ΑΝΩΤΑΤΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ
ΤΕΧΝΟΛΟΓΙΚΟΥ ΤΟΜΕΑ

ΑΥΤΟΝΟΜΗ ΠΛΟΗΓΗΣΗ ΜΗ ΕΠΑΝΔΡΩΜΕΝΩΝ ΑΕΡΟΣΚΑΦΩΝ

Ιούλιος 2016

ΤΣΙΛΙΒΗΣ ΠΑΝΑΓΙΩΤΗΣ

Α.Μ: 40311

Επιβλέποντας Καθηγητής:

ΝΙΚΟΛΑΟΥ ΓΡΗΓΟΡΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος ..... **Τσιλιβής Παναγιώτης** ....., του ..... Εμμανουήλ....., με αριθμό μητρώου .....40311..... φοιτητής του τμήματος Μηχανικών Αυτοματισμού του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

Ο Δηλών

Τσιλιβής Παναγιώτης

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Συντομογραφιών

- **ROS – Robot Operating System**
- **UAV – Unmanned Aerial Vehicle**
- **HAL – Hardware Abstraction Layer**
- **MAV – Micro Air Vehicle**
- **PoI – Point of Interest**
- **IRLS – Iteratively Reweighted Least Squares**
- **PDF – Probability Density Function**
- **SLAM – Simultaneous Localization & Mapping**
- **EKF – Extended Kalman Filter**
- **SSD – Sum of Squared Differences**
- **LSD-SLAM – Large-Scale Direct SLAM**

1. Κεφάλαιο 1: Εισαγωγή

1.1. Περίληψη Πτυχιακής Εργασίας

Στην παρούσα πτυχιακή εργασία, γίνεται θεωρητική ανάλυση για την αυτόνομη πλοήγηση σε μη επανδρωμένα αεροσκάφη, όπως και ανάπτυξη μίας πρακτικής εφαρμογής με θέμα την αυτόνομη επιθεώρηση πτερύγων ανεμογεννήτριας με τη χρήση μη επανδρωμένων αυτόνομων αεροσκαφών **(UAV)**.

Πιο αναλυτικά, στο πρώτο κεφάλαιο γίνεται μία εισαγωγή στις βασικές έννοιες της αυτόνομης πλοήγησης, όπως και μία περιγραφή των ειδών και της χρησιμότητας των μη επανδρωμένων αεροσκαφών. Με άλλα λόγια, ορίζονται και περιγράφονται τα είδη ενός μη επανδρωμένου αεροσκάφους όπως και η διαφορετική τους χρησιμότητα σε διάφορες εφαρμογές. Επίσης, γίνεται λόγος στις βασικές έννοιες της αυτόνομης πλοήγησης, ενώ αναλύεται και η δομή κατασκευής των διαφόρων ειδών UAV.

Στο δεύτερο κεφάλαιο, γίνεται η θεωρητική προσέγγιση της πτυχιακής εργασίας. Εκεί δηλαδή, αναλύονται το λογισμικό που χρησιμοποιήθηκε στην εργασία **(software)**, όπως και το υλικό **(hardware)**. Επίσης, περιγράφονται αναλυτικά οι μέθοδοι και αλγόριθμοι ελέγχου ενός μη επανδρωμένου αεροσκάφους, δηλαδή ο έλεγχος μέσω PIDs, μέσω του φίλτρου Bayes και του φίλτρου Kalman, αλλά γίνεται ανάλυση και των μεθόδων και αλγόριθμων αυτόνομης πλοήγησης, όπως είναι το SLAM, το monocular SLAM και το PTAM.

Στο κεφάλαιο 3, γίνεται αναλυτική περιγραφή της πρακτικής εφαρμογής για την αυτόνομη επιθεώρηση πτερύγων ανεμογεννήτριας με τη χρήση UAV. Πιο συγκεκριμένα, αναλύονται οι στόχοι αλλά και οι δυσκολίες της συγκεκριμένης εφαρμογής, όπως και οι πιθανές λύσεις για την αντιμετώπιση αυτών. Επιπλέον, γίνεται αναφορά στις βασικές έννοιες της επιθεώρησης μίας πτέρυγας με drones και περιγράφονται και κάποια σημαντικά προβλήματα αεροδυναμικής φύσεως που πρέπει να αντιμετωπίσει ένα σύστημα ελέγχου ενός drone. Ακόμα, αναλύεται ο πηγαίος κώδικας που αναπτύχθηκε για τη συγκεκριμένη εφαρμογή, επεξηγώντας κάθε συνάρτηση ξεχωριστά όσον αφορά τη λειτουργία της, ενώ περιγράφεται και η δομή των δύο πακέτων που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Τέλος, παρουσιάζονται τα αποτελέσματα της εφαρμογής κατά την εκτέλεσή της, μέσα από στιγμιότυπα και στο τέλος της εργασίας, βρίσκεται η βιβλιογραφία στην οποία βασίστηκε η ανάπτυξη της πτυχιακής εργασίας.

1.2. UAV – Μη Επανδρωμένα Αεροσκάφη

Στην παρούσα χρονική περίοδο, ο όρος μη επανδρωμένα αυτόνομα αεροσκάφη αναφέρεται σε μια καινοτόμα τεχνολογία, η οποία στο εγγύς μέλλον θα έχει σημαντική παρουσία στην καθημερινότητα των ανθρώπων. Η ανάπτυξη αυτής της τεχνολογίας και η μετάβαση της από στρατιωτικό σε πιο εμπορικό προϊόν, ξεκίνησε στις αρχές του 21^{ου} αιώνα. Σήμερα (2016), η εξέλιξη της τεχνολογίας τους έχει φτάσει σε ένα ικανοποιητικό επίπεδο έτσι ώστε να χρησιμοποιούνται για πρακτικές εφαρμογές· γι' αυτό το λόγο η επιστημονική κοινότητα θεωρεί τα drones μια από τις τεχνολογίες του μέλλοντος, λόγω της πληθώρας εφαρμογών στη βιομηχανία αλλά και την ιδιωτική χρήση. Ξεκινώντας από την απλή εναέρια επιτήρηση μεγάλων χώρων, όπως εργοστασίων, την επιθεώρηση δυσπρόσιτων από τον άνθρωπο εγκαταστάσεων και την κινηματογραφική λήψη τοπίων· μέχρι και τον εντοπισμό θυμάτων σε πυρκαγιές, την παροχή πρώτων βοηθειών σε επείγοντες καταστάσεις ή ακόμα φαρμάκων σε εμπόλεμες ζώνες, τα μη επανδρωμένα αυτόνομα αεροσκάφη μπορούν να αποτελέσουν ένα σημαντικό παράγοντα για την αποτελεσματική εφαρμογή των παραπάνω.

Εικόνα 1: Μη Επανδρωμένα Αυτόνομα Αεροσκάφη (UAV) *[63]*

Ετυμολογικά, ο όρος μη επανδρωμένα αεροσκάφη προέρχεται από τον αγγλικό όρο **Unmanned Aerial Vehicles (UAV)**. Συχνά, αναφερόμαστε σε αυτά χρησιμοποιώντας την αγγλική λέξη **drones** για να υποδηλώσουμε ότι το εκάστοτε αεροσκάφος είναι αυτόνομο ή απλά μη επανδρωμένο. Υπάρχουν όμως ποικίλες κατηγορίες drones, οι οποίες διαφοροποιούνται ανάλογα με την δομή και την χρησιμοποίηση τους, όπως επίσης και διάφοροι τρόποι ελέγχου ενός UAV.

Οι βασικές κατηγορίες στις οποίες μπορούν να χωριστούν τα drones είναι τρείς και βασίζονται στην κατασκευή τους και την διαφορετική τεχνική πτήσης και ανύψωσης [1] [2].

- **Σταθερής Πτέρυγας – Fixed-Wing**

  Αυτό το είδος των UAV έχει το γνωστό σχήμα του αεροπλάνου. Οι πτέρυγες του οχήματος είναι σταθερές στον κορμό του, και ως σύνολο δημιουργούν την απαραίτητη άνωση για να απογειωθεί το drone αλλά συμβάλλουν επίσης και στη διατήρηση του ύψους πτήσης του. Ακόμα, λόγω της δομής του, το παρόν drone υπακούει στους φυσικούς νόμους της αεροπλοΐας, με αποτέλεσμα να είναι πιο ελέγξιμο από έναν χειριστή και πιο ευέλικτο σε τυχόν χειριστικά και τεχνικά προβλήματα. Επίσης λόγω της κατασκευής του, τα συγκεκριμένα drones έχουν τη δυνατότητα να κουβαλάνε πιο βαρύ εξοπλισμό και για μεγαλύτερες αποστάσεις· κάτι το οποίο τα καθιστά ιδανικά για διανομές πακέτων, ιδίως σε απομακρυσμένα μέρη. Τέλος, το σημαντικό μειονέκτημα αυτού του είδους UAV είναι η αδυναμία σταθερής πτήσης πάνω από ένα σημείο – το γνωστό hover των ελικοπτέρων – το οποίο ωθεί το drone στην έλλειψη ακρίβειας θέσης.

Εικόνα 2: Παράδειγμα Μη Επανδρωμένου Αεροσκάφους Σταθερής Πτέρυγας *[59]*

- **Περιστροφικής Πτέρυγας – Rotary-Wing**

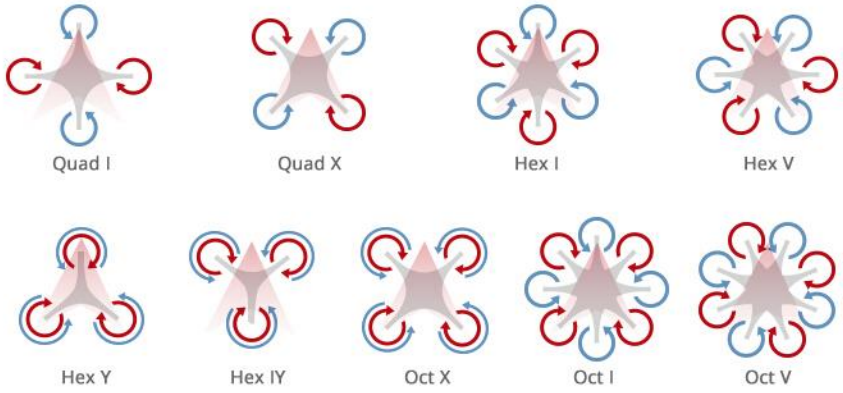
  Το συγκεκριμένο είδος UAV έχει τη δομή του συμβατικού ελικοπτέρου, με έναν κύριο περιστρεφόμενο έλικα, με τη μόνη διαφορά ότι είναι μικρότερης κλίμακας και μη επανδρωμένο αεροσκάφος. Το σημαντικό πλεονέκτημα που διαθέτει σε σχέση με την πρώτη κατηγορία, είναι η δυνατότητα κάθετης απογείωσης και προσγείωσης η οποία παρέχει στον χρήστη πιο εύκολο χειρισμό ακόμα και σε μικρούς χώρους. Βέβαια, λόγω πολύπλοκου χειρισμού το είδος αυτό εντάσσεται στην κατηγορία χειρισμού από απόσταση μέσω πιλότου και σε αυτήν της πλήρως αυτόνομης πλοήγησης, η οποία μπορεί να εφαρμοστεί στα άλλα δύο είδη. Βασικό χαρακτηριστικό αυτού του είδους είναι η ικανότητα σταθερής πτήσης πάνω από ένα σημείο – **hover** – κάτι το οποίο είναι ιδανικό για εφαρμογές όπως κινηματογραφική φωτογράφηση και βιντεοσκόπηση τοπίων.

