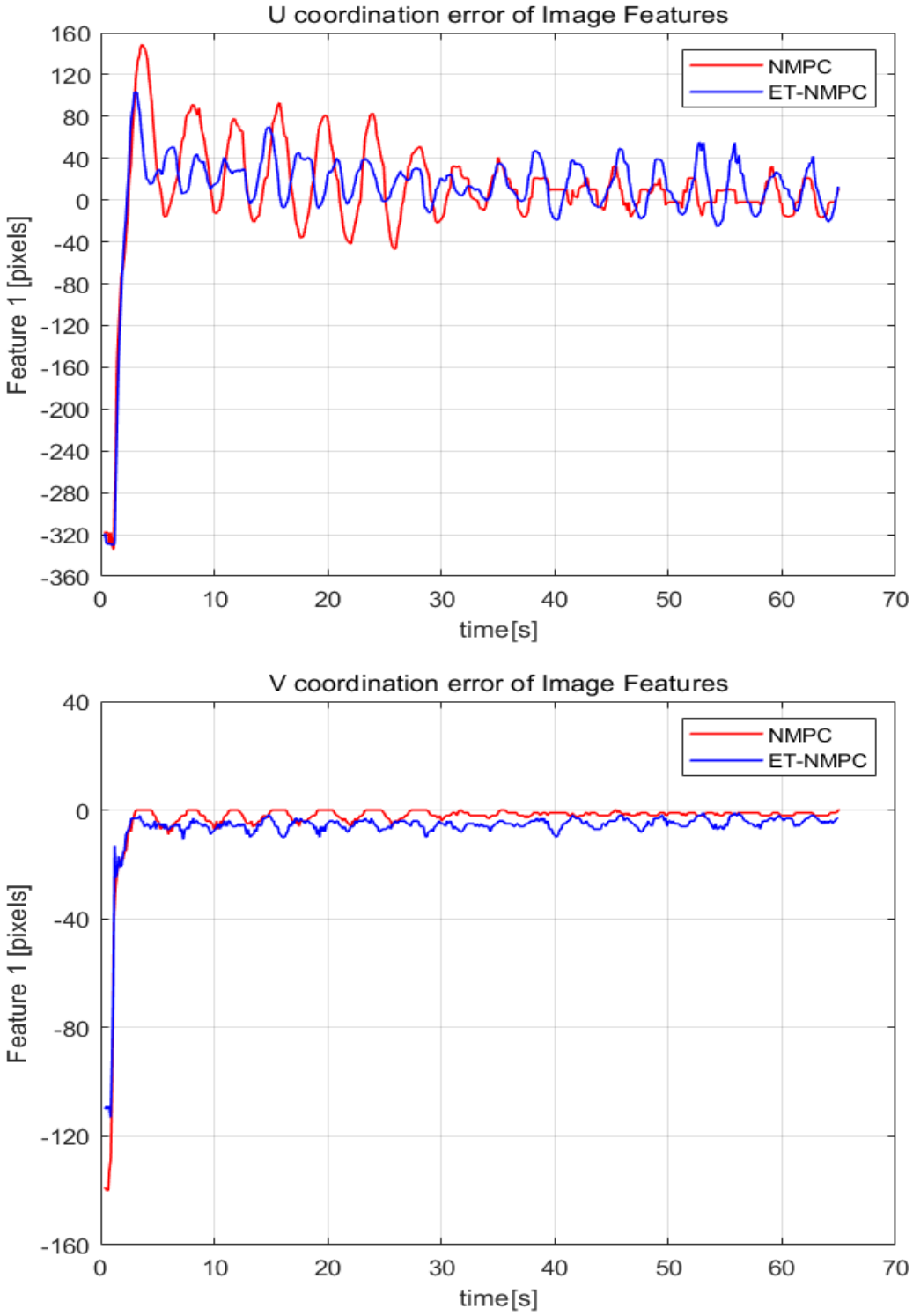

Figure 4.4 Simulation 1-Feature 1: The feature coordination errors under the NMPC and ET-NMPC

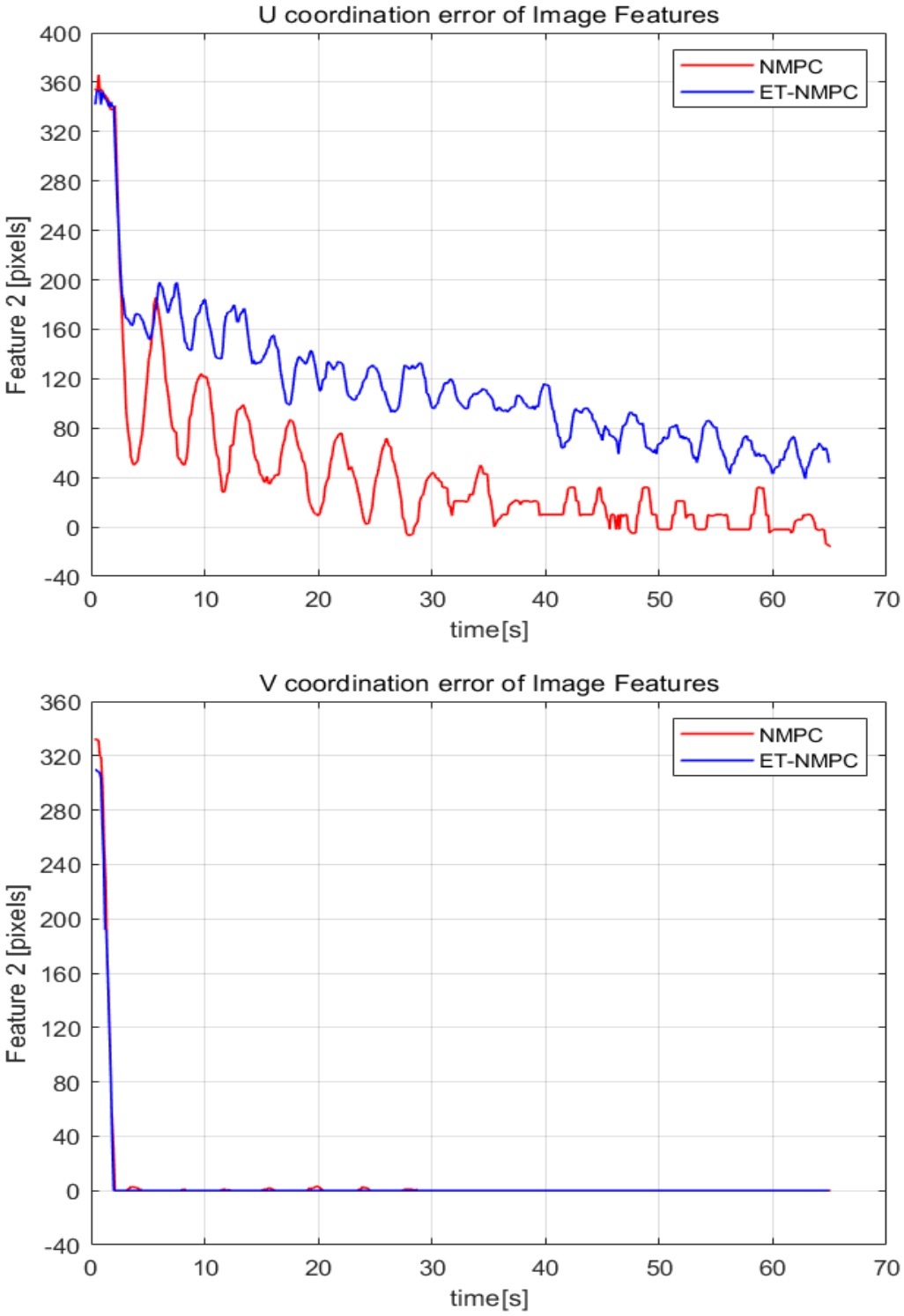

Figure 4.5 Simulation 1-Feature 2: The feature coordination errors under the NMPC and ET-NMPC

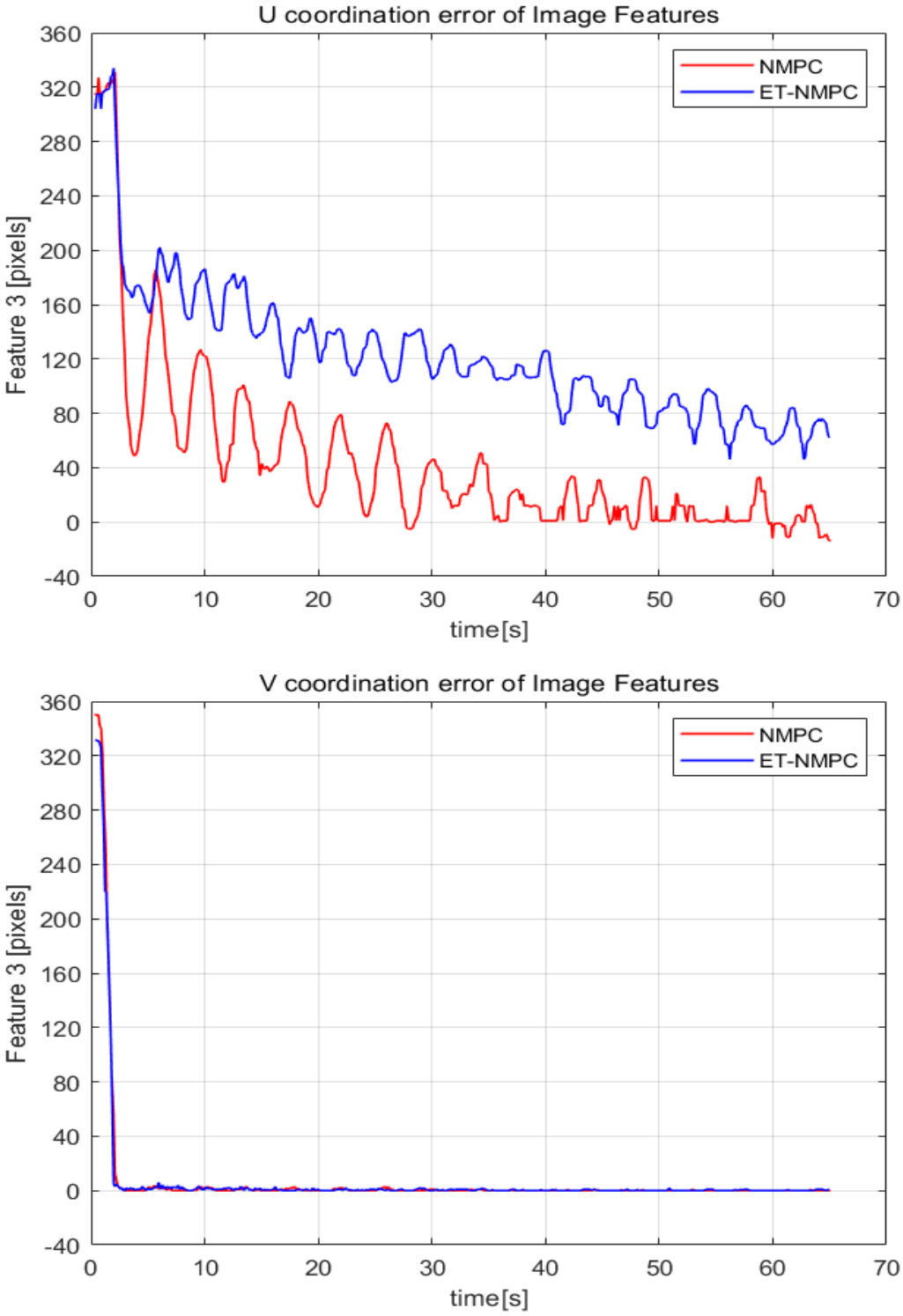

Figure 4.6 Simulation 1-Feature 3: The feature coordination errors under the NMPC and ET-NMPC

