ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Μελέτη και ανάπτυξη αλγορίθμου προβλεπτικού ελέγχου για την αυτόνομη κίνηση ρομποτικού πολυκοπτέρου

ΙΑΣΩΝΙΔΗΣ ΓΕΩΡΓΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΚΑΡΡΑΣ ΓΕΩΡΓΙΟΣ
Αναπληρωτής Καθηγητής

Λαμία ..... έτος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Μελέτη και ανάπτυξη αλγορίθμου προβλεπτικού ελέγχου για την αυτόνομη κίνηση ρομποτικού πολυκοπτέρου

ΙΑΣΩΝΙΔΗΣ ΓΕΩΡΓΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΚΑΡΡΑΣ ΓΕΩΡΓΙΟΣ
Αναπληρωτής Καθηγητής

Λαμία ..... έτος 2025

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

Study and development of a predictive control algorithm for the autonomous motion of a robotic multi-copter

IASONIDIS GEORGIOS

FINAL THESIS

ADVISOR

KARRAS GEORGIOS
Associate Professor

Lamia ..... year 2025

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: ……/…../20……

Ο – Η Δηλ.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία επικεντρώνεται στη μελέτη, ανάπτυξη και παρουσίαση σχήματος Προβλεπτικού Ελέγχου (Model Predictive Control – MPC) για τον έλεγχο θέσης και προσανατολισμού ενός πολυκοπτέρου (quadrotor). Στο **πρώτο κεφάλαιο** παρουσιάζεται το θεωρητικό υπόβαθρο του MPC, διατυπώνεται αναλυτικά το πρόβλημα ελέγχου και παρατίθενται σχετικές βιβλιογραφικές αναφορές που αναδεικνύουν την αποτελεσματικότητα του MPC στο βέλτιστο έλεγχο δυναμικών συστημάτων. Το **δεύτερο κεφάλαιο** επικεντρώνεται στη διατύπωση του προβλήματος στο πλαίσιο της συγκεκριμένης μελέτης καθώς και στην επίλυση του προβλήματος, γίνεται αναφορά στο περιβάλλον προσομοίωσης, περιλαμβάνοντας τη γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση του αλγορίθμου και το λογισμικό προσομοίωσης, ενώ παρέχεται και περιγραφή του εικονικού περιβάλλοντος (κόσμου) με το οποίο αλληλοεπιδρά το πολυκόπτερο. **Το τρίτο κεφάλαιο** περιλαμβάνει την τεχνική ανάλυση και υλοποίηση του αλγορίθμου, με παρουσίαση του κώδικα και της αρχιτεκτονικής του συστήματος. Ακολουθεί το **τέταρτο κεφάλαιο**, όπου παρουσιάζονται τα αποτελέσματα των προσομοιώσεων μέσα από γραφήματα και συγκριτικές απεικονίσεις. Τέλος, στο **πέμπτο κεφάλαιο** συνοψίζονται τα βασικά συμπεράσματα και προτείνονται μελλοντικές κατευθύνσεις.

ABSTRACT

This thesis focuses on the study, development, and implementation of a **Model Predictive Control (MPC) scheme** for the position and orientation control of a quadrotor. The **first chapter** presents the theoretical background of MPC, formulates the control problem in detail, and includes relevant literature references highlighting the effectiveness of MPC in the optimal control of dynamic systems. The **second chapter** focuses on the formulation of the specific control problem addressed in this study and its solution. It also introduces the simulation environment, including the programming language used for the implementation of the algorithm and the simulation software, while providing a description of the virtual environment (world) with which the quadrotor interacts. The **third chapter** includes the technical analysis and implementation of the algorithm, presenting both the code and the system architecture. The **fourth chapter** presents the results of the simulations through graphs and comparative visualizations. Finally, the **fifth chapter** summarizes the main conclusions and proposes future directions.

Table of Contents

Κεφάλαιο 1 - Εισαγωγή

1.1. Σκοπός της πτυχιακής εργασίας

Σκοπός της πτυχιακής εργασίας είναι **η μελέτη και η υλοποίηση** ενός σχήματος προβλεπτικού ελέγχου (Model Predictive Control - MPC) για τον ακριβή έλεγχο της θέσης και του προσανατολισμού ενός πολυκοπτέρου (quadrotor) σε περιβάλλον προσομοίωσης. Η εργασία αποσκοπεί στην κατανόηση και αξιολόγηση της αποδοτικότητας του MPC σε δυναμικά συστήματα, μέσω της χρήσης μοντέλου, της ανάπτυξης κατάλληλου ελεγκτή και της ανάλυσης αποτελεσμάτων προσομοίωσης. Επιπλέον, επιδιώκεται η ανάδειξη των πλεονεκτημάτων και των περιορισμών του MPC στην πρακτική εφαρμογή μη γραμμικών - γραμμικών συστημάτων, όπως ένα UAV, με στόχο τη διερεύνηση πιθανών μελλοντικών βελτιώσεων ή επεκτάσεων.

1.2. Τι είναι ο Προβλεπτικός Έλεγχος βάσει Μοντέλου – MPC

Το **Model Predictive Control (MPC)**, ή αλλιώς **ο Προβλεπτικός Έλεγχος Βάσει Μοντέλου** , αποτελεί μια από τις πλέον διαδεδομένες και ισχυρές μεθόδους ελέγχου για δυναμικά συστήματα. Η βασική φιλοσοφία του MPC στηρίζεται στη **χρήση ενός μαθηματικού μοντέλου** του υπό έλεγχο συστήματος το οποίο επιτρέπει την **πρόβλεψη της μελλοντικής του συμπεριφοράς** σε έναν πεπερασμένο χρονικό ορίζοντα. Η πρόβλεψη αυτή επιτυγχάνεται μέσω της επίλυσης ενός **προβλήματος βελτιστοποίησης**, το οποίο διαμορφώνεται ώστε να καθοδηγεί το σύστημα προς μια επιθυμητή κατάσταση ή τροχιά, με ελάχιστο κόστος και εντός συγκεκριμένων περιορισμών.
Στην πράξη, το MPC λειτουργεί επαναληπτικά, αναλύοντας σε κάθε χρονική στιγμή την **τρέχουσα κατάσταση** του συστήματος και προβλέποντας την μελλοντική του πορεία για τα επόμενα "N" χρονικά βήματα. Η προβλεπόμενη συμπεριφορά υπολογίζεται βάσει ενός **δυναμικού μοντέλου**, το οποίο μπορεί να είναι γραμμικό ή μη γραμμικό, ανάλογα με την πολυπλοκότητα και τη φύση του πραγματικού συστήματος.
Η κεντρική λειτουργία του MPC περιλαμβάνει την **ελαχιστοποίηση μιας αντικειμενικής συνάρτησης κόστους** (cost function), η οποία μετρά τη διαφορά μεταξύ της προβλεπόμενης κατάστασης και της επιθυμητής (αναφοράς), καθώς και το κόστος των ενεργειών ελέγχου που απαιτούνται. Τυπικά, η συνάρτηση αυτή έχει τη μορφή [2]:

$$
\begin{aligned}
U_t^*(x(t)) := \underset{U_t}{\operatorname{argmin}} \quad & \sum_{k=0}^{N-1} q(x_{t+k}, u_{t+k}) \\
\text{subj. to} \quad & x_t = x(t) \\
& x_{t+k+1} = Ax_{t+k} + Bu_{t+k} \\
& x_{t+k} \in \mathcal{X} \\
& u_{t+k} \in \mathcal{U} \\
& U_t = \{u_t, u_{t+1}, \ldots, u_{t+N-1}\}
\end{aligned}
$$

(**equation 1** : η μαθηματική διατύπωση ενός προβλήματος βελτιστοποίησης κόστους)

Όπου :

Βελτιστοποιήσεις :

- $U_t^*(x(t))$: Η βέλτιστη ακολουθία ενεργειών ελέγχου (inputs) που εξαρτάται από την τρέχουσα μέτρηση κατάστασης x(t).
- $\arg\min_{U_t}$: Εύρεση των Ut που ελαχιστοποιούν το άθροισμα.
- $\sum_{k=0}^{N-1} q(x_{t+k}, u_{t+k})$: Άθροισμα της "στιγμιαίας" συνάρτησης κόστους q για κάθε χρονικό βήμα k του prediction horizon (μέχρι το N - 1)

Περιορισμοί :

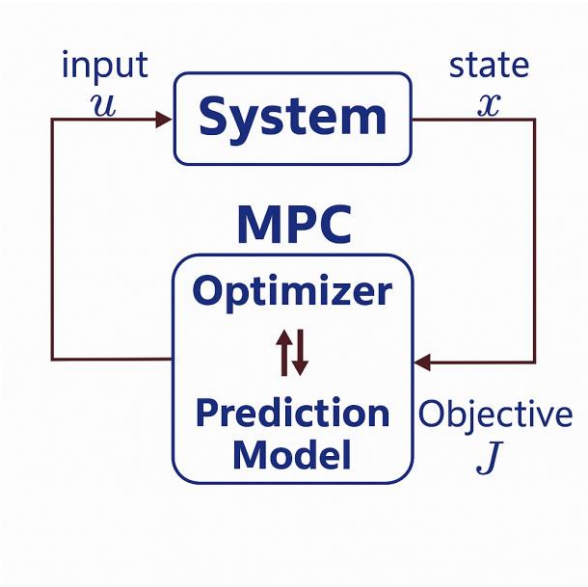
- $x_t = x(t)$: Η αρχική κατάσταση είναι η τρέχουσα μέτρηση του συστήματος.
- $x_{t+k+1} = Ax_{t+k} + Bu_{t+k}$: Το δυναμικό μοντέλο του συστήματος (γραμμική μορφή: A, B πίνακες κατάστασης και ελέγχου).
- $x_{t+k} \in \mathcal{X}$: Οι καταστάσεις του συστήματος που πρέπει να τηρούν τους φυσικούς/τεχνικούς περιορισμούς (state constraints).
- $u_{t+k} \in \mathcal{U}$: Τα σήματα ελέγχου (inputs) που επίσης έχουν όρια/περιορισμούς (input constraints).
- $U_t = \{u_t, u_{t+1}, \ldots, u_{t+N-1}\}$: Η ακολουθία των μεταβλητών βελτιστοποίησης (τι θα "επιλέξει" ο αλγόριθμος).

Αξίζει να σημειωθεί πως αυτή η διαρκής επαναξιολόγηση επιτρέπει στο MPC να προσαρμόζεται σε αλλαγές και αβεβαιότητες στο σύστημα. Όπως αναφέρεται χαρακτηριστικά από τους [1] Schwenzer et al, Ay M., Bergs T., Abel D - 2 Theory - (2021):

«Για αυτό το λόγο το MPC ονομάζεται και ως "receding horizon control". Ουσιαστικά η ιδέα είναι ότι μια βραχυπρόθεσμη (προβλεπτική) βελτιστοποίηση επιτυγχάνει βέλτιστη συμπεριφορά σε μακροπρόθεσμο χρονικό ορίζοντα».

Στην παρακάτω εικόνα (caption 1) παρουσιάζεται η αρχιτεκτονική της μεθόδου MPC η οποία αναλύεται από τον Lasse Peters [3] (MPC – Part1: Introduction to MPC) και αναφέρει χαρακτηριστικά ότι :

« Αντί να βελτιστοποιήσουμε ολόκληρη την τροχιά του συστήματος στο μέλλον, ξεκινάμε από το τρέχον χρονικό βήμα και λαμβάνουμε υπόψη μόνο την τρέχουσα κατάσταση. Από εκεί και πέρα, κοιτάμε ένα περιορισμένο χρονικό ορίζοντα – συνήθως για τα επόμενα 50 βήματα. Μετά τη μέτρηση της τρέχουσας κατάστασης, ζητάμε από τον αλγόριθμο βελτιστοποίησης να υπολογίσει τη βέλτιστη ακολουθία ενεργειών για αυτά τα επόμενα 50 χρονικά βήματα. Από αυτή την υπολογισμένη λύση, δεν εφαρμόζουμε ολόκληρη την ακολουθία. Αντίθετα, εφαρμόζουμε μόνο την πρώτη εντολή στο σύστημα. Στη συνέχεια, μετράμε ξανά τη νέα κατάσταση του συστήματος και επανασχεδιάζουμε τη λύση από αυτή τη νέα κατάσταση. Η βασική ιδέα είναι ότι, αντί να εκτελούμε τις εντολές σε ανοικτό βρόχο (open loop), πάντα εφαρμόζουμε μόνο την πρώτη εντολή, μετράμε που μας οδήγησε και επαναλαμβάνουμε τη διαδικασία σε κάθε βήμα »



(εικόνα 1)

(**Εικόνα 1**: Δομικό διάγραμμα του MPC όπου το σύστημα προβλέπει τη μελλοντική του συμπεριφορά και προσαρμόζει την είσοδο ώστε να ελαχιστοποιήσει τη συνάρτηση κόστους. [3])

1.3. Βασικά Πλεονεκτήματα και Μειονεκτήματα του MPC

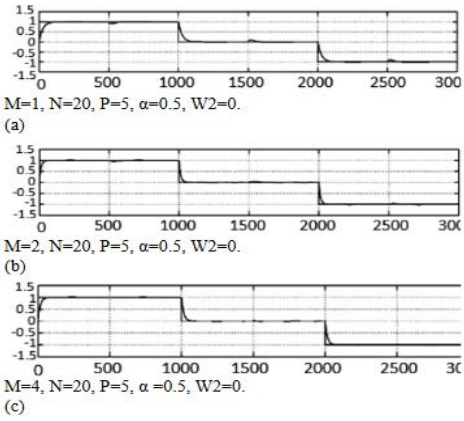
Η αξιολόγηση μιας μεθόδου ελέγχου όπως το Model Predictive Control (MPC) δεν μπορεί να στηρίζεται αποκλειστικά σε θεωρητικούς συλλογισμούς ή μαθηματικά μοντέλα, αλλά απαιτεί μια εμπεριστατωμένη ανάλυση των πρακτικών της συνέπειων. Οποιαδήποτε απόπειρα αξιολόγησης του MPC θα πρέπει να ενσωματώνει τόσο τα θεωρητικά του πλεονεκτήματα όσο και τις εμπειρικές παρατηρήσεις που απορρέουν από την εφαρμογή του σε πραγματικά δυναμικά περιβάλλοντα και βιομηχανικές διεργασίες.
Το MPC συνδυάζει την υψηλή απόδοση με την ευελιξία και την ασφάλεια, γεγονός που το έχει καταστήσει ευρέως αποδεκτό σε πολλούς βιομηχανικούς και ερευνητικούς τομείς. Ουσιαστικά, αποτελεί έναν από τους πιο ισχυρούς μηχανισμούς ελέγχου για πολύπλοκα, δυναμικά και πολυμεταβλητά συστήματα, οπού άλλες μέθοδοι ελέγχου είτε αποτυγχάνουν είτε παρουσιάζουν σημαντικές αδυναμίες όπως θα δούμε σε μια από τις παρακάτω ενότητες. Επιπρόσθετα, η ενσωμάτωση περιορισμών με φυσικό τρόπο στην επίλυση της βελτιστοποίησης και η δυνατότητα πρόβλεψης μελλοντικής συμπεριφοράς, του προσδίδουν μοναδικά χαρακτηριστικά ευφυΐας και προληπτικής παρέμβασης.
Ωστόσο, πάρα την υψηλή του θεωρητική και πρακτική αξία, η εφαρμογή του MPC δεν στερείται προκλήσεων. Ο υπολογιστικός φόρτος, η ανάγκη για ακριβή μοντέλα, η πολυπλοκότητα του σχεδιασμού και οι τεχνικές απαιτήσεις για την υλοποίηση αποτελούν ζητήματα που δεν μπορούν να αγνοηθούν. Συνεπώς, κάθε απόπειρα χρήσης του MPC πρέπει να λαμβάνει υπόψη ένα σαφή σχεδιασμό που εξισορροπεί τη θεωρητική απόδοση με την πρακτική εφαρμογή.
Στη συνέχεια, ακολουθεί αναλυτική παρουσίαση των βασικών πλεονεκτημάτων και μειονεκτημάτων που έχουν καταγραφεί στη σχετική βιβλιογραφία.

Πλεονεκτήματα χρήσης του MPC

- **Διαχείριση Περιορισμών σε Πραγματικό Χρόνο με MPC:**

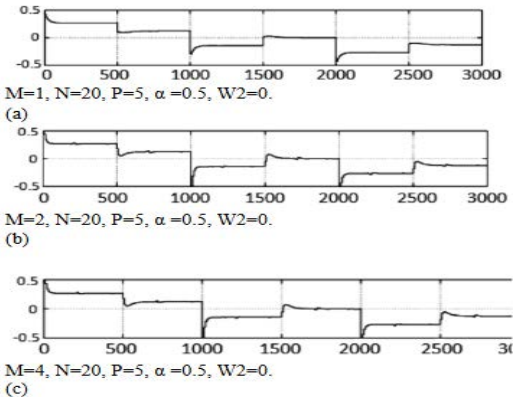
  Η ικανότητα του Model Predictive Control (MPC) να λαμβάνει υπόψη περιορισμούς σε πραγματικό χρόνο κατά τον σχεδιασμό του ελέγχου είναι ένα από τα βασικά του πλεονεκτήματα. Ο έλεγχος επιτυγχάνεται με τη διατύπωση και επίλυση ενός προβλήματος βελτιστοποίησης που λαμβάνει υπόψη τόσο το

μοντέλο του συστήματος όσο και τους περιορισμούς του – για παράδειγμα, περιορισμούς θέσης, ταχύτητας, ροπής, και αποφυγής συγκρούσεων.
Επιπλέον, μελέτες στη ρομποτική έχουν επιβεβαιώσει την αξιοπιστία του MPC στη διαχείριση σύνθετων περιορισμών. Για παράδειγμα, στο πλαίσιο της ρομποτικής πλοήγησης, το MPC χρησιμοποιήθηκε για να επιτρέψει σε κινητά ρομπότ να πλοηγούνται με ασφάλεια, διατηρώντας απόσταση από εμπόδια, τηρώντας τα όρια ταχύτητας και επιταχύνσεων [4] Rezaee A. – Model Predictive Controller for Mobile Robot (2017), όπου παρουσιάζεται το μοντέλο MPC με ενσωματωμένους περιορισμούς θέσης και ταχύτητας σε πραγματικό χρόνο (**βλ. Section 4**).



(εικόνα 2)

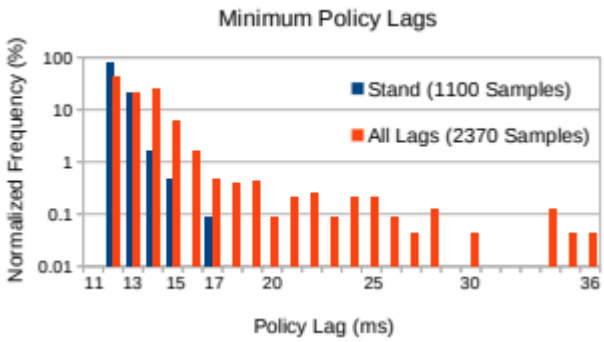
(**Εικόνα 2** : Η **εικόνα 2** παρουσιάζει τη δυναμική απόκριση του συστήματος για διαφορετικές τιμές του χρονικού ορίζοντα ελέγχου M, όπου η αύξηση του M μειώνει τον χρόνο εγκατάστασης και βελτιώνει τη σταθεροποίηση του συστήματος. [4])



(εικόνα 3)

(**Εικόνα 3**: Η **εικόνα 3** παρουσιάζει την επίδραση του χρονικού ορίζοντα ελέγχου M στις εντολές ελέγχου. Όσο αυξάνεται το MMM, οι εντολές γίνονται πιο έντονες και ο υπολογισμός πιο απαιτητικός. [4])

Αντίστοιχα, σε εφαρμογές με ρομποτικούς βραχίονες, όπως δείχνει η εργασία των Erez et al. [5] Erez T., Lowrey K., Tassa Y., Kumar V., Kolev S., Todorov E. – An integrated system for real-time Model Predictive Control of humanoid robots (2013), το MPC επέτρεψε όχι μόνο την ακριβή εκτέλεση κινήσεων αλλά και την αποφυγή μοναδικών θέσεων και υπερφόρτωσης των ενεργοποιητών, κάτι που αναλύεται αναλυτικά στην ενότητα 3 της εργασίας (**βλ. Section 3)**.

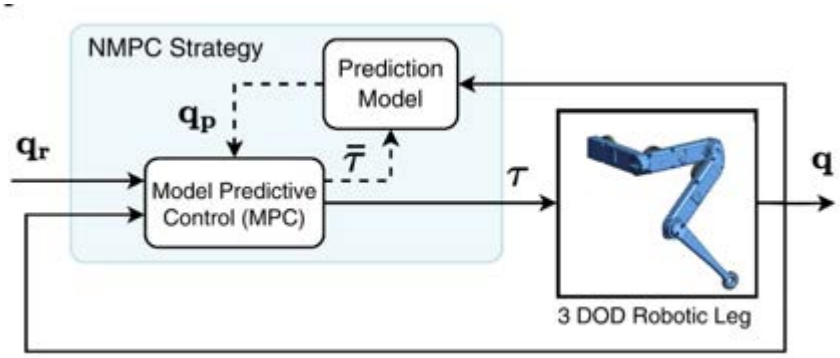


(εικόνα 4)

(**Εικόνα 4**: Ιστόγραμμα της καθυστέρησης πολιτικής (policy lag) στο MPC. Η καθυστέρηση παραμένει σε προβλέψιμα όρια (11–36 ms), επηρεαζόμενη από τη συνάρτηση κόστους. Η κατάσταση "Stand" εμφανίζει μικρότερη και πιο σταθερή καθυστέρηση λόγω σταθερού αριθμού επαφών. [5])

Η δυνατότητα αυτή είναι ζωτικής σημασίας για την ασφάλεια και την αξιοπιστία των συστημάτων. Όπως σημειώνουν οι [6] Singh S., Batra A. – Differentiable Robust Model Predictive Control (2021), οι νεότερες εκδόσεις του MPC είναι πλέον ικανές να διαχειρίζονται αβεβαιότητες στα δεδομένα αισθητήρων, προσαρμόζοντας έγκαιρα τα όρια στους περιορισμούς ανάλογα με τις περιβαλλοντικές συνθήκες (**βλ. Section 1**).

Τέλος, η συνδυαστική εφαρμογή τεχνικών deep learning έχει ενισχύσει περαιτέρω την ικανότητα του MPC να τηρεί περιορισμούς σε πραγματικό χρόνο. Σε μία πρόσφατη εργασία από τους [7] Wang J., Wu Z., Christofides P. D. – Learning-based MPC for robust motion control in legged robots (2024), ο συνδυασμός deep learning με MPC επέτρεψε τη συνεχή εκτίμηση περιορισμών ισορροπίας και επαφής, επιτυγχάνοντας ρομποτικό βάδισμα σε ασταθές έδαφος (**βλ. Section 5 - Deep learning-based MPC**).



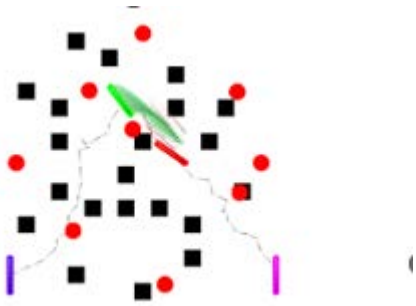
(εικόνα 5)

(Εικόνα 5 : Η εικόνα 5 παρουσιάζει το Δομικό διάγραμμα που απεικονίζει τη στρατηγική μη γραμμικού προγνωστικού ελέγχου (NMPC) για τον έλεγχο του ρομποτικού ποδιού με τρεις βαθμούς ελευθερίας. [7]**)**

- **Πρόβλεψη Μελλοντικής Συμπεριφοράς του συστήματος με MPC**

  Η δυνατότητα του MPC να προβλέπει τη μελλοντική δυναμική ενός συστήματος δίνει σημαντικό πλεονέκτημα σε εφαρμογές οπού απαιτείται προνοητικότητα, όπως η ρομποτική πλοήγηση, ο συντονισμός πολλαπλών ρομπότ κ.α.

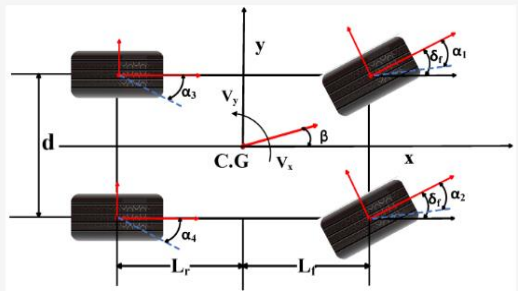
  Η έρευνα [8] Hatch&Boots, The value of planning for infinite-horizon MPC (2021) εστιάζει στην ενσωμάτωση πληροφοριών από αλγορίθμους σχεδιασμού διαδρομής (όπως το RRT#) στο πλαίσιο του MPC, με στόχο την προσέγγιση του απείρου χρονικού ορίζοντα πρόβλεψης. Συγκεκριμένα, προτείνεται η χρήση της συνάρτησης αξίας που προκύπτει από τον σχεδιασμό διαδρομής ως τελικό κόστος στο πρόβλημα βελτιστοποίησης του MPC. Αυτή η προσέγγιση επιτρέπει στον ελεγκτή να λαμβάνει αποφάσεις με μακροπρόθεσμη προοπτική, βελτιώνοντας τη σταθερότητα και την ευελιξία του συστήματος. **(βλ. Section 2)**



(εικόνα 6)

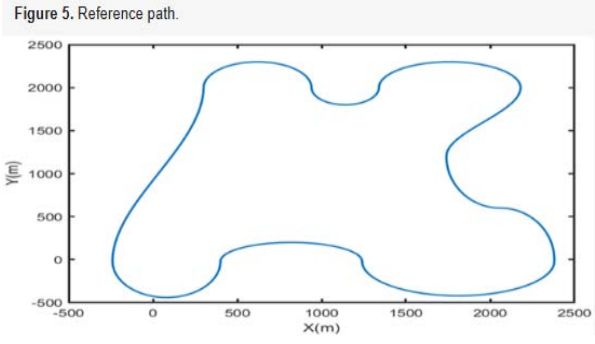
(**Εικόνα 6:** Η **εικόνα 6** παρουσιάζει μια οπτικοποίηση της εκτέλεσης MPC για το ρομπότ τύπου "stick" σε περιβάλλον με εμπόδια ("forest"). Η αρχική και τελική θέση φαίνονται με μπλε και ροζ χρώμα αντίστοιχα, η πράσινη γραμμή δείχνει την ονομαστική τροχιά, ενώ οι κόκκινοι κύκλοι αντιστοιχούν σε δυναμικά εμπόδια. [8])

Μια δεύτερη έρευνα [9] Chen et al. (2024), Prediction Horizon-Varying MPC for Autonomous Vehicle Control προτείνεται μια στρατηγική ελέγχου που συνδυάζει τον αλγόριθμο Proximal Policy Optimization (PPO) με το MPC, επιτρέποντας την προσαρμογή του prediction horizon ανάλογα με τις συνθήκες οδήγησης, όπως η καμπυλότητα του δρόμου και η ταχύτητα του οχήματος. Η μέθοδος αυτή, γνωστή ως PPO-MPC, βελτιώνει την απόδοση του ελεγκτή σε δυναμικά περιβάλλοντα, ενισχύοντας την ικανότητα του συστήματος να προβλέπει και να αντιδρά σε μεταβαλλόμενες συνθήκες. Παρακάτω δίνεται μια εικόνα ανάλυσης της δυναμικής ενός τροχοφόρου, ένα διάγραμμα της επιθυμητής αναφοράς και η μαθηματική διατύπωση της αντικειμενικής συνάρτησης (equation 2).



(εικόνα 7)

(Εικόνα 7: Η **εικόνα 7** δείχνει το δυναμικό του οχήματος τόσο για την διαμήκη όσο και την πλευρική κίνηση. [9])



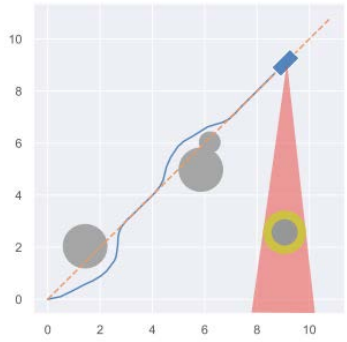
(εικόνα 8)

(**Εικόνα 8**: Η **εικόνα 8** παρουσιάζει το αποτέλεσμα της προ – υπολογισμένης επιθυμητής διαδρομής αναφοράς – path εντός ενός διαγράμματος [9])

$$L^{PPO}(\theta) = \mathbb{E}\left[\min\left(r_{t_2}(\theta)\cdot\hat{A}_{t_2}, clip\left(r_{t_2}(\theta), 1-\epsilon, 1+\epsilon\right)\cdot\hat{A}_{t_2}\right)\right]$$

(**equation 2**: Η μαθηματική εξίσωση που αποτελεί την έκφραση της αντικειμενικής συνάρτησης της PRO, όπου $rt2(\theta)$ δηλώνει την αναλογία της νέας στρατηγικής προς την παλιά στρατηγική και $\hat{A}\ t$ δηλώνει τη συνάρτηση πλεονεκτήματος. [9])

Στη συνέχεια η εργασία [10] Bohn et al. (2021), Reinforcement Learning of the Prediction Horizon in MPC προτείνει τη χρήση ενισχυτικής μάθησης (Reinforcement Learning) για την εκμάθηση του βέλτιστου prediction horizon ως συνάρτηση της κατάστασης του συστήματος. Η προσέγγιση αυτή επιτρέπει στο MPC να προσαρμόζει δυναμικά τον χρονικό ορίζοντα πρόβλεψης, βελτιώνοντας την απόδοση του ελεγκτή σε περιβάλλοντα με μεταβαλλόμενη πολυπλοκότητα και αβεβαιότητα. Τα αποτελέσματα δείχνουν σαφή βελτίωση σε σύγκριση με το MPC με σταθερό prediction horizon. **(βλ. Section 3)** Παρακάτω παρουσιάζεται ένα αποτέλεσμα της μεθόδου με τα συγκεκριμένα χαρακτηριστικά και η μαθηματικη διατυπωση της συναρτησης κοστους.