Εικόνα 3: Παράδειγμα Μη Επανδρωμένου Αεροσκάφους Περιστρεφόμενης Πτέρυγας *[60]*

- **Πολλαπλών Ελίκων - Multi-Rotor**

  Η κατηγορία αυτή είναι η πιο κοινή όσον αφορά τα εμπορικά micro UAV. Η δομή τους αποτελείται από πολλούς έλικες τοποθετημένους περιμετρικά από το κύριο σώμα. Τα drones αυτά χωρίζονται σε διάφορα είδη αναλόγως με τον αριθμό των ελίκων τους.

  - **3 έλικες (τρικόπτερο – tricopter)**
  - **4 έλικες (τετρακόπτερο – quadcopter)**
  - **6 έλικες (εξακόπτερο – hexacopter)**
  - **8 έλικες (οκτακόπτερο – octocopter)**



Εικόνα 4: Είδη multicopter σύμφωνα με τον αριθμό ελίκων και τον σχηματισμό τους *[61]*

Εκτός από τις παραπάνω κύριες κατηγορίες, υπάρχουν και πιο σπάνιες περιπτώσεις όπως drones με 12 ή 16 έλικες ή με 8 έλικες που σχηματίζουν το λατινικό γράμμα **V.** Γενικά, με την συνεχή ανάπτυξη αυτής της τεχνολογίας οι διαμορφώσεις της δομής των drones αλλάζουν συνεχώς έτσι ώστε κάθε φορά να εξυπηρετούν το σκοπό τους όσο το δυνατό πιο αποτελεσματικά.

Τα multi-rotor drones έχουν παρόμοια χαρακτηριστικά πτήσεις με τα UAV περιστρεφόμενης έλικας, μόνο που είναι ακόμα πιο αποδοτικά. Με άλλα λόγια, τα συγκεκριμένα drones έχουν σημαντικά μεγαλύτερη σταθερότητα στον αέρα κατά την διάρκεια της πτήσης τους, καθώς και όταν κάνουν hover. Επίσης, είναι αρκετά πιο εύκολα στον έλεγχο από απόσταση μέσω χειριστή, όπως και στον αυτόματο έλεγχο. Για το λόγω αυτό, χρησιμοποιούνται για εφαρμογές που απαιτούν ακρίβειας θέσης και ομαλή κίνηση, όπως επαγγελματική βιντεοσκόπηση, επίβλεψη και επιθεώρηση χώρων, εναέρια χαρτογράφηση και παρακολούθηση υποδομών. Δυστυχώς, η τεχνολογία ενέργειας δεν έχει φτάσει ακόμα στο βέλτιστο επίπεδο όσον αφορά την αυτονομία ενός τέτοιου drone, με την μπαταρία πλέον να διαρκεί περίπου 20 λεπτά στα εμπορικά και

μέχρι 40 λεπτά σε στρατιωτικού επιπέδου UAV. Έτσι, η αυτόνομη πλοήγηση αυτών των drone θέλει αρκετό προγραμματισμό έτσι ώστε να είναι όσο το δυνατό πιο αποτελεσματικό και γρήγορο στην αποστολή του.

Στην παρούσα διπλωματική εργασία, η ανάπτυξη της έρευνας που έχει πραγματοποιηθεί όσον αφορά την αυτόνομη πλοήγηση μη επανδρωμένων αεροσκαφών, βασίζεται στην κατηγορία των multicopter και πιο συγκεκριμένα γίνεται χρήση ενός τετρακόπτερου (quadcopter) κατάλληλο για ακαδημαϊκή χρήση.

1.3. Βασικές Έννοιες Αυτόνομης Πλοήγησης

Ο έλεγχος και ο χειρισμός ενός μη επανδρωμένου εναέριου αεροσκάφους είναι αρκετά περίπλοκος λόγω των πολλών παραμέτρων που υπάρχουν στο σύστημα, για το λόγο αυτό οι μέθοδοι ελέγχου ποικίλουν.

Ο πιο κοινός είναι αυτός του ελέγχου από απόσταση με τηλεχειρισμό. Αυτό φυσικά αναιρεί την ιδιότητα του πλήρως αυτόνομου, αλλά αυξάνει τον παράγοντα ασφάλειας για το drone κάτι το οποίο είναι αρκετά σημαντικό. Βέβαια, η παρουσία ενός έμπειρου χειριστή και ορισμένες φορές πιλότου, ανεβάζει αισθητά και το κόστος αλλά και το ποσοστό λάθους λόγω της αλληλεπίδρασης με τον ανθρώπινο παράγοντα.

Στο παρόν, ο έλεγχος για τον οποίο γίνεται η περισσότερη έρευνα και ανάπτυξη, έτσι ώστε να είναι εντελώς ασφαλής και αποδοτική η πτήση ενός drone κατά τη διάρκεια μιας αποστολής, είναι η μέθοδος της αυτόνομης πλοήγησης . Η αυτόνομη πλοήγηση βασίζεται κατά κύριο στο σύστημα ελέγχου του drone σε συνδυασμό με το σύστημα αισθητήρων του. Πιο συγκεκριμένα, το σύστημα ελέγχου επικοινωνεί με μεγάλη συχνότητα με τους αισθητήρες του drone, έτσι ώστε να δέχεται συνεχή ανάδραση των εξόδων του συστήματος για την καλύτερη αντιμετώπιση των σφαλμάτων. Έτσι μέσω αλγορίθμων ελέγχου και πλοήγησης, οι οποίοι θα αναλυθούν στη συνέχεια, αντιμετωπίζονται εγκαίρως σφάλματα που έχουν ήδη γίνει αντιληπτά, αλλά και μελλοντικά σφάλματα μέσω αλγορίθμων πρόβλεψης. Επίσης, το drone δημιουργεί ένα χάρτη πτήσης μέσω των αισθητήρων αναγνώρισής του, ο οποίος μπορεί να είναι απλός χάρτης συντεταγμένων ή ακόμα και κάποιος χάρτης προσομοίωσης. Το αποτέλεσμα ενός πλήρους συστήματος ελέγχου και πλοήγησης εφαρμοσμένο σε ένα drone, είναι μία ομαλή αυτόνομη πλοήγηση με δυνατότητα πρόβλεψης κινήσεων, αποφυγή εμποδίων και αναταράξεων και εφαρμογή ειδικού συστήματος πτήσης ανάλογα με την εκάστοτε αποστολή.

2. <u>Κεφάλαιο 2:</u> Θεωρητική Προσέγγιση

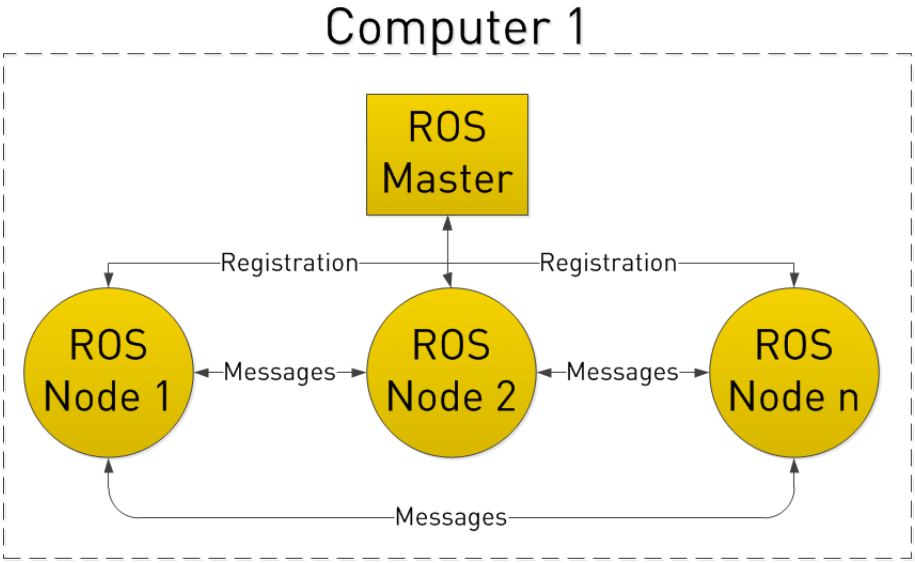
2.1. Σύνοψη Κεφαλαίου

Στη ανάπτυξη της παρούσας διπλωματικής εργασίας, το λογισμικό όπως και οι αλγόριθμοι πλοήγησης είχαν τον κυριότερο ρόλο. Η έρευνα βασίστηκε στο meta-operating λειτουργικό σύστημα ROS [3] το οποίο τρέχει σε περιβάλλον Linux Ubuntu. Ακόμα, ο πηγαίος κώδικας που αναπτύχθηκε είναι βασισμένος στη γλώσσα προγραμματισμού Python, με ορισμένες αναφορές σε C++. Στο κεφάλαιο αυτό, εκτός από την εκτενή περιγραφή του λογισμικού συστήματος **(software)** που χρησιμοποιήθηκε, θα αναπτυχθούν πλήρως και οι αλγόριθμοι ελέγχου και πλοήγησης ενός drone. Επίσης, θα περιγραφεί και ο εξοπλισμός **(hardware)** που χρησιμοποιήθηκε για την εφαρμογή των θεωρητικών προσεγγίσεων αλλά και το σύστημα προσομοίωσης.

2.2. Λογισμικό – Software

2.2.1. ROS – Robot Operating System

Ένα από τα πλέον διαδεδομένα λειτουργικά συστήματα ανάπτυξης λογισμικού σε ρομπότ, είναι το ***ROS*** [3]. Το ακρωνύμιο ROS προέρχεται από τις λέξεις *Robot Operating System* (το οποίο στα ελληνικά μεταφράζεται ως *Ρομποτικό Λειτουργικό Σύστημα)*. Είναι ένα *meta-operating* σύστημα το οποίο εκτός από το να προσφέρει τις γνωστές λειτουργίες ενός κλασικού λειτουργικού συστήματος, παρέχει και κάποιες επιπλέον χρήσιμες υπηρεσίες. Πιο συγκεκριμένα, παρέχει την δυνατότητα χρήσης ενός υποσυστήματος που ονομάζεται στρώμα αφαίρεσης υλικού **(HAL)**, το οποίο δίνει στο λογισμικό απευθείας πρόσβαση στο hardware κομμάτι του ρομπότ. Επίσης, υπάρχει η υπηρεσία επικοινωνίας μεταξύ διεργασιών με τη χρήση μηνυμάτων δημοσίευσης/εγγραφής **(Publish/Subscribe)**, η οποία επιτρέπει στο πρόγραμμα να επικοινωνεί σε πραγματικό χρόνο με διάφορες ξεχωριστές διεργασίες (nodes) μέσω διαφόρων **topics**· αλλά και η υπηρεσία διαχείρισης λογισμικού πακέτου. Πιο συγκεκριμένα, το σύστημα το διαχειρίζεται το **ROS Master**, το οποίο είναι το κεντρικό σύστημα που επιτρέπει στις υπόλοιπες διεργασίες **(nodes)** να επικοινωνούν μεταξύ τους χωρίς την ανάγκη αποστολής μηνυμάτων σε διάφορες πόρτες (π.χ. 127.0.0.32) από τον χρήστη.