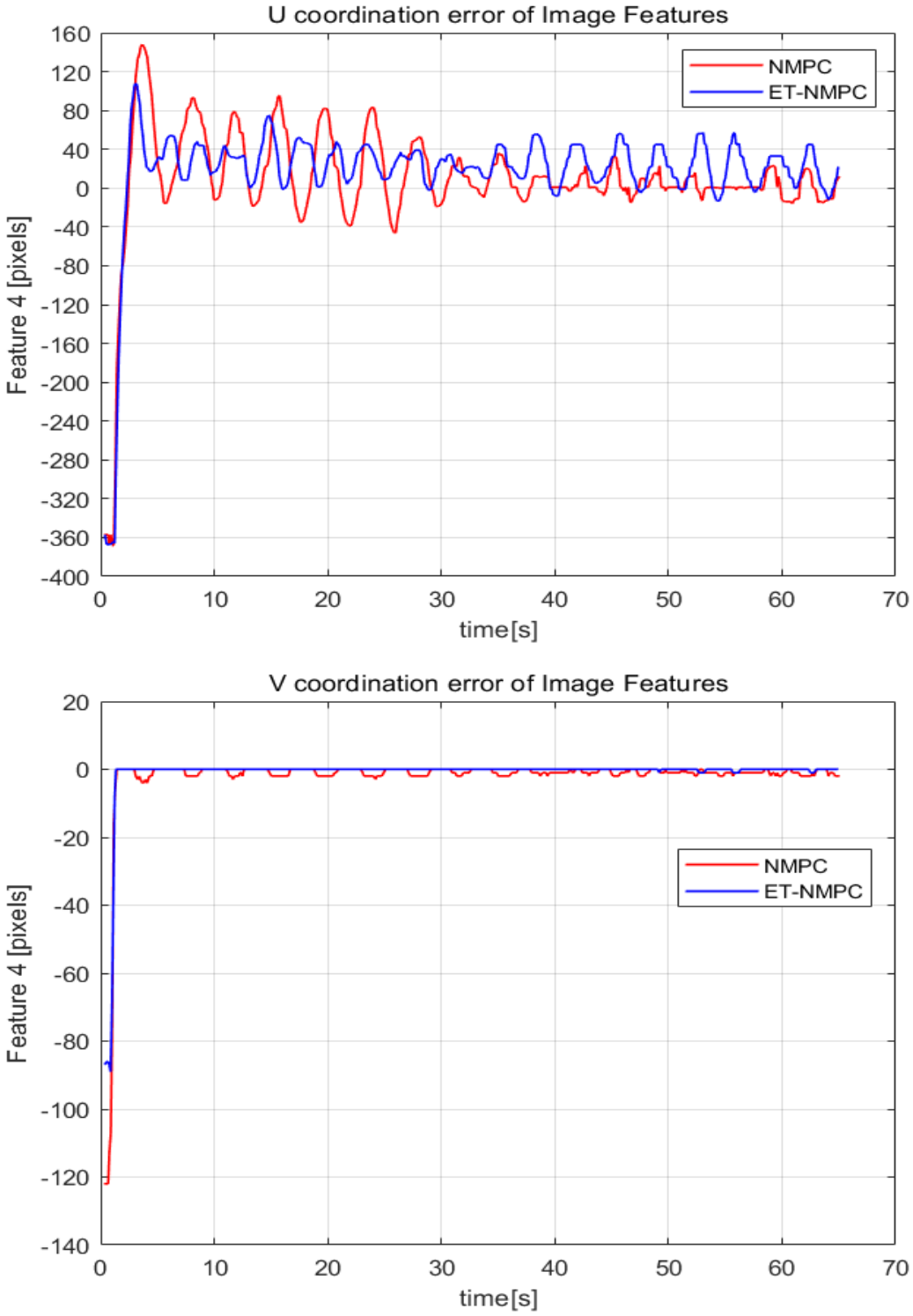

Figure 4.7 Simulation 1-Feature 4: The feature coordination errors under the NMPC and ET-NMPC

For the ET-NMPC case, as it can be clearly seen, the feature error on the v direction of the image frame converges to 0 for all four image features.For the u direction of the image frame, the coordination error converges to 0 following an sinusoidal fashion, which can be attributed to the motion of the waves in the u direction. The absolute value of the steady state error for features 1 and 4 is bounded by approximately 40 pixels while features 2 and 3 remain below 80 pixels. The steady state is achieved at approximately 55 seconds. Under the classic NMPC scheme, the errors follow a similar behaviour but converge close to 0 faster and are smaller in the steady state, which is reached at approximately 35 seconds.

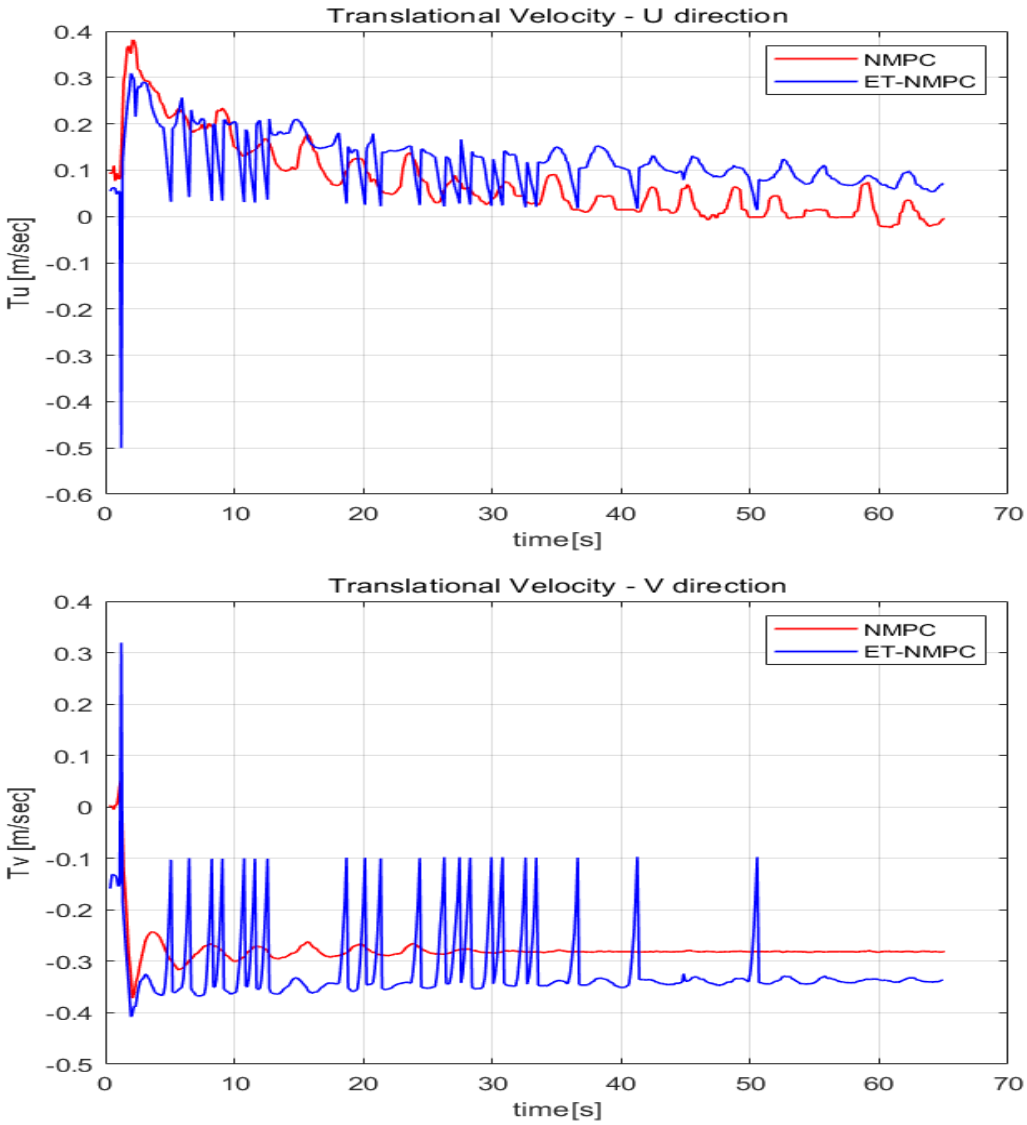

Figure 4.8 Simulation 1: The U and V Translational Camera Velocities under the NMPC and ET-NMPC

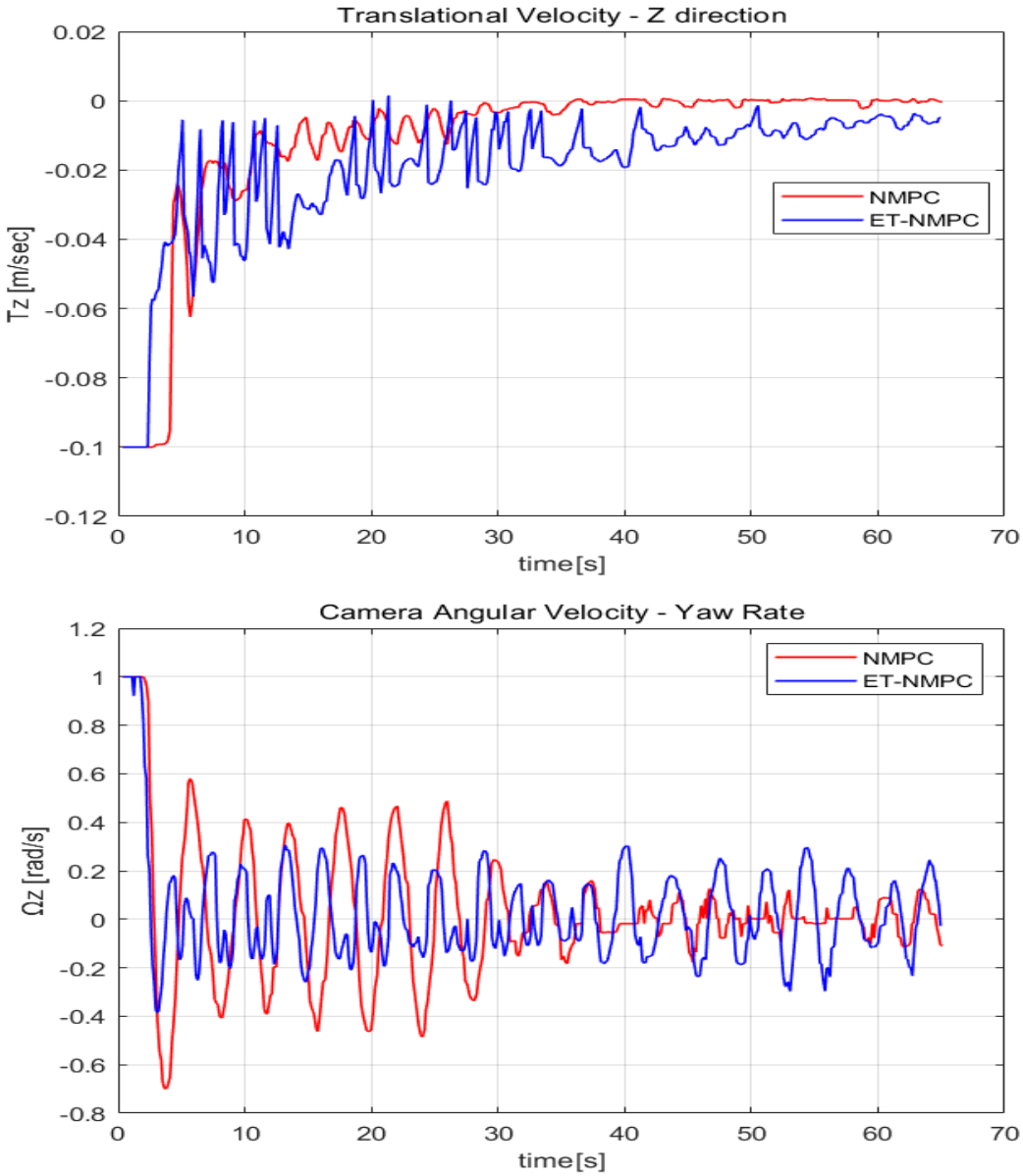

Figure 4.9 Simulation 1: The Z Translational and Ω_z Angular Camera Velocities under the NMPC and ET-NMPC

The velocities of the camera are presented in Figure 4.8. As we can seen, the velocities satisfy the input constraints throughout the flight. The camera translational velocity in the V direction is equal to the velocity with which the quadrotor traverses along the coastline. This velocity is larger in the ET-NMPC scenario, as expected due to the higher value of the α desired feature velocity in the v direction of the image frame. The translational and angular velocities of the camera in the z direction are presented in Figure 4.9.