(εικόνα 9)

(**Εικόνα 9**: Η **εικόνα 9** παραπέμπει σε ένα διάγραμμα που προβάλει το περιβάλλον αποφυγής εμποδίου. Πιο συγκεκριμένα απεικονίζεται το όχημα που παρακολουθεί την τροχιά (πορτοκαλί

γραμμή), τα εμπόδια (γκρι κύκλοι) καθώς και την διαδικασία αποφυγής (μπλε γραμμή). Ο κίτρινος περιμετρικός κύκλος γύρω από το εμπόδιο αντιπροσωπεύει την ανακρίβεια της δέσμης του αισθητήρα η οποία αυξάνεται με την απόσταση. [10])

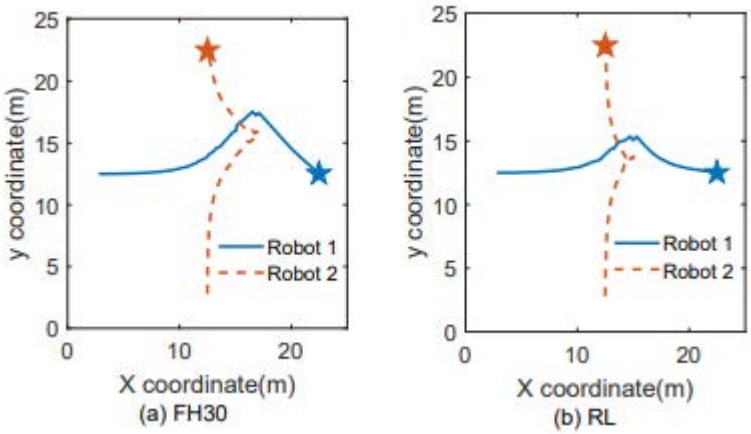
$$J(\theta) = \min_{\theta} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s, \pi_\theta(s))\right], \forall s_0 \in \mathcal{S}_0$$

(**equation 3**: Η μαθηματική διατύπωση της συνάρτησης κόστους που χρησιμοποιείται στο πλαίσιο του reinforcement learning. Εκφράζει το αναμενόμενο άθροισμα μελλοντικών κόστων που προκύπτει όταν εφαρμόζεται μια πολιτική (πθ)για τη λήψη αποφάσεων σε ένα άπειρο χρονικό ορίζοντα, με έκπτωση μέσω του συντελεστή γ [10]).

Τέλος στην εργασία [11] Gupta et al. (2023), RL-based Variable Horizon MPC of Multi-Robot Systems εφαρμόζεται το MPC με μεταβαλλόμενο χρονικό ορίζοντα σε συνεργατικά ρομπότ (multi-agent systems). Προτείνεται μια στρατηγική που συνδυάζει την ενισχυτική μάθηση με το MPC, επιτρέποντας σε κάθε ρομπότ να προσαρμόζει τον prediction horizon του με βάση την κατάσταση του συστήματος και των άλλων ρομπότ. Επιπλέον, εισάγεται η μέθοδος Versatile On-Demand Collision Avoidance (VODCA) για την αποφυγή συγκρούσεων σε πραγματικό χρόνο. Η προσέγγιση αυτή βελτιώνει τον συντονισμό και την ασφάλεια σε συνεργατικά ρομποτικά συστήματα. **(βλ. Section 4)**

$$J^{RL}(\theta) = \max_{\theta} \mathbb{E}\left[\sum_{k=0}^{K} \gamma^k R(s, \pi_\theta(s))\right], \forall s_0 \in \mathcal{S}_0.$$

(**equation 4**: Η παραπάνω μαθηματική διατύπωση αποτελεί την συνάρτηση ανταμοιβής – reward function όπου, η RL προσπαθεί να μεγιστοποιήσει την αναμενόμενη συνολική ανταμοιβή (αντί να ελαχιστοποιήσει κόστος όπως στο MPC)και καθοδηγεί τη μάθηση για τον βέλτιστο χρονικό ορίζοντα [11].)



(εικόνα 10)

(**Εικόνα 10**: Η **εικόνα 10** παρουσιάζει τις τροχιές από δυο ρομπότ σε ένα συνεργατικό σενάριο, όπου (α) απεικονίζεται η παρακολούθηση τροχιάς με MPC και (β) η βελτιωμένη συμπεριφορά με χρήση RL-based Variable Horizon MPC, επιδεικνύοντας βελτιωμένο συντονισμό και αποφυγή σύγκρουσης. [11])

Μειονεκτήματα του MPC

- **Υψηλό Υπολογιστικό Κόστος σε Πραγματικό Χρόνο**

  Ένα από τα σημαντικότερα πρακτικά μειονεκτήματα του MPC είναι το **υψηλό υπολογιστικό του κόστος**, ιδιαίτερα όταν εφαρμόζεται σε πραγματικό χρόνο. Η αρχή λειτουργίας του MPC βασίζεται στην επαναλαμβανόμενη επίλυση ενός προβλήματος βελτιστοποίησης κατά κάθε χρονικό βήμα ελέγχου. Σε αυτό το πρόβλημα λαμβάνονται υπόψη η δυναμική του συστήματος, οι περιορισμοί (φυσικοί και τεχνικοί), και η επιθυμητή συμπεριφορά, μέσω της ελαχιστοποίησης μιας συνάρτησης κόστους σε έναν χρονικό ορίζοντα πρόβλεψης.

  Η διαδικασία αυτή, αν και προσφέρει σημαντική προγνωστική ικανότητα και ευελιξία, είναι ιδιαίτερα απαιτητική από άποψη επεξεργαστικής ισχύος. Όπως αναφέρεται από τους [12] Nielsen & Axehill (2014), A parallel

algorithm for Newton step calculation in MPC, η υπολογιστική πολυπλοκότητα αυξάνεται ταχέως με το μέγεθος του ορίζοντα πρόβλεψης και την πολυπλοκότητα του μοντέλου. Η εργασία τους προτείνει αλγορίθμους που βασίζονται σε παράλληλη υπολογιστική επεξεργασία για την επιτάχυνση της διαδικασίας, όμως ακόμη και αυτές οι μέθοδοι απαιτούν εξειδικευμένο υλικό **(βλ. Section 5).**

Algorithm 1 Parallel reduction of MPC problem

1: Initiate level counter $k := 0$
2: Initiate the first number of subsystems $p_{-1} = N$
3: Set the minimal number of subproblems $p_{\min}$
4: **while** $p_k > p_{\min}$ **do**
5: Compute desired p_k to define the number of subproblems (with $p_k < p_{k-1}$)
6: Split the prediction horizon $0, \dots, p_{k-1}$ in $p_k + 1$ segments $0, \dots, N_0^k$ up to $0, \dots, N_{p_k}^k$
7: Create subproblems $i = 0, \dots, p_k$ for each time block
8: **parfor** $i = 0, \dots, p_k$ **do**
9: Solve subproblem i parametrically and store K_i^x, k_i^x, K_i^λ and k_i^λ
10: Compute $\hat{A}_i$, $\hat{B}_i$, $\hat{a}_i$, $\hat{H}_i$, $\hat{f}_i$ and $\hat{c}_i$ for the next level
11: Compute and store Z_i
12: **end parfor**
13: Update level counter $k := k + 1$
14: **end while**
15: Compute maximum level number $k := k - 1$

(εικόνα 11)

(**Εικόνα 11**: Στην παραπάνω εικόνα δίνεται ο αλγόριθμος για τη παράλληλη μείωση του προβλήματος MPC. Ο αλγόριθμος εφαρμόζεται ως πρώτο βήμα για τη δημιουργία του δέντρου που προκύπτει όταν τα προβλήματα MPC μειώνονται σε αρκετά βήματα [12])

Η πρόκληση αυτή γίνεται ακόμη πιο εμφανής σε ρομποτικά συστήματα όπου απαιτείται γρήγορη απόκριση και χαμηλή καθυστέρηση, ενώ οι υπολογιστικοί πόροι μπορεί να είναι περιορισμένοι. Στην έρευνα των [13] Salzmann et al. (2022), Real-Time Neural MPC: Deep Learning for Predictive Control in Robotics, η εφαρμογή νευρωνικών δικτύων στο MPC (Neural-MPC) επιχειρεί να προσεγγίσει την πρόβλεψη μέσω **deep learning** ώστε να μειωθεί ο χρόνος επεξεργασίας. Ωστόσο, η ίδια η χρήση τέτοιων τεχνικών αυξάνει την ανάγκη για εξοπλισμό με GPU και προγραμματιστικό φορτίο, περιορίζοντας τη δυνατότητα ενσωμάτωσης σε embedded real-time συστήματα **(βλ. Section 4).**

$$\min_{\Delta\omega^i} \sum_{k=0}^{N-1} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{r}_k \end{bmatrix}^\top \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix} + \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix}^\top \mathbf{H}_k \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix}$$

subject to

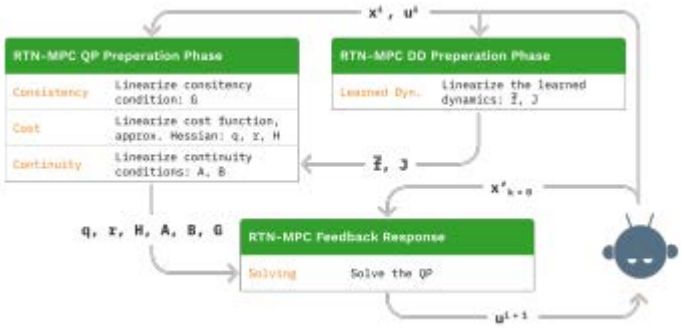
$$\Delta\mathbf{x}_{k+1} = \mathbf{A}_k\Delta\mathbf{x}_k + \mathbf{B}_k\Delta\mathbf{u}_k + \bar{\phi}_k - \mathbf{x}_{k+1}\,, \quad k = 0, \dots, N-1$$

$$-\bar{\mathbf{g}}_k \geq G_k^x\Delta\mathbf{x}_k + G_k^u\Delta\mathbf{u}_k\,, \quad k = 0, \dots, N\,,$$

(**equation 5**: Η παραπάνω μαθηματική διατύπωση περιγράφει το πρόβλημα Quadratic Programming (QP) που λύνει το SQP για την επίλυση του MPC [13])

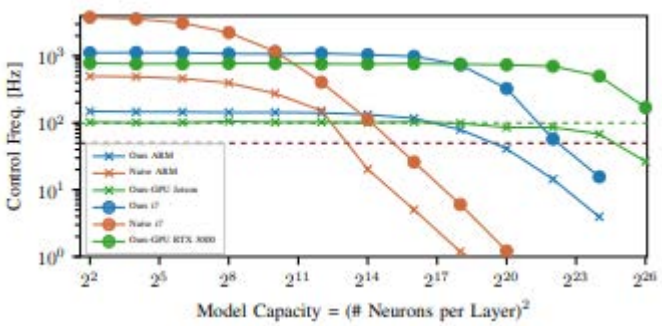
$$f^*_{\mathcal{D}}(\mathbf{x},\mathbf{u}) \approx \bar{\mathbf{f}}^i_{\mathcal{D}} + \mathbf{J}^i_{\mathcal{D},k}\begin{bmatrix}\mathbf{x}-\mathbf{x}^i_k\\ \mathbf{u}-\mathbf{u}^i_k\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\mathbf{x}-\mathbf{x}^i_k\\ \mathbf{u}-\mathbf{u}^i_k\end{bmatrix}^\top \mathbf{H}^i_{\mathcal{D},k}\begin{bmatrix}\mathbf{x}-\mathbf{x}^i_k\\ \mathbf{u}-\mathbf{u}^i_k\end{bmatrix}.$$

(**equation 6**: Η παραπάνω μαθηματική διατύπωση αποτελεί την εξίσωση του νευρωνικού μοντέλου δυναμικής, η οποία μειώνει το υπολογιστικό φορτίο στο πρόβλημα βελτιστοποίησης. [13])



(εικόνα 12)

(**Εικόνα 12**: Στην **εικόνα 12** παρουσιάζεται το διάγραμμα ροής της προετοιμασίας και επίλυσης του Quadratic Programming (QP) στο Real-Time Neural MPC (RTN-MPC), που συνδυάζει γραμμικές προσεγγίσεις με το νευρωνικό μοντέλο για αποδοτική υπολογιστική διαχείριση. [13])



(εικόνα 13)

(**Εικόνα 13**: Στην **εικόνα 13** παρουσιάζεται ένα γράφημα που αναλύει τη συχνότητα ελέγχου σε σχέση με την χωρητικότητα μοντέλου (αριθμός νευρώνων ανά στρώμα) σε διαφορετικές πλατφόρμες υλικού, αναδεικνύοντας την υπεροχή του RTN-MPC στην υλοποίηση σε embedded και GPU συστήματα. [13])

Η υπολογιστική απαίτηση καθιστά το MPC λιγότερο ελκυστικό σε εφαρμογές που απαιτούν άμεση απόκριση, όπως είναι η αποφυγή σύγκρουσης ή ο βηματισμός ρομπότ, και έχει οδηγήσει σε έρευνα για απλοποιημένες ή «explicit» παραλλαγές του MPC που προϋπολογίζουν τη λύση offline. **Συνεπώς, αν και το MPC θεωρείται εξαιρετικά ικανό, η πρακτική**

του εφαρμογή περιορίζεται από το τεχνολογικό πλαίσιο στο οποίο ενσωματώνεται.

- **Ευαισθησία σε Σφάλματα Μοντελοποίησης**

  Το MPC βασίζεται στην ακρίβεια του μοντέλου του συστήματος για να προβλέπει τη μελλοντική του συμπεριφορά. Συνεπώς, **η ποιότητα της απόδοσης του ελεγκτή εξαρτάται άμεσα από το πόσο καλά το μαθηματικό μοντέλο αντανακλά τη φυσική πραγματικότητα**. Σε εφαρμογές όπου το μοντέλο είναι ατελές, ή το περιβάλλον είναι μη στατικό και αβέβαιο (π.χ. δυναμικά εμπόδια ή μεταβαλλόμενες μάζες), το MPC μπορεί να οδηγήσει σε **ανακριβείς προβλέψεις**, υποβάθμιση της απόδοσης ή ακόμη και αστάθεια.

  Η εργασία των [14] Anand et al. (2024), On Model Mismatch in MPC, επισημαίνει ότι πολλές προσεγγίσεις σχεδιασμού MPC παραβλέπουν την ανάγκη για **μοντέλα υψηλής πιστότητας**, με αποτέλεσμα οι υποθέσεις που γίνονται για την αναπαράσταση της δυναμικής να επηρεάζουν αρνητικά τον ελεγκτή. Προτείνουν μάλιστα αναθεώρηση των prediction models με έμφαση στη βελτίωση της ρεαλιστικότητας του μοντέλου. **(βλ. Section 2)**

$$
\begin{aligned}
V^{\mathrm{MPC}}(\mathbf{s}_k) = \min_{\hat{\mathbf{s}},\mathbf{u}} \quad & \gamma^N T(\hat{\mathbf{s}_N}) + \sum_{i=0}^{N-1} \gamma^k L(\hat{\mathbf{s}}_i, \mathbf{u}_i) \\
\text{s.t.} \quad & \hat{\mathbf{s}}_{i+1} = \mathbf{f}(\hat{\mathbf{s}}_i, \mathbf{u}_i)\ , \\
& \mathbf{h}(\hat{\mathbf{s}}_i, \mathbf{u}_i) \leq 0\,, \\
& \hat{\mathbf{s}}_0 = \mathbf{s}_k, \quad \hat{\mathbf{s}}_N \in \mathbb{T}\,.
\end{aligned}
$$

(**equation 7**: Παραπάνω διατυπώνεται το πρόβλημα βελτιστοποίησης στο MPC, όπου ελαχιστοποιείται η συνάρτηση κόστους L σε χρονικό ορίζοντα N, λαμβάνοντας υπόψη τις δυναμικές εξισώσεις του συστήματος f και τους περιορισμούς h, με παράγοντα έκπτωσης γ και τελικό κόστος T. [14])

Για την αντιμετώπιση αυτών των προκλήσεων, έχουν αναπτυχθεί τεχνικές όπως το **Robust MPC** και το **Distributionally Robust MPC**, οι οποίες λαμβάνουν υπόψη την αβεβαιότητα στο μοντέλο. Στην εργασία των [15] Sung, K., Sinha, A., & Ramanan, J. (2024). Distributionally Robust Model Predictive Control with State-Input Estimation, προτείνεται ο συνδυασμός robust MPC με τεχνικές εκτίμησης κατάστασης και αβεβαιότητας (SIED-MPC), που αυξάνουν την **ανθεκτικότητα του ελεγκτή** σε σφάλματα μοντελοποίησης, βελτιώνοντας την ασφάλεια και την αξιοπιστία του ελέγχου. **(βλ. Section 4)**

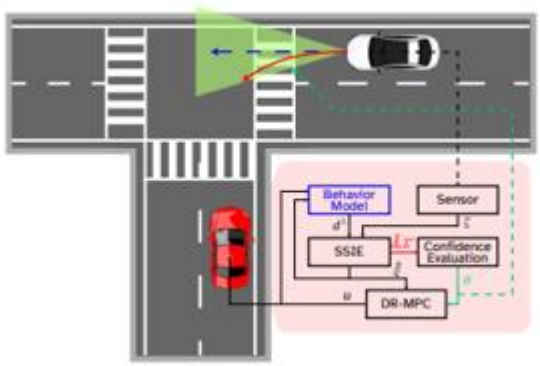
Algorithm 2: SIED-MPC

Given: $\mathbb{F} = \phi$, m, $\theta_{\max}$, τ, L

for $t = 1 : T$ **do**
- Observe x_t and ζ_t;
- $\hat{\xi}_t, \hat{\Delta}_{t-1}, \hat{\Sigma}^{\xi}_t, \Sigma^{\Delta}_{t-1} \leftarrow \texttt{SSIE}(\zeta_t)$;
- $\mathbb{F} \leftarrow \mathbb{F} \cup (\hat{\Delta}_{t-1}, \Sigma^{\Delta}_{t-1})$;
- **if** $|\mathbb{F}| > m$ **then**
  - Keep the most recent m pairs
- Evaluate F^{π}_t with $\mathbb{F}$ using (23);
- $\theta \leftarrow \theta_{\max} \tanh(\tau F^{\pi}_t)$;
- **for** l=1:L **do**
  - Compute $(\hat{\mu}^{\xi}_{t+l}, \hat{\Sigma}^{\xi}_{t+l})$ using (19),(20);
- Solve nonlinear MPC (27) to obtain $u^{\star}_t$;
- Apply $u^{\star}_t$ to the system;

(εικόνα 14)

(**Εικόνα 14**: Στον παραπάνω αλγόριθμο SIED-MPC περιγράφεται η διαδικασία συνδυασμένης εκτίμησης κατάστασης και εισόδου (State-Input Estimation) με Robust Model Predictive Control, όπου ο ελεγκτής προσαρμόζει τις εντολές του λαμβάνοντας υπόψη αβεβαιότητες μοντέλου και μετρήσεων για βελτιωμένη ασφάλεια και αξιοπιστία. [15])



(εικόνα 15)

(**Εικόνα 15**: Η **εικόνα 15** δίνει ένα διάγραμμα στο οποίο περιγράφεται ένα σενάριο. Παρουσιάζεται η αλληλεπίδραση μεταξύ του μοντέλου συμπεριφοράς του εμποδίου, του αισθητήρα, της εκτίμησης κατάστασης και εισόδου (SSIE), της αξιολόγησης εμπιστοσύνης και του Distributionally Robust MPC (DR-MPC) [15])

Παρ' όλα αυτά, ακόμη και οι robust παραλλαγές του MPC δεν μπορούν να υποκαταστήσουν πλήρως την ανάγκη για **σωστά μοντελοποιημένα δυναμικά συστήματα**, ειδικά σε γρήγορα μεταβαλλόμενα περιβάλλοντα όπως αυτά της ρομποτικής.

- **Δυσκολία Παραμετροποίησης και Σχεδιαστικής Ρύθμισης του MPC**

  Ένα από τα λιγότερο φανερά, αλλά ιδιαίτερα σημαντικά μειονεκτήματα του Model Predictive Control είναι η **πολυπλοκότητα στον σχεδιασμό και στην παραμετροποίηση του ελεγκτή**. Αν και το MPC είναι εξαιρετικά ευέλικτο, η ρύθμιση των παραμέτρων του όπως ο prediction horizon, ο control horizon, οι συντελεστές της συνάρτησης κόστους, και οι περιορισμοί απαιτούν προσεκτική, λεπτομερή και συχνά εμπειρική διαδικασία. Το γεγονός αυτό καθιστά τον σχεδιασμό του MPC χρονοβόρο και επιρρεπή σε σφάλματα, ειδικά σε συστήματα με μη γραμμική συμπεριφορά ή με πολλαπλές μεταβλητές εισόδου/εξόδου.

  Σύμφωνα με τους [16] Rawlings & Mayne (2014), Model Predictive Control: Theory and Design. Nob Hill Publishing, η διαδικασία επιλογής των οριζόντων πρόβλεψης και ελέγχου δεν είναι πάντα προφανής και επηρεάζει σημαντικά τόσο τη **σταθερότητα όσο και την υπολογιστική απόδοση του συστήματος**. Επιπλέον, η επιλογή των βαρών στη συνάρτηση κόστους (π.χ. Q και R matrices) είναι κρίσιμη για την εξισορρόπηση μεταξύ απόδοσης και κατανάλωσης ενέργειας ή μεταξύ ακρίβειας και ομαλότητας της συμπεριφοράς. Η παραμικρή αστοχία στη ρύθμιση μπορεί να οδηγήσει σε υπερβολικά επιθετική ή υποτονική συμπεριφορά του ελεγκτή. **(βλ. Section 1)**

$$V_N(x, \mathbf{u}) = (1/2)x'\mathbf{Q}x' + (1/2)\mathbf{u}'\mathbf{R}\mathbf{u} + \mathbf{u}'\mathbf{S}x$$

(**equation 8**: Στην παραπάνω εικόνα παρουσιάζεται η τυπική τετραγωνική συνάρτηση κόστους του MPC, όπου τα βάρη Q και R ρυθμίζουν τη σημαντικότητα της απόκλισης της κατάστασης και της εντολής ελέγχου αντίστοιχα, και αποτελούν κρίσιμους παράγοντες για τη ρύθμιση και την απόδοση του ελεγκτή.)

  Η εργασία των [17] Bemporad, A., & Morari, M. (2007). Robust Model Predictive Control: A Survey. Springer ενισχύει την άποψη αυτή, δείχνοντας ότι σε πολλά πρακτικά σενάρια ιδιαίτερα στην αυτοκινητοβιομηχανία και στη ρομποτική η παραμετροποίηση του MPC γίνεται με δοκιμή και σφάλμα **(trial-and-error)**, κάτι που έρχεται σε αντίθεση με την αυστηρά βελτιστοποιητική φύση της μεθόδου **(βλ. Section 2 - 5).**

  Η δυσκολία αυτή γίνεται ακόμη πιο έντονη στις περιπτώσεις **μη γραμμικού MPC (NMPC)**. Η εργασία των [18] Lucia, S., Andersson, J. A. E., Brandt, H., Diehl, M., & Engell, S. (2014). Handling Uncertainty in Economic Nonlinear Model Predictive Control επισημαίνει ότι ο σχεδιασμός NMPC σε περιβάλλοντα με αβεβαιότητες απαιτεί τεχνικές όπως multi-stage scenario-based optimization, που αυξάνουν σημαντικά την πολυπλοκότητα του

ελέγχου. Οι συγγραφείς δείχνουν ότι σε πραγματικές βιομηχανικές εφαρμογές, όπως η χημική επεξεργασία, η ακρίβεια του μοντέλου δεν είναι δεδομένη και η ρύθμιση του NMPC πρέπει να προσαρμόζεται συνεχώς στα νέα δεδομένα. Αυτό καθιστά την παραμετροποίηση του ελεγκτή όχι μόνο πιο απαιτητική αλλά και **ιδιαίτερα ευαίσθητη σε σφάλματα** σχεδίασης **(βλ. Section 2 - 4).**

1.4 Κατηγορίες MPC και εφαρμογές σε quadrotors

Ανάλογα με τη φύση του μοντέλου, την πολυπλοκότητα του προβλήματος και τις απαιτήσεις της εφαρμογής, έχουν αναπτυχθεί διάφορες παραλλαγές του MPC, οι οποίες διαφοροποιούνται τόσο ως προς το θεωρητικό υπόβαθρο όσο και ως προς τις δυνατότητές τους. Οι βασικές κατηγορίες που απαντώνται στη βιβλιογραφία και στην πράξη περιλαμβάνουν το **Linear MPC**, το **Nonlinear MPC**, το **Robust Adaptive MPC** και το **Stochastic MPC**. Επιπλέον, σε περιπτώσεις κατανεμημένων ή μεγάλης κλίμακας συστημάτων, εφαρμόζονται παραλλαγές όπως το Distributed/Decentralized MPC.

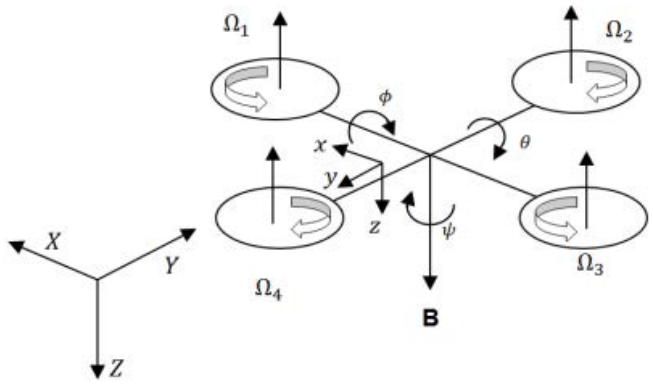
Το **Linear MPC** είναι η πιο κλασική και ευρέως διαδεδομένη εκδοχή, κατάλληλη για συστήματα των οποίων η δυναμική μπορεί να προσεγγιστεί γραμμικά. Το **Non Linear MPC** αποτελεί γενίκευση της μεθόδου και εφαρμόζεται σε συστήματα με έντονη μη γραμμικότητα, όπου απαιτείται η χρήση πολύπλοκων μαθηματικών μοντέλων και αριθμητικών τεχνικών. Το **Robust Adaptive MPC** στοχεύει στην εξασφάλιση της ευστάθειας και της επιθυμητής απόδοσης υπό αβεβαιότητες και διαταραχές, ενώ το **Stochastic MPC** ενσωματώνει πιθανοτικές περιγραφές των αβεβαιοτήτων και επιδιώκει βέλτιστες στρατηγικές ελέγχου υπό στατιστικούς περιορισμούς.

Η επιλογή της κατάλληλης παραλλαγής του MPC εξαρτάται από τις απαιτήσεις της εκάστοτε εφαρμογής, τη φύση των περιορισμών και την ακρίβεια του διαθέσιμου μοντέλου. Παρακάτω παρουσιάζονται αναλυτικά όλες οι βασικές κατηγορίες MPC πάνω σε εφαρμογές που διεξήχθησαν στα πλαίσια ερευνών με θέμα τα πολυκόπτερα - Quadrotors.