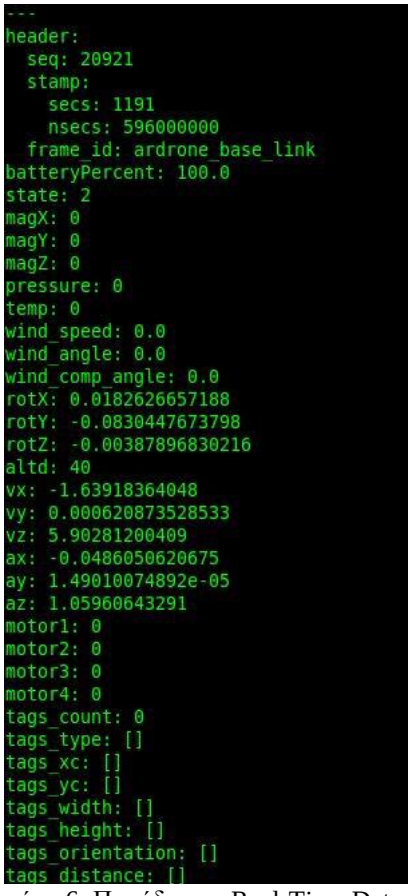
Εικόνα 5: Επικοινωνία μεταξύ των διεργασιών (nodes) στο σύστημα ROS *[62]*

Το παρόν λειτουργικό σύστημα έχει ευρεία εφαρμογή στο χώρο της ρομποτικής για τον λόγο του ότι είναι ένα λογισμικό ανοιχτού κώδικα **(open-source)**. Επιπλέον, προσφέρει στο χρήστη ένα σύνολο από εργαλεία και βιβλιοθήκες, τα οποία απλοποιούν σε μεγάλο βαθμό την δυσκολία ανάπτυξης μίας σύνθετης εφαρμογής για ρομπότ. Ακόμα, παρέχει στο χρήστη

την δυνατότητα επιλογής γλώσσας προγραμματισμού για την ανάπτυξη της εκάστοτε ρομποτικής εφαρμογής. Για έναν μηχανικό η δημιουργία και η ανάπτυξη πηγαίου κώδικα για ένα ρομπότ είναι μια αρκετά δύσκολη διαδικασία, η οποία περιλαμβάνει αρκετή έρευνα όπως και πρακτική εξάσκηση. Έτσι, η δυνατότητα επιλογής της γλώσσας που τον διευκολύνει, η ενσωμάτωση προ υπάρχοντών πακέτων όπως και η εξομοίωση του ρομπότ σε πραγματικό χρόνο, είναι λίγα από τα πλεονεκτήματα που διαθέτει το ROS.

Σε αυτή τη πτυχιακή εργασία, και κατ' επέκταση έρευνα, το σύστημα ROS χρησιμοποιείται κατά κύριο λόγο ως το βασικό μέσο επικοινωνίας μεταξύ του UAV και του ηλεκτρονικού υπολογιστή. Τα βασικά πλεονεκτήματα που προσφέρει σε αυτή την εργασία το ROS, είναι η πληθώρα επιλογών από προϋπάρχοντα πακέτα τα οποία συμβάλουν στην ανάπτυξη του λογισμικού της έρευνας. Έτσι, αποφεύγεται η ανάγκη δαπάνης χρόνου για την ανάπτυξη εργαλείων υποβοήθησης της έρευνας.

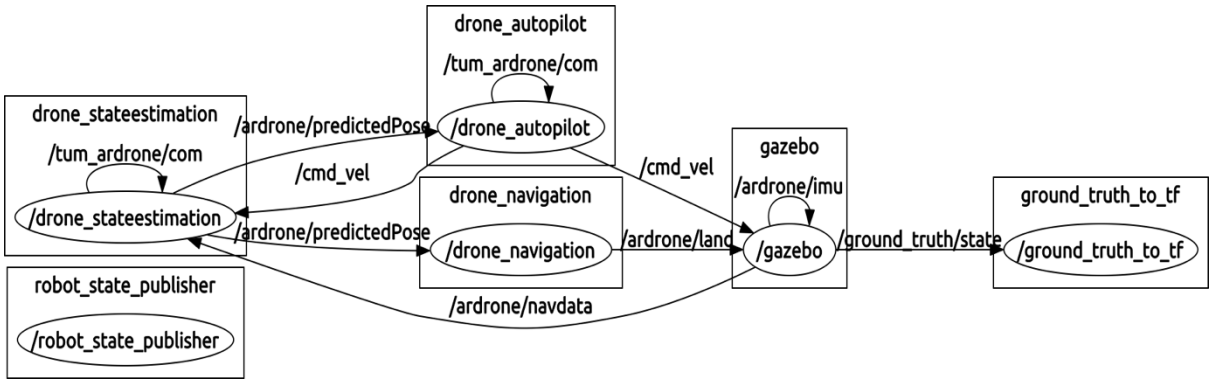
Για παράδειγμα, στα πλαίσια των επιπρόσθετων πακέτων, το ROS υποστηρίζει τη λειτουργία προσομοίωσης με τη χρήση της εφαρμογής **Gazebo.** Ακόμα, παρέχει δεδομένα σε πραγματικό χρόνο **(real-time data)** τα οποία είναι αρκετά σημαντικά για τον σωστό έλεγχο ενός ρομπότ, ιδίως ενός drone, για την αντιμετώπιση προβλημάτων αλλά και την παρακολούθηση της διαδικασίας εκτέλεσης της εκάστοτε εφαρμογής. Στην προκειμένη περίπτωση δε, τα δεδομένα αυτά ήταν οι βασικές παράμετροι στους αλγόριθμους ελέγχου τα οποία επέτρεπαν στο drone την σταθερή πτήση. Επίσης, όπως προαναφέρθηκε, το ROS προσφέρει εργαλεία παρακολούθησης και επεξεργασίας του συστήματος σε πραγματικό χρόνο όπως για παράδειγμα τη γραφική απεικόνιση της σύνδεσης των διεργασιών **(nodes)** μεταξύ τους ή ακόμα και την δυναμική επεξεργασία **(Dynamic Reconfigure)** κάποιων παραμέτρων εν ώρα πτήσης, κάτι το οποίο είναι σημαντικά χρήσιμο σε έναν μηχανικό.



Εικόνα 6: Παράδειγμα Real-Time Data

Τέλος, το ROS λόγω του ότι υποστηρίζει τη παράλληλη λειτουργία του διαφόρων διεργασιών το καθιστά ιδανικό για την παρούσα πτυχιακή, για το λόγο του ότι επιτρέπει τη συνεχή εκτέλεση του λογισμικού driver του drone μαζί με τις υπόλοιπες εξωτερικές διεργασίες, όπως για παράδειγμα την εκτέλεση ενός κώδικα **(script).**

Ως παράδειγμα των παράλληλων nodes, στην *Εικόνα 7* απεικονίζεται ολόκληρο το σύστημα των διεργασιών της εφαρμογής που αναπτύχθηκε στα πλαίσια της πτυχιακής εργασίας έτσι ώστε το drone να εκτελεί μία αυτόνομη πτήση γύρω από ένα σημείο. Είναι ξεκάθαρο ότι στην *Εικόνα 7* απεικονίζονται τα **topics** που επικοινωνούν μεταξύ τους, κάτι το οποίο χρειάζεται γνώση της εκάστοτε εφαρμογής για να είναι κατανοητό.



Εικόνα 7: Διάγραμμα απεικόνισης της επικοινωνίας των διεργασιών σε μία εφαρμογή

Παρακάτω, θα αναλυθούν περαιτέρω οι δυνατότητες που προσέφερε το ROS στην παρούσα πτυχιακή, με σκοπό την καλύτερη περιγραφή και κατανόηση του συστήματος.

2.2.2. Πακέτα Ανοιχτού-Λογισμικού – Open-Source ROS Packages

Όπως προαναφέρθηκε, το σύστημα ROS [3] προσφέρει την επιλογή ενσωμάτωσης διαφόρων πακέτων ανοιχτού λογισμικού στην εφαρμογή ενός χρήστη. Η δομή του λογισμικού που εμπεριέχεται στο ROS, είναι χωρισμένη σε πακέτα **(packages)**. Με τον όρο πακέτα εννοούμε ένα σύνολο από αρχεία κώδικα **(scripts),** βιβλιοθήκες **(libraries), ROS nodes**, αρχεία ρυθμίσεων **(configuration files)** κ.α. τα οποία αποτελούν ή μια ολοκληρωμένη εφαρμογή ή κάποιο βοήθημα **(plugin)**. Τα πακέτα αυτά είναι προσβάσιμα σε διαδικτυακούς αποθηκευτικούς χώρους **(repositories)** όπως είναι το **GitHub.** Μέσω αυτών των repositories, ο χρήστης κατεβάζει το πακέτο που τον διευκολύνει και το ενσωματώνει στην εφαρμογή του. Βέβαια, για εμπορικές εφαρμογές ο χρήστης είναι υποχρεωμένος να πάρει την έγκριση του δημιουργού του κάθε πακέτου, έτσι ώστε να αποφύγει την κατηγορία της αντιγραφής. Ο σκοπός ύπαρξης των πακέτων αυτών είναι η επαναχρησιμοποίηση χρήσιμων εργαλείων για την ανάπτυξη εφαρμογών, αποφεύγοντας την δημιουργία αυτών από την αρχή.

Πιο συγκεκριμένα, η δομή ενός τέτοιου πακέτου ακολουθεί την εξής διάταξη φακέλων και αρχείων:

- *include/όνομα_πακέτου:* Περιέχει **header files** όταν γίνεται χρήση της C++.
- *msg/:* Περιέχει τους τύπους μηνυμάτων που χρησιμοποιούνται για την περιγραφή δεδομένων όταν γίνονται **publish** από τα **ROS nodes**.
- *src/όνομα_πακέτου:* Περιέχει πηγαία αρχεία **(Source files)** και ιδίως αρχεία κώδικα Python.
- *srv/:* Περιέχει τους τύπους υπηρεσιών **(services)**, οι οποίοι σχετίζονται άμεσα με τους τύπους μηνυμάτων, επιτρέποντας την επικοινωνία μεταξύ των nodes.
- *scripts/:* Περιέχει εκτελέσιμα αρχεία κώδικα **(scripts)**.
- *CMakeLists.txt:* Βασικό αρχείο για την δημιουργία του λογισμικού πακέτου.
- *package.xml:* Αρχείο το οποίο περιέχει πληροφορίες για το λογισμικό πακέτο.

Είναι σημαντικό να αναλυθούν κάποιοι όροι οι οποίοι θα αναφέρονται και στην πορεία της εργασίας αυτής.