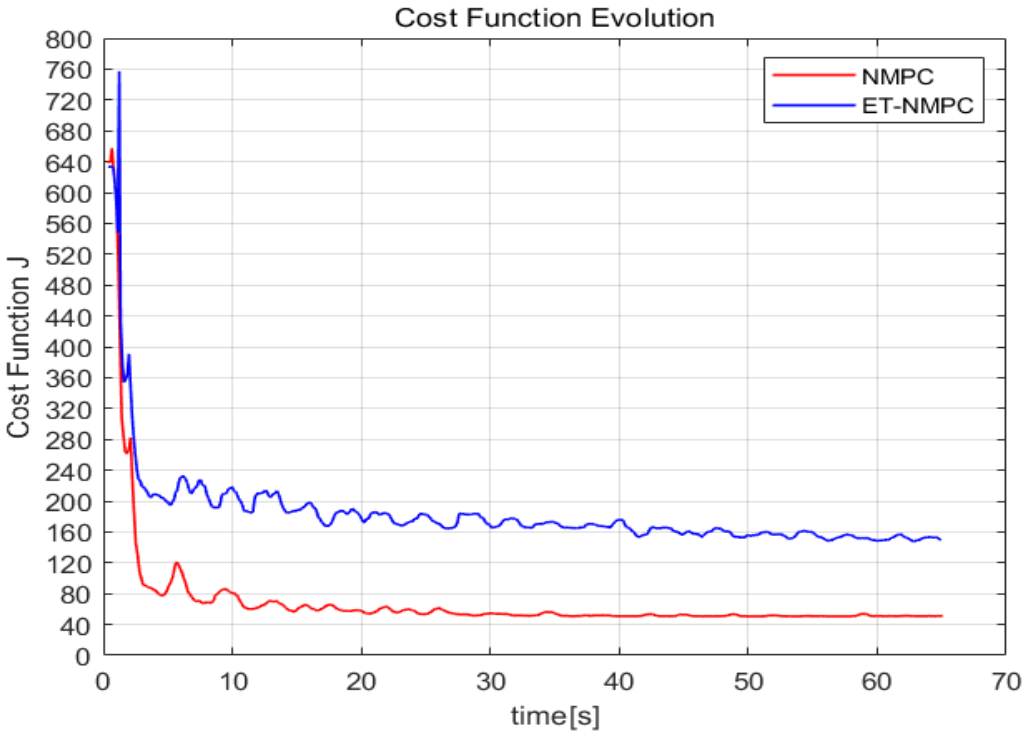

Figure 4.10 Simulation 1: The evolution of the Cost Function under the NMPC and ET-NMPC

The evolution of the cost function is presented in Figure 4.10. We can see that the cost function converges fast to a minimum value. In both cases, the cost function is lower bounded by a non-zero minimum value, which larger in the ET-NMPC case. The minimum value is not zero because the desired values of the v feature coordinates is larger the upper FOV bound, which can obviously never be reached. Furthermore, the minimum value in the ET-NMPC case is larger than in the classic NMPC due to the larger α desired velocity of the v feature coordinates. Small differences between the two schemes are also seen in the evolution of the quadrotor altitude, which remains in relatively low values and is presented in Figure 4.11. Moreover, the optimisation loop execution duration presented in Figure 4.12 proves the high speed of the C++ algorithm. As we can see, the execution duration for the optimisation loops are below 25 milliseconds which makes the algorithm viable for real-time applications.

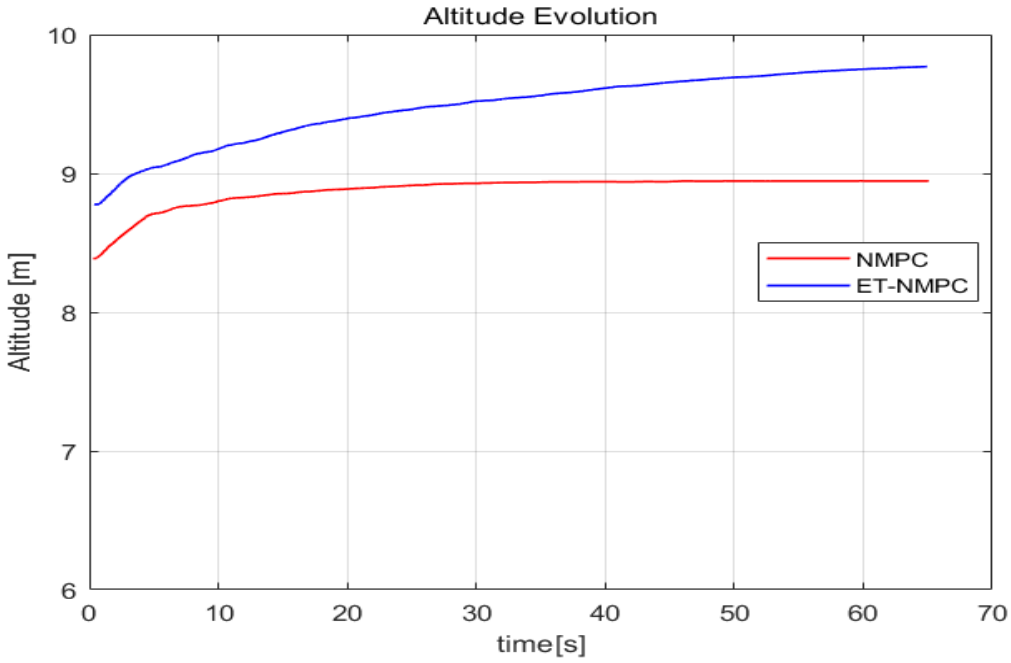

Figure 4.11 Simulation 1: The altitude of the quadrotor under the NMPC and ET-NMPC

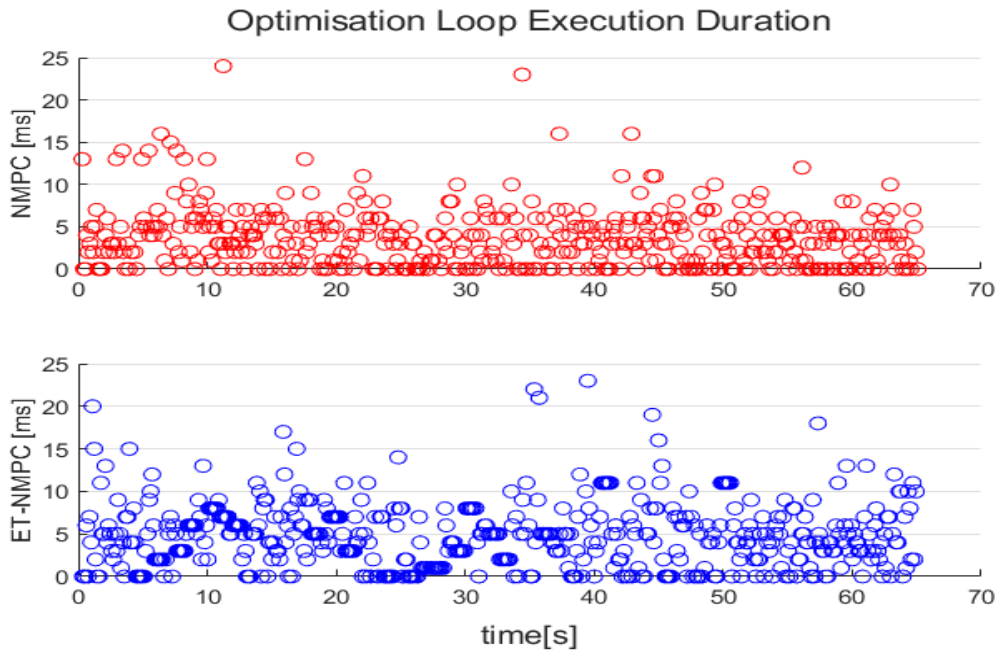

Figure 4.12 Simulation 1: Optimisation Loop Execution Duration under the NMPC and ET-NMPC

Lastly, in Figure 4.13 the triggering instances of the optimisation algorithm are depicted. As expected, as the camera moves closer to the desired state, the frequency of the events increases due to the higher accuracy that the system demands in order to stay close to the target in the presence of the additive noise and the field of view constraints. Hence, the system requires new measurements in order to generate a new control sequence that will stabilize the system to the desired state. In this scenario, the triggering of the MPC scheme is reduced by 28% with 468 triggering instances occuring instead of 600, as it happens in the classic NMPC scenario.

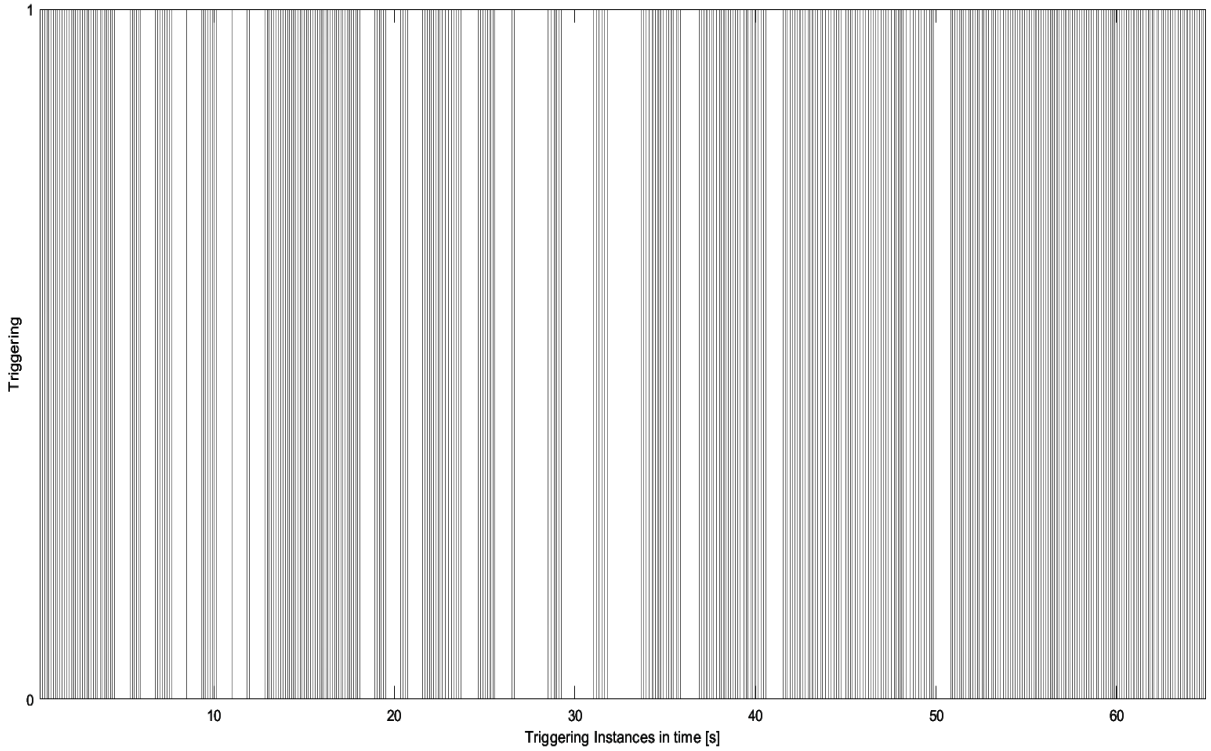

Figure 4.13 Simulation 1: Triggering Instances under the ET-NMPC scheme

4.3.2 Simulation 2

For simulation 2, the initial configuration, denoted as $p_{i,2}$, is shown in Figure 4.14 and described below:

$$p_{i,2} = [u_1, v_1, u_2, v_2, u_3, v_3, u_4, v_4] = [524, 465, 681, 267, 711, 291, 554, 480] \tag{4.6}$$

In this scenario, the α constant for the IBVS-NMPC and ET-IBVS-NMPC cases is considered equal and is presented below:

$$\begin{aligned} \dot{v}_{des}(t) &= \alpha_{IBVS-NMPC} = 30 \quad \text{pixels/sec} \\ \dot{v}_{des}(t) &= \alpha_{ET-IBVS-NMPC} = 30 \quad \text{pixels/sec} \end{aligned} \tag{4.7}$$