1.4.1 Linear MPC

Ο **Γραμμικός Προγνωστικός Έλεγχος με Βάση Μοντέλο (Linear Model Predictive Control – Linear MPC)** είναι μια προηγμένη μέθοδος ελέγχου που χρησιμοποιείται για τον έλεγχο δυναμικών συστημάτων με περιορισμούς. Η βασική αρχή του Linear MPC στηρίζεται στη χρήση ενός γραμμικού μαθηματικού μοντέλου για την πρόβλεψη της μελλοντικής συμπεριφοράς του συστήματος και στην επίλυση ενός προβλήματος βελτιστοποίησης σε κάθε χρονική στιγμή, με στόχο τον υπολογισμό της βέλτιστης ελεγκτικής δράσης. Η διαδικασία αυτή επαναλαμβάνεται διαρκώς, ακολουθώντας την αρχή του "οπισθοχωρούντος ορίζοντα" (receding horizon).
Η μεθοδολογία του Linear MPC μπορεί να εφαρμοστεί με επιτυχία ακόμη και σε σύνθετα, μη γραμμικά συστήματα, όπως τα μη επανδρωμένα εναέρια οχήματα τύπου **quadrotor**. Το quadrotor είναι ένα σύστημα με **έξι βαθμούς ελευθερίας ,** που σημαίνει ότι η κίνησή του περιγράφεται από έξι ανεξάρτητες μεταβλητές: τρεις μεταφορικές και τρεις στροφικές. Συγκεκριμένα, οι βαθμοί ελευθερίας είναι: **Μετακίνηση στον άξονα x (πλευρική μετατόπιση), Μετακίνηση στον άξονα y (διαμήκης μετατόπιση), Μετακίνηση στον άξονα z (κατακόρυφη μετατόπιση), Περιστροφή γύρω από τον άξονα x (roll), Περιστροφή γύρω από τον άξονα y (pitch), Περιστροφή γύρω από τον άξονα z (yaw).**

Η εργασία των [19] Islam, Okasha και Idres (2017), Dynamics and control of quadcopter using linear model predictive control approach, διερευνά ακριβώς την εφαρμογή του **Linear MPC** σε ένα τέτοιο σύστημα. Στη μελέτη αυτή, οι συγγραφείς αρχικά αναλύουν τη μη γραμμική δυναμική του quadrotor, και στη συνέχεια προβαίνουν στη γραμμικοποίηση του μοντέλου γύρω από σημεία ισορροπίας. Βάσει αυτού του γραμμικού μοντέλου, σχεδιάζουν έναν Linear MPC ελεγκτή ικανό να διαχειρίζεται την πλοήγηση του quadrotor σε διαφορετικές τροχιές αναφοράς.
Η διαδικασία περιλαμβάνει την πρόβλεψη της μελλοντικής συμπεριφοράς του συστήματος, την ελαχιστοποίηση μιας συνάρτησης κόστους που περιλαμβάνει την απόκλιση από τις επιθυμητές καταστάσεις και την προσπάθεια ελέγχου, και την ενσωμάτωση περιορισμών όπως οι κορεσμοί των κινητήρων. Η υλοποίηση και επαλήθευση πραγματοποιούνται σε περιβάλλον MATLAB/Simulink, με διάφορα σενάρια τροχιών, όπως ευθείες, κυκλικές και ελικοειδείς κινήσεις.
Τα αποτελέσματα δείχνουν ότι ο Linear MPC μπορεί να επιτύχει ακριβή παρακολούθηση τροχιάς και ταυτόχρονα να διαχειριστεί αποτελεσματικά φυσικούς και λειτουργικούς περιορισμούς, καθιστώντας τον κατάλληλο για τον έλεγχο UAV τύπου quadrotor.
Η εργασία αυτή αποτελεί σαφή απόδειξη ότι, παρά τη μη γραμμικότητα του φυσικού συστήματος, η γραμμικοποιημένη προσέγγιση του Linear MPC είναι σε θέση να παρέχει σταθερό και αποδοτικό έλεγχο υπό ρεαλιστικές συνθήκες. Παρακάτω παρουσιάζεται το σχήμα ενός πολυκοπτέρου τύπου quadrotor εστιάζοντας στους **έξι βαθμούς ελευθέριας**:



(εικόνα 16)

(**Εικόνα 16**: Η **εικόνα 16** αναπαριστά ένα πολυκόπτερο τύπου quadrotor και τους άξονες αναφοράς. Απεικονίζονται οι βασικές κινήσεις (roll φ, pitch θ, yaw ψ) και η διάταξη του σώματος (B) σε σχέση με το παγκόσμιο σύστημα (X, Y, Z) [19])

Όπου : $\dot{\boldsymbol{x}} = \boldsymbol{f}((\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{t}))$, το μη γραμμικό σύστημα και,

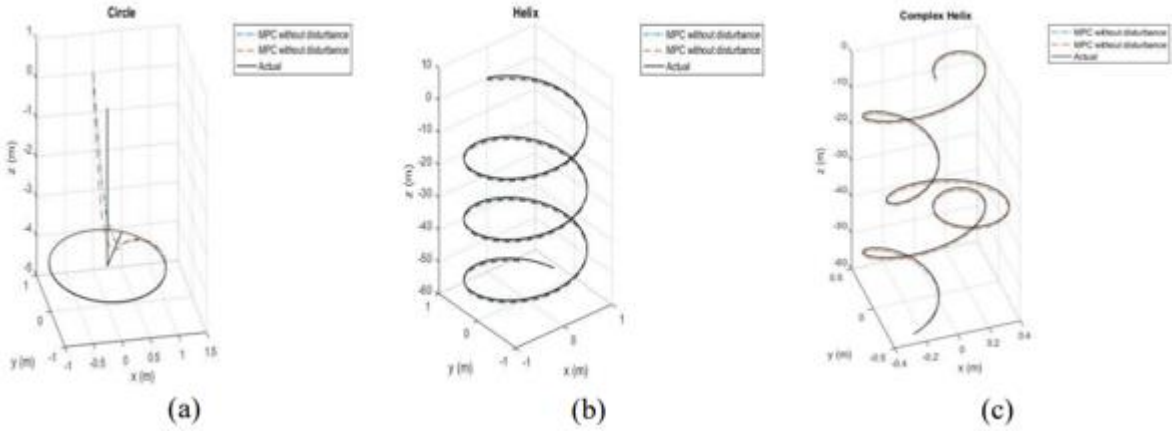
$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k$$
$$\Delta y_k = C\Delta x_k + D\Delta u_k$$
$$\Delta x_k = x_k - x_T$$
$$\Delta u_k = u_k - u_T$$

το δυναμικό μοντέλο του πολυκοπτέρου τύπου quadrotor **γραμμικοποιημένο**.

Επεκτείνοντας τις παραπάνω εξισώσεις μέχρι το χρονικό βήμα $k + N$, μπορούν να προσδιοριστούν οι μελλοντικές καταστάσεις και έξοδοι του συστήματος, ανάλογα με τις αρχικές καταστάσεις και τις μελλοντικές εισόδους, όπως δίνονται από την **παρακάτω εξίσωση** :

$$\Delta x_{k+N} = A^N\Delta x_k + A^{N-1}B\Delta u_k + A^{N-2}B\Delta u_{k+1} + \cdots + AB\Delta u_{k+N-2} + B\Delta u_{k+N-1}$$
$$\Delta y_{k+N} = CA^N\Delta x_k + C(A^{N-1}B\Delta u_k + A^{N-2}B\Delta u_{k+1} + \cdots + AB\Delta u_{k+N-2} + B\Delta u_{k+N-1})$$

Παρουσιάζονται επίσης αποτελέσματα γραφημάτων που σχετίζονται με την τροχιά σε διαφορετικά πειράματα :



(Εικόνα 17)

(**Εικόνα 17**: Το γράφημα **a** αναφέρεται στην κυκλική τροχιά, το γράφημα **b** παρουσιάζει την ελικοειδή τροχιά και το γράφημα **c** δείχνει την συνθέτη ελικοειδή τροχιά, όπου συγκρίνονται οι επιδόσεις του MPC με και χωρίς παρεμβολές, αποδεικνύοντας την ακρίβεια του ελεγκτή [19])

1.4.2 Non - Linear MPC

Όταν η δυναμική ενός συστήματος παρουσιάζει **έντονα μη γραμμικά χαρακτηριστικά**, η χρήση του κλασικού, γραμμικού MPC παύει να είναι ικανοποιητική. Σε αυτές τις περιπτώσεις, εφαρμόζεται ο **Μη Γραμμικός Προγνωστικός Έλεγχος με Βάση Μοντέλο (Nonlinear Model Predictive Control – Nonlinear MPC ή NMPC)**, ο οποίος βασίζεται σε πλήρως μη γραμμικά μαθηματικά μοντέλα του συστήματος.
Η λειτουργική φιλοσοφία του NMPC παραμένει ίδια με εκείνη του Linear MPC. Σε κάθε χρονική στιγμή, επιλύεται ένα πρόβλημα βελτιστοποίησης με χρονικό ορίζοντα πρόβλεψης, λαμβάνοντας υπόψη περιορισμούς και στόχους του συστήματος. Η σημαντική διαφορά εντοπίζεται στο γεγονός ότι **ο ελεγκτής χρησιμοποιεί το πλήρες, ακριβές, μη γραμμικό μοντέλο του συστήματος**, χωρίς να το απλοποιεί ή να το προσεγγίζει με γραμμικές εξισώσεις.. Αυτό καθιστά τον έλεγχο πιο ακριβή, αλλά αυξάνει και τις αριθμητικές απαιτήσεις της μεθόδου, αφού η επίλυση του προβλήματος βελτιστοποίησης είναι πλέον πιο πολύπλοκη και δεν εξασφαλίζει πάντοτε εύκολη ή μοναδική λύση.
Το NMPC είναι ιδανικός για εφαρμογές **όπου απαιτείται ακρίβεια σε ευρύτερο φάσμα συνθηκών λειτουργίας**. Ένα χαρακτηριστικό παράδειγμα είναι ο έλεγχος μη επανδρωμένων εναέριων οχημάτων τύπου quadrotor. Η λειτουργία αυτών των οχημάτων εμπλέκει μη γραμμικές σχέσεις μεταξύ εισόδων (όπως η ισχύς των

κινητήρων) και εξόδων (όπως οι κινήσεις στο χώρο), κάτι που καθιστά την προσέγγιση με γραμμικά μοντέλα περιορισμένης ακρίβειας. Σε τέτοια σενάρια, το NMPC επιτρέπει τον πιο ακριβή και σταθερό έλεγχο ακόμα και σε δυναμικές και ταχέως μεταβαλλόμενες συνθήκες, **όπως κατά τη διάρκεια ελιγμών, απότομων αλλαγών κατεύθυνσης ή ταχείας προσγείωσης.**

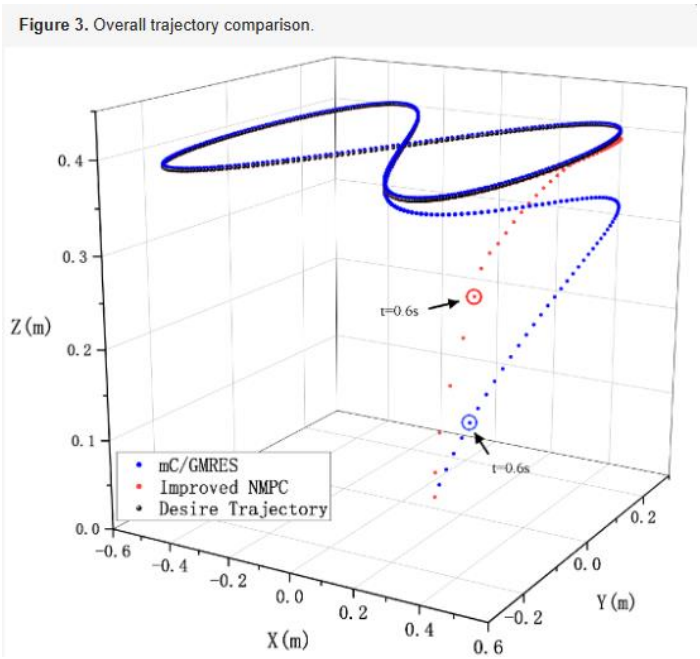
Η εφαρμογή του NMPC σε quadrotor UAVs που παρουσιάζεται στην εργασία των [20] Ma, H., Gao, Y., Yang, Y., & Xu, S. (2024). Improved Nonlinear Model Predictive Control Based Fast Trajectory Tracking for a Quadrotor Unmanned Aerial Vehicle, όπου αναπτύσσεται μια βελτιωμένη έκδοση του NMPC για παρακολούθηση τροχιάς με υψηλή ταχύτητα και ακρίβεια. Στην εργασία γίνεται χρήση ενός πλήρους μη γραμμικού μοντέλου του οχήματος, ενώ εξετάζεται η ικανότητα του ελεγκτή να διαχειρίζεται σύνθετους δυναμικούς περιορισμούς κατά την πτήση. Η απόδοση του ελεγκτή επιβεβαιώνεται μέσω προσομοιώσεων και συγκρίνεται με άλλες μεθόδους ελέγχου, αποδεικνύοντας τη σαφή υπεροχή του NMPC σε σενάρια αυξημένων απαιτήσεων.
Παρά τις αυξημένες υπολογιστικές απαιτήσεις που συνοδεύουν τη μέθοδο, η χρήση του NMPC κερδίζει έδαφος λόγω της **αυξανόμενης διαθεσιμότητας ισχυρών επεξεργαστικών μονάδων και των εξελίξεων σε αλγορίθμους αριθμητικής βελτιστοποίησης**.
Παρακάτω δίνονται ο αλγόριθμος που χρησιμοποιείται στην εργασία [20] καθώς επίσης και μερικά αποτελέσματα των προσημειώσεων που πραγματοποιήθηκαν.

Algorithm 2: Efficient-NMPC algorithm for UAV trajectory

1. **Initialize** $k_{max} = 0$, $t_{prev} = 0$, $x_0 = x(0)$, $U_0 = u(0)$, N_{min}, N_{max}, ΔN;
2. **While** $k < k_{max}$ **do**
3. $k = k + 1$; find $\tilde{\hat{U}}$ by $\left\| F\left(\hat{U}_0, x(t_0), 0\right)\right\| = 0$;
4. **End while**;
5. **While** $t < t_{end}$ **do**
6. $x_{k+1} = x(t + \delta)$ and $\Delta x_k = x_{k+1} - x_k$;
7. $\dot{U}^* = \text{FDGMRES}\left(\hat{U}, x, x/\delta, t, \tilde{\hat{U}}, k_{\max}, \rho, \varepsilon\right)$;
8. $U^* = \hat{U} + \dot{U}^* * \delta$;
9. **If** $\frac{\partial V}{\partial x} f(x(t), u(t)) \leq \frac{\partial V}{\partial x} f(\tilde{x}(t), \zeta(\tilde{x}(t)))$
10. $U = U^*$; **Else** $U = \zeta$;
11. **End If**
12. Decoupling angel $\phi_d = \arcsin\left(\frac{U_{a1} S\psi_d - U_{a2} C\psi_d}{\|U_a\|}\right)$, $\theta_d = \arctan\left(\frac{U_{a1} C\psi_d + U_{a2} S\psi_d}{U_{a3}}\right)$;
13. Attitude controller $\tau_I = K_P^I e_I + K_I^I \int_0^t e_I dt + K_D^I \frac{de_I}{dt}$;
14. **End While**

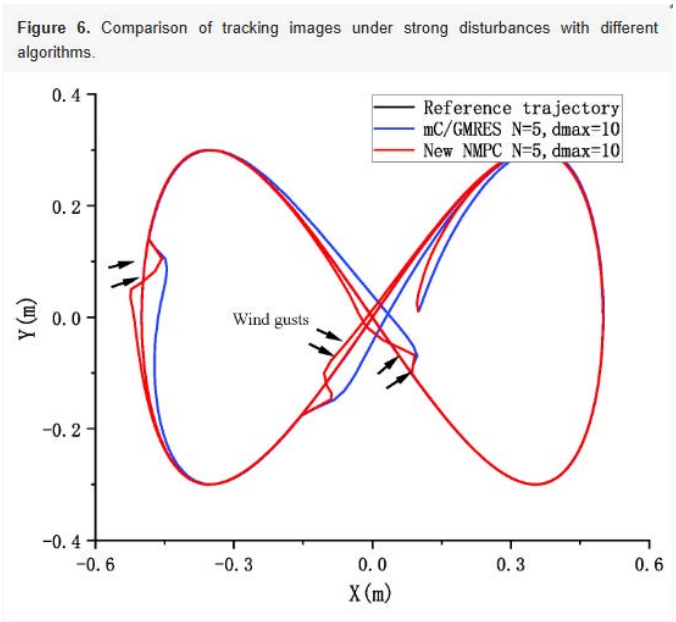
(εικόνα 18)

(**εικόνα 18**: Ο παραπάνω αλγόριθμος ονομάζεται Efficient – NMPC για την ταχεία παρακολούθηση τροχιάς UAV, που συνδυάζει βελτιστοποίηση μέσω FDGMRES με Lyapunov-based περιορισμούς σταθερότητας. [20])



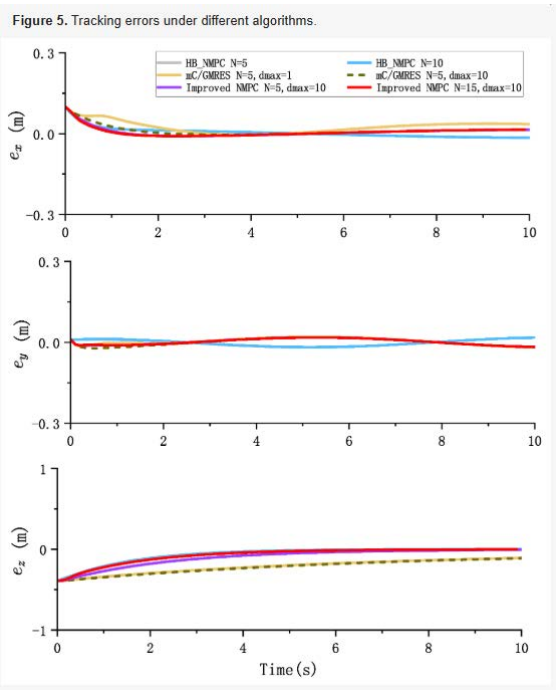
(εικόνα 19)

(**εικόνα 19**: Η **εικόνα 19** παρουσιάζει ένα γράφημα για τη σύγκριση της συνολικής τροχιάς μεταξύ της βελτιωμένης NMPC (μπλε) και του αλγορίθμου mC/GMRES (κόκκινο) σε σχέση με την επιθυμητή τροχιά (μαύρο), όπου η βελτιωμένη NMPC επιτυγχάνει γρηγορότερη και πιο ακριβή παρακολούθηση. [20])



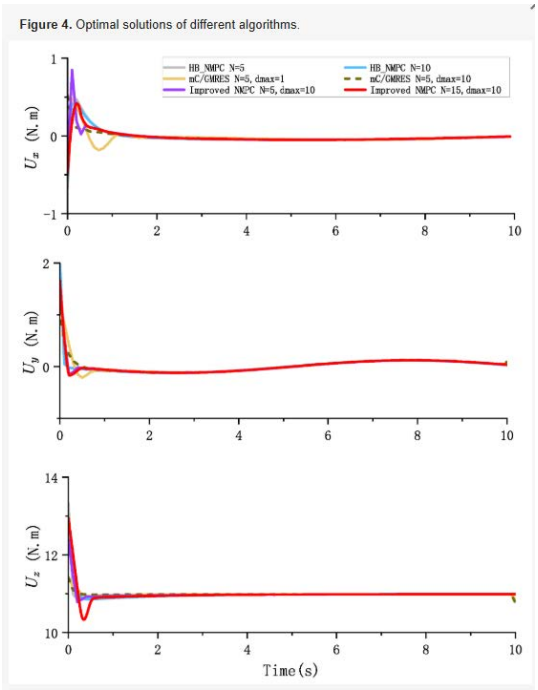
(εικόνα 20)

(**εικόνα 20**: Η **εικόνα 20** παρουσιάζει ένα γράφημα για τη σύγκριση της παρακολούθησης τροχιάς υπό ισχυρές διαταραχές ανέμου, όπου η βελτιωμένη NMPC (κόκκινη γραμμή) παρουσιάζει ταχύτερη ανάρρωση σε σχέση με τον αλγόριθμο mC/GMRES (μπλε γραμμή) [20])



(εικόνα 21)

(**Εικόνα 21**: Η **εικόνα 21** παρουσιάζει ένα γράφημα που περιγράφει τα σφάλματα παρακολούθησης τροχιάς σε διαφορετικούς αλγορίθμους, όπου η βελτιωμένη NMPC επιδεικνύει ταχύτερη σύγκλιση και μικρότερα σφάλματα σε όλες τις συνιστώσες. [20])



(εικόνα 22)

(**Εικόνα 22**: Η **εικόνα 22** παρουσιάζει ένα γράφημα που περιγράφει τις βέλτιστες λύσεις των ελεγκτικών εισόδων (δυνάμεων/ροπών) για διαφορετικούς αλγορίθμους, που δείχνουν την πιο ομαλή και σταθερή απόκριση της βελτιωμένης NMPC. [20])

Η επιλογή ανάμεσα σε Linear και Nonlinear MPC εξαρτάται εν τέλει από τον **βαθμό πολυπλοκότητας του συστήματος, τις απαιτήσεις ακρίβειας και τους υπολογιστικούς πόρους που είναι διαθέσιμοι.**

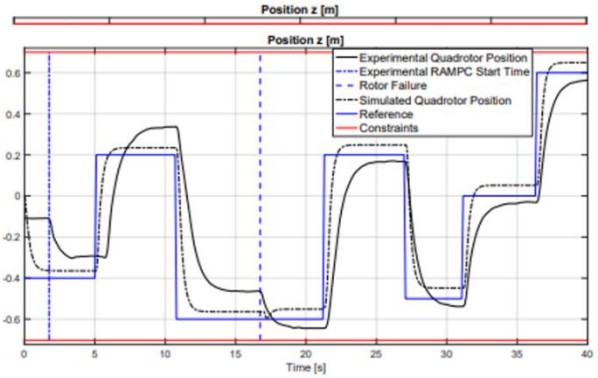
1.4.3 Robust Adaptive MPC

Ο **Ανθεκτικός Προσαρμοστικός Προβλεπτικός Έλεγχος με Βάση Μοντέλο (Robust Adaptive Model Predictive Control – RAMPC)** αποτελεί μια προηγμένη τεχνική ελέγχου που συνδυάζει την ανθεκτικότητα απέναντι σε αβεβαιότητες και διαταραχές με την προσαρμοστικότητα σε μεταβαλλόμενες συνθήκες λειτουργίας. Σε αντίθεση με τον κλασικό Προβλεπτικό Έλεγχο με Βάση Μοντέλο (MPC), ο οποίος βασίζεται σε ένα στατικό μοντέλο του συστήματος, το RAMPC ενσωματώνει μηχανισμούς που επιτρέπουν την online προσαρμογή του μοντέλου, διασφαλίζοντας την ευστάθεια και την απόδοση του συστήματος ακόμη και υπό αβεβαιότητες.
Στην εργασία των [21] Didier, A., Parsi, A., Coulson, J., & Smith, R. S. (2021). Robust Adaptive Model Predictive Control of Quadrotors, παρουσιάζεται η πρώτη πρακτική εφαρμογή του RAMPC σε quadrotor drone, χρησιμοποιώντας διακριτό γραμμικό μοντέλο κατάστασης. Οι συγγραφείς αντιμετωπίζουν προκλήσεις όπως η υπολογιστική πολυπλοκότητα και ο θόρυβος μετρήσεων, προτείνοντας λύσεις που καθιστούν τον αλγόριθμο εφαρμόσιμο σε πραγματικό χρόνο.

$$x_{k+1} = A(\theta)x_k + B(\theta)u_k + w_k,$$

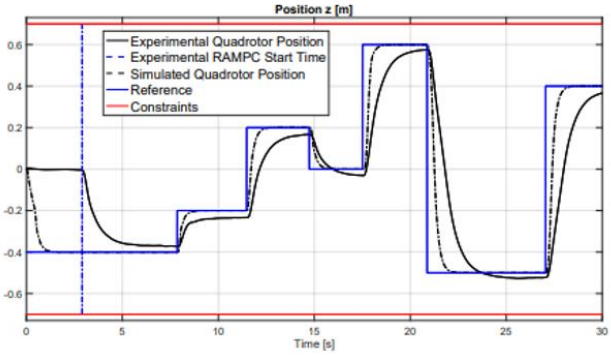
(**equation 9**: Στην παραπάνω εξίσωση περιγράφεται η διακριτή γραμμική δυναμική του συστήματος, όπου η επόμενη κατάσταση xk+1 υπολογίζεται από την τρέχουσα κατάσταση xk, την είσοδο ελέγχου uk και ο θόρυβος wk με πίνακες που εξαρτώνται από την παράμετρο θ. [21])

Συγκεκριμένα, εξετάζονται **δύο σενάρια**: (α) μεταβολές στη μάζα του quadrotor λόγω μεταφοράς φορτίου και παρουσία ανεμογενών διαταραχών, και (β) μείωση της απόδοσης όλων των ελίκων, προσομοιώνοντας προβλήματα στην παροχή ισχύος. Και στις δυο περιπτώσεις που εξετάζονται, το RAMPC επιτυγχάνει αξιόπιστη παρακολούθηση της επιθυμητής τροχιάς, προσαρμοζόμενο στις αβεβαιότητες και διατηρώντας την ευστάθεια του συστήματος.
Η βασική καινοτομία του RAMPC έγκειται στην **ενσωμάτωση μεθόδων ταυτοποίησης παραμέτρων με βάση το σύνολο αποδεκτών τιμών (set-membership identification)**, επιτρέποντας την online ανανέωση του μοντέλου του συστήματος. Επιπλέον, χρησιμοποιείται η τεχνική του σωληνωτού ελέγχου (tube-based control) για τη διατήρηση των καταστάσεων του συστήματος εντός προκαθορισμένων ορίων, παρά τις αβεβαιότητες.



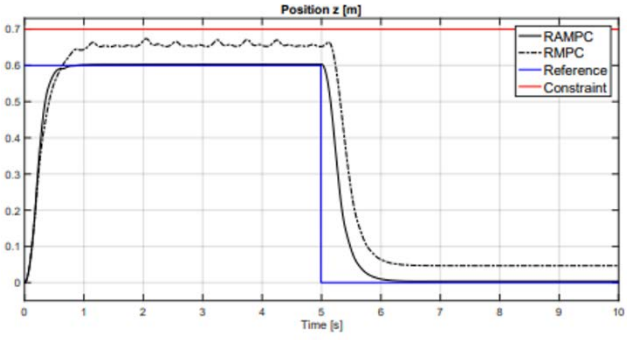
(εικόνα 23)

(**Εικόνα 23**: Η **εικόνα 23** παρουσιάζει ένα γράφημα για τη σύγκριση προσομοίωσης και υλοποίησης του RAMPC για σενάριο βλάβης παροχής ισχύος σε quadrotor, όπου ο ελεγκτής διατηρεί σταθερό τον έλεγχο ύψους παρά την απώλεια ισχύος στους ρότορες. [21])



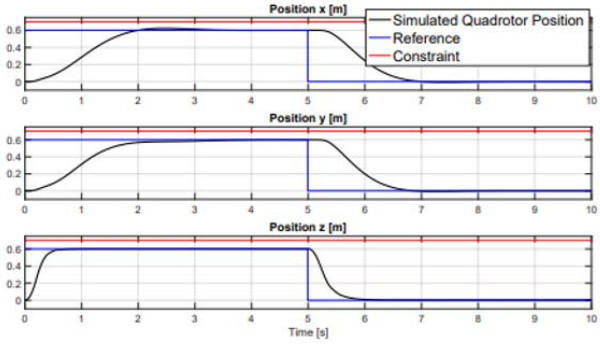
(εικόνα 24)

(**Εικόνα 24**: Η **εικόνα 24** παρουσιάζει ένα γράφημα για τη σύγκριση της απόδοσης μεταξύ Robust MPC και RAMPC σε προσομοίωση quadrotor με αβέβαιη μάζα και παρεμβολή ανέμου, δείχνοντας την παρακολούθηση της θέσης κατά μήκος του άξονα z με περιορισμούς. [21])



(εικόνα 25)

(**Εικόνα 25**: Η **εικόνα 25** παρουσιάζει ένα γράφημα παρόμοιας σύγκρισης όπως στην εικόνα **24**, με έμφαση στη διατήρηση της θέσης και του περιορισμού ύψους κατά τη διάρκεια διαταραχών, επιβεβαιώνοντας τη σταθερότητα του RAMPC. [21])



(εικόνα 26)

(**Εικόνα 26**: Η **εικόνα 26** παρουσιάζει την προσομοίωση του RAMPC εφαρμοσμένου σε quadrotor με αβέβαιη μάζα και σταθερή διαταραχή ανέμου, που απεικονίζει την εξέλιξη των καταστάσεων θέσης στους άξονες x, y και z σε σχέση με τις περιοριστικές συνθήκες και τις επιθυμητές αναφορές. [21])

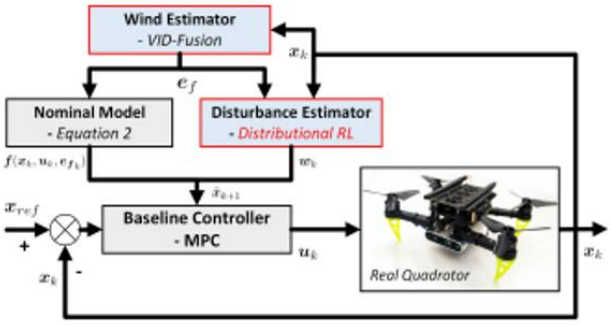
Η εργασία αυτή καταδεικνύει ότι ο RAMPC μπορεί να εφαρμοστεί επιτυχώς σε quadrotor drones, προσφέροντας **αυξημένη ανθεκτικότητα και προσαρμοστικότητα σε μεταβαλλόμενες συνθήκες λειτουργίας, καθιστώντας τον κατάλληλο για απαιτητικές εφαρμογές όπως η αυτόνομη πτήση σε αβέβαια περιβάλλοντα**.

1.4.4 Stochastic MPC

Ο **Στοχαστικός Προβλεπτικός Έλεγχος με Βάση Μοντέλο (Stochastic Model Predictive Control – SMPC)** αποτελεί μια προηγμένη τεχνική ελέγχου που επεκτείνει τον κλασικό Προβλεπτικό Έλεγχο με Βάση Μοντέλο (MPC) ενσωματώνοντας την αβεβαιότητα και τις στοχαστικές διαταραχές που επηρεάζουν τη δυναμική του συστήματος. Σε αντίθεση με τον κλασικό MPC, ο οποίος υποθέτει πλήρη γνώση του μοντέλου και των διαταραχών, ο SMPC λαμβάνει υπόψη την πιθανότητα εμφάνισης διαταραχών και αβεβαιοτήτων, επιτρέποντας τον σχεδιασμό ελεγκτών που είναι πιο ανθεκτικοί σε απρόβλεπτες συνθήκες.