- **Publish/Subscribe:** Η μέθοδος αυτή είναι η πιο σημαντική όσον αφορά την επικοινωνία μεταξύ των nodes. Για να διαμοιραστούν οι πληροφορίες μέσω των nodes χρησιμοποιούνται τα **μηνύματα** τα οποία μεταφέρουν πληροφορίες μέσω των **topics**. Ο τρόπος με τον οποίο δημοσιεύονται τα topics στον ευρύ κόμβο του συστήματος, έτσι ώστε να μπορεί να διαβαστεί από όλο το σύστημα ROS, είναι η μέθοδος **publish**. Η αντίθετη ακριβώς λειτουργία, αυτή του **subscribe**, είναι όταν γίνεται εγγραφή σε ένα topic με σκοπό την λήψη των δεδομένων που ανταλλάσσονται σε εκείνο το topic. Γενικά, η μέθοδος του publish/subscribe διευκολύνει όχι μόνο την εσωτερική επικοινωνία του συστήματος, αλλά και την αλληλεπίδραση του με τον χρήστη. Βασικό παράδειγμα για την περαιτέρω κατανόηση, είναι όταν ο χρήστης χρειάζεται να επεξεργαστεί ή απλά να ληφθεί τα real-time δεδομένα από το drone.
- **Topics:** Τα λεγόμενα topics είναι τα **μέσα** με τα οποία μεταφέρονται τα μηνύματα με τη μέθοδο **publish/subscribe**. Τα ονόματα τους έχουν μικρή σημασία στο σύστημα,, κάτι το οποίο τα κάνει αρκετά προσαρμόσιμα στις ανάγκες ενός χρήστη. Ακόμα, η διαφορά τους με τα **services** είναι ότι τα topics επικοινωνούν με μόνη κατεύθυνση μεταξύ των nodes και δεν μπορούν να δεχτούν απάντηση σε ένα request. Επίσης, ο τύπος του κάθε topic καθορίζεται από τον τύπο του μηνύματος το οποίο στέλνεται μέσω αυτού. Είναι σημαντικό δε να επισημανθεί ότι είναι συχνό λάθος σε μια ανάπτυξη λογισμικού να γίνεται **subscribe** σε ένα topic, με σκοπό να γίνει λήψη των δεδομένων που γίνονται **publish**, με λανθασμένο τύπο ο οποίος απλά διαφέρει από τον προκαθορισμένο τύπο μηνύματος και έτσι το σύστημα να χτυπάει λάθος. Τέλος, σε μία εφαρμογή, όπως είναι η πρακτική εφαρμογή αυτής της πτυχιακής, τα topics που τρέχουν στο παρασκήνιο είναι πολυάριθμα και αυτό απεικονίζεται και στην *Εικόνα 8*. Όπως είναι εμφανές, τα περισσότερα ξεκινάνε με το όνομα **/ardrone/**…, το οποίο είναι το όνομα του drone που χρησιμοποιήθηκε για την πρακτική εφαρμογή (και θα αναλυθεί στο επόμενο κεφάλαιο), και παρέχουν βασικές πληροφορίες για το drone αλλά και για τα εξωτερικά λογισμικά που εκτελούνται παράλληλα.

```
/altimeter
/ardrone/bottom/camera_info
/ardrone/bottom/image_raw
/ardrone/bottom/image_raw/compressed
/ardrone/bottom/image_raw/compressed/parameter_descriptions
/ardrone/bottom/image_raw/compressed/parameter_updates
/ardrone/bottom/image_raw/compressedDepth
/ardrone/bottom/image_raw/compressedDepth/parameter_descriptions
/ardrone/bottom/image_raw/compressedDepth/parameter_updates
/ardrone/bottom/image_raw/theora
/ardrone/bottom/image_raw/theora/parameter_descriptions
/ardrone/bottom/image_raw/theora/parameter_updates
/ardrone/bottom/parameter_descriptions
/ardrone/bottom/parameter_updates
/ardrone/camera_info
/ardrone/front/camera_info
/ardrone/front/image_raw
/ardrone/front/image_raw/compressed
/ardrone/front/image_raw/compressed/parameter_descriptions
/ardrone/front/image_raw/compressed/parameter_updates
/ardrone/front/image_raw/compressedDepth
/ardrone/front/image_raw/compressedDepth/parameter_descriptions
/ardrone/front/image_raw/compressedDepth/parameter_updates
/ardrone/front/image_raw/theora
/ardrone/front/image_raw/theora/parameter_descriptions
/ardrone/front/image_raw/theora/parameter_updates
/ardrone/front/parameter_descriptions
/ardrone/front/parameter_updates
/ardrone/image_raw
/ardrone/image_raw/compressed
/ardrone/image_raw/compressed/parameter_descriptions
/ardrone/image_raw/compressed/parameter_updates
/ardrone/image_raw/compressedDepth
/ardrone/image_raw/compressedDepth/parameter_descriptions
/ardrone/image_raw/compressedDepth/parameter_updates
/ardrone/image_raw/theora
/ardrone/image_raw/theora/parameter_descriptions
/ardrone/image_raw/theora/parameter_updates
/ardrone/imu
/ardrone/land
/ardrone/navdata
/ardrone/predictedPose
/ardrone/reset
/ardrone/takeoff
/clock
/cmd_vel
/drone_autopilot/parameter_descriptions
/drone_autopilot/parameter_updates
/drone_stateestimation/parameter_descriptions
/drone_stateestimation/parameter_updates
/fix
/fix_velocity
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
```

Εικόνα 8: Παράδειγμα λίστας από topics που τρέχουν στο παρασκήνιο για μια εφαρμογή αυτόνομης πλοήγησης.

- **<u>Μηνύματα – Messages:</u>** Τα nodes μίας εφαρμογής ROS επικοινωνούν μεταξύ τους μέσω δημοσίευσης μηνυμάτων σε topics. Τα μηνύματα αυτά είναι απλής δομής κείμενα τα οποία παρέχουν δεδομένα σε απλές μορφές, όπως είναι ακέραιοι ή Boolean, ή ακόμα και πιο σύνθετες, δηλαδή λίστες (arrays).
- **<u>Υπηρεσίες – Services:</u>** Λόγω του ότι τα topics έχουν μονής κατεύθυνσης επικοινωνία, δεν μπορούν να εξυπηρετήσουν περιπτώσεις που το σύστημα χρειάζεται επικοινωνία **request/reply**. Οι **υπηρεσίες** χρησιμοποιούνται για αυτό το λόγο και εκτός από το να υποστηρίζουν λειτουργίες του συστήματος ROS, παρέχουν και στο χρήστη επιλογές για άμεση επεξεργασία και αλλαγή δεδομένων όπως για παράδειγμα το *Dynamic Reconfigure* που προαναφέρθηκε.

Στη παρούσα πτυχιακή εργασία, χρησιμοποιήθηκε και μελετήθηκε εκτενώς το ανοιχτού-λογισμικού πακέτο ***tum_ardrone*** [4], το οποίο αποτέλεσε και σημαντικό κομμάτι της πρακτικής εφαρμογής για την αυτόνομη πλοήγηση ενός micro UAV. Επίσης το πακέτο ***ardrone_autonomy*** [5] αποτέλεσε ένα εξίσου σημαντικό εργαλείο μιάς και ήταν και το βασικό driver του drone. Τα δύο αυτά πακέτα θα αναλυθούν περαιτέρω στο κεφάλαιο 3 λόγω του ότι είναι μέρος της πρακτικής εφαρμογής.

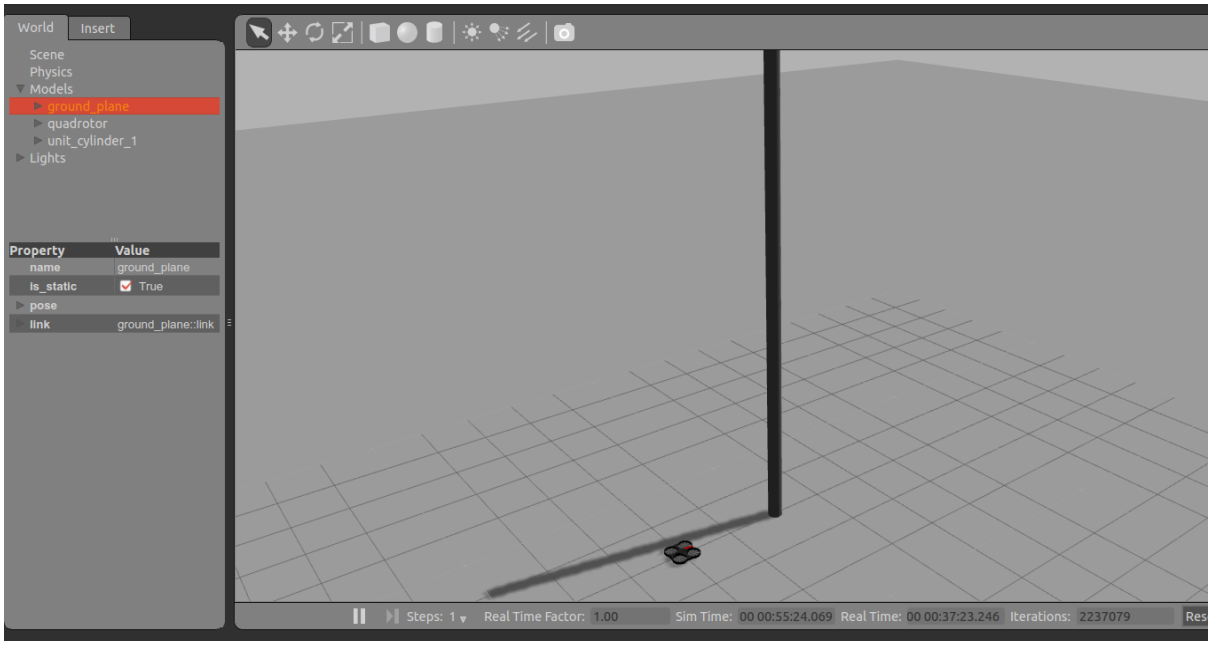
2.2.3. Λογισμικό Προσομοίωσης – Gazebo Simulator

Είναι γνωστό ότι στον επιστημονικό χώρο ανάπτυξης ρομποτικών εφαρμογών οι δοκιμές είναι ένα από τα πιο σημαντικά κομμάτια. Μπορεί θεωρητικά η εφαρμογή να δουλεύει χωρίς σφάλματα και στην πράξη να βγαίνουν προβλήματα που δεν είχαν προβλεφθεί. Η δυνατότητα, όμως, που υπάρχει τα τελευταία χρόνια του να γίνονται οι πρακτικές εφαρμογές ενός ρομπότ μέσα από προηγμένα λογισμικά προσομοίωσης δίνει άλλη διάσταση στην ανάπτυξη πρακτικών εφαρμογών. Πλέον, οι δοκιμές σε κάθε στάδιο ανάπτυξης γίνονται μέσα από προσομοιωτές με σκοπό να προβλεφθούν και να αποφευχθούν σφάλματα που μπορεί να επιφέρουν βλάβη στο εκάστοτε μηχάνημα.

Το βασικότερο λογισμικό που χρησιμοποιήθηκε για την προσομοίωση αυτής της πρακτικής εφαρμογής, ονομάζεται **Gazebo** [6]. Αποτελεί έναν τρισδιάστατο προσομοιωτή **(3D Dynamic Simulator)** ο οποίος τρέχει σε περιβάλλον Linux με την υποστήριξη του ROS. Λόγω της δομής ανοιχτού λογισμικού (**Open-Source)** που έχει, το gazebo είναι ένα από τα πλέον διαδεδομένα και χρησιμοποιούμενα λογισμικά προσομοίωσης που υπάρχουν. Το πλεονέκτημα του να μπορεί ένας χρήστης να δημιουργήσει τον δικό του εικονικό κόσμο, ο οποίος μπορεί

και μιμείται με μεγάλη ακρίβεια τις πραγματικές συνθήκες της εκάστοτε εφαρμογής, παρέχει στο χρήστη ένα σημαντικό βοήθημα για την ανάπτυξη του project του. Το gazebo έχει την δυνατότητα προσομοίωσης εσωτερικών και εξωτερικών χώρων, δίνοντας μεγάλη ακρίβεια και συνοχή στα τεχνικά ζητήματα της κάθε εφαρμογής και στη φυσική του εκάστοτε εικονικού κόσμου. Για παράδειγμα, το **gazebo simulator** χρησιμοποιείται για δοκιμές σε *αλγόριθμους ρομποτικής*, *σχεδίασης ρομπότ* και εφαρμογές *πλοήγησης οχημάτων*. Ακόμα, σημαντικά προτερήματα αποτελούν οι επιλογές σε πληθώρα διαφορετικών ιδιοτήτων φυσικής αναπαράστασης, η μεγάλης ποικιλίας open-source βιβλιοθήκη μοντέλων ρομπότ, περιβαλλόντων και αισθητήρων, και το φιλικό στο χρήστη περιβάλλον προσομοίωσης. Τέλος, είναι ουσιώδες να αναφερθεί ότι ο χρήστης έχει τη δυνατότητα ανάπτυξης και ενσωμάτωσης, στην open-source διαδικτυακή βιβλιοθήκη, μοντέλα ρομπότ και εικονικούς κόσμους που εφαρμόζουν κατάλληλα στο project του.