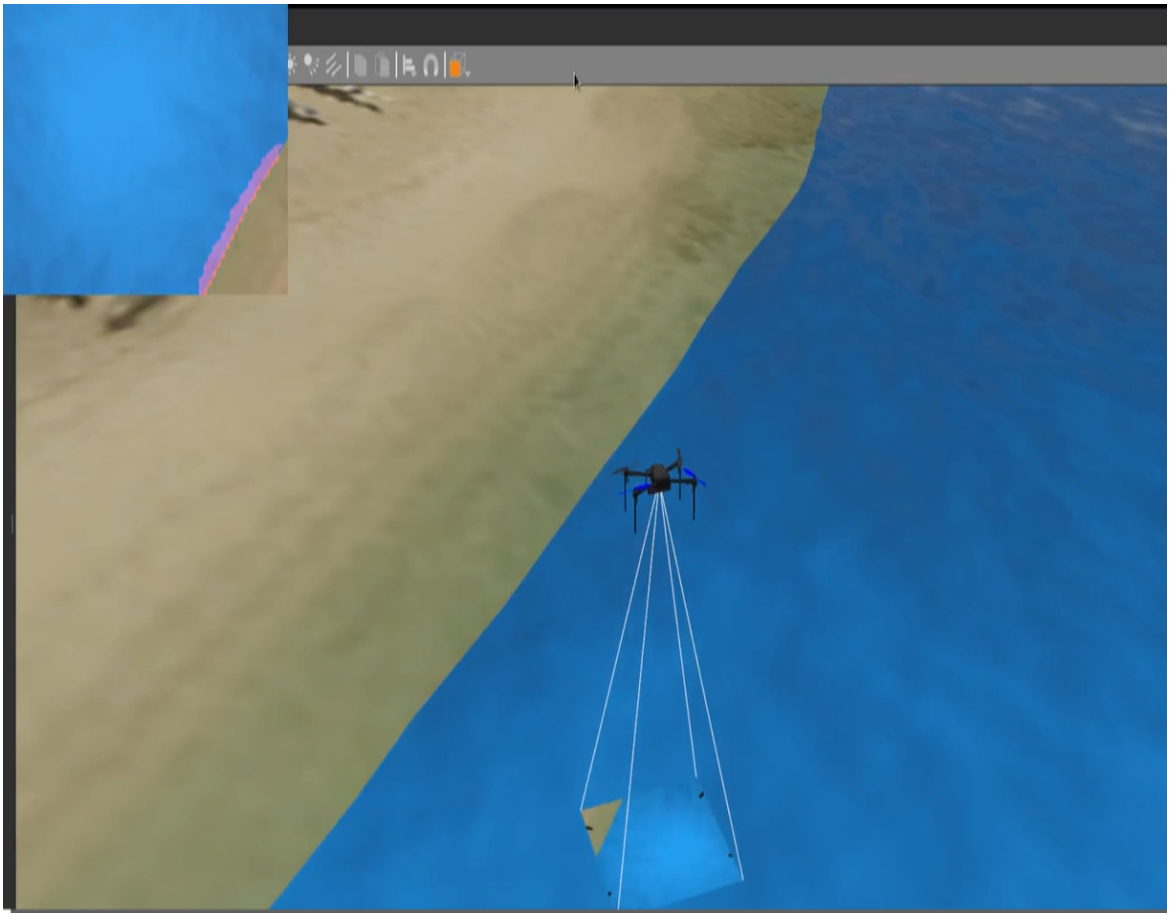

Figure 4.14 Simulation 2: Quadrotor Initial Configuration

The desired configuration $p_{i,des}$ is the same as in Simulation 1 and is presented below.

$$\begin{aligned} p_{i,des} &= [u_{1,des}, v_{1,des}, u_{2,des}, v_{2,des}, u_{3,des}, v_{3,des}, u_{4,des}, v_{4,des}] \\ &= [340, 480, 340, 0, 380, 0, 380, 480] \end{aligned} \tag{4.8}$$

In Figures 4.15, 4.16, 4.17 and 4.18, the image error evolution for feature 1,2,3 and 4 respectively is depicted under both the NMPC and ET-NMPC schemes.

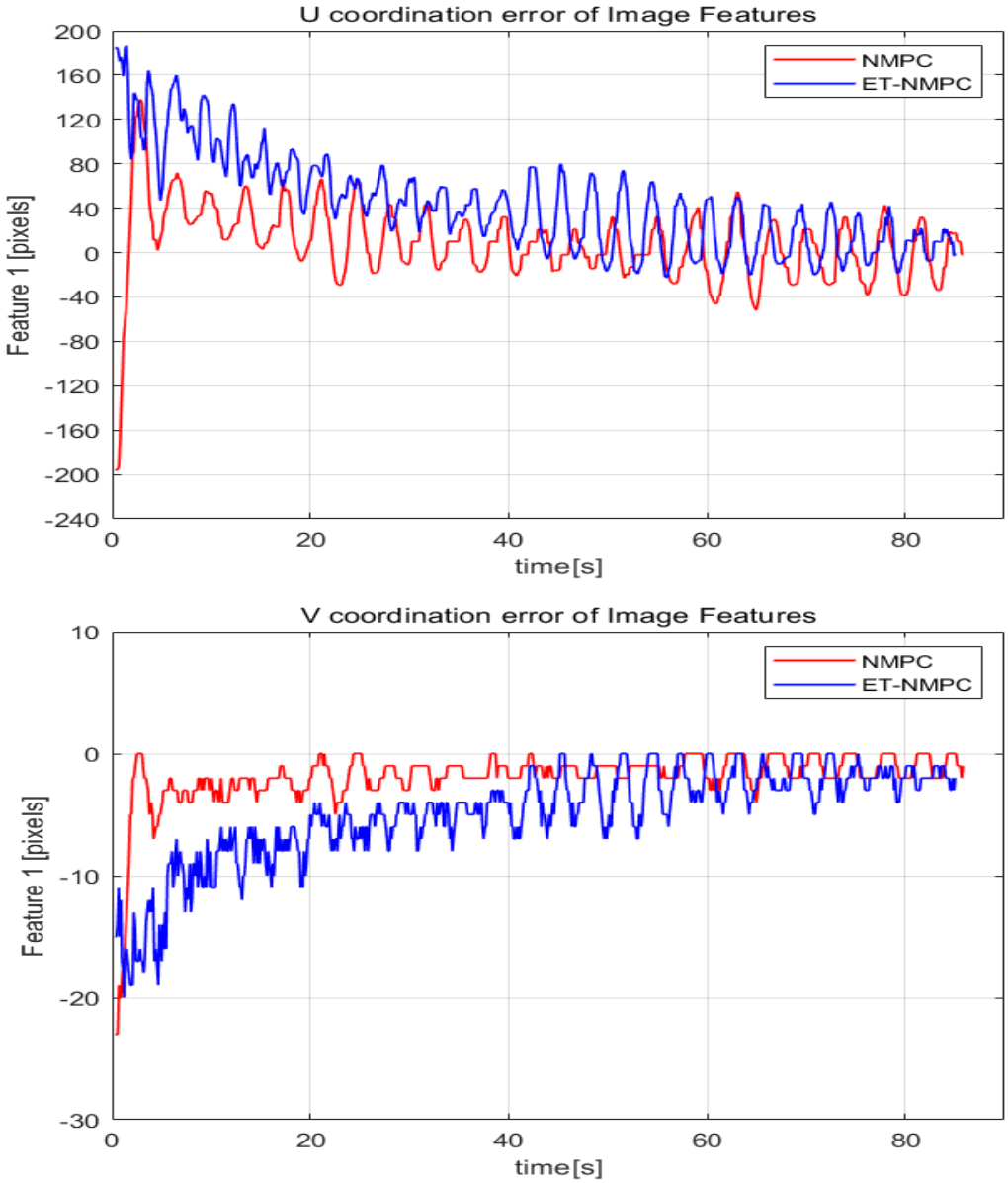

Figure 4.15 Simulation 2-Feature 1: The feature coordination errors under the NMPC and ET-NMPC

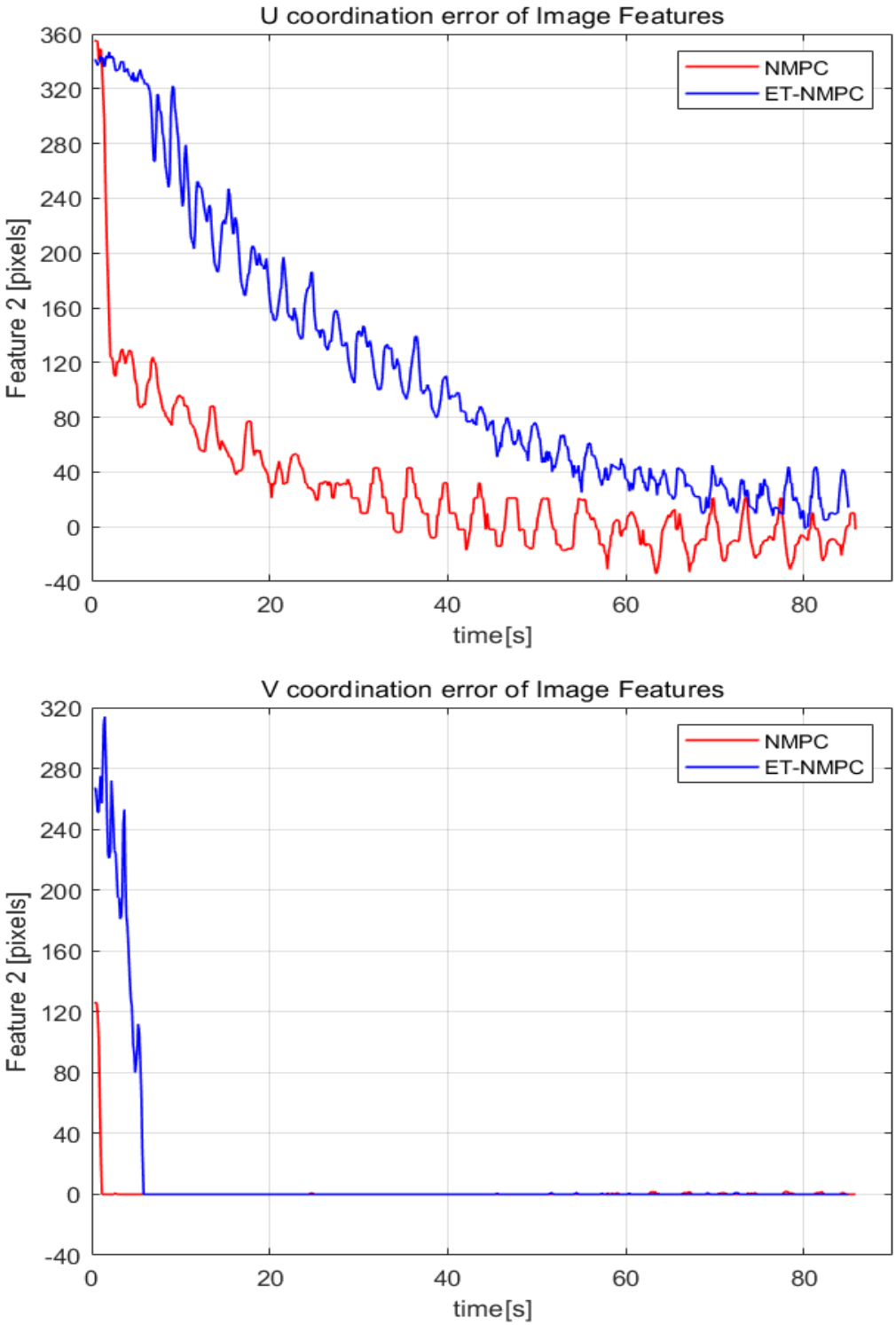

Figure 4.16 Simulation 2-Feature 2: The feature coordination errors under the NMPC and ET-NMPC