Στην εργασία των [22] Wang, Y., O'Keeffe, J., Qian, Q., & Boyle, D. (2022). Interpretable Stochastic Model Predictive Control using Distributional Reinforced Estimation for Quadrotor Tracking Systems, προτείνεται ένα νέο πλαίσιο ελέγχου για την παρακολούθηση τροχιάς από quadrotor drones σε δυναμικά και πολύπλοκα περιβάλλοντα. Το προτεινόμενο σύστημα ενσωματώνει έναν εκτιμητή διαταραχών βασισμένο σε κατανομή ενισχυτικής μάθησης (Distributional Reinforcement Learning) στον SMPC, επιτρέποντας την ακριβή εκτίμηση των αεροδυναμικών διαταραχών που είναι δύσκολο να μοντελοποιηθούν άμεσα και με ακρίβεια.
Συγκεκριμένα, οι συγγραφείς εισάγουν έναν εκτιμητή διαταραχών με βάση την προσέγγιση ποσοστών (Quantile-approximation-based Distributional Reinforced-disturbance-estimator), ο οποίος επιτρέπει την ακριβή αναγνώριση των αβεβαιοτήτων μεταξύ των πραγματικών και εκτιμώμενων τιμών των αεροδυναμικών επιδράσεων. Επιπλέον, χρησιμοποιείται μια απλοποιημένη παραμετροποίηση ελέγχου με βάση την ανάδραση διαταραχών (Simplified Affine Disturbance Feedback) για τη διασφάλιση της κυρτότητας, η οποία ενσωματώνεται στον SMPC για την επίτευξη επαρκών και μη συντηρητικών σημάτων ελέγχου.

Τα αποτελέσματα της μελέτης δείχνουν ότι το προτεινόμενο σύστημα **βελτιώνει τα σωρευτικά σφάλματα παρακολούθησης κατά τουλάχιστον 66%** σε σύγκριση με τις πρόσφατες προηγμένες μεθόδους, αντιμετωπίζοντας αποτελεσματικά τις άγνωστες και ποικίλες αεροδυναμικές αντιστάσεις. Επιπλέον, παρέχονται θεωρητικές εγγυήσεις για τη **σύγκλιση και την ευστάθεια τόσο της ενισχυτικής μάθησης όσο και του SMPC**, ακόμη και παρουσία διαταραχών με μη μηδενικό μέσο όρο.
Αυτή η προσέγγιση υπογραμμίζει τη σημασία της ενσωμάτωσης τεχνικών ενισχυτικής μάθησης στον σχεδιασμό ελεγκτών για μη επανδρωμένα εναέρια οχήματα, επιτρέποντας **την αντιμετώπιση των αβεβαιοτήτων και τη βελτίωση της απόδοσης σε πραγματικές συνθήκες λειτουργίας**.



(εικόνα 27)

(**Εικόνα 27**: Η **εικόνα 27** παρουσιάζει το συνολικό διάγραμμα ροής του συστήματος QuaDRED-SMPC, που ενσωματώνει έναν εκτιμητή ανέμου (Wind Estimator), έναν εκτιμητή διαταραχών βασισμένο σε distributional reinforcement learning, και έναν βασικό MPC ελεγκτή για τον έλεγχο της πορείας ενός πραγματικού quadrotor. [22])

Algorithm 1 SADF-SMPC

Get:
- the reference data $\boldsymbol{x}_{ref}$ from the quadrotor trajectory planning and generation module, i.e., Kino-JSS [11]
- the measurement state $\boldsymbol{x}_k$ from on-board sensors
- the wind estimation $\boldsymbol{e}_{fk}$ from VID-Fusion [38]

Initialize:
- the parameters θ^{μ} and θ^{Q} for the actor μ and the critic Q, respectively
- the decision variables $\boldsymbol{M}_0$ and $\boldsymbol{v}_0$ in Equation 8
- the initial state $\boldsymbol{s}_0$

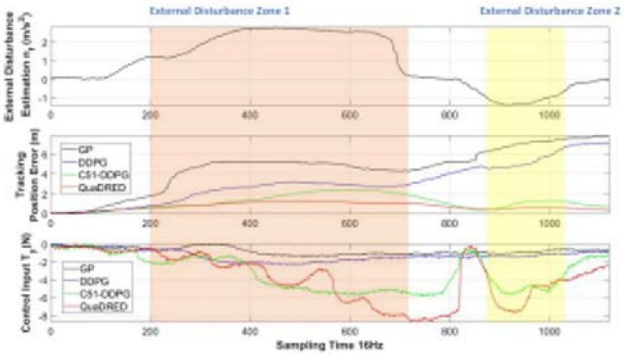
for each sampling timestamps k **do**
  Repeat
  $\boldsymbol{s}_k \leftarrow [\boldsymbol{x}_k, \boldsymbol{e}_{fk}]$
  Select an action vector $\boldsymbol{w}_k \leftarrow [w_{0|k}^{\mathrm{T}}, w_{1|k}^{\mathrm{T}}, ..., w_{N|k}^{\mathrm{T}}]^{\mathrm{T}}$ from $\boldsymbol{w}_k = \boldsymbol{a}_k \leftarrow \mu(\boldsymbol{a}_k|\boldsymbol{s}_k)$ in QuaDRED (Algorithm 2)
  $\boldsymbol{u}_{i|k} \leftarrow \sum_{l=0}^{i-1} \boldsymbol{M}_{i-l|k}\boldsymbol{w}_{l|k} + \boldsymbol{v}_{i|k}$
  $\boldsymbol{u}_k \leftarrow \boldsymbol{u}_{0|k}$, $\boldsymbol{w}_k \leftarrow \boldsymbol{w}_{0|k}$
  $\boldsymbol{x}_k \leftarrow \boldsymbol{A}x_{0|k} + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{G}\boldsymbol{w}_k$
  Solve optimization problem Equation 10 with nonlinear MPC solver
  Until convergence
  $\boldsymbol{u}_k \leftarrow \boldsymbol{v}_{0|k}$
  $\boldsymbol{x}_{k+1}, \boldsymbol{e}_{fk+1} \leftarrow \mathrm{RealQuadrotor}(\boldsymbol{u}_k)$
  $\boldsymbol{s}_{k+1} \leftarrow [\boldsymbol{x}_{k+1}, \boldsymbol{e}_{fk+1}]$
  $\boldsymbol{x}_{ref} \leftarrow$ Kino-JSS
  $k \leftarrow k + 1$
end for

(εικόνα 28)

(**Εικόνα 28**: Στον παραπάνω αλγόριθμο περιγράφεται η διαδικασία SADF-SMPC, όπου παρουσιάζονται τα βήματα για την λήψη δεδομένων, την αρχικοποίηση παραμέτρων, την επιλογή ενεργειών μέσω της πολιτικής που εκπαιδεύεται από το distributional RL, και την επίλυση του προβλήματος βελτιστοποίησης MPC σε κάθε χρονική στιγμή. [22])

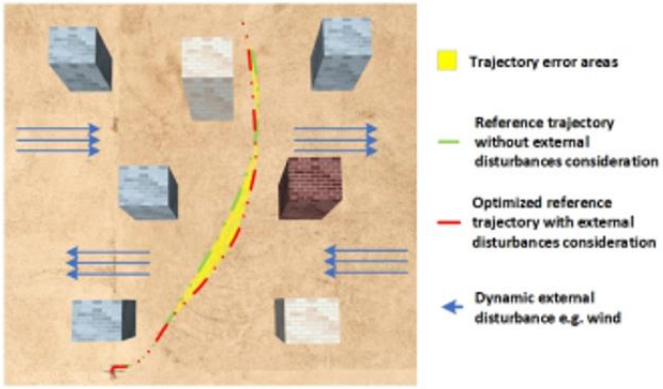
$$
\begin{aligned}
&\min_{\boldsymbol{M}_t, \boldsymbol{v}_t} \mathcal{L}_N(\boldsymbol{x}_t, \boldsymbol{M}_t, \boldsymbol{v}_t), \text{s.t.} \forall w_{i|t} \in \mathbb{W}, \forall i \in \mathbb{N}_{N-1} \\
&\text{subject to} \quad \boldsymbol{x}_t = \boldsymbol{A}x_{0|t} + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{G}\boldsymbol{w}_t \\
&\qquad \boldsymbol{u}_{i|t} = \sum_{k=0}^{i-1} \boldsymbol{M}_{i-k|t}\boldsymbol{w}_{k|t} + \boldsymbol{v}_{i|t} \\
&\qquad H_u \boldsymbol{M}\mathcal{G} + H_w \geq 0 \\
&\qquad (\boldsymbol{x}_t, \boldsymbol{u}_t) \in \mathbb{Z} \\
&\qquad \boldsymbol{x}_{N|t} \in \mathbb{X}_f \\
&\qquad \boldsymbol{x}_{0|t} = \boldsymbol{x}_t
\end{aligned}
$$

(**equation 10**: Η παραπάνω μαθηματική διατύπωση απεικονίζει το βελτιστοποιητικό πρόβλημα του SMPC με παράμετρο SADF, το οποίο ελαχιστοποιεί το κόστος LN υπό περιορισμούς που σχετίζονται με τις διαταραχές, τα όρια στα κράτη και τις εισόδους του συστήματος. [22])



(εικόνα 29)

(**Εικόνα 29**: Στην **εικόνα 29** εμφανίζονται τα αποτελέσματα συγκεκριμένων σεναρίων όπου απεικονίζεται η εκτίμηση ανέμου, το σφάλμα θέσης σε μέτρα και το έλεγχο στην είσοδο Y (εκφρασμένο στο σώμα του quadrotor), συγκρίνοντας διαφορετικές μεθόδους. [22])



(εικόνα 30)

(**Εικόνα 30**: Στην **εικόνα 30** παρουσιάζεται το σενάριο προσομοίωσης που χρησιμοποιείται στο RotorS, όπου φαίνονται οι τροχιές αναφοράς με και χωρίς εξωτερικές δυνάμεις, οι οποίες έχουν παραχθεί μέσω της μεθόδου Kino-JSS. [22])

Συνοψίζοντας, μέσα από την ανάλυση της βασικής αρχιτεκτονικής, των πλεονεκτημάτων και μειονεκτημάτων, καθώς και των διαφορετικών κατηγοριών του MPC, έγινε φανερό ότι η ικανότητα πρόβλεψης της μελλοντικής συμπεριφοράς και η διαχείριση περιορισμών προσφέρουν σημαντικά πλεονεκτήματα σε σχέση με τις κλασικές μεθόδους ελέγχου (όπως το PID). Παρότι το MPC απαιτεί αυξημένους υπολογιστικούς πόρους και κατάλληλο μοντέλο του συστήματος, η εφαρμογή του

στα quadrotors επιτρέπει την υλοποίηση πολύπλοκων αποστολών με ασφάλεια, ακρίβεια και προσαρμοστικότητα.

1.4.5 Gazebo και εφαρμογές με quadrotors

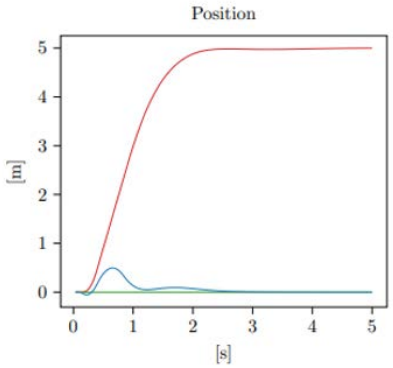
Σε αυτό το υποκεφάλαιο παρουσιάζονται 2 εργασίες που εφαρμόζουν πάνω σε πολυκόπτερα τύπου quadrotor τη μέθοδο MPC στον προσομοιωτή Gazebo με χρήση της γλώσσας python.

- **Εφαρμογή Nonlinear MPC σε quadrotor**

  Στην πτυχιακή εργασία του [25] Jan Hrebec *"Nonlinear Predictive Control of Unmanned Aerial Vehicle in Environments with Obstacles" (2023)*, παρουσιάζεται η ανάγκη για την ανάπτυξη ενός μη γραμμικού προγνωστικού ελεγκτή NMPC που να επιτρέπει σε μη επανδρωμένα εναέρια οχήματα UAVs να εκτελούν ευέλικτους ελιγμούς και να αποφεύγουν εμπόδια σε πραγματικό χρόνο.
  Το κίνητρο για την συγκεκριμένη υλοποίηση αποτελεί το γεγονός πως οι παραδοσιακοί ελεγκτές, όπως οι PID, παρουσιάζουν περιορισμούς στην αντιμετώπιση των μη γραμμικών δυναμικών και των περιορισμών που σχετίζονται με την πτήση UAVs σε περιβάλλοντα με εμπόδια. **Η χρήση του NMPC επιτρέπει την ενσωμάτωση αυτών των παραμέτρων στον σχεδιασμό του ελεγκτή, βελτιώνοντας την απόδοση και την ασφάλεια των πτήσεων**.
  Η εργασία εστιάζει στην ανάπτυξη ενός NMPC που να λαμβάνει υπόψη τα εμπόδια στο περιβάλλον πτήσης, επιτρέποντας στο UAV να εκτελεί ελιγμούς αποφυγής χωρίς να αποκλίνει σημαντικά από την επιθυμητή τροχιά. Η πρόκληση αφορά την ενσωμάτωση των περιορισμών αυτών στον ελεγκτή, διατηρώντας παράλληλα την υπολογιστική αποδοτικότητα που απαιτείται για λειτουργία σε πραγματικό χρόνο.
  Παρακάτω εντοπίζονται μερικά από τα αποτελέσματα κατά τη διάρκεια των προσομοιώσεων που "έτρεξαν".

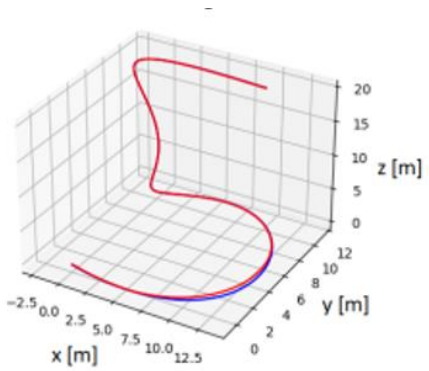
$$\begin{aligned} \min_{\mathbf{u}(t)} \quad & \int_{t_0}^{t_e} f_{\text{cost}}\left(\mathbf{x}(\tau), \mathbf{u}(\tau), \mathbf{y}_{\text{ref}}(\tau)\right) d\tau \\ \text{subject to} \quad & \dot{\mathbf{x}} = \mathbf{f}_{\text{dyn}}(\mathbf{x}, \mathbf{u}) \\ & \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}. \end{aligned}$$

(**equation 11**: Στην παραπάνω μαθηματική διατύπωση παρουσιάζεται το πρόβλημα βελτιστοποίησης που επιλύει ο NMPC, όπου ελαχιστοποιείται η συνάρτηση κόστους που εξαρτάται από την κατάσταση, την είσοδο ελέγχου και την επιθυμητή αναφορά, υπό τους δυναμικούς περιορισμούς του συστήματος. [25])



(εικόνα 31)

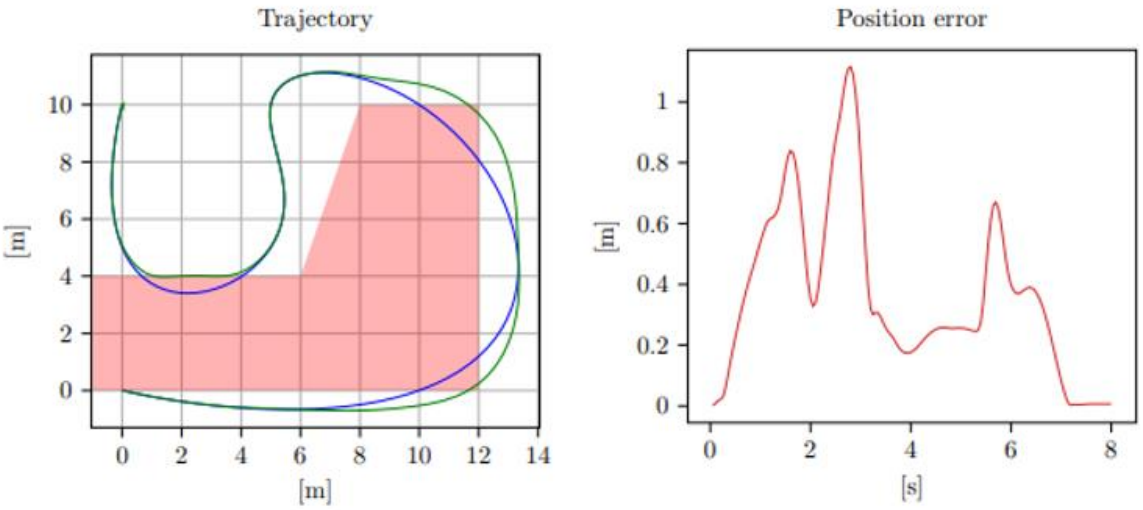
(Εικόνα 31: Στην **εικόνα 31** παρουσιάζεται ένα γράφημα που αφορά την εξέλιξη των συντεταγμένων θέσης x (κόκκινο), y (πράσινο) και z (μπλε) του drone ως απάντηση σε μια βηματική αλλαγή αναφοράς, δείχνοντας την απόδοση του προτεινόμενου NMPC σε απλή προσομοίωση. [25])



(εικόνα 32)

(**Εικόνα 32**: Στην **εικόνα 32** παρουσιάζεται ένα γράφημα που απεικονίζει την επιθυμητή τροχιά (μπλε γραμμή) και της πραγματικής τροχιάς που ακολουθήσε το πολυκόπτερο (κόκκινη γραμμή), υπογραμμίζοντας την ικανότητα του NMPC να παρακολουθεί με ακρίβεια πολύπλοκες τροχιές. [25])

Οι παρακάτω εικόνες δείχνουν και την αποφυγή εμποδίων που πραγματοποιείται :



(εικόνες 32,33)

(**Εικόνες 32,33**: Στις παραπάνω δυο **εικόνες 32 και 33**, παρακολουθείται η τροχιά με περιορισμένη περιοχή (κόκκινη σκιασμένη), όπου η μπλε γραμμή αντιπροσωπεύει την τροχιά αναφοράς και η πράσινη την τροχιά που ακολούθησε το drone. Στο δεξιό γράφημα φαίνεται το σφάλμα θέσης κατά την παρακολούθηση της τροχιάς με τον περιορισμό. [25])

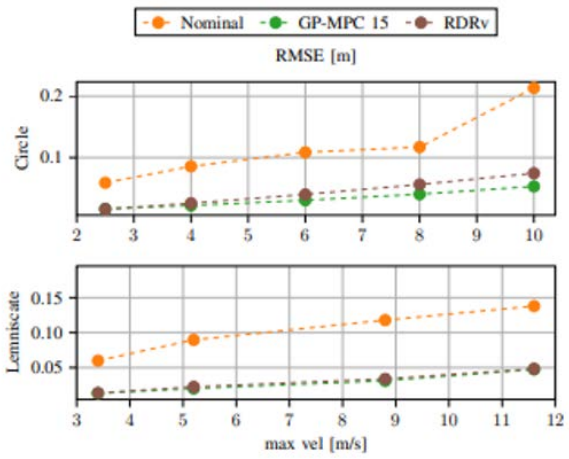
- **Εφαρμογή Linear MPC σε quadrotor**

Στην εργασία των [26] Torrente, A., Kaufmann, E., Loquercio, A., Gehrig, M., Foehn, P., & Scaramuzza, D. (2021). Data-Driven MPC for Quadrotors, παρουσιάζεται μια μέθοδος ελέγχου πολυκοπτέρου τύπου quadrotor που βασίζεται στον Γραμμικό Προγνωστικό Έλεγχο (Linear Model Predictive Control – LMPC), με στόχο την **ακριβή παρακολούθηση τροχιάς ακόμη και σε δυναμικά απαιτητικά σενάρια πτήσης**.
Ο LMPC αξιοποιείται εδώ ως βασικός μηχανισμός λήψης αποφάσεων, με προβλεπτικό ορίζοντα που επιτρέπει τον σχεδιασμό ακολουθίας εντολών, ελαχιστοποιώντας αποκλίσεις από την επιθυμητή τροχιά υπό λειτουργικούς περιορισμούς. Η ακρίβεια του ελέγχου ενισχύεται μέσω της ενσωμάτωσης ενός μαθηματικού μοντέλου που εκπαιδεύεται από δεδομένα πτήσης, ώστε να καλύψει τις ελλείψεις του βασικού μοντέλου δυναμικής.
Η αξιολόγηση της μεθόδου πραγματοποιείται αρχικά στο περιβάλλον προσομοίωσης **Gazebo**, το οποίο χρησιμοποιείται για την ανάπτυξη και δοκιμή του quadrotor σε ελεγχόμενα σενάρια, με ρεαλιστική αναπαράσταση φυσικών φαινομένων όπως η βαρύτητα, η αεροδυναμική τριβή και οι συγκρούσεις. Μέσω της διασύνδεσης με το πακέτο RotorS, το Gazebo επιτρέπει την ακριβή μοντελοποίηση του πολυκόπτερου και τη δοκιμή του LMPC σε συνθήκες που προσομοιάζουν την πραγματική πτήση. Η μεθοδολογία αξιολογείται και σε πραγματικές πτήσεις, όπου η χρήση του LMPC αποδεικνύει εξαιρετική ακρίβεια στην παρακολούθηση τροχιάς, ακόμα και σε υψηλές ταχύτητες και επιταχύνσεις.

Η συγκεκριμένη εργασία αποδεικνύει την αποτελεσματικότητα του LMPC ως μέθοδο ελέγχου για quadrotors, όταν συνδυάζεται με προσομοιωτικά εργαλεία υψηλής πιστότητας όπως το Gazebo. Η ικανότητα του LMPC να ενσωματώνει ρεαλιστικά μοντέλα και να προσαρμόζεται σε μεταβαλλόμενες συνθήκες λειτουργίας τον καθιστά ιδανική επιλογή για ρομποτικά συστήματα όπου απαιτείται ακρίβεια και προβλεψιμότητα.
Παρακάτω δίνονται εικόνες από γραφήματα που αποδεικνύουν την αποτελεσματικότητα της συγκεκριμένης μεθόδου.

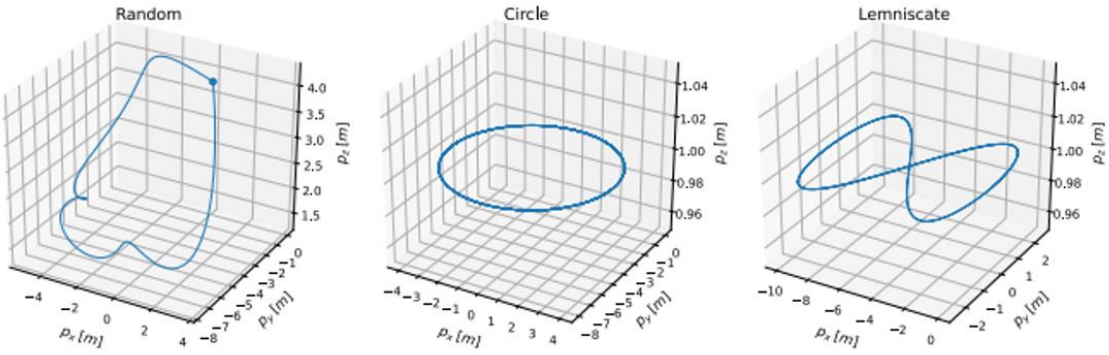
$$\begin{aligned} \min_{u} \; & \boldsymbol{x}_N^T Q \boldsymbol{x}_N + \sum_{k=0}^{N} \boldsymbol{x}_k^T Q \boldsymbol{x}_k + \boldsymbol{u}_k^T R \boldsymbol{u}_k \\ \text{subject to} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}_{RK4}(\boldsymbol{x}_k, \boldsymbol{u}_k, \delta t) \\ & \boldsymbol{x}_0 = \boldsymbol{x}_{init} \\ & u_{min} \leq \boldsymbol{u}_k \leq u_{max} \end{aligned}$$

(**equation 12**: Η παραπάνω μαθηματική διατύπωση παρουσιάζεται το πρόβλημα βελτιστοποίησης του LMPC, όπου η συνάρτηση κόστους ελαχιστοποιείται υπό τους δυναμικούς περιορισμούς του συστήματος, με όρια στα ελεγκτικά σήματα. [26])



(εικόνα 34)

(**Εικόνα 34**: Στην **εικόνα 34** παρουσιάζεται ένα γράφημα που παρουσιάζει το σφάλμα θέσης (RMSE) ως συνάρτηση της μέγιστης ταχύτητας σε προσομοιώσεις με το RotorS Gazebo simulator για τις τροχιές κύκλου και λημνισκάτης, όπου συγκρίνονται διαφορετικοί ελεγκτές. [26])



(εικόνα 35)

(**Εικόνα 35**: Στην **εικόνα 35** παρουσιάζονται τρεις τροχιές που εξετάστηκαν στην εργασία: τυχαία πολυωνυμική (Random), κυκλική (Circle) και λημνισκάτης (Lemniscate), όπου αποτυπώνονται οι θέσεις που πρέπει να ακολουθήσει το quadrotor. [26])

Ο προσομοιωτής Gazebo αναλύεται περισσότερο στο Κεφάλαιο 2.

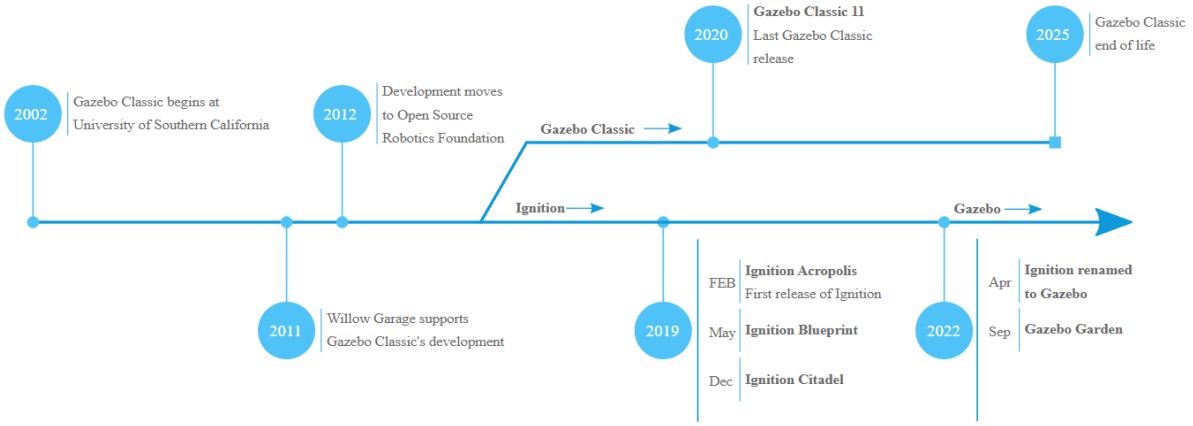
Κεφάλαιο 2 - Περιβάλλον προσομοίωσης

2.1. Λογισμικό προσομοίωσης – ιστορική αναδρομή

Στην παρούσα μελέτη που διεξάχθηκε, χρησιμοποιήθηκε το λογισμικό προσομοίωσης **«Gazebo» και η προγραμματιστική γλώσσα «Python 3» για την επίλυση του προβλήματος.** Στη συνέχεια γίνεται μια ιστορική αναφορά του προσομοιωτή η οποία βρίσκεται διαθέσιμη στην επίσημη σελίδα του Gazebo.