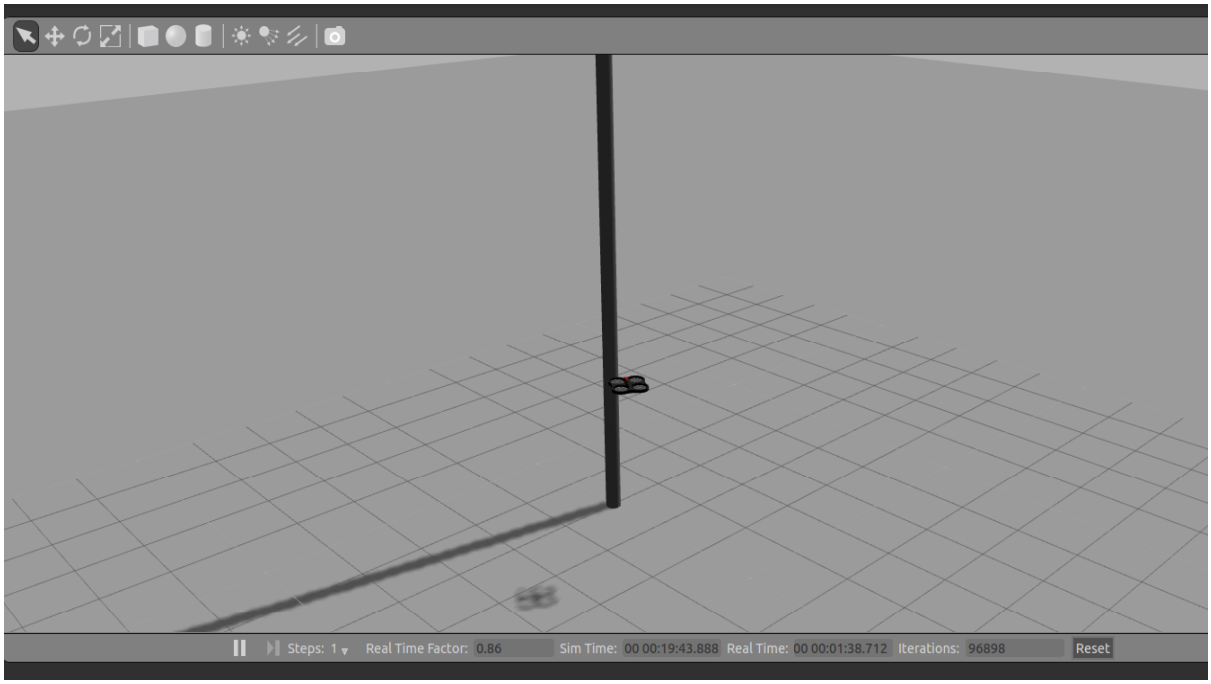
Σε αυτήν την πτυχιακή εργασία, η πρακτική εφαρμογή αναπτύχθηκε σε μεγάλο βαθμό με βάση του **gazebo simulator,** λόγω των συχνών δοκιμών, για να προβλεφθούν τυχόν λάθη αλλά και επειδή εκμηδένιζε το πρόβλημα ενέργειας (φόρτιση, εκφόρτιση μπαταρίας του drone). Επομένως, μερικά από τα πλεονεκτήματα που πρόσφερε ο προσομοιωτής αυτός στην εφαρμογή, ήταν η επίλυση του προβλήματος ενέργειας του drone, η λύση στην έλλειψη χώρου δοκιμών και η δημιουργία τεχνιτών αναταράξεων με σκοπό την βελτιστοποίηση ελέγχου πτήσης του drone. Στην παρακάτω *Εικόνα 9* φαίνεται ο εικονικός κόσμος, στην απλούστερη μορφή του, ο οποίος χρησιμοποιούταν για τις συχνές δοκιμές κατά τη διάρκεια ανάπτυξης της εφαρμογής αυτόνομης πλοήγησης του τετρακόπτερου.



Εικόνα 9: Επισκόπηση εικονικού κόσμου της πρακτικής εφαρμογής στο Gazebo.

Όπως είναι εμφανές στην *Εικόνα 9*, στο αριστερό σημείο του παραθύρου υπάρχουν όλες οι επιλογές της προσομοίωσης. Ο χρήστης έχει τη δυνατότητα να επεξεργαστεί τις φυσικές ιδιότητες του εικονικού αυτού κόσμου αλλά και τις ιδιότητες των μοντέλων και αισθητήρων που παίρνουν μέρος στην εφαρμογή. Επίσης, στη δεύτερη καρτέλα υπάρχει η επιλογή εκχώρησης ήδη υπάρχοντων μοντέλων και σκηνικών, τα οποία είναι αποθηκευμένα σε έναν διαδικτυακό χώρο **(repository)**.

Στην *Εικόνα* 10 απεικονίζεται μία δοκιμή της εφαρμογής αυτόνομης επιθεώρησης μίας φτερωτής σε ανεμογεννήτρια, η οποία για λόγους ευκολίας έχει αντικατασταθεί από έναν κάθετο στύλο.



Εικόνα 10: Στιγμιότυπο από δοκιμή της εφαρμογής αυτόνομης επιθεώρησης με drone

Εν τέλει, το Gazebo Simulator προσφέρει αρκετά πλεονεκτήματα σε μία ανάπτυξη ρομποτικής εφαρμογής, σαν και αυτή. Το open-source σύστημα του μαζί με την πληθώρα των χρηστών που το χρησιμοποιούν, το καθιστούν ιδανικό μέσω προσομοίωσης εφαρμογών για λόγους όπως πρόσβαση σε ήδη υπάρχοντα μοντέλα, ευκολία για συχνές δοκιμές (ιδίως όταν το εκάστοτε ρομπότ έχει σημαντική αξία), και ακριβής αναπαράσταση σκηνικών και γενικώς απαιτητικών εφαρμογών.

2.3. Hardware

Στη παρούσα πτυχιακή εργασία, ο εξοπλισμός που χρησιμοποιήθηκε κατά τη διάρκεια της έρευνας, αλλά και για τη πρακτική εφαρμογή, ήταν ένα ακαδημαϊκού επιπέδου μη επανδρωμένο αεροσκάφος με την εμπορική ονομασία **Parrot AR Drone 2.0** [7]. Παρακάτω θα αναλυθεί περαιτέρω η δομή του drone, όπως και τα εξαρτήματα αυτού, λαμβάνοντας υπόψιν πληροφορίες από την δημοσίευση [8].

2.3.1. Parrot AR Drone 2.0

Το **AR Drone** [7] είναι ένα τετρακόπτερο αεροσκάφος μικρής κλίμακας το οποίο κατά την εμπορική του χρήση ελέγχεται από απόσταση μέσω κινητού τηλεφώνου και της αντίστοιχης εφαρμογής του. Το όνομα του προέρχεται από της αγγλικές λέξεις *Augmented Reality Drone*, το οποίο στα ελληνικά μεταφράζεται ως *"Μη Επανδρωμένο Αεροσκάφος Επαυξημένης Πραγματικότητας"*, και αναπτύχθηκε με απώτερο στόχο τόσο την εμπορική του χρήση, όσο και την ακαδημαϊκή και στρατιωτική του χρησιμοποίηση.

Εικόνα 11: Parrot AR Drone 2.0. *[7]*

Το παρόν micro UAV (ή αλλιώς **MAV)** είναι σχεδιασμένο για να μπορεί να πετάει σε εσωτερικούς και εξωτερικούς χώρους με αυξημένη σταθερότητα βάσει του εργονομικού σχεδιασμού του, ο οποίος το καθιστά αρκετά ανθεκτικό σε κραδασμούς και πτώσεις. Πιο συγκεκριμένα, το **ardrone (AR Drone)** αποτελείται κατά κύριο λόγο από ειδικό φελιζόλ το οποίο παίζει σημαντικό ρόλο στο βάρος του αεροσκάφους αλλά και στο μειωμένο του κόστος. Εν αντιθέσει όμως με το ότι είναι αρκετά ελαφρύ, είναι σταθερό κόντρα σε ανέμους του περιβάλλοντος όπως και σε πτήσεις κλειστού χώρου. Βέβαια, λόγω του ότι το συγκεκριμένο drone φτιάχτηκε για ερασιτεχνική χρήση, με χειρισμό από απόσταση, το open-loop σύστημα του είναι αρκετά ασταθές με σκοπό η ανατροφοδότηση **(feedback)** να ελέγχεται πλήρως από το χρήστη για περισσότερη ευχαρίστηση. Αυτό όμως τροφοδοτεί και το προγραμματισμό αυτού σε πλήρως αυτόνομο με βέλτιστο έλεγχο. Έτσι, το μεγαλύτερο πλεονέκτημα του ardrone

είναι ότι η εταιρεία Parrot [7] έχει αναπτύξει το λογισμικό του συγκεκριμένου drone ως open-source, με αποτέλεσμα να έχουν ήδη αναπτυχθεί αναρίθμητα πακέτα για βέλτιστο έλεγχο, αυτόνομη πλοήγηση, ή ακόμα και πιο σπάνια όπως τον εντοπισμό συγκεκριμένων αντικειμένων από το ardrone.

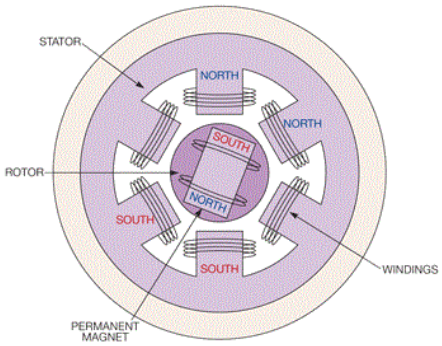
Σημαντικό είναι να αναφερθεί η εσωτερική δομή του ardrone, δηλαδή τα εξαρτήματα που το απαρτίζουν (τα σημαντικότερα από αυτά θα αναλυθούν στη πορεία λεπτομερώς):

- 4 Μοτέρ χωρίς ψήκτρα **(brushless motors)**
- Επεξεργαστής: Parrot P6 ARM9-core **(Parrot P6 processor)**
- Σύστημα Πλοήγησης: 16bits PIC micro-controller
- 2 Κάμερες Μονού Οφθαλμού **(Monocular Cameras)**
- Αδρανειακό Σύστημα Μέτρησης **(IMU – Inertial Measurement Unit)**
- 2 Υπερηχητικοί Αισθητήρες **(Ultrasonic Sensors)**

2.3.2. Αισθητήρες & Μοτέρ – Sensors & Motors

Κινητήρες: Τα μοτέρ που χρησιμοποιούνται σε ένα πολυκόπτερο είναι συνήθως μοτέρ χωρίς ψήκτρες, τα οποία βοηθούν το drone στην απόδοση και την εξοικονόμηση ενέργειας. Με άλλα λόγια, τα **brushless motors** αποδίδουν καλύτερα με ίδια παροχή συνεχόμενου ρεύματος σε σχέση με τα **brushed motors** και επίσης είναι πιο ανθεκτικά αφού έχουν χαμηλότερη ευαισθησία σε μηχανικές βλάβες. Ωστόσο, λόγω της ανεπτυγμένης τεχνολογίας τους τα συγκεκριμένα μοτέρ έχουν υψηλότερο κόστος συγκριτικά με τα κανονικά, όπως επίσης υψηλότερο κόστος έχουν και τα ηλεκτρονικά εξαρτήματα γι' αυτά. Γενικά, τα brushless motors έχουν πολλά πλεονεκτήματα σε σχέση με τα συμβατικά μοτέρ όπως υψηλή ροπή σε αναλογία με το βάρος, περισσότερη ροπή ανά Watt, λιγότερο θόρυβο και αυξημένη αξιοπιστία, όμως αυτό εξισορροπείται με το υψηλό κόστος τους.