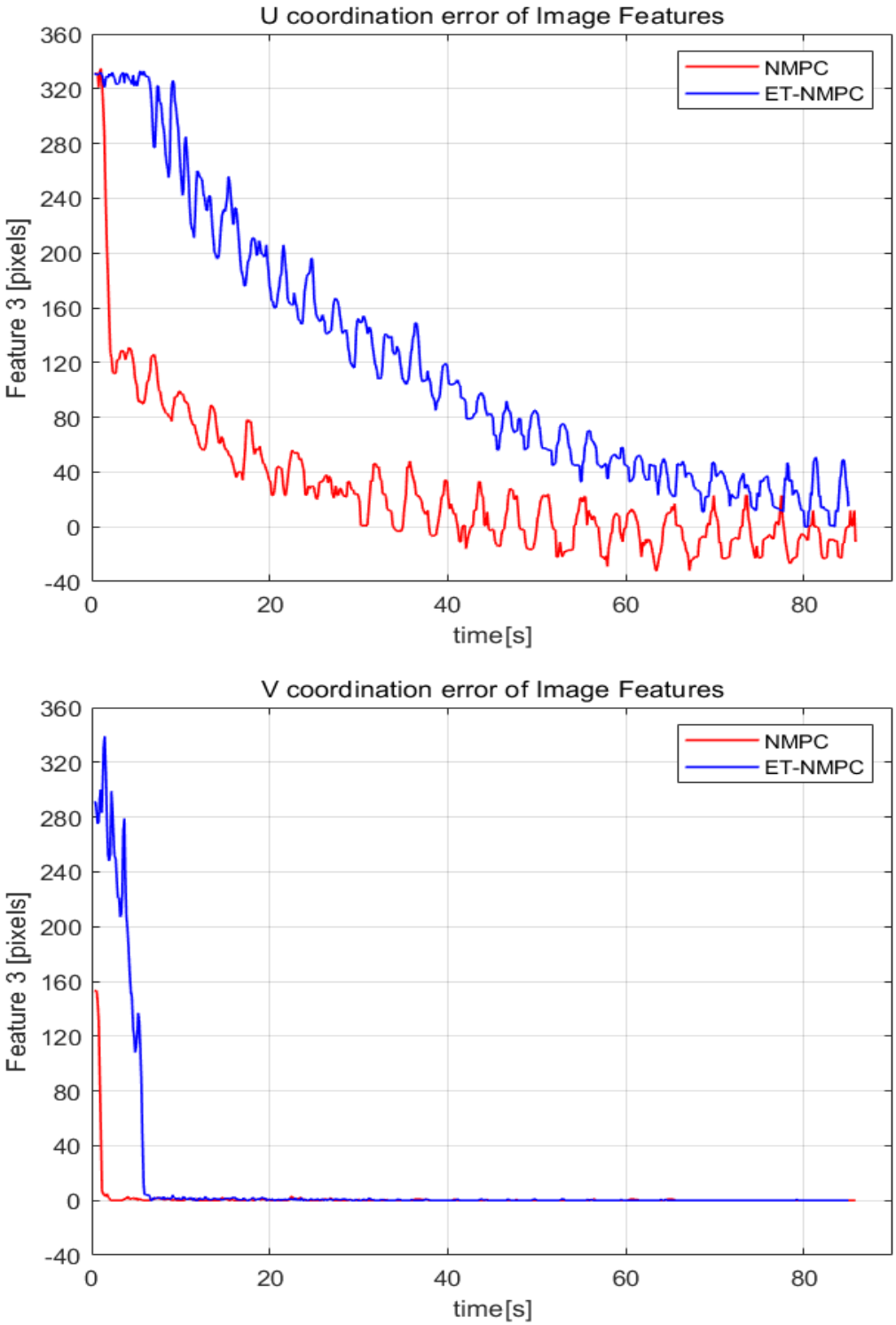

Figure 4.17 Simulation 2-Feature 3: The feature coordination errors under the NMPC and ET-NMPC

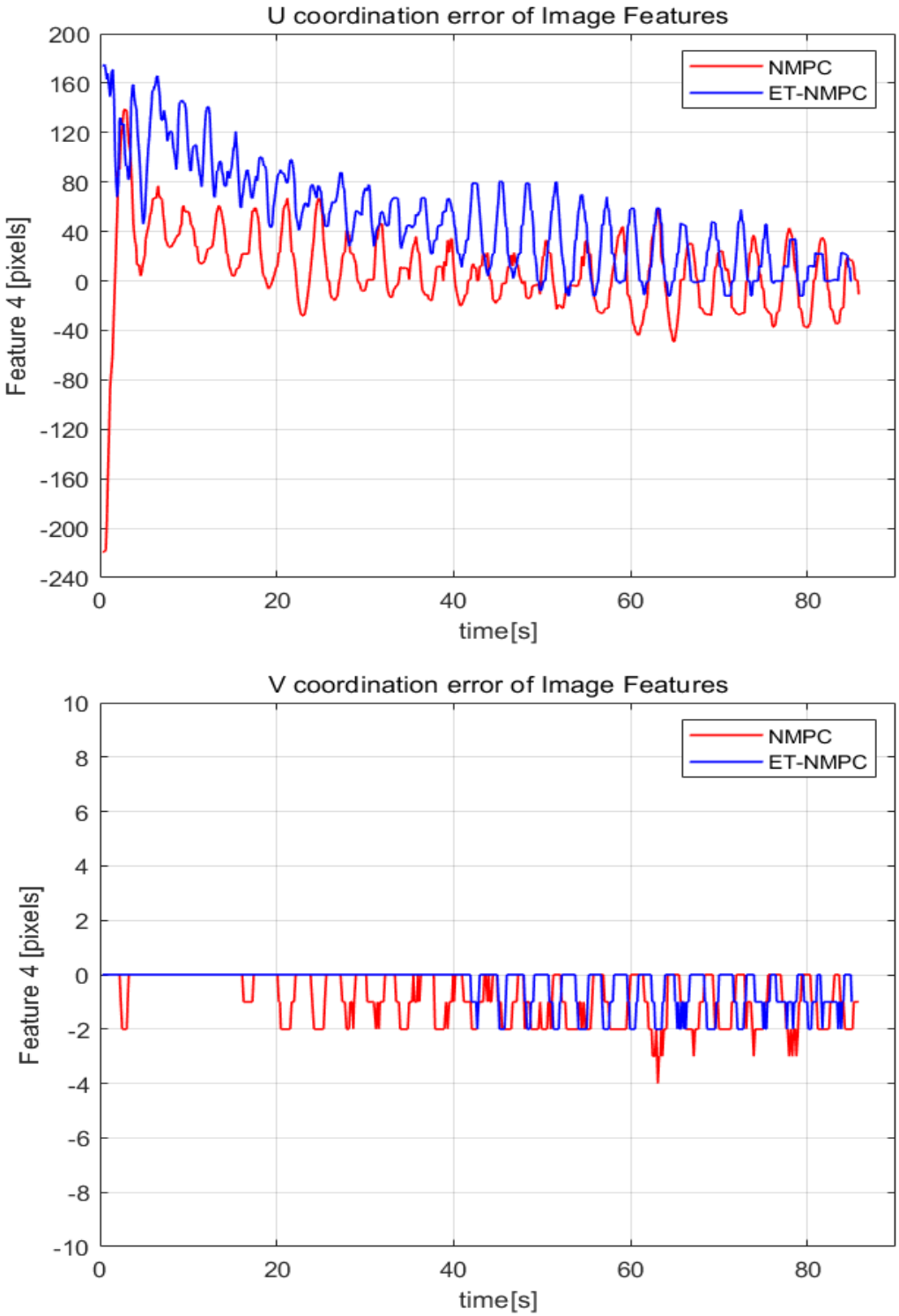

Figure 4.18 Simulation 2-Feature 4: The feature coordination errors under the NMPC and ET-NMPC

The v feature coordination error converges to 0 for all the features and under both control schemes. For the ET-NMPC, the u feature coordination error for features 1 and 4 also converges to 0 in a sinusoidal fashion and its steady state absolute value is bounded by approximately 20 pixels. Features 2 and 3 follow a similar converging trajectory and their steady state oscillates between approximately 0 and 50 pixels. Similarly, the u feature coordination errors for features 1 and 4 under the NMPC approach follow a similar trajectory to those under the ET-NMPC scheme. On the other hand, features 2 and 3 converge to the steady state at a significantly slower rate and are characterized by larger steady state errors. Specifically, the steady state for the NMPC approach is reached at approximately 45 seconds, while under the ET-NMPC scheme the system stabilized at the steady state at 65 seconds.

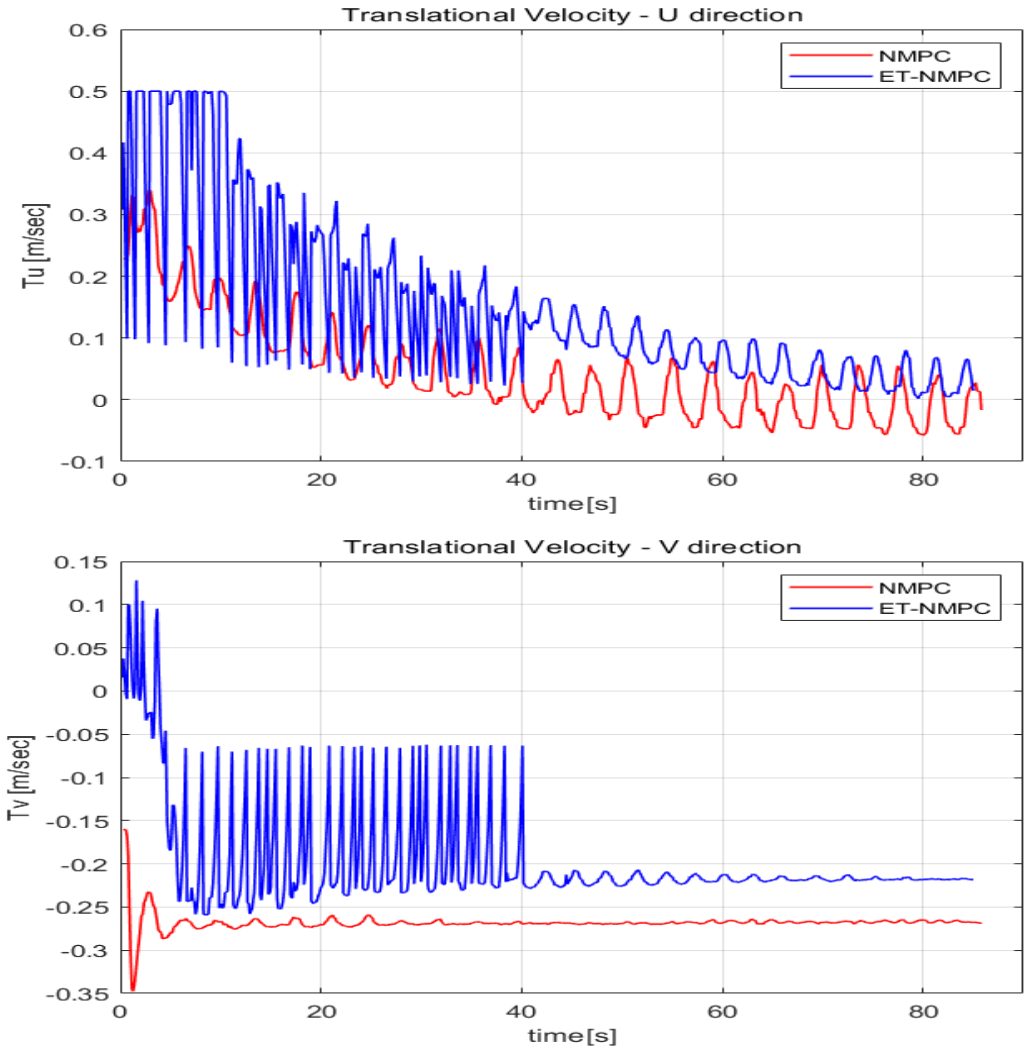

Figure 4.19 Simulation 2: The U and V Translational Camera Velocities under the NMPC and ET-NMPC