« **Το Gazebo ξεκίνησε την ανάπτυξή του το 2002**. Ύστερα από περισσότερα από 15 χρόνια εξέλιξης, κρίθηκε αναγκαία μια σημαντική αναβάθμιση και εκσυγχρονισμός της πλατφόρμας. Αυτή η αναβάθμιση έδωσε επίσης τη δυνατότητα να εγκαταλειφθεί η μονολιθική αρχιτεκτονική και να υιοθετηθεί μια προσέγγιση που βασίζεται σε μια συλλογή ανεξάρτητων βιβλιοθηκών, οι οποίες συνεργάζονται χαλαρά μεταξύ τους.
Για να διαφοροποιηθεί το νέο αυτό εγχείρημα από το αρχικό Gazebo, το οποίο πλέον αναφέρεται ως **Gazebo Classic**, το νέο έργο έλαβε την ονομασία **Ignition**. Η μετάβαση αυτή σημείωσε μεγάλη επιτυχία, κυρίως χάρη στη συμβολή και τη στήριξη της παγκόσμιας κοινότητας χρηστών και ενδιαφερομένων μερών.
Το 2022 προέκυψε ένα ζήτημα σχετικά με τα εμπορικά δικαιώματα του ονόματος "Ignition". Η εξέλιξη αυτή αποτέλεσε αφορμή για να επιστρέψει το εγχείρημα στην

ευρέως αναγνωρισμένη ονομασία "Gazebo". Έτσι, από το 2022 και έπειτα, η σύγχρονη συλλογή λογισμικού για ρομποτικά συστήματα, που παλαιότερα ήταν γνωστή ως "Ignition", φέρει πλέον και πάλι την επωνυμία **Gazebo** ».
(βλ. Section "History" [23])



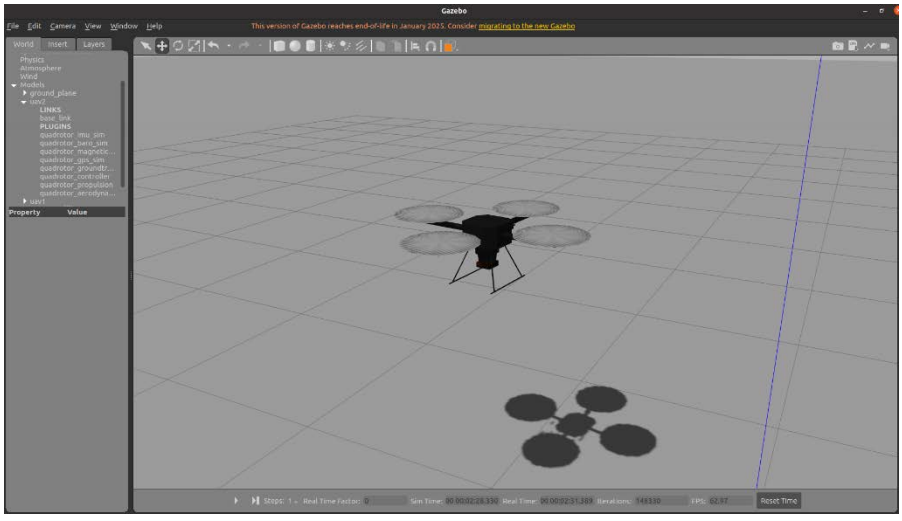
(εικόνα 1)

(**Εικόνα 1**: Στην **εικόνα 1** παρουσιάζεται το ιστορικό ανάπτυξης του Gazebo, από την αρχική έκδοση του 2002 μέχρι και το 2025. Το διάγραμμα με μπλε χρώμα απεικονίζει τις σημαντικές εξελίξεις και ορόσημα, όπως η μετάβαση στο Open Source Robotics Foundation το 2012, η ανάπτυξη της πλατφόρμας Ignition και η επαναφορά του ονόματος Gazebo το 2022, ενώ καταγράφεται και το τέλος ζωής της έκδοσης Gazebo Classic το 2025. [23])

2.1.1 Gazebo – Τι ακριβώς είναι ;

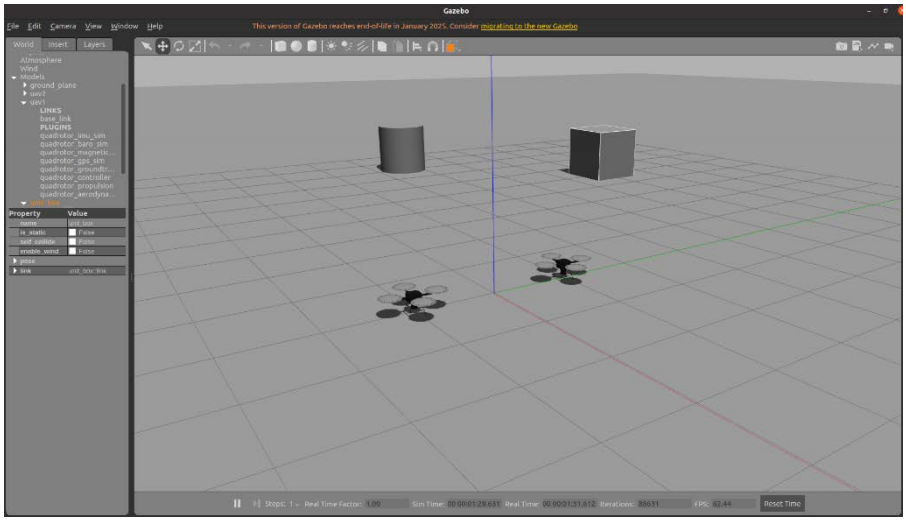
Το Gazebo αποτελεί ένα σύγχρονο και ισχυρό **εργαλείο προσομοίωσης** για τη ρομποτική, το οποίο διατίθεται ελεύθερα ως λογισμικό ανοιχτού κώδικα. Ο σχεδιασμός του επικεντρώνεται στη ρεαλιστική αναπαράσταση τόσο της φυσικής συμπεριφοράς όσο και της αλληλεπίδρασης ρομποτικών συστημάτων με το περιβάλλον τους. Μέσα σε τρισδιάστατους εικονικούς κόσμους που δημιουργούνται στο Gazebo, ο **χρήστης μπορεί να ενσωματώσει πλήθος ρομπότ, αισθητήρων και αντικειμένων**, επιτυγχάνοντας την προσομοίωση ακόμη και πολύπλοκων φυσικών φαινομένων, όπως οι δυνάμεις, οι συγκρούσεις, η τριβή και οι μεταβαλλόμενες περιβαλλοντικές συνθήκες.
Η ευελιξία και η προσαρμοστικότητα του Gazebo οφείλονται στην αρχιτεκτονική του και στη χρήση σύγχρονων βιβλιοθηκών φυσικής, όπως οι ODE, Bullet και DART. Χάρη στη δομή του, κάθε επιμέρους βιβλιοθήκη μπορεί να αξιοποιηθεί αυτόνομα σε διάφορα στάδια ανάπτυξης, από απλούς μαθηματικούς μετασχηματισμούς και κωδικοποίηση βίντεο, μέχρι τη διαχείριση προσομοιώσεων υψηλής πολυπλοκότητας. Ένα **σημαντικό πλεονέκτημα** είναι ότι ο χρήστης μπορεί να επιλέξει μόνο τα στοιχεία του Gazebo που χρειάζεται για το δικό του project, χωρίς να είναι υποχρεωμένος να χρησιμοποιεί ολόκληρη την πλατφόρμα.

Ένα από τα ισχυρά χαρακτηριστικά του Gazebo είναι η στενή διασύνδεσή του με το Robot Operating System (ROS), που διευκολύνει την υλοποίηση, τη δοκιμή και τη βελτιστοποίηση αλγορίθμων ελέγχου και αντίληψης, χωρίς να απαιτείται πραγματικό υλικό. Το Gazebo υιοθετεί πρακτικές ανάπτυξης που διασφαλίζουν ότι το λογισμικό μπορεί να ενημερώνεται, να βελτιώνεται και να διορθώνεται εύκολα, ακόμα και με την πάροδο του χρόνου, με αυστηρούς ελέγχους και συνεχή αναβάθμιση, ενώ προσφέρει ευχρηστία τόσο για αρχάριους όσο και για προχωρημένους χρήστες.
Η αξιοπιστία του Gazebo έχει αναδειχθεί μέσα από τη μακρόχρονη χρήση του στην ερευνητική κοινότητα και τη βιομηχανία. Επιτρέπει τη δοκιμή και τη βελτιστοποίηση σύνθετων συστημάτων ελέγχου, όπως αλγορίθμων Model Predictive Control (MPC) σε εφαρμογές quadrotor, σε ελεγχόμενο και ασφαλές περιβάλλον προτού οι λύσεις μεταφερθούν σε πραγματικές συνθήκες. Έτσι, το Gazebo συμβάλλει ουσιαστικά στην επιτάχυνση της έρευνας και της ανάπτυξης προηγμένων ρομποτικών εφαρμογών.
Οι παραπάνω πληροφορίες αναφέρονται στην επίσημη σελίδα του Gazebo [23] καθώς επίσης και στο Wikipedia. [24]



(εικόνα 2)

(**εικόνα 2**: Στην **εικόνα 2** παρουσιάζεται το περιβάλλον του προσομοιωτή Gazebo και το quadrotor πολυκόπτερο της μελέτης μου)



(εικόνα 3)

(**Εικόνα 3**: Στην **εικόνα 3** , παρόμοια με την **εικόνα 2**, παρουσιάζεται ο κόσμος στον προσομοιωτή Gazebo με δυο quadrotors και δυο εμπόδια με διαφορετικό σχήμα στο βάθος.)

2.2 Πακέτο hector_quadrotor_noetic

Στον παρόν υποκεφάλαιο 2.2 θα ειπωθούν μερικά λογία για το repository το οποίο χρησιμοποιήθηκε στη συγκεκριμένη μελέτη ως προς την ανάπτυξη αλγορίθμου προβλεπτικού ελέγχου για την αυτόνομη κίνηση ενός ρομποτικού πολυκοπτέρου τύπου quadrotor.

Το repository **hector_quadrotor_noetic** [27] αποτελεί μια ολοκληρωμένη συλλογή πακέτων ROS σχεδιασμένη για την προσομοίωση, τον έλεγχο και την πλοήγηση quadrotors στο περιβάλλον ROS Noetic. Το πακέτο παρέχει ακριβή μοντέλα δυναμικής του quadrotor, υποστηρίζει διάφορους αισθητήρες όπως IMU, LIDAR και κάμερες, και ενσωματώνει βασικούς ελεγκτές για την επίτευξη σταθερής και ρεαλιστικής πτήσης.
Η δομή του repository είναι οργανωμένη ώστε να διευκολύνει την ανάπτυξη και πειραματισμό με προηγμένους αλγορίθμους ελέγχου και πλοήγησης, επιτρέποντας την εύκολη ενσωμάτωση νέων ελεγκτών ή αισθητήρων. Προσφέρει ένα πλήρως διαδραστικό περιβάλλον προσομοίωσης βασισμένο στο Gazebo. Επιπλέον περιλαμβάνει έτοιμες ρυθμίσεις και παραμέτρους για το Gazebo, επιτρέποντας την εκτέλεση ρεαλιστικών σεναρίων πτήσης και τη δοκιμή σε εικονικούς χώρους με εμπόδια, καθιστώντας την προσομοίωση και τη δοκιμή εφικτή χωρίς τη χρήση πραγματικού υλικού.

Ένα από τα βασικά πλεονεκτήματα του repository είναι η υποστήριξη της διαχείρισης φυσικών περιορισμών, όπως κορεσμοί στους κινητήρες και τα όρια λειτουργίας, που εξασφαλίζουν ασφαλή και ρεαλιστική λειτουργία κατά την προσομοίωση.
Η χρήση του **hector_quadrotor_noetic** στην παρούσα εργασία ήταν καθοριστική για την ανάπτυξη ενός αλγορίθμου Model Predictive Control (MPC). Με τη βοήθεια του πακέτου, πραγματοποιήθηκαν προσομοιώσεις που αποτύπωσαν με ακρίβεια τη δυναμική συμπεριφορά του quadrotor υπό πραγματικές συνθήκες, συμπεριλαμβανομένων περιορισμών.
Συνολικά, το **hector_quadrotor_noetic** αποτελεί ένα πολύτιμο εργαλείο για την έρευνα και την ανάπτυξη ελέγχου μη επανδρωμένων εναέριων οχημάτων, ειδικά όταν συνδυάζεται με περιβάλλοντα προσομοίωσης όπως το Gazebo και το ROS.

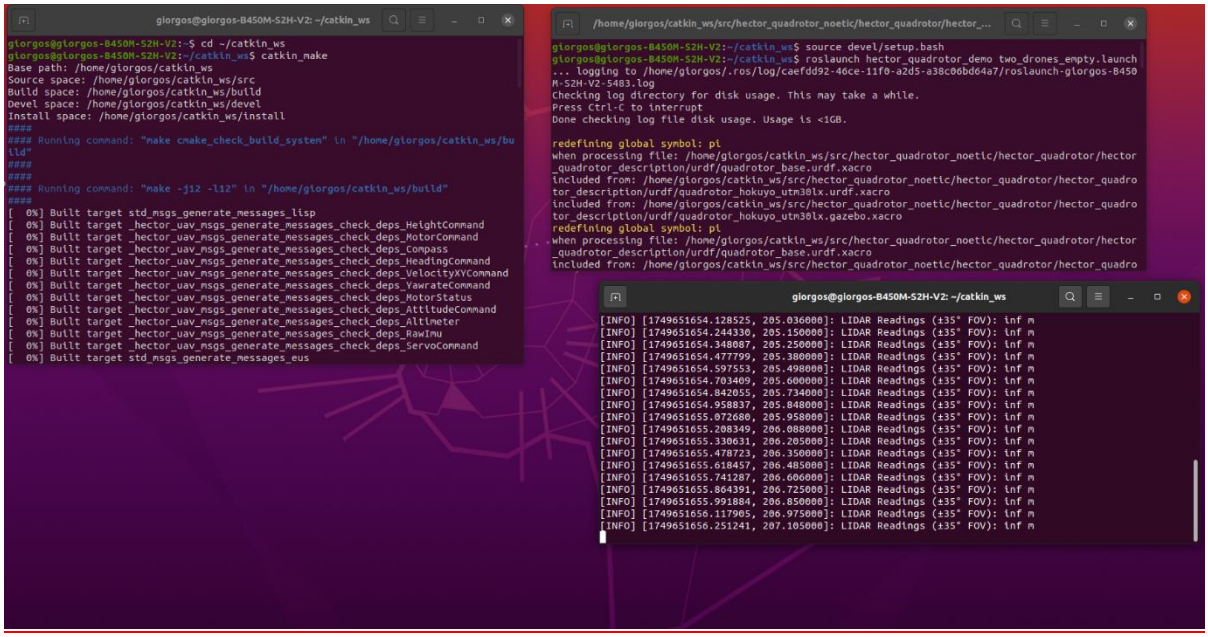
2.2.1 Εγκατάσταση hector_quadrotor_noetic

Η εργασία ξεκίνησε με τη δημιουργία ενός νέου ROS workspace σε λειτουργικό σύστημα **ubuntu 22.04** χρησιμοποιώντας την εντολή **mkdir -p ~/catkin_ws/src**. Το επόμενο βήμα ήταν η μεταφορά στον φάκελο **/src** με χρήση της εντολής **cd** . Έπειτα ακολουθήθηκε ολόκληρη η διαδικασία εγκατάστασης των απαιτούμενων πακέτων όπως διευκρινίζεται και στο **repository**. Τέλος αφού έγινε η επιτυχής ολοκλήρωση της εγκατάστασης των πακέτων, με τη χρήση της εντολής **catkin_make** (cd ~/catkin_ws && catkin_make) δημιούργησα το ROS πακέτο (“building package”).
Τέλος για την ενεργοποίηση του workspace χρησιμοποίησα την εντολή **source devel/setup bash**. Αφού πραγματοποιήθηκαν όλα τα προηγούμενα βήματα με επιτυχία ήμουν σε θέση να εκτελέσω τις απαραίτητες εντολές **roslaunch** όπως δίνονται στο repository. Συγκεκριμένα εργάστηκα με τις εντολές :

1) roslaunch hector_quadrotor_demo two_drones_empty.launch

2) rosrun hector_ui ui_hector_quad_leader.py

3) rosrun hector_ui ui_hector_quad_follower.py



(εικόνα 4)

(**Εικόνα 4**: Στην **εικόνα 4** παρουσιάζονται οι βασικές εντολές οι οποίες είναι υπεύθυνες για το άνοιγμα του προσομοιωτή με τις απαραίτητες παραμέτρους του, τη δημιουργία του ROS πακέτου και το build του workspace καθώς και το execution του κώδικα βλέποντας τα μηνύματα εξόδου που αφορούν το διάβασμα από τον αισθητήρα lidar. [27])

Κατά τη διάρκεια ανάπτυξης του κώδικα, τροποποιήθηκε το αρχείο : quadrotor_hokuyo_utm30lx.urdf.xarco, ώστε να μειωθεί η **εμβέλεια** (μοίρες) του αισθητήρα. Η εφαρμογή της αλλαγής αποτελεί βοήθεια για τη συνολική συμπεριφορά του πολυκοπτέρου κατά τον εντοπισμό ενός εμποδίου, στατικού ή δυναμικού.

(εικόνα 5)

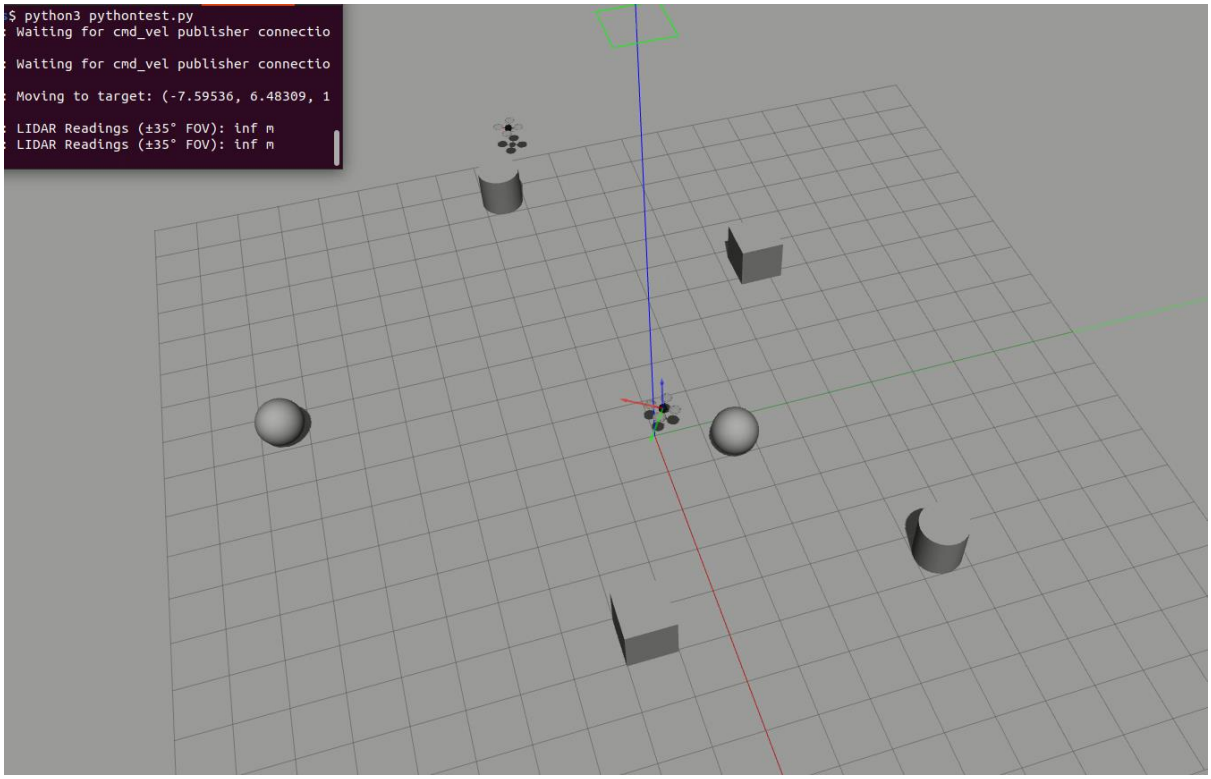
(**Εικόνα 5**: Στην **εικόνα 5**, παρουσιάζεται το αρχείο που αφορά τον αισθητήρα που ελέγχει την εμβέλεια του laser scan σε μοίρες. Το αρχικό αρχείο είχε τιμές σε min και max angle = - 135 / 135 και με την τροποποίηση η εμβέλεια έπεσε στις -35 / 35 μοίρες)

2.3 Πρόβλημα βελτιστοποίησης, επίλυση, map

Στη συνέχεια του κεφαλαίου 2 και συγκεκριμένα στο υποκεφάλαιο 2.3 θα περιγράφει και θα αναλυθεί ο **χάρτης της προσομοίωσης** (map) και πως αυτός είναι διαμορφωμένος σε διαφορετικά σενάρια παρουσιάζοντας μερικά βίντεο και εικόνες του κόσμου, το **πρόβλημα βελτιστοποίησης** στη συγκεκριμένη μελέτη και ο επιλεγμένος **τρόπος** με τον οποίο επιλύθηκε το πρόβλημα βελτιστοποίησης.

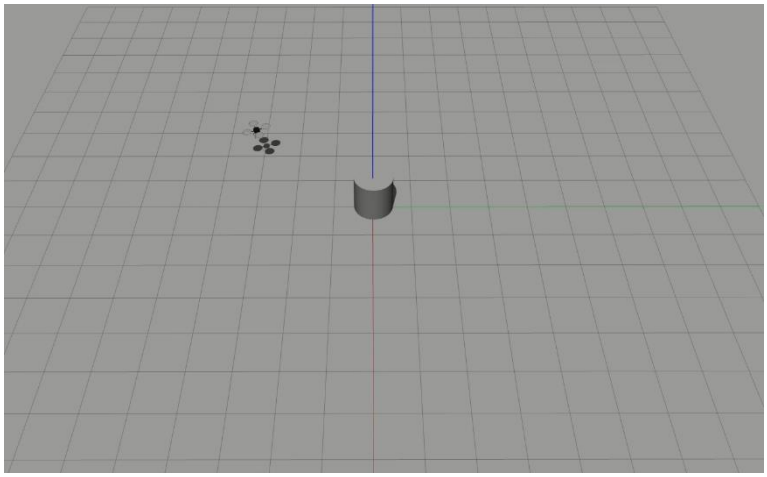
2.3.1 Ο χάρτης της προσομοίωσης

Στη παρούσα εργασία, ο κόσμος της προσομοίωσης αποτελείται από έναν κενό (empty) χώρο με επίπεδο δάπεδο και βασικά γεωμετρικά εμπόδια, όπως κύβους και κυλίνδρους, που χρησιμεύουν ως στατικά αντικείμενα προς αποφυγή.
Στον κόσμο τοποθετούνται δύο μη επανδρωμένα εναέρια οχήματα (UAVs). Το πρώτο UAV (**uav1**) αποτελεί το κύριο όχημα στο οποίο υλοποιείται και εφαρμόζεται ο αλγόριθμος Model Predictive Control (MPC), ο οποίος υπολογίζει σε πραγματικό χρόνο τις βέλτιστες εντολές κίνησης λαμβάνοντας υπόψη την τρέχουσα θέση, τον προσανατολισμό και τις αποστάσεις από τα εμπόδια μέσω αισθητήρων LIDAR. Το δεύτερο UAV (**uav2**) λειτουργεί ως δυναμικό εμπόδιο, κινείται απρόβλεπτα στον χώρο χωρίς να διαθέτει κάποιον αλγόριθμο ελέγχου ή συνεργασίας, προσφέροντας ρεαλιστικές προκλήσεις στην πλοήγηση του uav1.
Η παρακολούθηση της θέσης και του προσανατολισμού του uav1 πραγματοποιείται μέσω των ROS topics που συνδέονται με το Gazebo **(/gazebo/model_states)**, ενώ τα δεδομένα από τον αισθητήρα LIDAR συλλέγονται από το topic **/uav1/scan**. Ο κώδικας Python που υλοποιήθηκε χρησιμοποιεί τη βιβλιοθήκη **Casadi** για την επίλυση του προβλήματος βελτιστοποίησης του MPC και αποστέλλει εντολές κίνησης μέσω του topic **/uav1/cmd_vel**.
Ο κόσμος προσομοίωσης, με την παρουσία τόσο στατικών όσο και δυναμικών εμποδίων, επιτρέπει την πλήρη αξιολόγηση της αποτελεσματικότητας του ελεγκτή MPC σε ρεαλιστικές συνθήκες. Τα βίντεο που καταγράφηκαν αποτυπώνουν ποικίλες περιπτώσεις πτήσης, από απλή κίνηση σε ανοιχτό χώρο, μέχρι προχωρημένες καταστάσεις αποφυγής εμποδίων με ελιγμούς και στάσεις.
Συνολικά, η διαμόρφωση του κόσμου στο Gazebo, σε συνδυασμό με την υλοποίηση του MPC μέσω του Python κώδικα και του πακέτου **hector_quadrotor_noetic**, παρέχει ένα αξιόπιστο και ευέλικτο περιβάλλον προσομοίωσης για τη μελέτη και ανάπτυξη ελέγχου μη επανδρωμένων εναέριων οχημάτων.



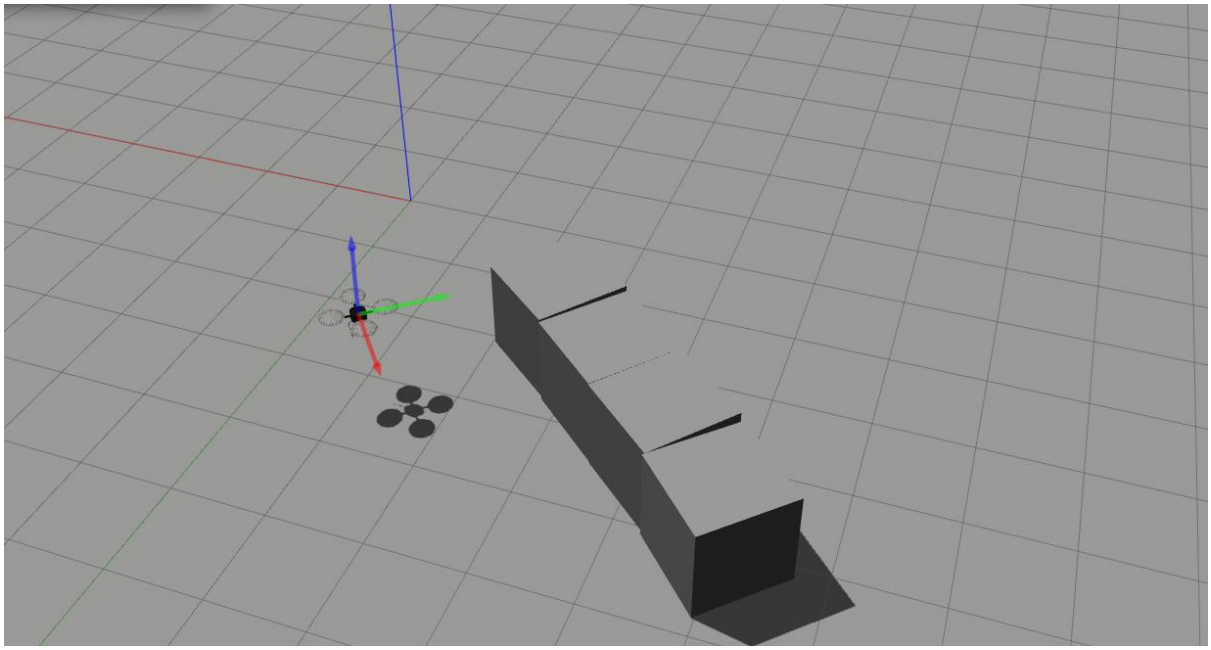
(εικόνα 6)

(**Εικόνα 6**: Στην **εικόνα 6** παρουσιάζεται το συνολικό σενάριο χάρτη προσομοίωσης. Ο χάρτης αναδεικνύει το βασικό drone uav1 στο κέντρο του επιπέδου με πολλαπλά στατικά εμπόδια διαφορετικού σχήματος να βρίσκονται γύρω από το uav1 καθώς και ένα δυναμικό εμπόδιο το οποίο αποτελεί το δεύτερο drone της προσομοίωσης. Στην εικόνα , φαίνεται επίσης και μηνύματα εξόδου από την χρήση του κυρίως κώδικα πάνω στο uav1.)



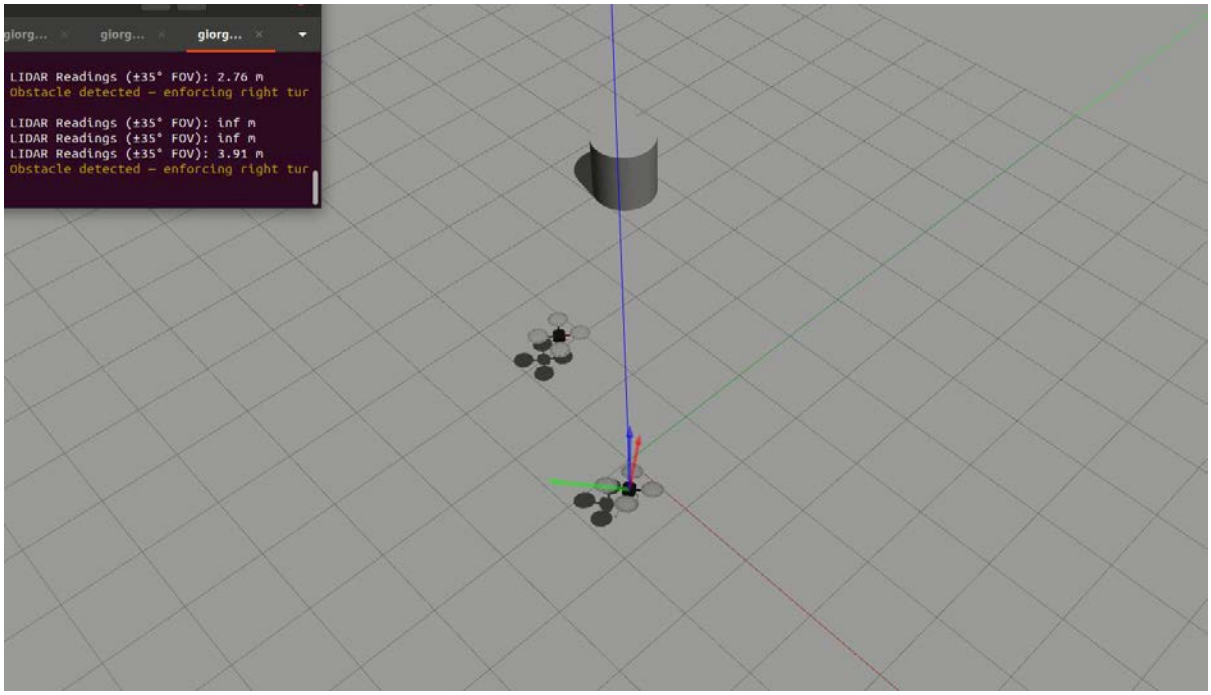
(εικόνα 7)

(**Εικόνα 7**: Στην **εικόνα 7** παρουσιάζεται ένα διαφορετικό σενάριο, πιο απλό όπου το uav1 έχει να αντιμετωπίσει ένα εμπόδιο κυλινδρικού σχήματος το οποίο βρίσκεται στο κέντρο του επιπέδου.)