Η δομή των συγκεκριμένων μοτέρ είναι αρκετά απλή και ακολουθεί τη βασική ηλεκτρομαγνητική. Πιο συγκεκριμένα, στο εσωτερικό του μοτέρ υπάρχουν μόνιμοι ηλεκτρομαγνήτες **(electromagnets)** οι οποίοι είναι τοποθετημένοι σε έναν ρότορα **(rotor)**, αυτοί περιστρέφονται στο κέντρο ενός στάτωρα **(stator)** στον οποίο είναι τοποθετημένες σταθερές περιελίξεις· εκμηδενίζοντας έτσι προβλήματα σύνδεσης ηλεκτρικού ρεύματος με τον οπλισμό. Το συλλέκτη ηλεκτρικού ρεύματος **(brush)** τον αντικαθιστά σε αυτή τη



Εικόνα 12: Δομή ενός Brushless Motor *[58]*

περίπτωση ένα ηλεκτρονικό σύστημα ελέγχου, το οποίο αλλάζει με σταθερή χρονική κατανομή ισχύος τη φάση στις περιελίξεις με σκοπό τη συνεχή περιστροφή του κινητήρα.

Το **ardrone** είναι εξοπλισμένο με **4 brushless motors** τα οποία έχουν ενσωματωμένο το ολοκληρωμένο ηλεκτρονικό σύστημα ελέγχου **(BLCB – Brushless Control Board)**, έτσι ώστε να είναι πιο εύκολη και αποτελεσματική η επικοινωνία αυτών με την κεντρική μονάδα. Πιο συγκεκριμένα, το κάθε μοτέρ είναι συνδεδεμένο και με τον δικό του μικροεπεξεργαστή **(microcontroller)**, ο οποίος ονομάζεται **ATMEGA8L 8bit**, όπως και με το δικό του σύστημα αποκοπής **(cutout system)** έτσι ώστε να είναι δυνατό το σταμάτημα της περιστροφής σε περίπτωση σύγκρουσης της προπέλας με αντικείμενο. Στην *Εικόνα 13* φαίνεται το ολοκληρωμένο σύστημα ενός από τα τέσσερα μοτέρ που υπάρχουν στο ardrone.

Εικόνα 13: Ολοκληρωμένο Σύστημα BLCB για το AR Drone 2.0 *[9]*

<u>Αισθητήρες:</u> Στην περίπτωση αυτή, που -όπως προαναφέρθηκε- υπάρχουν 2 υπερηχητικοί αισθητήρες, το **ardrone** τους χρησιμοποιεί στο σύστημα πλοήγησης του για τον υπολογισμό ύψους από την επιφάνεια της γης, όταν αυτό βρίσκεται σε σχετικά μικρό ύψος. Για μεγαλύτερο από 6 μέτρα ύψος, το ardrone χρησιμοποιεί το αλτίμετρο που διαθέτει, έτσι ώστε να έχει μεγαλύτερη ακρίβεια στους υπολογισμούς του.

Η βασική λειτουργία ενός **ultrasonic** αισθητήρα είναι η εκπομπή και λήψη σήματος. Ένας ultrasonic αισθητήρας έχει σαν δομή δύο διαφορετικά εξαρτήματα, εκ' των οποίων το ένα είναι υπεύθυνο για την εκπομπή ενός ακουστικού κύματος και το άλλο για την λήψη αυτού. Για τον εντοπισμό της απόστασης, ο αισθητήρας αυτός εκπέμπει το ακουστικό κύμα προς την εκτιμώμενη κατεύθυνση και υπολογίζει το χρόνο που κάνει το σήμα μέχρι να ληφθεί η ανάκλαση του από το δέκτη. Με αυτή τη διαδικασία, ο αισθητήρας μπορεί και εκτιμά την απόσταση με μεγάλη ακρίβεια και μικρό κόστος ενέργειας. Το **ardrone** διαθέτει δύο **Prowave Ultrasonic Sensors** τα οποία είναι συνδεδεμένα με τον **PIC μικρο-ελεγκτή** για την διαχείριση του εκπομπού και την ψηφιοποίηση του λαμβάνοντος σήματος. Εκτός από την εκτίμηση του

ύψους, τα αισθητήρια αυτά χρησιμοποιούνται και για τον εντοπισμό μη επιθυμητών μετατοπίσεων του UAV όπως και για την εκτίμηση του βάθους της περιοχής που βρίσκεται το drone. Τέλος, τα συγκεκριμένα αισθητήρια έχουν συχνότητα συντονισμού τα 40kHz και μπορούν να υπολογίσουν με ακρίβεια απόσταση μέχρι και 6 μέτρα.

Στην κατηγορία των αισθητηρίων του συστήματος πλοήγησης ανήκουν και τα επιταχυνσιόμετρο και γυροσκόπιο, τα οποία αναλυθούν στην περιγραφή του συστήματος IMU.

2.3.3. IMU – Inertial Measurement Unit

Το σύστημα **IMU**, το οποίο σε ελεύθερη ελληνική μετάφραση σημαίνει *"Σύστημα Αδρανειακής Εκτίμησης"*, είναι το κυριότερο ηλεκτρονικό εξάρτημα σε ένα μη επανδρωμένο αεροσκάφος. Αποτελείται από αισθητήρες όπως γυροσκόπια **(Gyroscope)** και επιταχυνσιόμετρα **(Accelerometer)** τα οποία παρέχουν real-time δεδομένα με σκοπό την εκτίμηση της θέσης του drone στον τρισδιάστατο χώρο.

Στην προκειμένη περίπτωση, το **AR Drone** διαθέτει ένα σύστημα IMU το οποίο περιέχει ένα επιταχυνσιόμετρο **Bosch BMA150** τριών διαστάσεων συνδεδεμένο με έναν μετατροπέα **A/D** 10 δυφίων, και δύο γυροσκόπια τα οποία είναι υπεύθυνα το ένα για τις διαστάσεις **x** και **y**, και το άλλο για την κάθετη διάσταση ύψους **z.** Το IMU αυτό, τρέχει στη συχνότητα των 200Hz έτσι ώστε να τροφοδοτεί το drone με συνεχόμενες μετρήσεις. Πιο συγκεκριμένα, το γυροσκόπιο των οριζόντιων διαστάσεων **(x,y)** ονομάζεται **Invensense IDG500** και είναι ένας αναλογικός αισθητήρας του οποίου το σήμα επεξεργάζεται μέσω του PIC 12 A/D μετατροπέα και γίνεται ψηφιακό. Ο αισθητήρας αυτός έχει τη δυνατότητα να μετρήσει έως και 500 μοίρες το δευτερόλεπτο με μεγάλη ακρίβεια, η οποία παρέχει στο σύστημα πλοήγησης του drone σημαντική ανάδραση. Επίσης, στον κατακόρυφο άξονα το γυροσκόπιο **Epson XV3700** παρέχει στο σύστημα ακόμα μεγαλύτερης ακρίβειας μετρήσεις, με την έξτρα δυνατότητα της μεθόδου αυτόματης εκμηδένισης για τυχόν θόρυβο στις μετρήσεις του.

Γενικότερα, ένα IMU δίνει πληροφορίες στο σύστημα πλοήγησης ενός drone όσον αφορά τη γωνιακή του ταχύτητα, το μαγνητικό πεδίο το οποίο κάποιες φορές δημιουργείται γύρω του, και την ειδική δύναμη που ασκείται πάνω του. Η ειδική δύναμη **(Specific Force)** αποτελείται από το πηλίκο της μη βαρυτικής δύναμης προς τη συνολική μάζα του drone.

$$\boldsymbol{Ειδικη\ Δυναμη} = \frac{\boldsymbol{Δυναμη_{μη-βαρυτικη}}}{\boldsymbol{Μαζα}}$$

Η μονάδα μέτρησής της είναι $meters/second^2$ και υποδηλώνει την επιτάχυνση της βαρύτητας σε σχέση με τη μάζα του drone, γι' αυτό αναφέρεται συχνά και ως **g-force**.

Έτσι, μέσω αυτού του ηλεκτρονικού εξαρτήματος το αεροσκάφος μπορεί να εντοπίσει τη θέση του και την κατάσταση του. Η μέθοδος που ακολουθεί το σύστημα κατά την επεξεργασία των δεδομένων που λαμβάνει από το IMU, ονομάζεται **dead reckoning**. Στην ουσία το drone εντοπίζει, με τη βοήθεια του ηλεκτρονικού του συστήματος και των μετρήσεων από το IMU, την παρούσα θέση του με βάση την προηγούμενη. Πιο συγκεκριμένα, η μέθοδος αυτή δέχεται ως δεδομένα την *επιτάχυνση*, την *ταχύτητα*, την *πορεία* του αεροσκάφους και την *απόσταση* που έχει διανύσει, και υπολογίζει την πιο πιθανή *θέση* του drone. Στα συστήματα πλοήγησης οχημάτων, η συγκεκριμένη μέθοδος εφαρμόζεται ευρέως λόγω του ότι αντικαθιστά το σύστημα **GPS** σε σημεία που είναι αδύνατο να πιάσει **(GPS-denied areas)**. Έτσι συνδυάζοντας την τεχνολογία GPS με τη μέθοδο αυτή, τα συστήματα πλοήγησης έχουν τη δυνατότητα ακρίβειας όσον αφορά την παρούσα θέση του οχήματος.

Βέβαια, όπως κάθε σύστημα που δρα σε πραγματικό χρόνο και χώρο, έτσι και το αεροσκάφος έχει απώλειες στις μετρήσεις του λόγω διαταραχών. Ως εκ τούτου και η μέθοδος dead reckoning υπόκειται σε σημαντικά λάθη ορισμένες φορές, τα οποία αν δεν υπάρχει βέλτιστος έλεγχος του συστήματος μπορούν να επιφέρουν βλάβες. Για παράδειγμα, αν το σύστημα λάβει μετρήσεις οι οποίες απέχουν από την πραγματικότητα, όσον αφορά την παρούσα θέση και κατάσταση του οχήματος, τότε το αποτέλεσμα του dead reckoning θα είναι ανακριβές και λανθασμένο· οδηγώντας έτσι το αεροσκάφος σε λάθος θέση.