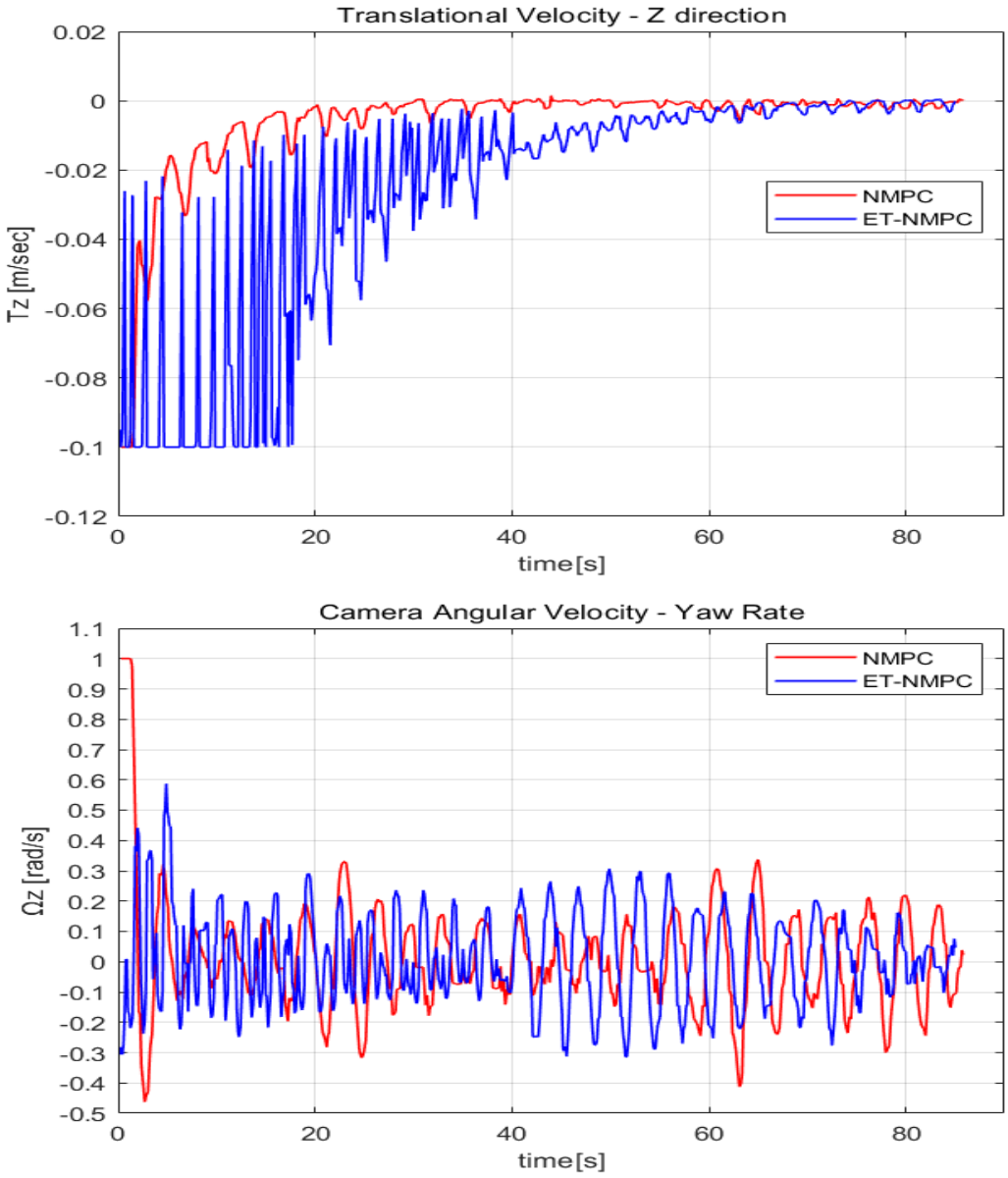

Figure 4.20 Simulation 2: The Z Translational and Ω_z Angular Camera Velocities under the NMPC and ET-NMPC

The velocities of the camera are presented in Figures 4.19 and 4.20. As we can see, the velocities satisfy the input constraints throughout the flight in this simulation as well. In both approaches, the velocities are characterized by similar oscillatory fashioned trajectories with differences in the amplitude. More specifically, in the ET-NMPC approach, the velocities showcase oscillations with larger amplitude and reach the steady state at a slower rate.

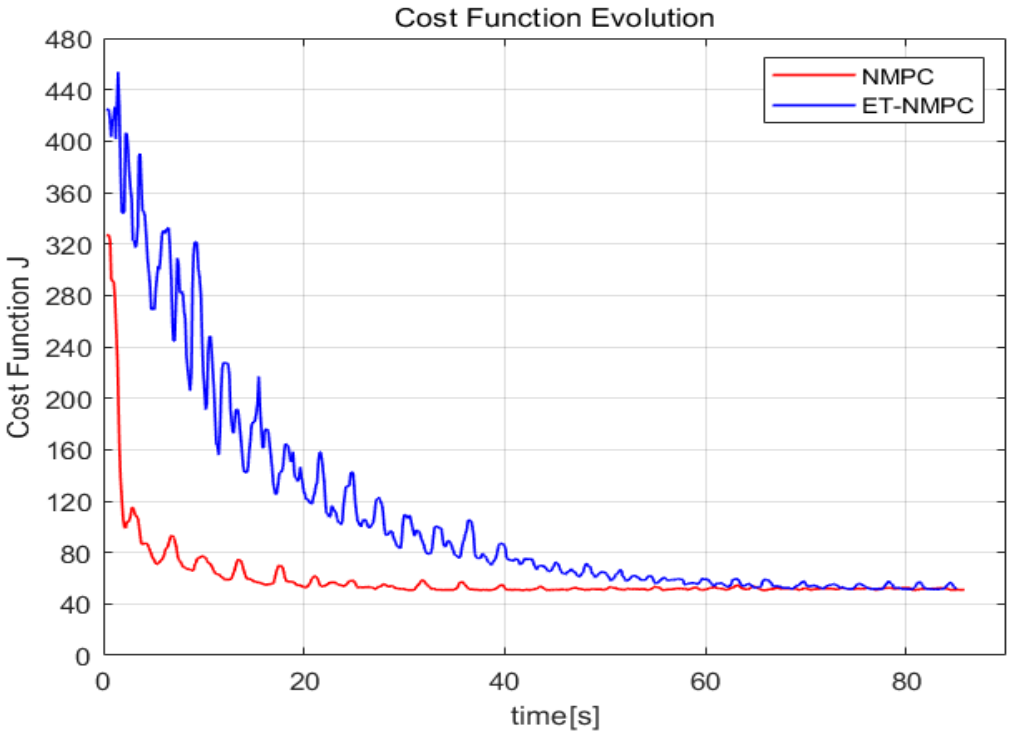

Figure 4.21 Simulation 2: The evolution of the Cost Function under the NMPC and ET-NMPC

The evolution of the cost function is presented in Figure 4.21. We can see that the cost function in the NMPC case converges to its minimum value at approximately 20 seconds while in the ET-NMPC case, it takes approximately 60 seconds to reach the steady-state. The steady-state value of the Cost Function is the same for both the ET-NMPC and NMPC case due to the same α desired velocity of the v feature coordinates. Similar to Simulation 1, very small differences between the two schemes are also seen in the evolution of the quadrotor altitude, which is presented in Figure 4.22. In the ET-NMPC approach a total increase of about 1.5 meters is seen while the altitude in the NMPC scheme remains almost stagnant. Similarity to Simulation 1 is also noticed in the optimisation loop execution duration, which is presented in Figure 4.23 and remains below 25 milliseconds.

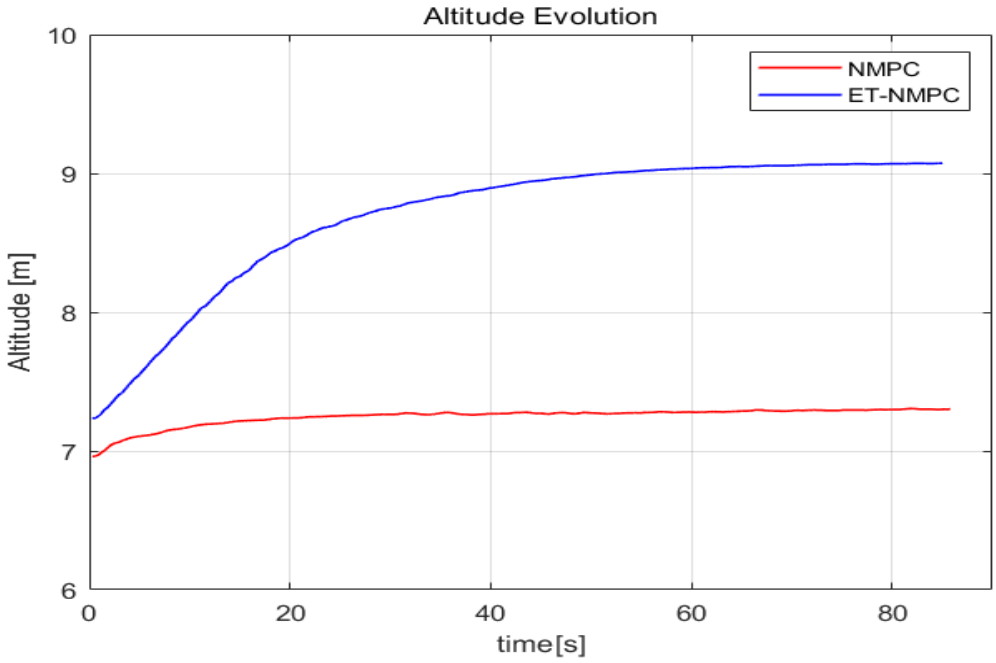

Figure 4.22 Simulation 2: The altitude of the quadrotor under the NMPC and ET-NMPC

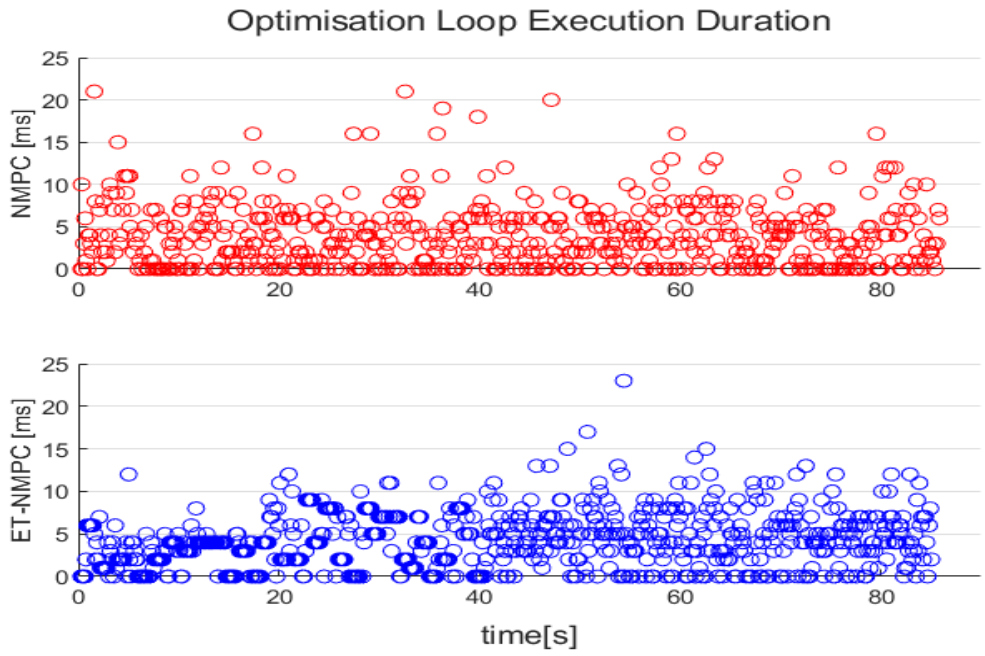

Figure 4.23 Simulation 2: Optimisation Loop Execution Duration under the NMPC and ET-NMPC

To conclude, in Figure 4.24 the triggering of the optimisation algorithm in Simulation 2 are depicted. Similarly to Simulation 1, events become more frequent near the desired states. The triggering of the MPC scheme is reduced by 28.6% with 607 triggering instances occuring instead of 850, as it happens in the classic NMPC scenario.