(εικόνα 8)

(**Εικόνα 8**: Στην **εικόνα 8** παρουσιάζεται ένα σενάριο το οποίο περιέχει ένα ενιαίο τοίχος φτιαγμένος από μικρότερα εμπόδια τετραγωνικού σχήματος. Το Drone καλείται να εφαρμόσει συνεχή αποφυγή.)



(εικόνα 9)

(**Εικόνα 9**: Το σενάριο στην **εικόνα 9** αποτελεί ένα συνθετικό πιο προχωρημένο σενάριο, παρουσιάζοντας το uav1 και το δυναμικό εμπόδιο uav2 καθώς και ένα στατικό εμπόδιο κυλινδρικού σχήματος. Το uav1 καλείται να αποφύγει τη σύγκρουση με το δυναμικό και αμέσως μετά με το στατικό εμπόδιο.)

Στο επόμενο υποκεφάλαιο θα παρουσιαστούν video τα οποία δείχνουν σε πραγματικό χρόνο την λειτουργία του drone uav1.

2.3.2 Ο χάρτης της προσομοίωσης - video

Όπως αναφέρθηκε στο προηγούμενο υποκεφάλαιο, στο υποκεφάλαιο 2.3.2 θα παρουσιαστούν video τα οποία παρουσιάζουν τη συμπεριφορά και την αλληλεπίδραση του κυρίως drone uav1 με το περιβάλλον του όπως με δυναμικά, στατικά εμπόδια αλλά και χωρίς.

- Στο πρώτο video ο **uav1** κινείται μέσα σε χώρο με τέσσερα στατικά εμπόδια κυλινδρικής μορφής. Ο έλεγχος βασίζεται στην απόσταση που μετρά το lidar, και όταν τα εμπόδια πλησιάζουν εντός των ορίων, επιβάλλεται περιστροφή προς τα δεξιά για αποφυγή. Η παράλληλη χρήση MPC επιτρέπει την ακριβή τήρηση της βασικής πορείας, με το σύστημα αποφυγής να λειτουργεί ως προσαρμοστικό φίλτρο για την αποφυγή συγκρούσεων. Σε όλη τη διάρκεια του video παρατηρούμε πάνω αριστερά τα μηνύματα εξόδου που λαμβάνουμε για όταν εντοπίζει εμπόδιο ή όχι το drone :

Simple 4 obstacles avoidance for graph

Video 1

- Στο δεύτερο video, το **uav1** συναντά ένα στατικό εμπόδιο σε μορφή τοίχου. Η απόσταση lidar ανιχνεύει το εμπόδιο και επιβάλλει στο σύστημα περιορισμό στροφής, με το uav1 να εκτελεί ομαλή παράκαμψη. Παράλληλα, ο MPC συνεχίζει να υπολογίζει τη βέλτιστη πορεία προς τον στόχο, δείχνοντας ότι η αποφυγή εμποδίων γίνεται μέσω συνδυασμού MPC και μετρήσεων από το lidar. Σε όλη τη διάρκεια του video παρατηρούμε πάνω αριστερά τα μηνύματα εξόδου που λαμβάνουμε για όταν εντοπίζει εμπόδιο ή όχι το drone :

Obstacle_wall avoidance (1).mp4

Video 2

- Στο τρίτο video, το **uav1** κινείται ελεύθερα προς προκαθορισμένους στόχους σε περιβάλλον χωρίς εμπόδια. Η λειτουργία MPC εξασφαλίζει ομαλή και ακριβή παρακολούθηση τροχιάς, ενώ δεν απαιτείται κανένα επιπλέον μέτρο αποφυγής, δίνοντας έμφαση στην ακριβή εκτέλεση της κίνησης. Σε όλη τη διάρκεια του video παρατηρούμε κάτω δεξιά τα μηνύματα εξόδου που λαμβάνουμε για όταν εντοπίζει εμπόδιο ή όχι το drone :

MoveToTarget_0
obstacles for graphs

Video 3

- Στο τέταρτο video, παρόμοια με το προηγούμενο, το **uav1** πλοηγείται με ακρίβεια χωρίς εμπόδια. Αυτό το σενάριο χρησιμεύει ως βάση σύγκρισης, επιβεβαιώνοντας την ορθή λειτουργία του MPC σε συνθήκες ελεύθερου πεδίου. Σε όλη τη διάρκεια του video παρατηρούμε κάτω αριστερά τα μηνύματα εξόδου που λαμβάνουμε για όταν εντοπίζει εμπόδιο ή όχι το drone :

MoveToTarget_0
obstacles for graphs

Video 4

- Στο πέμπτο video, λειτουργεί στο ίδιο σενάριο στόχων με το video 4. Η λειτουργία αποφυγής εμποδίων ενεργοποιείται και απενεργοποιείται κατά περίπτωση, επιτρέποντας τη σύγκριση της συμπεριφοράς του **uav1** με και χωρίς τον επιπλέον έλεγχο αποφυγής. Όταν η αποφυγή είναι ενεργή, το **uav1** προσαρμόζει την τροχιά του ομαλά ώστε να αποφύγει εμπόδια, ενώ όταν είναι απενεργοποιημένη, το **uav1** κινείται ευθέως προς τον στόχο χωρίς προσαρμογές. Προς το τέλος του video 5, παρατηρείται αρχικά μια παύση του drone δείχνοντας ότι υπάρχει ένα εμπόδιο το οποίο είναι στατικό. Το Drone έχει ως τελικό στόχο το κέντρο του χώρου. Αρά έτσι διασφαλίζεται ότι αν το βρεθεί ένα στατικό εμπόδιο μπροστά στο drone και βρίσκεται πάνω στον τελικό στόχο τότε το drone σταματάει και δε συγκρούεται με το στατικό εμπόδιο (αυτό φαίνεται παράλληλα και στα μηνύματα εξόδου κάτω αριστερά του video). Τέλος παρατηρούνται επίσης κάποια διαγράμματα αποτελεσμάτων τα οποία θα αναλυθούν στο τέταρτο κεφάλαιο της πτυχιακής εργασίας.

enable_disabling_o
bstacle for movTOta

Video 5

- Στο <u>έκτο video</u>, το **uav1** κινείται προς τον προκαθορισμένο στόχο και παρακολουθεί διαρκώς τις αποστάσεις μπροστά του μέσω του αισθητήρα lidar. Όταν το lidar ανιχνεύσει το δυναμικό εμπόδιο τύπου drone σε απόσταση μικρότερη από το όριο που έχει τεθεί, η λογική ελέγχου εφαρμόζει έναν προκαθορισμένο περιορισμό στροφής προς τα δεξιά, αποφεύγοντας έτσι το εμπόδιο με απλό και ασφαλή τρόπο. Η απόκριση αυτή είναι ανεξάρτητη από τον MPC, ο οποίος συνεχίζει να υπολογίζει τη βέλτιστη κίνηση προς τον στόχο :

Dynamic obstacle
avoidance_basic avo

Video 6

- Στο <u>έβδομο video</u>, η κατάσταση είναι παρόμοια με το video 6. Η διαφορά είναι ότι σε αυτή την περίπτωση παρουσιάζεται μια επιπλέον λειτουργία για την αντιμετώπιση πιθανής σύγκρουσης με ένα δυναμικό εμπόδιο (λειτουργεί και με στατικά εμπόδια). Η λειτουργεία αυτή αποτελεί μια παύση του drone, μηδενίζοντας όλες τις ταχύτητες για κάποιο χρονικό διάστημα, εφαρμόζει μετά μια μικρή περιστροφή προς τα δεξιά και αφού ανιχνευθεί από το lidar πως δεν υπάρχει πλέον εμπόδιο μπροστά από το drone, τότε το drone συνεχίζει με τη μέθοδο MPC να κινείται προς τον επιθυμητό στόχο. Το αποτέλεσμα φαίνεται και στα μηνύματα εξόδου πάνω αριστερά του video :

dynamic obstacle
avoidance_critical st

Video 7

- Στο <u>όγδοο, ένατο και δέκατο video</u>, παρουσιάζονται 3 διαφορετικές περιπτώσεις στις οποίες το uav1- drone καλείται να “αντιμετωπίσει” πιο προχωρημένες καταστάσεις εμποδίων. Αυτές οι περιπτώσεις παρέχουν μια εικόνα του χώρου που περιλαμβάνει πολλαπλά εμπόδια διαφορετικού σχήματος και όγκου. Ταυτόχρονα υπάρχουν προχωρημένοι συνδυασμοί εμποδίων. Στο παρακάτω video παρατηρείται ένας συνδυασμός δυναμικού εμποδίου με στατικό εμπόδιο. Το drone καλείται να αντιμετωπίσει το εμπόδιο προχωρημένου επιπέδου και να καταλήξει στον επιθυμητό στόχο του.

Advanced dynamic obstacle avoidance

Video 8

Advanced obstacle avoidance 2 (1).mp4

Video 9

Advanced obstacle avoidance 3 (1).mp4

Video 10

2.3.3 Ποιο είναι το πρόβλημα βελτιστοποίησης ;

Το πρόβλημα που αντιμετωπίζεται αφορά τον υπολογισμό της βέλτιστης ακολουθίας εντολών κίνησης για το UAV (uav1), έτσι ώστε να μετακινηθεί με ακρίβεια προς προκαθορισμένους στόχους, λαμβάνοντας υπόψη τις φυσικές του περιοριστικές συνθήκες. Στην υλοποίηση αυτή, το UAV μοντελοποιείται με ένα **κινηματικό μοντέλο μονόκυκλου**, το οποίο απλοποιεί τη δυναμική του οχήματος περιορίζοντας το σε τέσσερις καταστάσεις (x, y, z, yaw) και τρεις ελεγκτικές εισόδους (εμπρόσθια ταχύτητα, κάθετη ταχύτητα και γωνιακή ταχύτητα). Η επιλογή αυτού του απλουστευμένου μοντέλου ήταν συνειδητή, προκειμένου να διατηρηθεί η υπολογιστική αποδοτικότητα και η ευκολία εφαρμογής του προγνωστικού ελεγκτή. Παράλληλα, το σύστημα πρέπει να λαμβάνει υπόψη την παρουσία εμποδίων που ανιχνεύονται από τον αισθητήρα LIDAR, ώστε να διασφαλίζει την αποφυγή συγκρούσεων.

Παρακάτω παρουσιάζεται η μαθηματική διατύπωση του προβλήματος βελτιστοποίησης MPC :

Έστω $x_t = [x_t, y_t, z_t, \psi_t]^T$ η κατάσταση του UAV στο χρόνο **t**, όπου xt,yt,zt οι τρισδιάστατες συντεταγμένες και **ψt** η γωνιά προσανατολισμού (yaw). Οι μεταβλητές ελέγχου είναι $u_t = [v_t, v_{z,t}, \omega_t]^T$ όπου **ut** η εμπρόσθια ταχύτητα, **uzt** η κάθετη ταχύτητα και **ωt** η γωνιακή ταχύτητα yaw.
Το κινηματικό μοντέλο μονοκύκλου περιγράφεται διακριτά ως :

$$\begin{cases} x_{t+1} = x_t + \Delta t \cdot v_t \cos(\psi_t) \\ y_{t+1} = y_t + \Delta t \cdot v_t \sin(\psi_t) \\ z_{t+1} = z_t + \Delta t \cdot v_{z,t} \\ \psi_{t+1} = \psi_t + \Delta t \cdot \omega_t \end{cases}$$

όπου :

- x_t, y_t, z_t: Θέσεις του drone στον χώρο κατά τον χρόνο t, στους άξονες X, Y, Z αντίστοιχα.
- ψ_t : Γωνία προσανατολισμού (yaw) του drone στο επίπεδο XY, δηλαδή η γωνία που δείχνει προς τα πού κοιτάει το drone στο οριζόντιο επίπεδο.
- Δt: Χρονικό βήμα, δηλαδή το διάστημα χρόνου μεταξύ των διαδοχικών βημάτων t και t +1.
- v_t : Εμπρόσθια ταχύτητα του drone στον οριζόντιο επίπεδο (ταχύτητα κινήσεως κατά τη διεύθυνση που κοιτάει το drone).
- $v_t \cos(\psi_t)$ γινόμενο : δίνει **πόσο** το drone κινείται κατά τον άξονα x στο συγκεκριμένο χρονικό βήμα. Αντίστοιχα, η συνάρτηση sin δίνει την προβολή στον άξονα y.
- $v_{z,t}$: Κατακόρυφη ταχύτητα
- ω_t : Γωνιακή ταχύτητα γύρω από τον κάθετο άξονα (yaw rate), μετράει δηλαδή πόσο γρήγορα αλλάζει ο προσανατολισμός του drone σε γωνία.

<u>Το πρόβλημα βελτιστοποίησης για ορίζοντα N δίνεται από</u> :

$$\min_u \sum_{t=0}^{N-1} \|x_t - x_{ref}\|_2^2 + \|u_t\|_2^2$$

όπου :

- $\min_u$: Ο στόχος είναι να βρεθεί η ακολουθία εντολών ελέγχου utu_tut που ελαχιστοποιεί το συνολικό κόστος (δηλαδή το άθροισμα απόκλισης κατάστασης από το στόχο και "κόστους" ελέγχου) σε όλο το χρονικό ορίζοντα.

- $\sum_{t=0}^{N-1}$: Το άθροισμα αυτών των δύο όρων σε όλα τα χρονικά βήματα από $t=0$ εως $N-1$, δηλαδή κατά τον ορίζοντα βελτιστοποίησης N.

- x_t : Το διάνυσμα κατάστασης του συστήματος στο χρονικό βήμα t. Στην περίπτωσή μας, περιλαμβάνει τις μεταβλητές θέσης και προσανατολισμού του drone.

- x_{ref} : Η επιθυμητή (αναφορική) κατάσταση, δηλαδή η κατάσταση-στόχος προς την οποία θέλουμε να οδηγηθεί το σύστημα (π.χ. η θέση και ο προσανατολισμός στόχος του drone).

- $\|x_t - x_{ref}\|_2^2$: Το τετράγωνο της ευκλείδειας απόστασης μεταξύ της τρέχουσας κατάστασης και της αναφορικής κατάστασης στο χρονικό βήμα t. Μετράει το πόσο μακριά βρίσκεται το drone από το στόχο του. Το τετράγωνο χρησιμοποιείται για να επιβαρύνει περισσότερο τις μεγαλύτερες αποκλίσεις.

- u_t : Το διάνυσμα εντολών ελέγχου (π.χ. ταχύτητες και ρυθμοί περιστροφής) που εφαρμόζονται στο drone στο χρονικό βήμα t.

- $\|u_t\|_2^2$: Το τετράγωνο του μεγέθους των εντολών ελέγχου στο χρονικό βήμα t. Αυτός ο όρος επιβαρύνει μεγάλες τιμές ελέγχου, προάγοντας πιο ομαλή και λιγότερο απότομη συμπεριφορά του drone.

Υπό τους περιορισμούς :

$$x_0 = x_{\text{init}}$$

$$0 \leq v_t \leq v_{\max}$$

$$|v_{z,t}| \leq v_{z,\max}$$

$$|\omega_t| \leq \omega_{\max}$$

όπου :

- $x_0 = x_{\text{init}}$: Η τρέχουσα κατάσταση του συστήματος.

- $0 \leq v_t \leq v_{\max}$: Περιορισμός στην εμπρόσθια ταχύτητα (όχι αρνητική).

- $|v_{z,t}| \leq v_{z,\max}$: Περιορισμός στην κάθετη ταχύτητα

- $|\omega_t| \leq \omega_{\max}$: Περιορισμός στην γωνιακή ταχύτητα (ρυθμός περιστροφής γύρω από τον κατακόρυφο άξονα).

Στην περίπτωση ανίχνευσης εμποδίου σε απόσταση μικρότερη από ένα όριο, εφαρμόζονται επιπλέον περιορισμοί στην ωt για την υποχρεωτική δεξιά στροφή.
Συνολικά, οι περιορισμοί αυτοί διασφαλίζουν ότι οι εντολές ελέγχου που προκύπτουν από τη βελτιστοποίηση θα είναι εφικτές και ασφαλείς.

2.3.4 Μέθοδος επίλυσης;

Το πρόβλημα βελτιστοποίησης που ορίζεται από τον αλγόριθμο MPC επιλύθηκε μέσω της χρήσης του επιλυτή IPOPT (Interior Point Optimizer), ο οποίος είναι ένας αλγόριθμος κατάλληλος για μη γραμμικά προβλήματα βελτιστοποίησης με περιορισμούς. Η επιλογή του IPOPT βασίστηκε στην ικανότητά του να αντιμετωπίζει αποτελεσματικά το πρόβλημα βελτιστοποίησης με τις μη γραμμικές δυναμικές εξισώσεις του μοντέλου και τους περιορισμούς στις μεταβλητές κατάστασης και ελέγχου.
Ο αλγόριθμος MPC διατυπώνει το πρόβλημα ως ελαχιστοποίηση μιας αντικειμενικής συνάρτησης κόστους, που περιλαμβάνει την απόκλιση της προβλεπόμενης κατάστασης από τον επιθυμητό στόχο και το κόστος των ενεργειών ελέγχου, υπό τους περιορισμούς που απορρέουν από το κινηματικό μοντέλο τύπου μονόκυκλου, το οποίο επιλέχθηκε σκόπιμα για την απλότητα και αποτελεσματικότητά του. Η δυναμική αυτή προβλέπει την κίνηση του drone σε σχέση με τις γραμμικές ταχύτητες και τη γωνιακή ταχύτητα (yaw rate).
Κατά την επίλυση, ο IPOPT λαμβάνει υπόψη τα μη γραμμικά ισοδύναμα που περιγράφουν την εξέλιξη της κατάστασης και εφαρμόζει εσωτερικό σημείο για να βρει την βέλτιστη ακολουθία εντολών ελέγχου (ταχύτητες και γωνιακή ταχύτητα) μέσα στο χρονικό ορίζοντα πρόβλεψης (horizon). Το MPC λειτουργεί σε επαναληπτικό βρόχο, όπου σε κάθε βήμα υπολογίζεται και εφαρμόζεται μόνο η πρώτη εντολή ελέγχου, ενώ στη συνέχεια το πρόβλημα επιλύεται ξανά με την ενημερωμένη κατάσταση του drone.
Επιπλέον, η διαδικασία αποφυγής εμποδίων δεν ενσωματώνεται απευθείας στο MPC, αλλά υλοποιείται με έναν ανεξάρτητο μηχανισμό που ανιχνεύει την παρουσία εμποδίων μέσω LIDAR και επιβάλλει περιορισμούς ή τροποποιεί τις εντολές περιστροφής ώστε το drone να στρίβει δεξιά για την αποφυγή σύγκρουσης. Αυτός ο συνδυασμός εξασφαλίζει την ασφάλεια στην πλοήγηση χωρίς να επιβαρύνει το πρόβλημα βελτιστοποίησης.
Συνολικά, η χρήση του IPOPT στο πλαίσιο του MPC, μαζί με το απλοποιημένο κινηματικό μοντέλο και τον ανεξάρτητο μηχανισμό αποφυγής εμποδίων, παρέχει μια αποδοτική και πρακτικά εφαρμόσιμη λύση για την αυτόνομη πλοήγηση του drone στο προσομοιωμένο περιβάλλον.

Στο επόμενο **κεφάλαιο 3**, θα παρουσιαστεί αναλυτικά η τεχνική περιγραφή του κώδικα που υλοποιεί τον αλγόριθμο επίλυσης του προβλήματος βελτιστοποίησης μέσω MPC. Θα αναλυθούν οι βασικές δομές, οι μεταβλητές και οι λειτουργίες που διαχειρίζονται την πρόβλεψη της κίνησης και τον υπολογισμό των εντολών ελέγχου. Επιπλέον, θα περιγραφεί ο τυπικός κώδικας του δυναμικού εμποδίου, το οποίο χρησιμοποιείται στο περιβάλλον προσομοίωσης για την απεικόνιση και αλληλεπίδραση με το κυρίως drone.

ΚΕΦΑΛΑΙΟ 3 – Τεχνική ανάλυση κώδικα

Το κεφάλαιο 3 χωρίζεται σε δυο υπό ενότητες. Η πρώτη υπό ενότητα 3.1 αναλύει τεχνικά τον κώδικα που εφαρμόζεται ο αλγόριθμος βελτιστοποίησης MPC. Ενώ η δεύτερη υπό ενότητα 3.2 αναλύει τον κώδικα του δυναμικού εμποδίου τύπου quadrotor.

3.1 Βασικός Κώδικας MPC

Αρχικά ο κώδικας που εφαρμόζεται ο αλγόριθμος βελτιστοποίησης MPC σε γλώσσα python 3 :

```python
#!/usr/bin/env python3

import rospy
import time
import math
import casadi as ca
import numpy as np
from geometry_msgs.msg import Twist
from gazebo_msgs.msg import ModelStates
from sensor_msgs.msg import LaserScan
from tf.transformations import euler_from_quaternion
import matplotlib.pyplot as plt

class DroneController:
    def __init__(self):
        rospy.init_node('drone_control', anonymous=True)

        self.cmd_vel_pub = rospy.Publisher('/uav1/cmd_vel', Twist, queue_size=10)
        self.drone_position = None
        self.current_yaw = 0.0
        self.lidar_ranges = []
        self.lidar_max_range = 4.0

        rospy.Subscriber("/gazebo/model_states", ModelStates, self.model_states_callback)
        rospy.Subscriber("/uav1/scan", LaserScan, self.lidar_callback)

        while self.cmd_vel_pub.get_num_connections() == 0 and not rospy.is_shutdown():
            rospy.loginfo("Waiting for cmd_vel publisher connection...")
            rospy.sleep(0.1)

        self.path_x = []
        self.path_y = []
        self.path_z = []
        self.yaw_list = []
        self.cmd_linear_x = []
        self.cmd_linear_z = []
        self.cmd_angular_z = []

    def model_states_callback(self, data):
        try:
            idx = data.name.index("uav1")
            self.drone_position = data.pose[idx].position
            orientation = data.pose[idx].orientation
            _, _, yaw = euler_from_quaternion([orientation.x, orientation.y, orientation.z, orientation.w])
            self.current_yaw = yaw
        except ValueError:
            pass

    def lidar_callback(self, data):
        ranges = np.array(data.ranges)
        valid_ranges = np.where(
            (ranges >= 0.15) & (ranges <= self.lidar_max_range),
            ranges,
            np.inf
        )
```

```
        )
        self.lidar_ranges = valid_ranges

    def log_lidar_info(self):
        if len(self.lidar_ranges) == 0:
            return float('inf')
        filtered_ranges = np.where(
            (self.lidar_ranges >= 0.15) & (self.lidar_ranges <= self.lidar_max_range),
            self.lidar_ranges,
            np.inf
        )
        min_front = np.nanmin(filtered_ranges)
        rospy.loginfo(f"LIDAR Readings (±35° FOV): {min_front:.2f} m")
        return min_front

    def mpc_controller(self, current_position, current_yaw, target_position, min_front, max_speed=1.0, horizon=10, dt=0.5):
        opti = ca.Opti()

        x = opti.variable(4, horizon + 1)
        u = opti.variable(3, horizon)

        dx = target_position[0] - current_position.x
        dy = target_position[1] - current_position.y

        yaw_desired = math.atan2(dy, dx)
        yaw_error = yaw_desired - current_yaw
        yaw_error = np.arctan2(np.sin(yaw_error), np.cos(yaw_error))

        x_ref = np.array([target_position[0], target_position[1], target_position[2], current_yaw + yaw_error])
        x_init = np.array([current_position.x, current_position.y, current_position.z, current_yaw ])

        objective = 0

        for t in range(horizon):
            objective += ca.sumsqr(x[:, t] - x_ref)
            objective += ca.sumsqr(u[:, t])

        opti.minimize(objective)
        opti.subject_to(x[:, 0] == x_init)

        for t in range(horizon):
            opti.subject_to(x[0, t + 1] == x[0, t] + dt * (u[0, t] * ca.cos(x[3, t])))
            opti.subject_to(x[1, t + 1] == x[1, t] + dt * (u[0, t] * ca.sin(x[3, t])))
            opti.subject_to(x[2, t + 1] == x[2, t] + dt * u[1, t])
            opti.subject_to(x[3, t + 1] == x[3, t] + dt * u[2, t])

            opti.subject_to(0.0 <= u[0, t])
            opti.subject_to(u[0, t] <= max_speed)
            opti.subject_to(ca.fabs(u[1, t]) <= 1.0)
            opti.subject_to(ca.fabs(u[2, t]) <= 1.0)

        if min_front < self.lidar_max_range:
            rospy.logwarn("Obstacle detected – enforcing right turn.")
            for t in range(horizon):
```

```
            opti.subject_to(u[2, t] <= 1.0)
        opti.solver('ipopt', {'ipopt.print_level': 0, 'print_time': 0, 'ipopt.sb': 'yes'})
        try:
            sol = opti.solve()
            return sol.value(u)[:, 0]
        except RuntimeError as e:
            rospy.logwarn("MPC solver failed, using zero command.")
            return np.zeros(3)

    def move_to_target(self, target_x: float, target_y: float, target_z: float):
        rate = rospy.Rate(10)
        twist = Twist()
        start_time = time.time()

        while self.drone_position is None and not rospy.is_shutdown():
            if time.time() - start_time > 5.0:
                rospy.logerr("Timeout waiting for drone position!")
                return
            rospy.sleep(0.1)

        rospy.loginfo(f"Moving to target: ({target_x}, {target_y}, {target_z})")

        while not rospy.is_shutdown():
            dx = target_x - self.drone_position.x
            dy = target_y - self.drone_position.y
            dz = target_z - self.drone_position.z

            distance = math.sqrt(dx**2 + dy**2 + dz**2)
            min_front = self.log_lidar_info()

            if min_front < 1.5:
               rospy.logwarn("Obstacle extremely close — pausing for 2 seconds.")
               twist = Twist()
               self.cmd_vel_pub.publish(twist)
               rospy.sleep(1.0)
               twist.angular.z = -1.0
               self.cmd_vel_pub.publish(twist)
               rospy.sleep(1.0)
               twist.angular.z = 0.0
               self.cmd_vel_pub.publish(twist)
               continue

            if distance < 0.2:
                rospy.loginfo("Reached target.")
                break

            control_command = self.mpc_controller(
                self.drone_position, self.current_yaw,
                (target_x, target_y, target_z),
                min_front=min_front
            )

            twist.linear.x = control_command[0]
            twist.linear.z = control_command[1]
```

```
            twist.angular.z = control_command[2]

            self.cmd_vel_pub.publish(twist)

            self.path_x.append(self.drone_position.x)
            self.path_y.append(self.drone_position.y)
            self.path_z.append(self.drone_position.z)

            self.yaw_list.append(self.current_yaw)

            self.cmd_linear_x.append(twist.linear.x)
            self.cmd_linear_z.append(twist.linear.z)
            self.cmd_angular_z.append(twist.angular.z)

            rate.sleep()

        twist.linear.x = twist.linear.y = twist.linear.z = twist.angular.z = 0
        self.cmd_vel_pub.publish(twist)

    def plot_data(self):
        plt.figure(figsize=(12, 6))
        plt.subplot(2, 2, 1)
        plt.plot(self.path_x, self.path_y, '-o', label='Path (x,y)')
        plt.xlabel('X position (m)')
        plt.ylabel('Y position (m)')
        plt.title('Drone Path (XY)')
        plt.grid()
        plt.legend()
        plt.subplot(2, 2, 2)
        plt.plot(self.path_z, label='Altitude (z)')
        plt.xlabel('Time steps')
        plt.ylabel('Altitude (m)')
        plt.title('Altitude over time')
        plt.grid()
        plt.legend()
        plt.subplot(2, 2, 3)
        plt.plot(self.cmd_linear_x, label='Linear X velocity')
        plt.plot(self.cmd_linear_z, label='Linear Z velocity')
        plt.xlabel('Time steps')
        plt.ylabel('Velocity (m/s)')
        plt.title('Linear Velocities')
        plt.grid()
        plt.legend()
        plt.subplot(2, 2, 4)
        plt.plot(self.cmd_angular_z, label='Angular Z velocity')
        plt.xlabel('Time steps')
        plt.ylabel('Angular Velocity (rad/s)')
        plt.title('Angular Velocity')
        plt.grid()
        plt.legend()
        plt.tight_layout()
        plt.show()

def main():
    controller = DroneController()
    waypoints = [
        (-5.503114, -5.595720, 1),
        (5.384420, 3.581098, 1),
        (5.396716, -5.445508, 1),
        (-5.571792, 3.501312, 1),
        (0, 0, 1)
    ]
    try:
        for wp in waypoints:
            controller.move_to_target(*wp)
            rospy.sleep(1)
    except rospy.ROSInterruptException:
        pass
    controller.plot_data()

if __name__ == "__main__":
    main()
```

Συνάρτηση <u>main</u>

- Η συνάρτηση **main** δε δέχεται κάποια είσοδο ούτε υπάρχει κάποια έξοδος από τη συνάρτηση. Δημιουργεί το αντικείμενο **DroneController**, ορίζει μια λίστα από σημεία στόχους (**waypoints**) και καλεί διαδοχικά τη **move_to_target** για κάθε σημείο. Μετά την ολοκλήρωση, καλεί τη συνάρτηση **plot_data** για ανάλυση πορείας και εντολών σε σχήμα γραφήματος.