Το παραπάνω παράδειγμα είναι και ένα από τα σημαντικότερα μειονεκτήματα του IMU. Λόγω των διαταραχών που υπάρχουν κατά τη διάρκεια μίας πτήσης, το IMU είναι πολύ πιθανό να δώσει στιγμιαίες μετρήσεις ακατάλληλες για το σύστημα με αποτέλεσμα να επηρεάσουν την κίνηση του drone. Δυστυχώς, στην αυτόνομη πλοήγηση αν μία μέτρηση επηρεάσει το σύστημα μία χρονική στιγμή, η αλληλουχία των σφαλμάτων θα είναι μεγάλη οδηγώντας έτσι το εκάστοτε UAV σε μία σειρά από λανθασμένες κινήσεις. Σε πολλές περιπτώσεις δε, το σφάλμα μπορεί να μεγιστοποιηθεί από την μία θέση στην άλλη λόγω του σφάλματος Abbe **(Abbe error)** [10]. Με άλλα λόγια, το Abbe error περιγράφει την αύξηση του γωνιακού σφάλματος με βάση την απόσταση ως εξής:

$$\boldsymbol{\varepsilon = h * sin(\theta)} \qquad \textbf{(2.1)}$$

Όπου:

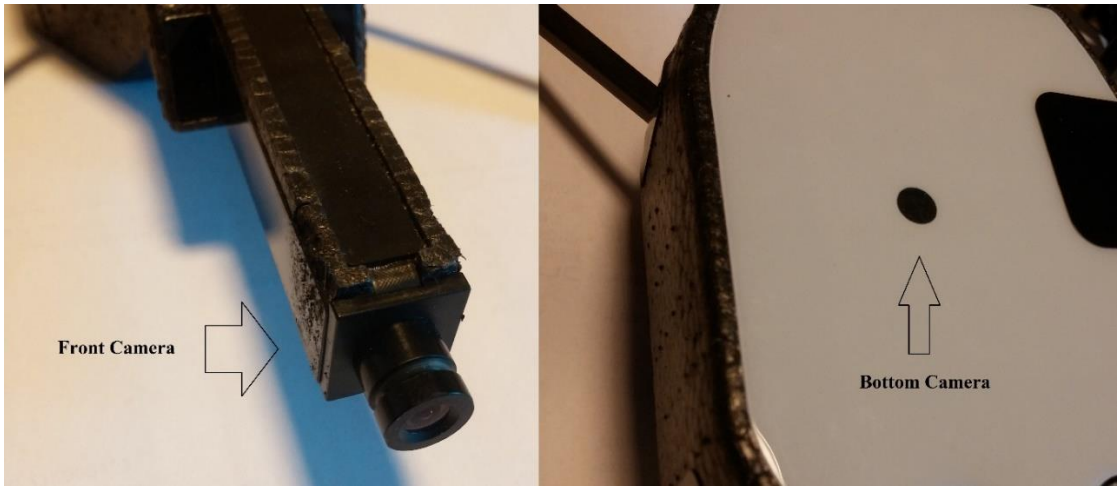
- **ε** = το σφάλμα
- **h =** η απόσταση από την αρχική θέση
- **θ** = η γωνία με βάση την αρχική θέση

Ως αποτέλεσμα των παραπάνω, το IMU είναι ένα αναπόσπαστο εξάρτημα σε ένα σύστημα πλοήγησης, και ιδίως όταν αυτή είναι αυτόνομη. Βέβαια, μπορεί να επιφέρει σημαντικά προβλήματα, λόγω των μεθόδων πρόβλεψης που βασίζονται σε αυτό, αν δεν υπάρχει βέλτιστος έλεγχος στο σύστημα πλοήγησης.

2.3.4. Κάμερα Μονού Οφθαλμού – Monocular Camera

Λόγω της συνεχής εξέλιξης της τεχνολογίας, πολλά ηλεκτρονικά εξαρτήματα τα οποία σε προηγούμενα χρόνια ήταν δύσκολο να εγκατασταθούν πάνω σε οχήματα, πλέον μπορούν. Πιο συγκεκριμένα, η ανεπτυγμένη τεχνολογία στις τυπικές κάμερες μονού οφθαλμού **(Monocular Camera)** επιτρέπει πλέον να κατασκευάζονται σε μικρό μέγεθος, με εξίσου καλή ανάλυση και μικρό κόστος. Βάσει αυτού, οι κάμερες μονού οφθαλμού χρησιμοποιούνται πλέον ως βασικό ηλεκτρονικό εξάρτημα σε συστήματα αυτόνομης και μη πλοήγησης, με αποτέλεσμα να έχουν αναπτυχθεί και ειδικοί αλγόριθμοι πλοήγησης βασισμένοι σε αυτές τις κάμερες.

Το **AR Drone,** στην προκειμένη περίπτωση, διαθέτει δύο κάμερες μονού οφθαλμού οι οποίες είναι τοποθετημένες σε διαφορετική θέση και για διαφορετικό σκοπό. Η κύρια κάμερα **(front camera)** του ardrone είναι τοποθετημένη έτσι ώστε το τετρακόπτερο να "κοιτάει" μπροστά και να μπορεί κατά κύριο λόγο να βιντεοσκοπεί με ευκολία. Τεχνικά, η κάμερα αυτή χρησιμοποιεί ένα φακό ευρείας γωνίας ***93 μοιρών*** με ανάλυση κλίμακας ***VGA 640x480*** και ***15 frame/second***. Από την άλλη, η δεύτερη κάμερα **(vertical ή bottom camera)** είναι τοποθετημένη στην "κοιλιά" του drone με σκοπό να είναι προσανατολισμένη προς το έδαφος, για λόγους πλοήγησης και μέτρησης της γραμμικής ταχύτητας του αεροσκάφους. Η κάμερα αυτή διαθέτει φακό ***64 μοιρών*** με χαμηλότερης κλίμακας ανάλυση, ενώ μπορεί να καταγράφει μέχρι και ***60 frames/second***.



Εικόνα 14: Η front και η bottom camera του AR.Drone 2.0.

Οι εν λόγω οπτικοί αλγόριθμοι **(Vision Algorithms)**, που χρησιμοποιούνται με σκοπό το drone να είναι ικανό να πλοηγείτε αυτόνομα, βασίζονται στο περιεχόμενο του σκηνικού που καταγράφουν μέσω των καμερών και των μετρήσεων του αεροσκάφους και βάσει αυτών, επιλέγονται αναλόγως.

Ο πρώτος αλγόριθμος υπολογίζει την οπτική ροή **(optical flow)** όλης της εικόνας μέσα από ένα σύστημα πολλαπλής ανάλυσης **(multi-resolution)** και χρησιμοποιεί τη μεθοδολογία που περιγράφεται στη θεωρία που αναπτύχθηκε από τους (B. D. Lucas και T. Kanade) [11, pp. 121-130] όπως εφαρμόζει και ένα σύστημα εξομάλυνσης χωρικών και χρονικών διαφοροποιήσεων σύμφωνα με τους (B. G. Schunck και B. K. Horn) [12, pp. 81-87]. Με άλλα λόγια, το σύστημα στην αρχή λαμβάνει δύο φωτογραφίες του τοπίου και αναλύει τυχόν διαφορές. Κατά τη διάρκεια προσπάθειας βελτίωσης της ανάλυσης του οπτικού πεδίου, όμως, οι αλλαγές χαρακτηριστικών δεν λαμβάνονται υπόψιν με σκοπό να εξαλειφθεί το σφάλμα. Για την επίτευξη αυτού, οι εικόνες αναλύονται ως προς την μετατόπιση του κέντρου εικόνας τους, η οποία έχει δημιουργηθεί λόγω της αλλαγής των χαρακτηριστικών σημείων μέσα στο τοπίο, και έπειτα την αφαίρεση των διαφορών κέντρου μέσω υπολογισμών.

Ο δεύτερος αλγόριθμος, ο οποίος ονομάζεται και "αλγόριθμος εντοπισμού γωνιών" **(corner tracking)**, υπολογίζει την μετατόπιση διαφόρων σημαντικών στοιχείων **(points of interest – PoI)** μέσα στην εικόνα του drone. Μία συνεχής διαδικασία υπολογισμού με τη χρήση της μεθόδου σταθμισμένων ελαχίστων τετραγώνων **(Iteratively reweighted least squares – IRLS)**, είναι αυτή που προσδιορίζει και την ταχύτητα της εικόνας του drone και το εύρος μετατόπισης των PoI. Η κύρια λειτουργία του αλγόριθμου αυτού γίνεται μέσω ανιχνευτών γωνίας **(FAST corner trackers)**, που περιγράφονται στην δημοσίευση των (M. Trajkovic και M. Hedley) [13, pp. 75-87], οι οποίοι στέλνουν πληροφορίες στο σύστημα και σύμφωνα με την ποιότητα εντοπισμού, τοποθετείται ένας καθορισμένος αριθμός από trackers πάνω από κάθε γωνία αμέσως μετά την τελευταία ανάλυση περιεχομένου των εικόνων. Τέλος, η εκτίμηση IRLS λαμβάνεται θέτοντας το βάθος του σκηνικού ως ομοιόμορφο, λόγω της έλλειψης που έχει η κάμερα μονού οφθαλμού στο να εντοπίζει βάθος.

Εάν συγκριθούν τα πλεονεκτήματα που προσφέρει ο κάθε αλγόριθμος, η έκβαση που προκύπτει για τον πρώτο είναι ότι παρέχει έγκυρα αποτέλεσμα μέχρι και σε χαμηλής κλίμακας αντίθεση **(contrast)**, κάτι το οποίο επιτρέπει στο αεροσκάφος μεγαλύτερη ταχύτητα. Από την άλλη, ο δεύτερος αλγόριθμος χρησιμοποιείται κυρίως για ακρίβεια **(accuracy)** εντοπισμού γωνιών στο σκηνικό, με την προϋπόθεση ότι το αεροσκάφος κινείται με ταχύτητα κάτω από μία προκαθορισμένη τιμή και ότι το τοπίο είναι διαυγές από την κάμερα. Τέλος, οι δύο αυτοί αλγόριθμοι χρησιμοποιούνται εναλλάξ ανάλογα με την κατάσταση του drone.

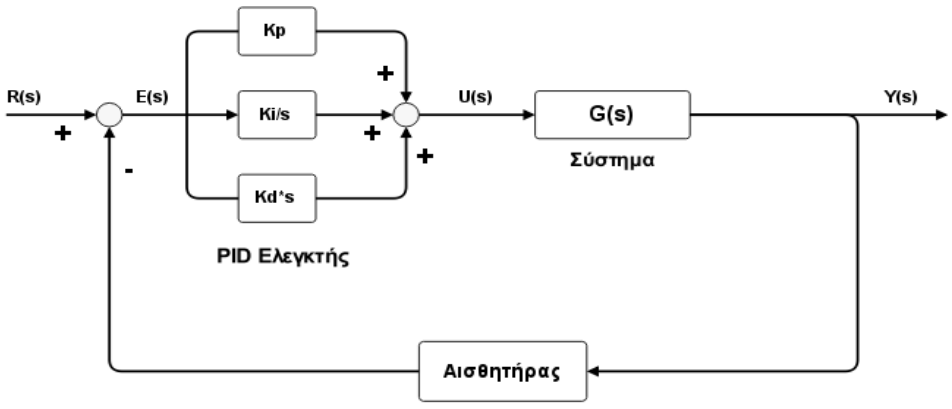
2.4. Μέθοδοι και Αλγόριθμοι Ελέγχου

2.4.1. Ελεγκτής PID

Οι ελεγκτές PID είναι τα πιο διαδεδομένα συστήματα ελέγχου στο χώρο της βιομηχανίας, όπως και τον ακαδημαϊκό χώρο. Ξεκίνησαν την ανάπτυξη τους εν έτη 1930, πρώτα με πνευματικό τρόπο λειτουργίας και αργότερα μέσω ηλεκτρονικών συστημάτων. Μέχρι και σήμερα οι ελεγκτές αυτοί χρησιμοποιούνται για τον βασικό έλεγχο ενός συστήματος, χωρίς όμως να αποτελούν τη βέλτιστη λύση σε απαιτητικές εφαρμογές. Συνήθως, οι ελεγκτές αυτοί προτιμάτε να χρησιμοποιούνται μαζί με κάποια ψηφιακά φίλτρα (π.χ. Kalman Filter [14]) ή ακόμα και με πιο έξυπνα συστήματα, όπως ελεγκτές ασαφούς λογικής, έτσι ώστε να γίνεται πιο σωστός έλεγχος του συστήματος.