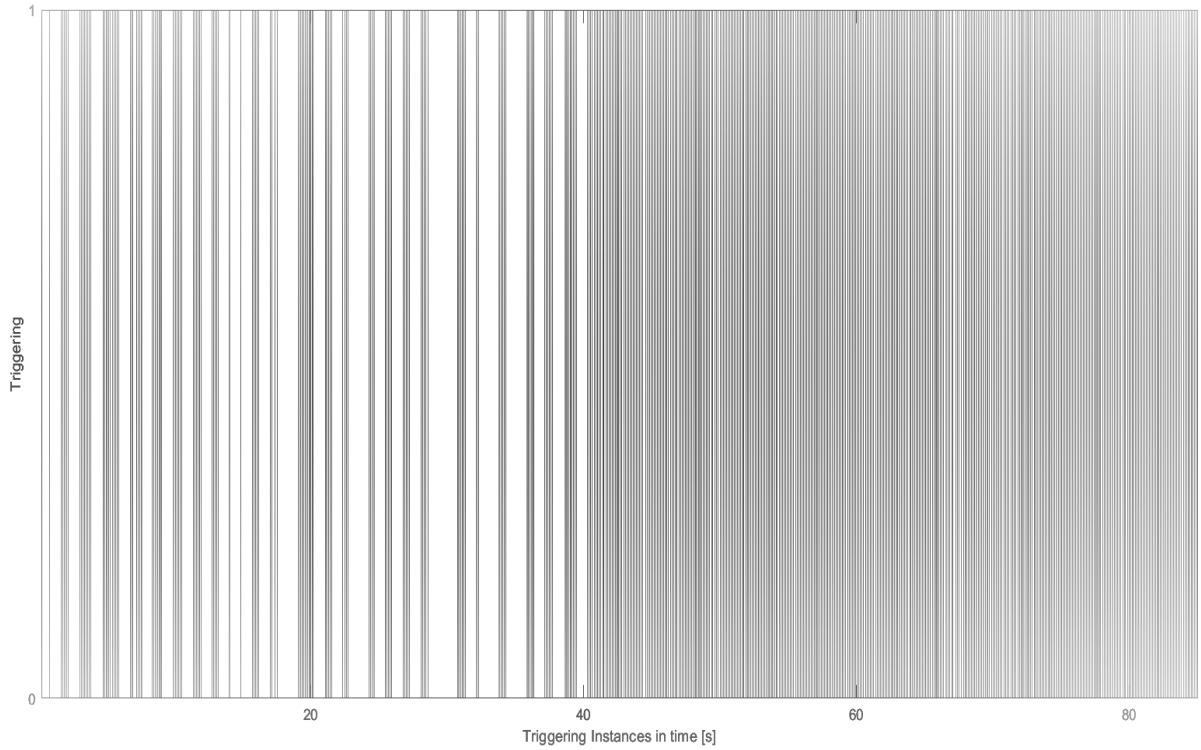

Figure 4.24 Simulation 2: Triggering Instances under the ET-NMPC scheme

Chapter 5

Conclusions

5.1 Assessment

In this thesis, we presented an Event-Triggered-IBVS-NMPC strategy for coastline tracking using a quadrotor. At first, a CNN was employed to detect the coastline and generate the ROI which encompasses it. The four corners of the detected ROI were used as image features to be used in the IBVS scheme. The IBVS framework was incorporated within an MPC problem in order to track the coastline under the effect of waves while ensuring the satisfaction of input and FOV constraints. A C++ code , incorporating the aforementioned parts of the control scheme, was developed and tested in a synthetic simulation environment. The simulations comparing the Event-Triggered NMPC scheme to the Classic MPC were conducted using ROS and Gazebo, in which the efficiency and performance of the scheme was demonstrated. It was shown that under the ET-NMPC formulation, the scheme showcased similar performance with relation to the image feature coordination errors while triggering the solution of the OCP less frequently. As a result, less energy and computational effort was required. Furthermore, relatively small changes in the parameter values of the waves in the Gazebo environment did not cause considerable difference in the u feature coordination errors.

5.2 Future Work

The ET-IBVS-NMPC for coastline tracking is an interesting research topic with many possible experimental applications. An interesting future direction includes the investigation of the robustness of the scheme to different levels of disturbances and uncertainty. Furthermore, it would be interesting to explore scenarios in more challenging environmental conditions where the effect of wind, low visibility due to fog and waves of higher magnitude are considered. At last, an interesting future research option includes the experimental validation of this approach in a real-coastline environment using an on-board computational unit, in which the autonomy of the quadrotor under the ET-IBVS-NMPC scheme can be tested by measuring the maximum possible flight duration of the quadrotor.

Bibliography

[1] L. Villani G. Oriolo B. Siciliano, L. Sciavicco. *Robotics - Modelling, Planning and Control, Visual Servoing*. Springer London, London, 2009.

[2] J. M. McCarthy, L. D. Earnest, D. R. Reddy, and P. J. Vicens. A computer with hands, eyes, and ears. In *American Federation of Information Processing Societies: Proceedings of the AFIPS '68 Fall Joint Computer Conference*, volume 33, pages 329–338, Thomson Book Company–Washington DC, San Fransisco, California, USA, 1968.

[3] L. L. Sutro and W. L. Kilmer. Assembly of computers to command and control a robot. In *Proceedings of the May 14-16, 1969, Spring Joint Computer Conference*, volume 33 of *AFIPS '69 (Spring)*, pages 113–137, Association for Computing Machinery, Boston, Massachusetts, USA, 1969.

[4] M. Ejiri et al. Development of an adaptive robot with visual sensor. In *Tokyo Conference of the Institute of Energy Economics IEEJ*, Tokyo, Japan, 1970.

[5] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5(2):99–108, 1973.

[6] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The confluence of vision and control*, pages 66–78, Springer London, London, 1998.

[7] N. Guenard, T. Hamel, and R. Mahony. A practical visual servo control for an unmanned aerial vehicle. *IEEE Transactions on Robotics*, 24(2):331–340, 2008.

[8] M. Kazemi, K. Gupta, and M. Mehrandezh. Global path planning for robust visual servoing in complex environments. In *2009 IEEE International Conference on Robotics and Automation*, pages 326–332, 2009.

[9] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4):534–549, 2002.

[10] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics Automation Magazine*, 13(4):82–90, 2006.

[11] F. Chaumette and S. Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *IEEE Robotics Automation Magazine*, 14(1):109–118, 2007.

[12] C. Kanellakis and G. Nikolakopoulos. Survey on computer vision for uavs: Current developments and trends. *Journal of Intelligent Robotic Systems*, 87, 07 2017.

[13] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck. Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle. *IEEE Transactions on Robotics*, 25(3):743–749, 2009.

[14] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti, and S. Longhi. A single-camera feature-based vision system for helicopter autonomous landing. pages 1 – 6, 07 2009.

[15] Erdinç A., J. P. Ostrowski, and C. J. Taylor. Control of a quadrotor helicopter using dual camera visual feedback. *The International Journal of Robotics Research*, 24(5):329–341, 2005.

[16] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar. Toward image based visual servoing for aerial grasping and perching. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2113–2118, 2014.

[17] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar. Visual servoing of quadrotors for perching by hanging from cylindrical objects. *IEEE Robotics and Automation Letters*, 1:1–1, 01 2015.

[18] G. Loianno, C. Brunner, G. McGrath, and V. Kumar. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. *IEEE Robotics and Automation Letters*, 2(2):404–411, 2017.

[19] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette. Vision-based minimum-time trajectory generation for a quadrotor uav. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6199–6206, 2017.

[20] B. Penin, P. R. Giordano, and F. Chaumette. Minimum-time trajectory planning under intermittent measurements. *IEEE Robotics and Automation Letters*, 4(1):153–160, 2019.

[21] H. Jabbari, G. Oriolo, and H. Bolandi. Dynamic ibvs control of an underactuated uav. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1158–1163, 2012.

[22] M. Sauvee, P. Poignet, E. Dombre, and E. Courtial. Image based visual servoing through nonlinear model predictive control. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1776–1781, 2006.

[23] G. Allibert, E. Courtial, and Y. Touré. A flat model predictive controller for trajectory tracking in image based visual servoing. *IFAC Proceedings Volumes, 7th IFAC Symposium on Nonlinear Control Systems*, 40(12):993–998, 2007.

[24] G. Allibert, E. Courtial, and Y. Touré. Real-time visual predictive controller for image-based trajectory tracking of a mobile robot. *IFAC Proceedings Volumes*, 41:11244–11249, 2008.

[25] G. Allibert, E.Courtial, and F. Chaumette. Visual servoing via nonlinear predictive control. *G. Chesi and K. Hashimoto, Visual Servoing via Advanced Numerical Methods, LNCIS 401, Springer- Verlag*, pages 375–394, 2010.

[26] G. Allibert, E. Courtial, and F. Chaumette. Predictive control for constrained image-based visual servoing. *IEEE Transactions on Robotics*, 26(5):933–939, 2010.

[27] P. Roque, E. Bin, P. Miraldo, and D. Dimarogonas. Fast model predictive image-based visual servoing for quadrotors*. pages 7566–7572, 10 2020.

[28] D. Lee, H. Lim, and H. J. Kim. Obstacle avoidance using image-based visual servoing integrated with nonlinear model predictive control. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 5689–5694, 2011.

[29] K.-H. Chen and W.-H. Tsai. Vision-based autonomous land vehicle guidance in outdoor road environments using combined line and road following techniques. *Journal of Robotic Systems*, 14(10):711–728, 1997.

[30] R. Hartley, B. Kamgar-Parsi, and C. Narber. Using roads for autonomous air vehicle guidance. *IEEE Transactions on Intelligent Transportation Systems*, 19(12):3840–3849, 2018.

[31] E. Frew, T. McGee, ZuWhan K., Xiao X., S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta. Vision-based road-following using a small autonomous aircraft. In *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720)*, volume 5, pages 3006–3015 Vol.5, 2004.

[32] S. Rathinam, P. Almeida, Z. Kim, S. Jackson, A. Tinka, W. Grossman, and R. Sengupta. Autonomous searching and tracking of a river using an uav. In *2007 American Control Conference*, pages 359–364, 2007.

[33] R. Frezza, G. Picci, and S. Soatto. *A lagrangian formulation of nonholonomic path following*, volume 237, pages 118–133. 11 2007.

[34] P. Baker and B. Kamgar-Parsi. Using shorelines for autonomous air vehicle guidance. *Computer Vision and Image Understanding*, 114:723–729, 06 2010.

[35] Anqi X. and G. Dudek. A vision-based boundary following framework for aerial vehicles. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 81–86, 2010.

[36] C-S Lee and F-B Hsiao. Implementation of vision-based automatic guidance system on a fixed-wing unmanned aerial vehicle. *The Aeronautical Journal (1968)*, 116(1183):895–914, 2012.

[37] C. Bordons E. F. Camacho. *Model Predictive Control in the Process Industry*. Springer-Verlag London, London, 1995.

[38] J. M. Maciejowski. *Predictive Control: With Contraints*. Pearson Education, London, 2002.

[39] S. Han W. H. Kwon. *Receding Horizon Control: Model Predictive Control for State Models*. Springer-Verlag London, London, 2005.

[40] S. Han W. H. Kwon. *Receding Horizon Control: Model Predictive Control for State Models*. Springer-Verlag London, London, 2005.

[41] M. Cannon B. Kouvaritakis. *Model predictive control*. Springer International Publishing, Switzerland, 2016.

[42] M. Morari F. Borelli, A. Bemporad. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, Cambridge, 2017.

[43] J. Pannek L. Grüne. *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer Science Business Media, Berlin, 2011.

[44] F. Allgöwer L. Magni, D. M. Raimondo. *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Springer Science Business Media, Berlin, 2009.

[45] K. J. Åström and B. Bernhardsson. Comparison of periodic and event based sampling for first-order stochastic systems. *IFAC Proceedings Volumes*, 32(2):5006–5011. 14th IFAC World Congress, Beijing, China, 5-9 July, 1999.

[46] K.J. Astrom and B.M. Bernhardsson. Comparison of riemann and lebesgue sampling for first order stochastic systems. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 2, pages 2011–2016 vol.2, 2002.

[47] K. Johan Åström. *Event Based Control*, pages 127–147. Springer, Germany, 2008.

[48] D. Hristu-Varsakelis and P.R. Kumar. Interrupt-based feedback control over a shared communication medium. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pages 3223–3228 vol.3, 2002.

[49] K.-E. Åarzén. A simple event-based pid controller. *IFAC Proceedings Volumes*, 32(2):8687–8692. 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July, 1999.

[50] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

[51] W. P. M. H. Heemels, J. H. Sandee, and P. P. J. Van Den Bosch. Analysis of event-driven controllers for linear systems. *International Journal of Control*, 81(4):571–590, 2008.

[52] W.P.M.H. Heemels, K.H. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3270–3285, 2012.

[53] X. Wang and M. D. Lemmon. Event design in event-triggered feedback control systems. In *2008 47th IEEE Conference on Decision and Control*, pages 2105–2110, 2008.

[54] P. Tabuada and X. Wang. Preliminary results on state-trigered scheduling of stabilizing control tasks. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 282–287, 2006.

[55] M. Lemmon, T. Chantem, X. Hu, and M. Zyskowski. On self-triggered full-information h-infinity controllers. pages 371–384, 04 2007.

[56] D. Lehmann and J. Lunze. Event-based control: A state-feedback approach. In *2009 European Control Conference (ECC)*, pages 1716–1721, 2009.

[57] M. Velasco, P. Colom, and C. Lozoya. On the timing of discrete events in event-driven control systems. volume 4981, April 2008.

[58] L. Li and M. Lemmon. Event-triggered output feedback control of finite horizon discrete-time multi-dimensional linear processes. In *49th IEEE Conference on Decision and Control (CDC)*, pages 3221–3226, 2010.

[59] L. Grüne, S. Jerg, O. Junge, D. Lehmann, J. Lunze, F. Müller, and M. Post. Two complementary approaches to event-based control zwei komplementäre zugänge zur ereignisbasierten regelung. *at - Automatisierungstechnik*, 58(4):173–182, 2010.

[60] W.P.M.H. Heemels and M.C.F. Donkers. Model-based periodic event-triggered control for linear systems. *Automatica*, 49(3):698–711, 2013.

[61] S. Di Cairano, A. Bemporad, and J. Júlvez. Event-driven optimization-based control of hybrid systems with integral continuous-time dynamics. *Automatica*, 45:1243–1251, 05 2009.