Συνάρτηση <u>lidar_callback</u>

- Είσοδος →
  1. **data** (τύπου LaserScan), Δεδομένα από τον αισθητήρα LIDAR, που περιλαμβάνουν αποστάσεις σε πολλαπλές κατευθύνσεις.
  2. **self**: Αναφορά στο αντικείμενο της κλάσης.

- Έξοδος → Δεν επιστρέφει κάποια τιμή (**είναι callback**), αλλά ενημερώνει την εσωτερική μεταβλητή self.lidar_ranges με τις έγκυρες αποστάσεις.

- Περιγραφή →
  Επεξεργάζεται τα raw **δεδομένα αποστάσεων** από το LIDAR, μετατρέποντάς τα **σε πίνακα NumPy**. Φιλτράρει τις αποστάσεις ώστε να διατηρεί μόνο αυτές που βρίσκονται εντός του εύρους **[0.15, lidar_max_range].** Τιμές εκτός αυτού του εύρους αντικαθίστανται με άπειρο (**np.inf**), υποδεικνύοντας μη έγκυρες ή απροσπέλαστες αποστάσεις.

Συνάρτηση <u>log_lidar_info</u>

- Είσοδος →
  1. **self**: Αναφορά στο αντικείμενο της κλάσης.

- Έξοδος →
  1. Επιστρέφει την ελάχιστη απόσταση που ανιχνεύεται μπροστά από το drone (τύπος **float**).

- Περιγραφή →
  Ελέγχει εάν υπάρχουν έγκυρα δεδομένα από το LIDAR. Αν όχι, επιστρέφει άπειρο (). Διαχωρίζει το πεδίο θέασης μπροστά (+- 35 μοίρες) και υπολογίζει την ελάχιστη απόσταση εμποδίου σε αυτό το πεδίο. Χρησιμοποιείται για την ανίχνευση εμποδίων και την ενεργοποίηση λογικής αποφυγής.

Συνάρτηση <u>mpc_controller</u>

- Είσοδος →
    1. **current_position**: Αντικείμενο θέσης με πεδία x, y, z.
    2. **current_yaw (float):** Τρέχουσα γωνία προσανατολισμού yaw του drone.
    3. **target_position (tuple ή λίστα):** Στόχος (x, y, z) προς επίτευξη.
    4. **min_front (float):** Ελάχιστη μπροστινή απόσταση εμποδίου από το LIDAR.
    5. **max_speed:** Ανώτατο όριο για την εμπρόσθια ταχύτητα του drone.
    6. **horizon:** Ορίζοντας χρονικού ορίζοντα πρόβλεψης του MPC (πόσα βήματα στο μέλλον προβλέπει).
    7. **dt**: Χρονικό βήμα μεταξύ διαδοχικών προβλέψεων.
    8. **self**

- Έξοδος →

  Πίνακας τριών διαστάσεων, που αντιπροσωπεύει την πρώτη βέλτιστη εντολή ελέγχου που πρέπει να εφαρμοστεί.
    1. **v**: Εμπρόσθια ταχύτητα (linear velocity κατά μήκος του άξονα x).
    2. **vz**: Κάθετη ταχύτητα (κατακόρυφη κίνηση κατά άξονα z).
    3. **ω**: Γωνιακή ταχύτητα (yaw rate), δηλαδή ρυθμός περιστροφής γύρω από τον κατακόρυφο άξονα.

- Περιγραφή →

  Η συνάρτηση δημιουργεί ένα μοντέλο βελτιστοποίησης (με τη βοήθεια της **Casadi**) για το προγνωστικό έλεγχο (MPC), που προβλέπει την εξέλιξη της κατάστασης του drone (θέση και προσανατολισμό) για τα επόμενα horizon χρονικά βήματα με βήμα dt.
  Στόχος της βελτιστοποίησης είναι **η ελαχιστοποίηση ενός αντικειμενικού κόστους**, το οποίο περιλαμβάνει:

    - Την τετραγωνική απόκλιση των προβλεπόμενων καταστάσεων από τις επιθυμητές καταστάσεις (στόχος).
    - Την τιμωρία των εντολών ελέγχου ώστε να αποφευχθούν απότομες ή υπερβολικές κινήσεις.

  Η πρόβλεψη της κατάστασης γίνεται μέσω κινηματικού μοντέλου μονοκύκλου, όπου η θέση και ο προσανατολισμός ενημερώνονται με βάση τις εντολές ελέγχου.
  Επιπλέον, τίθενται περιορισμοί στις μεταβλητές:

    - Θετική εμπρόσθια ταχύτητα κάτω από max_speed.
    - Περιορισμοί στα όρια της κατακόρυφης και γωνιακής ταχύτητας.
      Όταν ανιχνεύεται εμπόδιο μπροστά (ελάχιστη απόσταση μικρότερη από lidar_max_range), εφαρμόζεται πρόσθετος

περιορισμός που υποχρεώνει το drone να εκτελεί δεξιά στροφή (αρνητικός ρυθμός yaw) σε όλη τη διάρκεια του ορίζοντα.

Η επίλυση του προβλήματος γίνεται με τον αριθμητικό επιλυτή IPOPT, ο οποίος είναι κατάλληλος για μη γραμμικά προβλήματα βελτιστοποίησης με περιορισμούς. Το αποτέλεσμα είναι η ακολουθία βέλτιστων εντολών ελέγχου, από την οποία επιστρέφεται η πρώτη εντολή για εφαρμογή στο τρέχον χρονικό βήμα.

Συνάρτηση <u>move_to_target</u>

- Είσοδος →
  1. **self**
  2. **target_x:** Συντεταγμένη στόχος στον άξονα x
  3. **target_y:** Συντεταγμένη στόχος στον άξονα y
  4. **target_z:** Συντεταγμένη στόχος στον άξονα z

- Έξοδος → Καμία επιστροφή τιμής. Η συνάρτηση χειρίζεται **εσωτερικά την εκτέλεση της κίνησης.**

- Περιγραφή →

  Η συνάρτηση υλοποιεί το βρόχο εκτέλεσης της πλοήγησης προς έναν συγκεκριμένο στόχο. Αρχικά, περιμένει να ληφθεί η τρέχουσα θέση και προσανατολισμός του drone μέσω ROS topics. Στη συνέχεια, σε κάθε επανάληψη βρόχου:

  - Υπολογίζει την ευκλείδεια απόσταση από το drone μέχρι τον στόχο.
  - Αν η απόσταση είναι μικρότερη από ένα μικρό όριο (π.χ. 0.2 m), θεωρείται ότι ο στόχος επιτεύχθηκε και ο βρόχος τερματίζει.
  - Ανιχνεύει την ελάχιστη μπροστινή απόσταση εμποδίου από τα δεδομένα LIDAR, καλώντας τη log_lidar_info.
  - Αν το εμπόδιο είναι πολύ κοντά (π.χ. < 1.5 m), εκτελείται χειρισμός αποφυγής: προσωρινό πάγωμα κίνησης, δεξιά στροφή για να αποφευχθεί το εμπόδιο, και συνέχιση του βρόχου.
  - Διαφορετικά, καλείται η mpc_controller για να υπολογιστούν οι βέλτιστες εντολές ελέγχου με βάση την τρέχουσα θέση, προσανατολισμό, στόχο και εμπόδια.
  - Οι εντολές ελέγχου εφαρμόζονται μέσω δημοσίευσης σε ROS topic /uav1/cmd_vel.
  - Καταγράφονται οι παράμετροι θέσης και εντολών για μελλοντική ανάλυση.

Όταν ολοκληρωθεί η πλοήγηση, όλες οι εντολές κινητικής κίνησης μηδενίζονται για να σταματήσει το drone.

Συνολικά ο κώδικας υλοποιεί μια ολοκληρωμένη μέθοδο ελέγχου κίνησης και πλοήγησης μέσα σε περιβάλλον προσομοίωσης χρησιμοποιώντας ROS και το Gazebo. Με την αρχικοποίηση του κόμβου ROS και τη δημιουργία των κατάλληλων publishers και subscribers, ο ελεγκτής λαμβάνει σε πραγματικό χρόνο τη θέση και τον προσανατολισμό του drone μέσω του topic /uav1/model_states και δεδομένα από αισθητήρα LIDAR μέσω του /uav1/scan. Η βασική λογική ελέγχου βασίζεται στην επίλυση ενός προβλήματος βελτιστοποίησης Model Predictive Control (MPC) μέσω της βιβλιοθήκης CasADi, η οποία υπολογίζει τη βέλτιστη ακολουθία κινήσεων που φέρνει το drone κοντά στον επιθυμητό στόχο ενώ λαμβάνει υπόψη περιορισμούς ταχύτητας και αποφυγής εμποδίων. Η συνάρτηση MPC ορίζει τις μεταβλητές κατάστασης και ελέγχου, τη συνάρτηση κόστους που περιλαμβάνει την απόκλιση από την επιθυμητή θέση και την τιμωρία των ενεργειών, καθώς και περιορισμούς που μοντελοποιούν τις δυναμικές ιδιότητες του drone. Επιπλέον, σε περίπτωση που ο αισθητήρας LIDAR ανιχνεύσει εμπόδιο εντός ορισμένης απόστασης, εφαρμόζεται περιορισμός στον ρυθμό περιστροφής ώστε το drone να πραγματοποιεί στροφή δεξιά για αποφυγή. Η κύρια λειτουργία move_to_target εκτελεί βρόχο ελέγχου όπου διαρκώς υπολογίζονται οι κατάλληλες εντολές κίνησης και αποστέλλονται μέσω ROS, μέχρι την επίτευξη του στόχου ή τη διακοπή του προγράμματος. Ταυτόχρονα, συλλέγονται και αποθηκεύονται δεδομένα θέσης και εντολών για ανάλυση. Η δομή του κώδικα και η χρήση του ROS εξασφαλίζουν την αδιάλειπτη ενημέρωση και εκτέλεση εντολών σε πραγματικό χρόνο, επιτρέποντας την αποτελεσματική πλοήγηση και βασική αποφυγή εμποδίων μέσα στο προσομοιωμένο περιβάλλον.

3.2 Κώδικας δυναμικού εμποδίου τύπου quadrotor

Ο δεύτερος κώδικας υλοποιεί τον ελεγκτή του δυναμικού εμποδίου τύπου drone, που λειτουργεί ως κινητό εμπόδιο στο περιβάλλον προσομοίωσης. Αποτελεί ένα απλό σύστημα ελέγχου κίνησης, βασισμένο σε ROS, το οποίο επιτρέπει την αυτόνομη πτήση και κίνηση του drone-εμποδίου μέσα στον χώρο.

```python
#!/usr/bin/env python3

import rospy
from geometry_msgs.msg import Twist
from gazebo_msgs.msg import ModelStates

class SecondDroneController:
    def __init__(self):
        rospy.init_node('drone_control', anonymous=True)
        self.cmd_vel_pub = rospy.Publisher('/uav2/cmd_vel', Twist, queue_size=10)
        self.drone_position = None
        rospy.Subscriber("/gazebo/model_states", ModelStates, self.model_states_callback)

        while self.cmd_vel_pub.get_num_connections() == 0 and not rospy.is_shutdown():
            rospy.loginfo("Waiting for cmd_vel publisher connection...")
            rospy.sleep(0.1)

        rospy.loginfo("Controller initialized for /uav2.")

    def model_states_callback(self, data):
        try:
            idx = data.name.index("uav2")
            self.drone_position = data.pose[idx].position
        except ValueError:
            pass

    def move_x_timed(self, direction=1, speed=1.0, duration=1.0):
        rate = rospy.Rate(50)
        twist = Twist()
        twist.linear.x = speed * direction
        twist.linear.y = 0.0
        twist.linear.z = 0.0
        twist.angular.z = 0.0

        start_time = rospy.Time.now().to_sec()
        while not rospy.is_shutdown():
            now = rospy.Time.now().to_sec()
            if now - start_time >= duration:
                break
            self.cmd_vel_pub.publish(twist)
            rate.sleep()

        self.cmd_vel_pub.publish(Twist())

    def takeoff(self, target_z=0.88, speed=0.5):
        rospy.loginfo("Taking off...")
        rate = rospy.Rate(50)
        twist = Twist()

        while self.drone_position is None and not rospy.is_shutdown():
            rospy.sleep(0.1)

        while not rospy.is_shutdown():
            current_z = self.drone_position.z
            error_z = target_z - current_z
```

```
56
57              if self.drone_position.z >= target_z:
58                  break
59
60              twist.linear.z = speed * (1 if error_z > 0 else -1)
61              self.cmd_vel_pub.publish(twist)
62              rate.sleep()
63
64          self.cmd_vel_pub.publish(Twist())
65
66      def move_in_circle(self, radius=1.0, linear_speed=0.5, duration=20.0):
67          rospy.loginfo("Starting circular movement...")
68          rate = rospy.Rate(50)
69
70          twist = Twist()
71          twist.linear.x = linear_speed
72          twist.angular.z = linear_speed / radius
73
74          start_time = rospy.Time.now().to_sec()
75          while not rospy.is_shutdown():
76              now = rospy.Time.now().to_sec()
77              if now - start_time >= duration:
78                  break
79              self.cmd_vel_pub.publish(twist)
80              rate.sleep()
81
82          self.cmd_vel_pub.publish(Twist())
83          rospy.loginfo("Circle complete.")
84
85  def main():
86      controller = SecondDroneController()
87
88      try:
89          controller.takeoff(target_z=0.88)
90
91          while not rospy.is_shutdown():
92              controller.move_in_circle(radius=3.0, linear_speed=1.0, duration=20.0)
93             # rospy.sleep(2.0)
94
95      except rospy.ROSInterruptException:
96          pass
97
98  if __name__ == "__main__":
99      main()
100
```

Συνάρτηση: move_x_timed

- Είσοδος →
    1. self
    2. direction (κατεύθυνση κίνησης στον άξονα x, +1 ή -1)
    3. speed (ταχύτητα γραμμικής κίνησης σε m/s)
    4. duration (διάρκεια κίνησης σε δευτερόλεπτα)

- Έξοδος → Καμία (εκτελεί κίνηση).

- Περιγραφή →

    Εκτελεί ευθεία κίνηση κατά τον άξονα x με την προκαθορισμένη ταχύτητα και κατεύθυνση για χρονικό διάστημα duration. Στέλνει συνεχώς εντολές τύπου Twist στο topic /uav2/cmd_vel με τη γραμμική ταχύτητα στον άξονα x ενεργή και όλες τις άλλες ταχύτητες μηδενικές. Μετά τη λήξη του χρόνου, σταματάει την κίνηση στέλνοντας μηδενικές εντολές.

Συνάρτηση: takeoff

- Είσοδος →
    1. self
    2. target_z (επιθυμητό ύψος απογείωσης σε μέτρα)
    3. speed (ταχύτητα ανόδου σε m/s)

- Έξοδος → Καμία (εκτελεί απογείωση).

- Περιγραφή →

    Υλοποιεί διαδικασία απογείωσης του drone έως το ύψος target_z. Ελέγχει συνεχώς τη σημερινή θέση z από το self.drone_position και στέλνει γραμμική ταχύτητα στον άξονα z ώστε να αυξάνει ή να μειώνει το ύψος με σταθερή ταχύτητα speed μέχρι να φτάσει στο επιθυμητό επίπεδο. Εφόσον επιτευχθεί το ύψος, σταματά την κίνηση.

Συνάρτηση: move_in_circle

- Είσοδος →
    1. self
    2. radius (ακτίνα κύκλου σε μέτρα)
    3. linear_speed (γραμμική ταχύτητα κίνησης σε m/s)
    4. duration (διάρκεια κίνησης σε δευτερόλεπτα)

- Έξοδος → Καμία (εκτελεί κυκλική κίνηση).
- Περιγραφή →

  Εκτελεί κίνηση σε κυκλική τροχιά με ακτίνα radius και σταθερή γραμμική ταχύτητα linear_speed. Υπολογίζει την γωνιακή ταχύτητα ως linear_speed / radius και στέλνει συνδυασμένες εντολές γραμμικής κίνησης στον άξονα x και περιστροφής γύρω από τον άξονα z (yaw). Η κίνηση εκτελείται για το χρονικό διάστημα duration. Μετά τη λήξη, σταματά την κίνηση.

Συνάρτηση: <u>main</u>

- Είσοδος → Καμία
- Έξοδος → Καμία
- Περιγραφή →

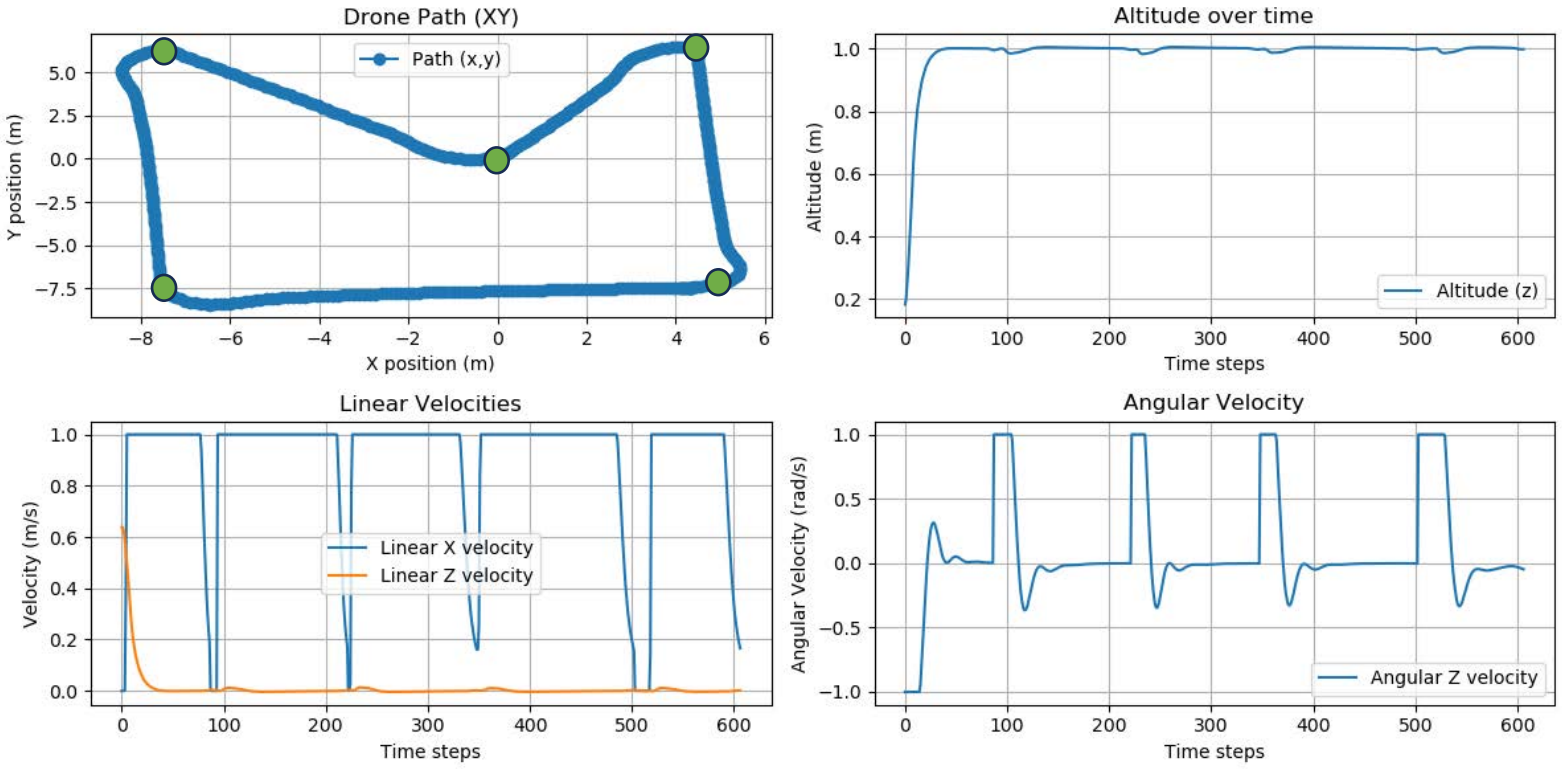
  Δημιουργεί το αντικείμενο SecondDroneController και εκκινεί την απογείωση του drone μέχρι ύψος 0.88 μέτρα. Στη συνέχεια εισέρχεται σε ατέρμονα βρόχο, όπου το drone εκτελεί κυκλική κίνηση με ακτίνα 3 μέτρα και ταχύτητα 1 m/s για 20 δευτερόλεπτα κάθε φορά. Σε περίπτωση διακοπής (π.χ. χειροκίνητη διακοπή ROS), ο βρόχος τερματίζεται ομαλά.

Συνολικά ο κώδικας του δυναμικού εμποδίου uav2, αποτελεί μια βασική και λειτουργική εφαρμογή ελέγχου κίνησης μέσα σε περιβάλλον προσομοίωσης Gazebo, με χρήση του ROS. Αρχικά, ο κόμβος ROS αρχικοποιείται και δημιουργείται publisher που αποστέλλει εντολές κίνησης τύπου Twist στο topic του δεύτερου drone, καθώς και subscriber που λαμβάνει σε πραγματικό χρόνο τη θέση και τον προσανατολισμό του από το Gazebo μέσω του topic /gazebo/model_states. Η θέση αυτή ενημερώνεται συνεχώς μέσα από callback συναρτήσεις και χρησιμοποιείται για την καθοδήγηση των εντολών κίνησης. Ο έλεγχος περιλαμβάνει βασικές κινήσεις όπως απογείωση, ευθύγραμμη κίνηση στον άξονα x και κυκλικές τροχιές, που υλοποιούνται μέσω κατάλληλων γραμμικών και γωνιακών ταχυτήτων, οι οποίες στέλνονται περιοδικά με ρυθμό 10 Hz. Η απογείωση πραγματοποιείται με συνεχή έλεγχο του ύψους και ρύθμιση της κατακόρυφης ταχύτητας μέχρι να επιτευχθεί το προκαθορισμένο ύψος, ενώ οι υπόλοιπες κινήσεις εκτελούνται με χρονικό έλεγχο και παρακολούθηση της απόστασης ή της γωνίας περιστροφής. Η δομή αυτή επιτρέπει στο drone-εμπόδιο να κινείται αυτοματοποιημένα μέσα στο προσομοιωμένο περιβάλλον

ΚΕΦΑΛΑΙΟ 4 Αποτελέσματα

Στο κεφάλαιο 4 παρουσιάζονται τέσσερις προσομοιώσεις. Η κάθε προσομοίωση θα αναδεικνύει τα αποτελέσματα που προέκυψαν από μια σειρά δοκίμων οπού πραγματοποιήθηκαν με σκοπό την αξιολόγηση της απόδοσης του συστήματος ελέγχου του drone. Μέσα από την ανάλυση των δεδομένων κίνησης, των εντολών ελέγχου και της συμπεριφοράς του drone σε διαφορετικά σενάρια πλοήγησης και αποφυγής εμποδίων, γίνεται δυνατή η αξιολόγηση της αποτελεσματικότητας του προτεινόμενου αλγορίθμου Model Predictive Control (MPC) και της ενσωματωμένης λογικής αποφυγής εμποδίων. Τα γραφήματα αποτυπώνουν παραμέτρους όπως η τροχιά κίνησης, το ύψος, οι γραμμικές και γωνιακές ταχύτητες, προσφέροντας μια ολοκληρωμένη εικόνα της συμπεριφοράς του drone και της προσαρμοστικότητάς του στο προσομοιωμένο περιβάλλον.

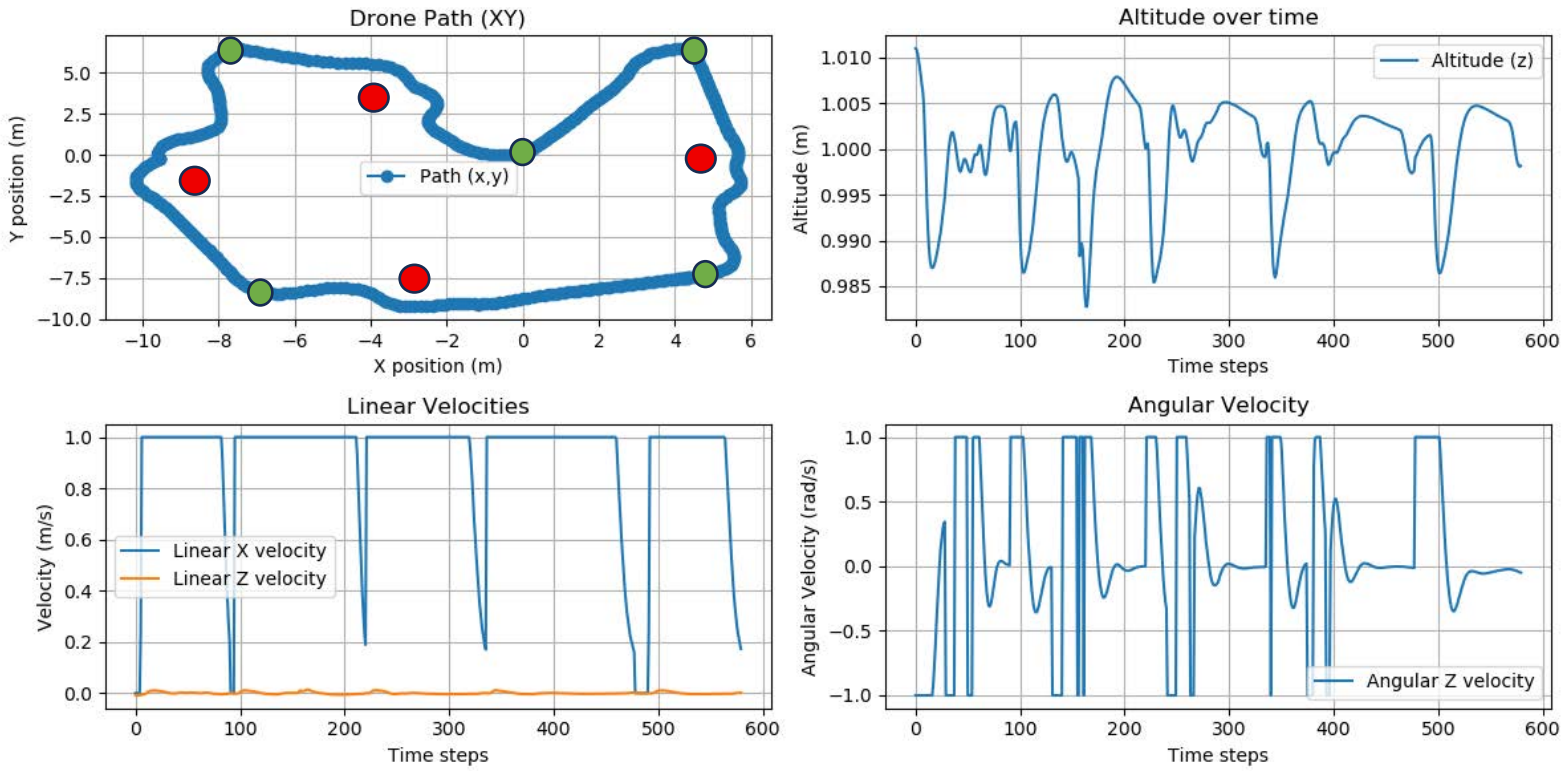
4.1 Προσομοιώσεις



Προσομοίωση 1: Απόκριση του συστήματος κατά τη διάρκεια της προσομοίωσης 1.
Η συνολική εικόνα παρουσιάζει την απόκριση του συστήματος ελέγχου του drone κατά την πρώτη προσομοίωση, όπου το drone κινείται σε περιβάλλον χωρίς εμπόδια και εκτελεί μία διαδρομή μεταξύ πέντε προκαθορισμένων σημείων στόχων. Στα γραφήματα απεικονίζονται τόσο η τροχιά του drone (μπλε γραμμή), οι μεταβολές του υψομέτρου του, όσο και οι εντολές γραμμικών και γωνιακών ταχυτήτων που παράγονται από τον αλγόριθμο MPC για την επίτευξη των στόχων με ακρίβεια και ομαλή κίνηση. Οι πράσινοι κύκλοι αντιπροσωπεύουν τους στόχους.