[62] M. Miskowicz. Send-on-delta concept: An event-based data reporting strategy. *Sensors*, 6, 01 2006.

[63] M. Mazo and P. Tabuada. Decentralized event-triggered control over wireless sensor/actuator networks. *IEEE Transactions on Automatic Control*, 56(10):2456–2461, 2011.

[64] M. Mazo and M. Cao. Decentralized event-triggered control with asynchronous updates. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 2547–2552, 2011.

[65] X. Wang and M. D. Lemmon. Event-triggered broadcasting across distributed networked control systems. In *2008 American Control Conference*, pages 3139–3144, 2008.

[66] X. Wang and M. D. Lemmon. Event-triggering in distributed networked control systems. *IEEE Transactions on Automatic Control*, 56(3):586–601, 2011.

[67] D. V. Dimarogonas and K. H. Johansson. Event-triggered cooperative control. In *2009 European Control Conference (ECC)*, pages 3015–3020, 2009.

[68] D. V. Dimarogonas and K. H. Johansson. Event-triggered control for multi-agent systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 7131–7136, 2009.

[69] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson. Distributed self-triggered control for multi-agent systems. In *49th IEEE Conference on Decision and Control (CDC)*, pages 6716–6721, 2010.

[70] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson. Event-based broadcasting for multi-agent average consensus. *Automatica*, 49(1):245–252, 2013.

[71] A. Bemporad, S. Di Cairano, and J. Júlvez. Event-based model predictive control and verification of integral continuous-time hybrid automata. volume 3927, pages 93–107, 04 2006.

[72] L. Grüne and F. Müller. An algorithm for event-based optimal feedback control. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 5311–5316, 2009.

[73] D. Bernardini and A. Bemporad. Energy-aware robust model predictive control based on wireless sensor feedback. In *2008 47th IEEE Conference on Decision and Control*, pages 3342–3347, 2008.

[74] L. Grüne, J. Pannek, and K. Worthmann. A prediction based control scheme for networked systems with delays and packet dropouts. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 537–542, 2009.

[75] P. Ge, B. Chen, and F. Teng. Event-triggered distributed model predictive control for resilient voltage control of an islanded microgrid. *International Journal of Robust and Nonlinear Control*, 31(6):1979–2000, 2021.

[76] D.L. Marruedo, T. Alamo, and E.F. Camacho. Input-to-state stable mpc for constrained discrete-time nonlinear systems with bounded additive uncertainties. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 4, pages 4619–4624 vol.4, 2002.

[77] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos. Novel event-triggered strategies for model predictive controllers. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 3392–3397, 2011.

[78] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos. Event-triggered control for discrete-time systems. In *Proceedings of the 2010 American Control Conference*, pages 4719–4724, 2010.

[79] J. Sun, M. Wang, and J. Chen. Event-based model predictive control of discrete-time nonlinear systems with external disturbances. *IET Control Theory Applications*, 13, 10 2018.

[80] F. D. Brunner, W.P.M.H. Heemels, and F. Allgöwer. Robust event-triggered mpc for constrained linear discrete-time systems with guaranteed average sampling rate. *IFAC-PapersOnLine*, 48(23):117–122, 2015. 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015.

[81] Z. Sun, L. Dai, K. Liu, Y. Xia, and K. H. Johansson. Robust mpc for tracking constrained unicycle robots with additive disturbances. *Automatica*, 90:172–184, 2018.

[82] Z. Cai, H. Zhou, J. Zhao, K. Wu, and Y. Wang. Formation control of multiple unmanned aerial vehicles by event-triggered distributed model predictive control. *IEEE Access*, 6:55614–55627, 2018.

[83] D. Jang, C. Y. Son, J. Yoo, H. J. Kim, and K. H. Johansson. Efficient networked uav control using event-triggered predictive control. *IFAC*, 52(15):412–417, 2019. 8th IFAC Symposium on Mechatronic Systems Mechatronics 2019.

[84] J.F. Guerrero-Castellanos, A. Vega-Alonzo, N. Marchand, S. Durand, J. Linares-Flores, and G. Mino-Aguilar. Real-time event-based formation control of a group of vtol-uavs. In *2017 3rd International Conference on Event-Based Control, Communication and Signal Processing (EBCCSP)*, pages 1–8, 2017.

[85] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics Automation Magazine*, 19(3):20–32, 2012.

[86] M. Lazar, W. P. M. H. Heemels, and A. R. Teel. Further input-to-state stability subtleties for discrete-time systems. *IEEE Transactions on Automatic Control*, 58(6):1609–1613, 2013.

[87] A. Eqtami. *Event-Based Model Predictive Controllers*. PhD thesis, School of Mechanical Engineering, National Technical University of Athens, Athens, Greece, 2013.

[88] Gazebo. Available at http://gazebosim.org/.

[89] D. Hinsinger, F. Neyret, and M.-P. Cani. Interactive animation of ocean waves. page 161, 01 2002.

[90] B. Bingham, C. Agüero, M. McCarrin, J. Klamo, J. Malia, K. Allen, T. Lum, M. Rawson, and R. Waqar. Toward maritime robotic simulation in gazebo. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–10, 2019.

[91] J. Tessendorf. Simulating ocean water. *SIG-GRAPH'99 Course Note*, 01 2001.

[92] 3DR Iris Quadrotor. Iris. Available at https://www.arducopter.co.uk/iris-quadcopter-uav.html.

[93] ROS. Robot operating system. Available at https://www.ros.org/.

[94] ROS Melodic. Robot operating system. Available at http://wiki.ros.org/melodic.

[95] Ubuntu. Canonical. Available at https://ubuntu.com/.

[96] Mavros. Available at http://wiki.ros.org/mavros.

[97] MAVLINK. Micro air vehicle communication protocol. Available at https://mavlink.io/en/.

[98] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui. Micro air vehicle link (mavlink) in a nutshell: A survey. *IEEE Access*, 7:87658–87680, 2019.

[99] Mavlink micro air vehicle communication protocol. Available at http://qgroundcontrol.org/mavlink/start.

[100] Eigen. Available at https://eigen.tuxfamily.org/index.php?title=Main_Page.

[101] NLopt. Nonlinear optimization library. Available at https://nlopt.readthedocs.io/en/latest/.

[102] M. Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Technical Report, Department of Applied Mathematics and Theoretical Physics*, 01 2009.

[103] OpenCV. Open source computer vision library. Available at https://opencv.org/.

[104] Keras. Python open-source software library for artificial neural networks. Available at https://keras.io/.

[105] ArduPilot. Open-source autopilot software system. Available at https://ardupilot.org/.

[106] Event-Triggered IBVS-NMPC Gazebo Simulation Flight Recordings Mario Sinani. Available at shorturl.at/iuPQ1.

Appendix A

Lipschitz Constants

For the Lipschitz constant, we need to express our system function as:

$$f(s_k,V) = \begin{bmatrix} x_k \\ \gamma_k \end{bmatrix} + \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x_k}{Z} & \gamma_k \\ 0 & \frac{-1}{Z} & \frac{\gamma_k}{Z} & -x_k \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \\ \Omega_z \end{bmatrix} dt =$$

$$\begin{bmatrix} x_k \\ \gamma_k \end{bmatrix} + \begin{bmatrix} -\frac{T_x}{Z}dt + \frac{x_k T_z}{Z}dt + \gamma_k \Omega_z dt \\ -\frac{T_y}{Z}dt + \frac{\gamma_k T_z}{Z}dt - x_k \Omega_z dt \end{bmatrix} =$$

$$\begin{bmatrix} x_k - \frac{T_x}{Z}dt + \frac{x_k T_z}{Z}dt + \gamma_k \Omega_z dt \\ \gamma_k - \frac{T_y}{Z}dt + \frac{\gamma_k T_z}{Z}dt - x_k \Omega_z dt \end{bmatrix} \iff$$

$$f(s_k,V) = \begin{bmatrix} x_k(1+\frac{T_z}{Z}dt) - \frac{T_x}{Z}dt + \gamma_k \Omega_z dt \\ \gamma_k(1+\frac{T_z}{Z}dt) - \frac{T_y}{Z}dt - x_k \Omega_z dt \end{bmatrix}$$

For the euclidean norm, we can express the difference as:

$$||f(s_1,V) - f(s_2,V)||^2 = \left\| \begin{matrix} (x_1-x_2)(1+\frac{T_z}{Z}dt) + (\gamma_1-\gamma_2)\Omega_z dt \\ (\gamma_1-\gamma_2)(1+\frac{T_z}{Z}dt) - (x_1-x_2)\Omega_z dt \end{matrix} \right\|^2 =$$

$$|(x_1-x_2)(1+\frac{T_z}{Z}dt) + (\gamma_1-\gamma_2)\Omega_z dt|^2 + |(\gamma_1-\gamma_2)(1+\frac{T_z}{Z}dt) - (x_1-x_2)\Omega_z dt|^2$$

From the first part we can derive:

$$|(x_1 - x_2)(1 + \frac{T_z}{Z}dt) + (\gamma_1 - \gamma_2)\Omega_z dt|^2 \leq$$

$$[2max\{(1 + \frac{T_z}{Z}dt)|x_1 - x_2|, \Omega_z dt|\gamma_1 - \gamma_2|\}]^2 \leq$$

$$4max\{(1 + \frac{T_z}{Z}dt)^2(x_1 - x_2)^2, (\Omega_z dt)^2(\gamma_1 - \gamma_2)^2\} \leq$$

$$max\{4(1 + \frac{T_z}{Z}dt)^2, 4(\Omega_z dt)^2\}max\{(x_1 - x_2)^2, (\gamma_1 - \gamma_2)^2\} \leq$$

$$max\{4(1 + \frac{\overline{T_z}}{Z}dt)^2, 4(\overline{\Omega_z}dt)^2\}[(x_1 - x_2)^2 + (\gamma_1 - \gamma_2)^2]$$

Similarly, we can derive:

$$|(\gamma_1 - \gamma_2)(1 + \frac{T_z}{Z}dt) - (x_1 - x_2)\Omega_z dt|^2 =$$

$$|(\gamma_1 - \gamma_2)(1 + \frac{T_z}{Z}dt) + (x_2 - x_1)\Omega_z dt|^2 \leq$$

$$[2max\{(1 + \frac{T_z}{Z}dt)|\gamma_1 - \gamma_2|, \Omega_z dt|x_2 - x_1|\}]^2 \leq$$

$$4max\{(1 + \frac{T_z}{Z}dt)^2(\gamma_1 - \gamma_2)^2, (\Omega_z dt)^2(x_2 - x_1)^2\} \leq$$

$$max\{4(1 + \frac{T_z}{Z}dt)^2, 4(\Omega_z dt)^2\}max\{(\gamma_1 - \gamma_2)^2, (x_2 - x_1)^2\} \leq$$

$$max\{4(1 + \frac{\overline{T_z}}{Z}dt)^2, 4(\overline{\Omega_z}dt)^2\}[(\gamma_1 - \gamma_2)^2 + (x_2 - x_1)^2] =$$

$$max\{4(1+\frac{\overline{T_z}}{Z}dt)^2, 4(\overline{\Omega_z}dt)^2\}[(x_1-x_2)^2+(\gamma_1-\gamma_2)^2]$$

Therefore, we end up with the following expression for the Lipschitz constant:

$$||f(s_1,V)-f(s_2,V)||^2 \leq 2max\{4(1+\frac{\overline{T_z}}{Z}dt)^2, 4(\overline{\Omega_z}dt)^2\}[(x_1-x_2)^2+(\gamma_1-\gamma_2)^2] \iff$$

$$||f(s_1,V)-f(s_2,V)|| \leq [2max\{4(1+\frac{\overline{T_z}}{Z}dt)^2, 4(\overline{\Omega_z}dt)^2\}]^{\frac{1}{2}}[(x_1-x_2)^2+(\gamma_1-\gamma_2)^2]^{\frac{1}{2}}$$

$$||f(s_1,V)-f(s_2,V)|| \leq [2max\{4(1+\frac{\overline{T_z}}{Z}dt)^2, 4(\overline{\Omega_z}dt)^2\}]^{\frac{1}{2}}||s_1-s_2||$$

$$||f(s_1,V)-f(s_2,V)|| \leq C_f||s_1-s_2||$$

$$L_f = [2max\{4(1+\frac{\overline{T_z}}{Z}dt)^2, 4(\overline{\Omega_z}dt)^2\}]^{\frac{1}{2}}$$