Κατά την προσομοίωση 1:

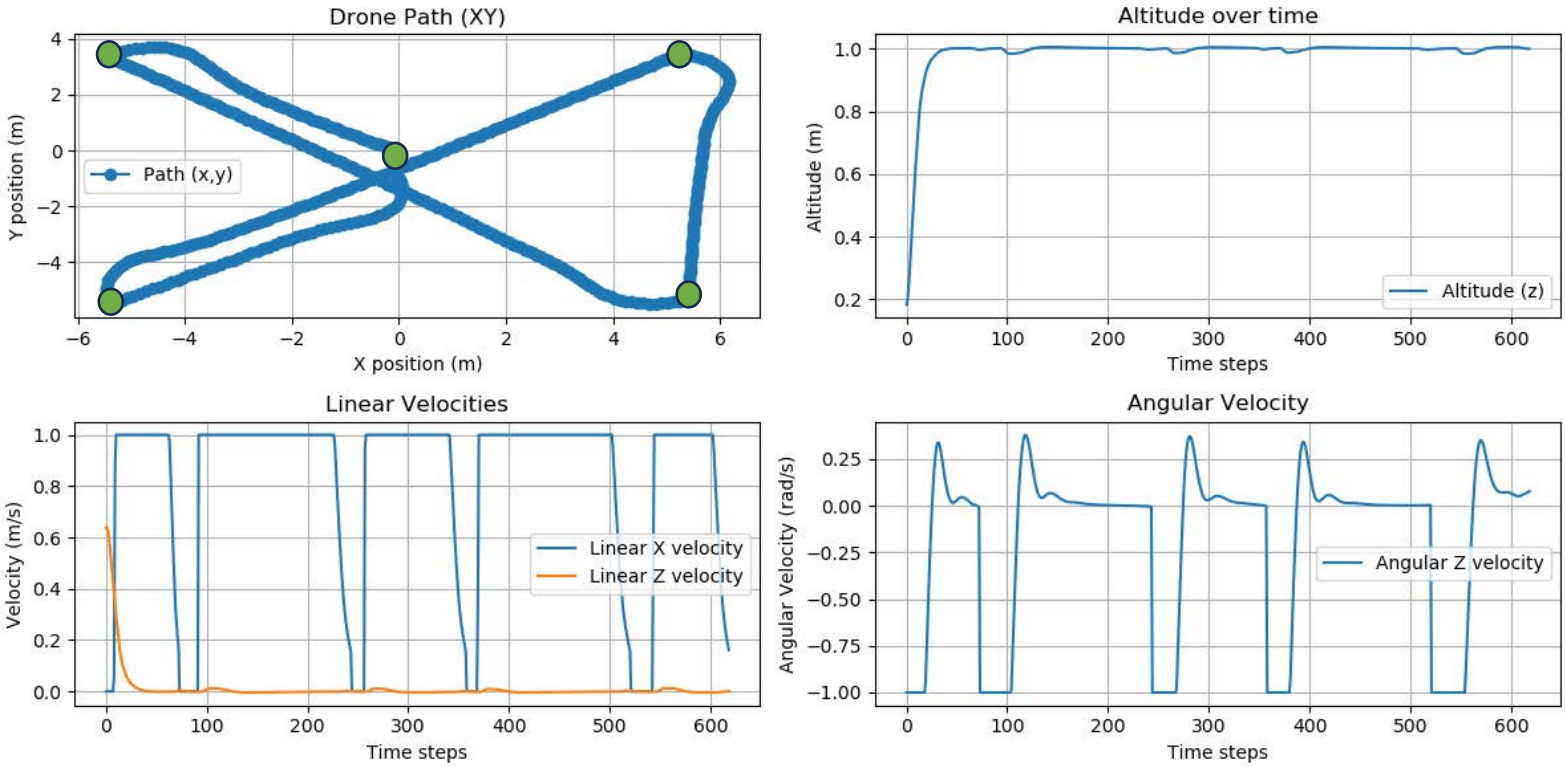
Στο γράφημα της πορείας (**πάνω αριστερά**) φαίνεται η κίνηση του drone στο επίπεδο XY, όπου διακρίνεται ότι το drone ακολουθεί με ακρίβεια το προκαθορισμένο μονοπάτι, πραγματοποιώντας ομαλές στροφές και προσαρμογές κατεύθυνσης ώστε να φτάσει στους στόχους του. Στο γράφημα του υψομέτρου (**πάνω δεξιά**) παρατηρείται γρήγορη ανάβαση στην επιθυμητή στάθμη ύψους περίπου 1 μέτρου, που στη συνέχεια διατηρείται σταθερή με μικρές διακυμάνσεις, γεγονός που δείχνει σταθερότητα και σωστό έλεγχο της κατακόρυφης θέσης του drone. Το γράφημα των γραμμικών ταχυτήτων (**κάτω αριστερά**) παρουσιάζει την εμπρόσθια ταχύτητα στον άξονα X, η οποία μεταβάλλεται ανάλογα με την κίνηση και τις αλλαγές στην πορεία, ενώ η κάθετη ταχύτητα στον άξονα Z παραμένει χαμηλή, υποδεικνύοντας ότι το drone διατηρεί το ύψος του σταθερό κατά τη διάρκεια της διαδρομής. Τέλος, στο γράφημα της γωνιακής ταχύτητας (**κάτω δεξιά**) απεικονίζονται οι στροφές του drone γύρω από τον άξονα Z (yaw), όπου οι θετικές και αρνητικές αιχμές αντιστοιχούν σε στροφές δεξιά και αριστερά αντίστοιχα, επιτρέποντας στο drone να αλλάζει ομαλά κατεύθυνση και να παραμένει ευθυγραμμισμένο με την πορεία του. Συνολικά, τα γραφήματα επιβεβαιώνουν τη σωστή λειτουργία του αλγορίθμου ελέγχου και την επιτυχή εκτέλεση των κινήσεων του drone σύμφωνα με τους στόχους της προσομοίωσης. **(βλ. video 3, κεφ 2.3.2)**



Προσομοίωση 2: <u>Απόκριση του συστήματος κατά τη διάρκεια της προσομοίωσης 2.</u>
Η προσομοίωση 2 βασίστηκε στην ιδιά πορεία που σχημάτισε το πείραμα 1. Η συνολική εικόνα παρουσιάζει τα αποτελέσματα της πτήσης του drone κατά τη διάρκεια μιας προσομοίωσης όπου εφαρμόστηκε η διαδικασία αποφυγής τεσσάρων απλών εμποδίων. Διακρίνονται η διαδρομή που ακολούθησε το drone στον οριζόντιο χώρο με μπλε χρώμα, οι αλλαγές στο υψόμετρο, οι γραμμικές ταχύτητες κατά τους άξονες X και Z, καθώς και η γωνιακή ταχύτητα (yaw rate). Τα γραφήματα αποτυπώνουν με σαφήνεια τις προσαρμογές που πραγματοποίησε το σύστημα για την αποφυγή εμποδίων, ενώ παράλληλα διατηρεί την κατεύθυνση προς τους πέντε προκαθορισμένους στόχους. Οι πράσινοι κύκλοι αντιπροσωπεύουν τους στόχους ενώ οι κόκκινοι τα εμπόδια.

Κατά την προσομοίωση 2:

Στο γράφημα της πορείας (**πάνω αριστερά**), το drone ακολουθεί μία διαδρομή που παρουσιάζει χαρακτηριστικές αποκλίσεις και στροφές, αντιπροσωπεύοντας τις αλλαγές κατεύθυνσης που πραγματοποιεί προκειμένου να αποφύγει τα εμπόδια, διατηρώντας ταυτόχρονα συνοχή στην πορεία προς τους προκαθορισμένους στόχους. Στο γράφημα του υψομέτρου (**πάνω δεξιά**), παρατηρούνται μικρές διακυμάνσεις γύρω από το 1 μέτρο, οι οποίες υποδεικνύουν προσαρμογές στο ύψος ώστε να διατηρηθεί η σταθερότητα κατά την αποφυγή εμποδίων. Στο γράφημα των γραμμικών ταχυτήτων (**κάτω αριστερά**), η εμπρόσθια ταχύτητα στον άξονα X παρουσιάζει διακοπές και μεταβολές, ένδειξη των διακοπών στην κίνηση και των αλλαγών ταχύτητας κατά την αποφυγή, ενώ η κάθετη ταχύτητα στον άξονα Z παραμένει χαμηλή και σχετικά σταθερή. Τέλος, το γράφημα της γωνιακής ταχύτητας (**κάτω δεξιά**) παρουσιάζει έντονες και συχνές εναλλαγές μεταξύ θετικών και αρνητικών τιμών, αντανακλώντας τις συχνές στροφές δεξιά και αριστερά που εκτελεί το drone για να αποφύγει τα εμπόδια και να παραμείνει ευθυγραμμισμένο με την επιθυμητή πορεία. Συνολικά, τα γραφήματα δείχνουν την επιτυχή εφαρμογή του αλγορίθμου αποφυγής εμποδίων και τη ικανότητα του drone να προσαρμόζεται δυναμικά στο περιβάλλον του. **(βλ. video 1, κεφ 2.3.2)**

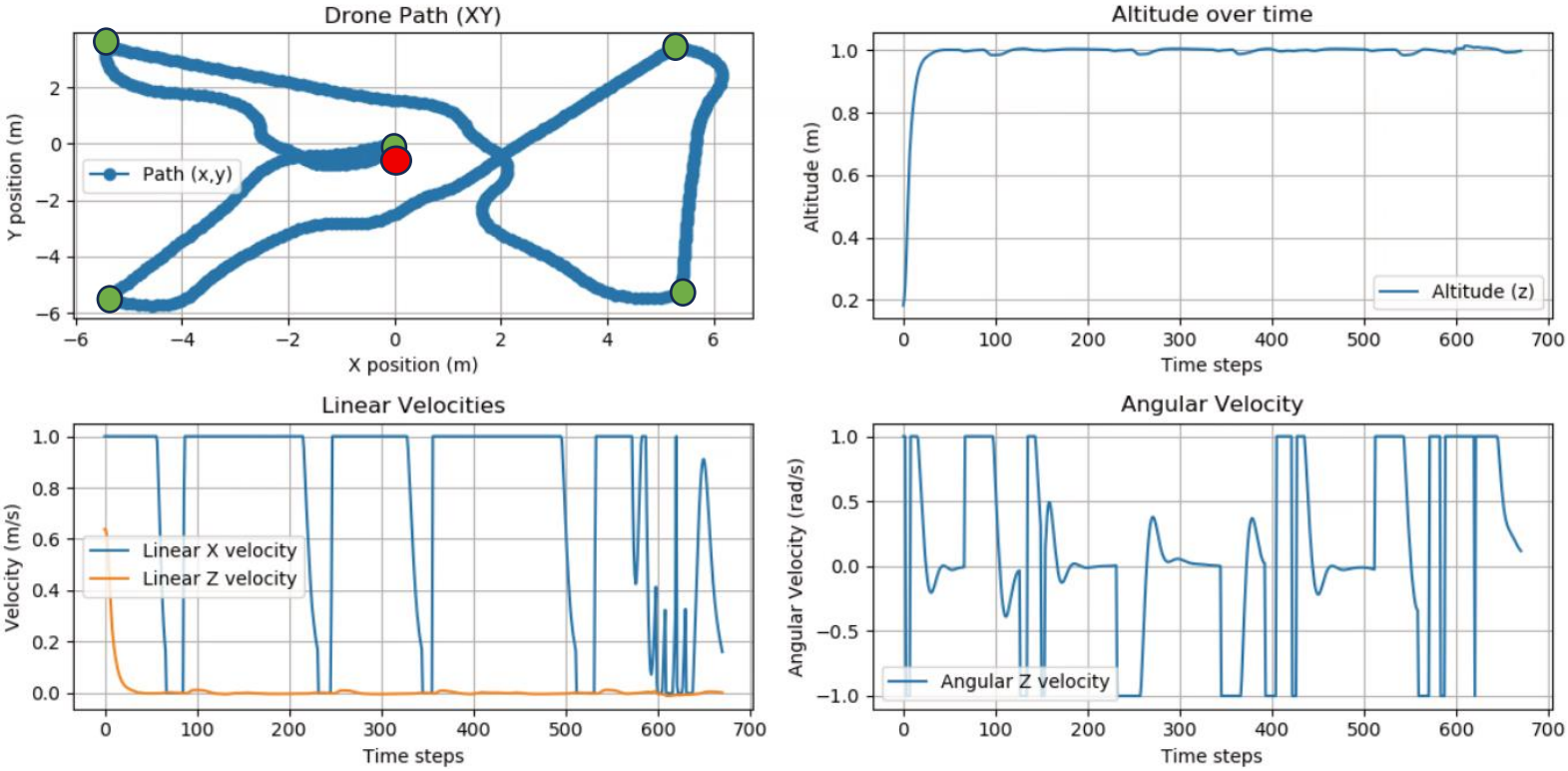


Προσομοίωση 3: <u>Απόκριση του συστήματος κατά τη διάρκεια της προσομοίωσης 3.</u>
Η προσομοίωση 3 διαφοροποιείται των προηγουμένων προσομοιώσεων σχετικά με την πορεία που σχηματίζει το drone. Η συνολική εικόνα απεικονίζει την συμπεριφορά του drone κατά τη διάρκεια της τρίτης προσομοίωσης, όπου το σύστημα εκτελεί πτήσεις σε πέντε προκαθορισμένους στόχους χωρίς την παρουσία εμποδίων. Τα γραφήματα προσφέρουν μία πλήρη οπτική της κίνησης, περιλαμβάνοντας την πορεία στο επίπεδο XY (μπλε γραμμή), τις μεταβολές στο υψόμετρο, καθώς και τις γραμμικές και γωνιακές ταχύτητες που υλοποιήθηκαν κατά τη διάρκεια της πτήσης. Οι πράσινοι κύκλοι αντιπροσωπεύουν τους στόχους.

Κατά την προσομοίωση 3:

Στο γράφημα της πορείας (**πάνω αριστερά**), απεικονίζεται η κίνηση του drone στον διοδιάστατο χώρο XY, όπου διακρίνεται ότι το drone ακολουθεί ένα σύνθετο και συμμετρικό μονοπάτι, με σαφείς αλλαγές κατεύθυνσης και διασταυρώσεις, χωρίς να παρατηρούνται αποκλίσεις ή αστάθειες. Η διαδρομή περιλαμβάνει πολλαπλές καμπύλες και ευθύγραμμα τμήματα, ενώ το drone καταφέρνει να παραμείνει ευθυγραμμισμένο με την προκαθορισμένη τροχιά σε όλα τα κρίσιμα σημεία της διαδρομής. Η ακρίβεια της πορείας σε αυτή την προσομοίωση καταδεικνύει την ικανότητα του ελεγκτή να παράγει κατάλληλες εντολές σε πραγματικό χρόνο και να προσαρμόζεται άμεσα στις απαιτήσεις του σχήματος πτήσης. Στο γράφημα του υψομέτρου (**πάνω δεξιά**), φαίνεται ότι το drone επιτυγχάνει γρήγορα το επιθυμητό ύψος και το διατηρεί σταθερό καθ' όλη τη διάρκεια της πορείας, με ελάχιστες αποκλίσεις. Στο γράφημα των γραμμικών ταχυτήτων (**κάτω αριστερά**), η εμπρόσθια ταχύτητα μεταβάλλεται με ρυθμικό τρόπο ανάλογα με τις απαιτήσεις της διαδρομής, ενώ η κάθετη ταχύτητα παραμένει πολύ χαμηλή, επιβεβαιώνοντας τη σταθερότητα του ύψους. Το γράφημα της γωνιακής ταχύτητας (**κάτω δεξιά**) δείχνει διαδοχικές περιστροφικές μεταβολές που σχετίζονται με τις απαραίτητες στροφές

κατά την πλοήγηση, χωρίς να εμφανίζονται ενδείξεις απότομης ή ασταθούς συμπεριφοράς. (**βλ. video 4, κεφ 2.3.2**)



Προσομοίωση 4: Απόκριση του συστήματος κατά τη διάρκεια της προσομοίωσης 4.
Η εικόνα παρουσιάζει τη συνολική απόκριση του UAV κατά τη διάρκεια μιας πιο απαιτητικής προσομοίωσης, όπου ενεργοποιούνται και απενεργοποιούνται δυναμικά εμπόδια κατά την πλοήγηση. Το drone επιδεικνύει αξιοσημείωτη ικανότητα προσαρμογής, αντιδρώντας άμεσα στις μεταβολές του περιβάλλοντος. Η πορεία του χαρακτηρίζεται από προσαρμοστικές αλλαγές διεύθυνσης και ταχύτητας, διατηρώντας παράλληλα τη σταθερότητα στο υψόμετρο και την πλοήγηση προς τους στόχους του. Η γενική εικόνα αποτυπώνει την αποτελεσματικότητα του ελεγκτικού αλγορίθμου και την ομαλή λειτουργία του συστήματος σε σενάρια με online τροποποίηση του περιβάλλοντος. Οι πράσινοι κύκλοι αντιπροσωπεύουν τους στόχους ενώ ο κόκκινος τα εμπόδια.

Κατά την προσομοίωση 4:

Στο γράφημα της πορείας (**πάνω αριστερά**) φαίνεται η συνολική τροχιά του drone στον χώρο XY. Παρατηρείται ότι το drone ακολουθεί μία τροχιά με σαφείς αποκλίσεις από τις ευθείες γραμμές, με καμπυλώσεις και επιτόπιες διορθώσεις που υποδεικνύουν την ενεργή παρουσία μεταβαλλόμενων εμποδίων. Σε διάφορα σημεία της πορείας, το drone μεταβάλλει κατεύθυνση ή απομακρύνεται από την ιδανική γραμμή πλεύσης, γεγονός που οφείλεται στην online ενεργοποίηση και απενεργοποίηση εμποδίων, στα οποία το σύστημα ανταποκρίνεται προσαρμόζοντας τη διαδρομή του. Στο γράφημα του υψομέτρου (**πάνω δεξιά**), παρατηρείται μία γρήγορη αρχική ανάβαση στο επιθυμητό επίπεδο ύψους και στη συνέχεια διατηρείται ένα σχεδόν σταθερό υψόμετρο κοντά στο 1 μέτρο. Οι μικρές διακυμάνσεις υποδεικνύουν προσαρμογές στις κάθετες κινήσεις του drone χωρίς να διακυβεύεται η σταθερότητα του συστήματος. Το γράφημα των γραμμικών ταχυτήτων (**κάτω αριστερά**) δείχνει ότι η ταχύτητα στον άξονα X μεταβάλλεται έντονα, με συχνές επιταχύνσεις και επιβραδύνσεις. Το γεγονός αυτό αντανακλά την προσπάθεια του συστήματος να αποφύγει εμπόδια και να εκτελέσει διορθώσεις πορείας. Αντίθετα, η κάθετη ταχύτητα στον άξονα Z παραμένει χαμηλή και σχεδόν σταθερή, γεγονός που υποστηρίζει τη σταθερή πτήση ως προς το ύψος. Στο γράφημα της γωνιακής ταχύτητας (**κάτω δεξιά**), διακρίνονται επαναλαμβανόμενες αιχμές και απότομες μεταβολές. Οι μεταβολές αυτές αποτελούν ενδείξεις ελιγμών στροφής (yaw) που απαιτούνται για να ακολουθηθεί η δυναμικά μεταβαλλόμενη πορεία, λόγω της εμφάνισης ή εξαφάνισης εμποδίων κατά την διάρκεια της προσομοίωσης. Οι αλλαγές στο πρόσημο της ταχύτητας δείχνουν στροφές είτε δεξιόστροφα είτε αριστερόστροφα, προσδίδοντας ευελιξία και ικανότητα άμεσης αντίδρασης στο σύστημα. **(βλ. video 5, κεφ 2.3.2)**

ΚΕΦΑΛΑΙΟ 5 Συμπεράσματα και μελλοντικές κατευθύνσεις

Η παρούσα πτυχιακή εργασία επικεντρώθηκε στη μελέτη, υλοποίηση και αξιολόγηση ενός αλγορίθμου Προβλεπτικού Ελέγχου (Model Predictive Control – MPC) για την αυτόνομη πλοήγηση ενός πολυκοπτέρου (drone) σε περιβάλλον προσομοίωσης. Με βάση το θεωρητικό πλαίσιο του MPC και αξιοποιώντας τεχνικές ανάλυσης της δυναμικής συμπεριφοράς, σχεδιάστηκε και υλοποιήθηκε ένας ελεγκτής ικανός να κατευθύνει το drone προς προκαθορισμένους στόχους, ενώ ταυτόχρονα αποφεύγει στατικά και δυναμικά εμπόδια. Στο πλαίσιο του έργου πραγματοποιήθηκε ενδελεχής τεχνική ανάλυση του λογισμικού, με τη δημιουργία δομών δεδομένων, υποσυστημάτων ελέγχου και επεξεργασίας αισθητήρων (LiDAR), καθώς και με υλοποίηση μονάδων απόφασης για την πλοήγηση. Μέσα από πολλαπλές προσομοιώσεις σε συνθετικά σενάρια, τεκμηριώθηκε η αποτελεσματικότητα του συστήματος, καθώς και η ικανότητά του να ανταποκρίνεται σε διαφορετικές συνθήκες πλοήγησης.

Σε επίπεδο προοπτικής, ο αλγόριθμος που αναπτύχθηκε παρουσιάζει σημαντικές δυνατότητες αξιοποίησης και επαναχρησιμοποίησης σε ένα ευρύ φάσμα ρομποτικών και αυτόνομων εφαρμογών. Εκτός από τη χρήση σε **σμήνη UAVs (multi-agent drones)** όπου απαιτείται συνεργατική αποφυγή εμποδίων και συντονισμένη πλοήγηση, μπορεί να επεκταθεί και σε **αυτόνομα εναέρια ρομπότ για εσωτερικούς χώρους (π.χ. αποθήκες ή βιομηχανικές εγκαταστάσεις)** όπου απαιτείται ακρίβεια και ευελιξία στον ελιγμό. Αντίστοιχα, ο ίδιος αλγόριθμος δύναται να εφαρμοστεί σε **ρομποτικά συστήματα επιθεώρησης υποδομών**, όπως αγωγοί, γέφυρες ή εργοτάξια, όπου απαιτείται ασφαλής πλοήγηση σε δυναμικά ή μη-δομημένα περιβάλλοντα. Επιπλέον, το ίδιο πλαίσιο μπορεί να υιοθετηθεί σε **αυτόνομα ρομπότ καθαρισμού** ή **συστήματα delivery μικρής κλίμακας** εντός αστικού περιβάλλοντος, καθώς και σε **αποστολές εξερεύνησης ή χαρτογράφησης** σε μη προσβάσιμες περιοχές (π.χ. σπήλαια, τούνελ ή σεισμόπληκτες ζώνες). Οι μελλοντικές επεκτάσεις του συστήματος περιλαμβάνουν την ενσωμάτωση εξελιγμένων δυναμικών μοντέλων, την υποστήριξη αβεβαιοτήτων μέσω stochastic ή robust MPC, καθώς και τη δυναμική παραμετροποίηση μέσω reinforcement learning.

Η εφαρμογή του αλγορίθμου σε πραγματικές συνθήκες θα απαιτήσει ορισμένες στοχευμένες βελτιστοποιήσεις. Πρώτον, η αντικατάσταση του απλοποιημένου κινηματικού μοντέλου από ένα πλήρες δυναμικό μοντέλο του drone είναι απαραίτητη ώστε να αντικατοπτρίζει πιο πιστά τη φυσική συμπεριφορά του UAV. Δεύτερον, η υποστήριξη πραγματικών αισθητήρων (π.χ. lidar, κάμερες, IMU) θα πρέπει να ενσωματωθεί, με παράλληλη χρήση φίλτρων εκτίμησης κατάστασης (όπως Extended Kalman Filter) για την αποφυγή θορύβου και αβεβαιότητας στα δεδομένα. Επιπλέον, απαιτείται προσεκτική διαχείριση του υπολογιστικού κόστους για χρήση σε embedded πλατφόρμες με περιορισμένους πόρους. Σε σενάρια όπως αυτόνομη επιτήρηση γεωργικών καλλιεργειών, αποστολές έρευνας και διάσωσης σε δύσβατες περιοχές, real-time παρακολούθηση εγκαταστάσεων ή αποστολές παράδοσης σε αστικές περιοχές, ο παρών αλγόριθμος μπορεί να αποτελέσει τον πυρήνα του

συστήματος πλοήγησης, προσφέροντας ακρίβεια, ευελιξία και ασφαλή απόκριση υπό πραγματικές επιχειρησιακές συνθήκες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. **Schwenzer M, Ay M, Bergs T, Abel D (2021)** Review on model predictive control: an engineering perspective. *Int J Adv Manuf Technol* 115:3899–3938. https://doi.org/10.1007/s00170-021-07682-3

2. **Jones C, Borrelli F, Morari M (2015)** Model Predictive Control Part I – Introduction. Institut für Automatik, ETH Zurich, Spring Semester 2015. https://ceid.utsa.edu/ataha/wp-content/uploads/sites/38/2017/10/MPC_Intro.pdf

3. **Peters L. (2021)**. Model Predictive Control – Part 1: Introduction to MPC [Βίντεο Διάλεξης]. YouTube Channel: **Cyrill Stachniss**. https://youtu.be/XaD8Lngfkzk

4. **Rezaee, A. (2017)**. Model Predictive Controller for Mobile Robot. Transactions on Environment and Electrical Engineering, 2(2), 17–22. https:// www.researchgate.net/publication/318003444

5. **Erez, T., Lowrey, K., Tassa, Y., Kumar, V., Kolev, S., & Todorov, E. (2013).** An integrated system for real-time Model Predictive Control of humanoid robots. Proceedings of IEEE Humanoids. https://roboti.us/lab/papers/ErezHumanoids13.pdf

6. **Singh, S., & Batra, A. (2021)**. Differentiable Robust Model Predictive Control. Proceedings of Robotics: Science and Systems. https://roboticsproceedings.org/rss20/p003.pdf

7. **Wang, J., Wu, Z., Christofides, P. D. (2024)**. Learning-based MPC for robust motion control in legged robots. Scientific Reports (Nature). https://www.nature.com/articles/s41598-024-66104-y

8. **Hatch & Boots (2021).** The Value of Planning for Infinite-Horizon MPC. https://arxiv.org/pdf/2104.02863

9. **Chen et al. (2024).** Prediction Horizon-Varying MPC for Autonomous Vehicle Control. https://doi.org/10.3390/electronics13081442

10. **Bohn et al. (2021).** Reinforcement Learning of the Prediction Horizon in MPC. https://www.sciencedirect.com/science/article/pii/S2405896321013380

11. **Gupta et al. (2023).** RL-based Variable Horizon MPC of Multi-Robot Systems. https://arxiv.org/pdf/2308.07071

12. **Nielsen, C., & Axehill, D. (2014).** A parallel algorithm for Newton step calculation in MPC. https://arxiv.org/pdf/1401.7882

13. **Salzmann, T., Rothe, A., & Hirche, S. (2022).** Real-Time Neural MPC: Deep Learning for Predictive Control in Robotics. https://arxiv. org/abs/2203.07747

14. **Anand, A., Rai, A., & Krishnamurthy, D. (2024).** On Model Mismatch in MPC: Analysis and Design Considerations**.** https://arxiv.org/pdf/2412.18268

15. **Sung, K., Sinha, A., & Ramanan, J. (2024).** Distributionally Robust Model Predictive Control with State-Input Estimation. https://arxiv.org/pdf/2407.19071

16. **Rawlings, J. B., & Mayne, D. Q. (2014).** Model Predictive Control: Theory and Design. Nob Hill Publishing. https://sites.engineering.ucsb.edu/~jbraw/mpc/MPC-book-2nd-edition-1st-printing.pdf

17. **Bemporad, A., & Morari, M. (2007).** Robust Model Predictive Control: A Survey. Springer. https://cse.lab.imtlucca.it/~bemporad/publications/papers/survey-robust-mpc.pdf

18. **Lucia, S., Andersson, J. A. E., Brandt, H., Diehl, M., & Engell, S. (2014).** Handling Uncertainty in Economic Nonlinear Model Predictive Control. Journal of Process Control. https://cdn.syscop.de/publications/Lucia2014.pdf

19. **Islam, M., Okasha, M., & Idres, M. M. (2017).** Dynamics and control of quadcopter using linear model predictive control approach. IOP Conference Series: Materials Science and Engineering. https://iopscience.iop.org/article/10.1088/1757-899X/270/1/012007/pdf

20. **Ma, H., Gao, Y., Yang, Y., & Xu, S. (2024).** Improved Nonlinear Model Predictive Control Based Fast Trajectory Tracking for a Quadrotor Unmanned Aerial Vehicle. https://doi.org/10.3390/drones8080387

21. **Alexandre Didier, Anilkumar Parsi1, Jeremy Coulson, Roy S. Smith. (2021).** Robust Adaptive Model Predictive Control of Quadrotors. https://arxiv.org/pdf/2102.13544

22. **Wang, Y., O'Keeffe, J., Qian, Q., & Boyle, D. (2022).** Interpretable Stochastic Model Predictive Control using Distributional Reinforced Estimation for Quadrotor Tracking Systems. https://arxiv.org/pdf/2205.07150

23. **Official Gazebo web information.** https://gazebosim.org/about

24. **Wiki Gazebo web information.** https://en.wikipedia.org/wiki/Gazebo_(simulator)

25. **Jan Hrebec (2023)**. Nonlinear Predictive Control of Unmanned Aerial Vehicle in Environments with Obstacles. https://dspace.cvut.cz/bitstream/handle/10467/108738/F3-BP-2023-Hrebec-Jan-Hrebec_Jan_Bachelor_Thesis.pdf

26. **Torrente, A., Kaufmann, E., Loquercio, A., Gehrig, M., Foehn, P., & Scaramuzza, D. (2021).** Data-Driven MPC for Quadrotors. https://rpg.ifi.uzh.ch/docs/RAL21_Torrente.pdf

27. **Rafalamao (2020).** hector_quadrotor_noetic package. https://github.com/RAFALAMAO/hector-quadrotor-noetic?tab=readme-ov-file