Ετυμολογικά, ο όρος **PID** προέρχεται από τις λέξεις **Proportional (P) – Integral (I) – Derivative (D)** οι οποίοι στα ελληνικά μεταφράζονται ως **Αναλογικοί – Ολοκληρωτικοί – Διαφορικοί** ελεγκτές. Σύμφωνα με τη λειτουργία του κάθε ελεγκτή, μπορούν να χρησιμοποιηθούν ξεχωριστά ή ανά διαφορετικούς τύπους οι οποίοι είναι: **PI, PD** και **PID.** Αναλόγως με την εκάστοτε εφαρμογή, συνήθως, επιλέγεται και ο καταλληλότερος τύπος έτσι ώστε να γίνεται ο σωστός έλεγχος του συστήματος και ταυτόχρονα να αποφεύγεται η περίπλοκη ρύθμιση τους **(tuning)**.

Ο βασικός μηχανισμός λειτουργίας ενός PID ελεγκτή είναι *η συνεχής σύγκριση της επιθυμητής τιμής εξόδου του συστήματος* ***Y(s)*** *με την επιθυμητή τιμή εισόδου* ***R(s)***. Το αποτέλεσμα αυτής της σύγκρισης δημιουργεί τη μεταβλητή **E(s) – Error**, δηλαδή το σφάλμα του συστήματος, το οποίο το δέχεται σαν δεδομένο ο ελεγκτής και προσαρμόζει τις τιμές του με στόχο τον μηδενισμό του. Αξίζει να σημειωθεί, ότι η μέθοδος ελέγχου PID είναι ένα κλειστού βρόχου σύστημα το οποίο τροφοδοτεί το συγκριτή με τη τιμή Y(s) μέσω της ανάδρασης **(feedback)** που υπάρχει.



Εικόνα 15: Σύστημα με Ελεγκτή PID

Στην *Εικόνα 15* απεικονίζεται ένα απλό σύστημα με έλεγχο μέσω του PID, το οποίο δέχεται σαν ανάδραση την τιμή ενός αισθητήρα. Για παράδειγμα, ένα τέτοιο σύστημα (δεδομένης της παρούσας πτυχιακής εργασίας) μπορεί να είναι ο έλεγχος διατήρησης ύψους ενός drone. Ο χρήστης ορίζει στο ξεκίνημα της πτήσης το επιθυμητό ύψος R(s) και μέσω του υπερηχητικού αισθητηρίου **(ultrasonic)**, αν βρίσκεται σε εσωτερικό χώρο, ή του αλτίμετρου, αν βρίσκεται σε εξωτερικό χώρο, δέχεται το σφάλμα. Έτσι, μέσω των μεθόδων βέλτιστης ρύθμισης του PID (Ziegler – Nichols, Cohen – Coon) το σφάλμα απλοποιείται σε σημαντικό βαθμό και πετυχαίνετε η διατήρηση ύψους του drone.

Παρακάτω απεικονίζονται οι εξισώσεις στο πεδίο του χρόνου **(2.2)** και στο πεδίο συχνότητας **(2.3)** οι οποίες περιγράφουν τη λειτουργία του PID Controller:

$$\mathbf{u(t) = K_P\, e(t) + K_I \int e(\tau)d\tau + K_D\, de(t)/dt} \qquad \mathbf{(2.2)}$$

$$\mathbf{C(s) = K_P + K_I/s + K_D\, s} \qquad \mathbf{(2.3)}$$

Όπου:

- K_P → Αναλογικό Κέρδος (Proportional Gain)
- K_I → Ολοκληρωτικό Κέρδος (Integral Gain)
- K_D → Διαφορικό Κέρδος (Differential Gain)

Η εξίσωση στο πεδίο συχνότητας **(2.2)** αποτελεί και τη συνάρτηση μεταφοράς του συστήματος, η οποία χρησιμοποιείται κατά κόρον όταν ρυθμίζεται ο ελεγκτής PID μέσω των μεθόδων που προαναφέρθηκαν.
Γνωρίζοντας τη θεωρητική προσέγγιση ενός ελεγκτή PID, προκύπτει ένας εμπειρικός κανόνας ο οποίος διευκρινίζει ότι:

- Το $\mathbf{K_P}$ (Proportional Gain) επηρεάζεται άμεσα από το σφάλμα κατά τη διάρκεια της εκτέλεσης.
- Το $\mathbf{K_I}$ (Integral Gain) επηρεάζεται από το σύνολο των προηγούμενων σφαλμάτων.
- Το $\mathbf{K_D}$ (Differential Gain) προβλέπει τα σφάλματα που μπορεί να προκύψουν.

Συνήθως, μία καλή βαθμονόμηση των παραμέτρων ενός ελεγκτή PID θέτει το κέρδος K_I μικρό, διότι είναι ο πιο ευαίσθητος παράγοντας για την ευστάθεια του συστήματος. Βέβαια, δεν μπορεί να πάρει τη τιμή μηδέν διότι το σύστημα δεν θα μπορεί να φτάσει την επιθυμητή

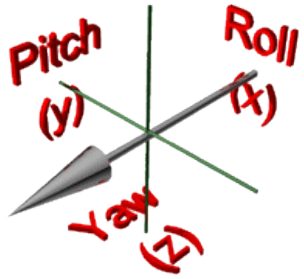
απόκριση. Επίσης, το κέρδος K_D είναι επιρρεπές σε θόρυβο και γι' αυτό σε πολλά απλά συστήματα η τιμή του τίθεται μηδέν και ο ελεγκτής μετατρέπεται σε PD. Για την περαιτέρω κατανόηση των παραμέτρων PID, ακολουθεί ο *Πίνακας 1* ο οποίος περιγράφει την επίδραση κάθε κέρδους σε ένα σύστημα με έλεγχο PID.

Κέρδος Ελεγκτή	**Χρόνος Ανύψωσης (Rise Time)**	**Υπερύψωση (Overshoot)**	**Χρόνος Αποκατάστασης (Settling Time)**	**Μόνιμο Σφάλμα (Error)**
Κέρδος – P	Μείωση	Αύξηση	Μικρή Αλλαγή	Μείωση
Κέρδος – I	Μείωση	Αύξηση	Αύξηση	Εξάλειψη
Κέρδος – D	Μικρή Αλλαγή	Μείωση	Μείωση	Μικρή Αλλαγή

Πίνακας 1: Εμπειρική επίδραση των κερδών ενός PID σε ένα σύστημα

Στην παρούσα πρακτική εφαρμογή της αυτόνομης πλοήγησης ενός Parrot AR.Drone 2.0. [7], οι ελεγκτές PID είχαν αρκετά σημαντικό ρόλο. Είναι γνωστό ότι σε ένα πολυκόπτερο αεροσκάφος οι διαστάσεις που πρέπει να ελεγχθούν είναι τέσσερις:

- **Roll (x) – Περιστροφή κατά μήκος του άξονα**
- **Pitch (y) – Περιστροφή γύρω από τον πλάγιο άξονα**
- **Yaw (yaw) – Περιστροφή γύρω από τον κάθετο άξονα**
- **Altitude (z) – Ύψος**



Εικόνα 16: Γραφική Απεικόνιση των Roll, Pitch, Yaw

Έτσι, για να υπάρχει βέλτιστος έλεγχος σε όλες τις διαστάσεις, στο σύστημα του drone εφαρμόζεται ένας PID ελεγκτής για κάθε μία από αυτές, με αποτέλεσμα η τελική ανάδραση των εξόδων του αεροσκάφους να φιλτράρεται μέσα από τέσσερις διαφορετικούς ελεγκτές.

Πιο συγκεκριμένα, ο ελεγκτής του ύψους $(\boldsymbol{PID_Z})$, ο οποίος είναι υπεύθυνος στο να εξομαλύνει τις διαφοροποιήσεις της κάθετης τοποθέτησης του drone, δέχεται σαν είσοδο τα σφάλματα των μετρήσεων μετατόπισης $\boldsymbol{e_Z}$, αλλά και την κάθετη γραμμική ταχύτητα $\boldsymbol{U_Z}$. Έτσι, οι συναρτήσεις που προκύπτουν [15] για τον $\boldsymbol{PID_Z}$ είναι:

$$\boldsymbol{e_Z = Z_{ΕΠΙΘΥΜΗΤΟ} - Z} \qquad \textbf{(2.4)}$$

$$\boldsymbol{U_Z = k_P * e_Z + k_I * e_Z - k_D * \dot{Z}} \qquad \textbf{(2.5)}$$

Η συνάρτηση μεταφοράς του συγκεκριμένου ελεγκτή προκύπτει να μοιάζει στην γενική:

$$\boldsymbol{PID_{(S)} = k_P + \frac{k_I}{s} + k_D s} \qquad \textbf{(2.6)}$$

Τα κέρδη της κάθε παραμέτρου $(\boldsymbol{k_P, k_I, k_D})$ ρυθμίζονται με real-time διαδικασία, μέσω του συστήματος δυναμικής διαμόρφωσης **(Dynamic Reconfigure)**. Το πακέτο *tum_ardrone* [4] που αναφέρθηκε σε προηγούμενη ενότητα της εργασίας, παρέχει real-time αλληλεπίδραση με το συγκεκριμένο εργαλείο έτσι ώστε η ρύθμιση του συστήματος ελέγχου του drone να μπορεί να προσαρμόζεται στις απαιτήσεις του χρήστη. Στην *Εικόνα 17* φαίνεται το εργαλείο ***rqt_reconfigure***, το οποίο έχει φορτωμένες τις βέλτιστες τιμές των ελεγκτών PID μέσω επικοινωνίας με τα nodes του *tum_ardrone* [16].

Εικόνα 17: Τιμές παραμέτρων των PIDs σύμφωνα με τις εκτιμήσεις του tum_ardrone.

Όπως είναι εμφανές από την *Εικόνα 17*, τα κέρδη των ελεγκτών είναι χωρισμένα σε τρείς βασικές κατηγορίες, τις **yaw, gaz** και **rp**.

- Η πρώτη ομάδα τιμών αφορά τον ελεγκτή για το **yaw** του αεροσκάφους, όπου τα $\boldsymbol{k_{P\_yaw}, k_{I\_yaw}, k_{D\_yaw}}$ είναι και τα κέρδη του $\boldsymbol{PID_{yaw}}$. Η συνάρτηση μεταφοράς του ελεγκτή αυτού προκύπτει από την γενική συνάρτηση με αντικατάσταση των τιμών:

  $$\boldsymbol{PID_{yaw(S)} = 0.5} \qquad \textbf{(2.7)}$$

  Έτσι, η συνάρτηση μεταφοράς έχει απλή μορφή, λόγω του ότι η οριζόντια περιστροφή του drone δεν χρειάζεται κάποια σύνθετη ρύθμιση για να εκτελεστεί σωστά.

- Η επόμενη ομάδα ονομάζεται **gaz** και είναι οι παράμετροι που είναι υπεύθυνοι για τον ελεγκτή $\boldsymbol{PID_Z}$, ο οποίος ρυθμίζει την κάθετη ταχύτητα του drone και άρα το ύψος του. Η συνάρτηση μεταφοράς στην προκειμένη περίπτωση παίρνει την εξής μορφή:

$$\boldsymbol{PID_{z(S)} = 0,6 + \frac{0,001}{s} + 0,1s} \qquad \textbf{(2.8)}$$

  Εκτός βέβαια από τους παραμέτρους των PIDs ($\boldsymbol{k_{P\_gaz}, k_{I\_gaz}, k_{D\_gaz}}$), υπάρχουν και τιμές που ρυθμίζουν άλλες λειτουργίες, όπως για παράδειγμα το "**max_gaz_rise**" το οποίο αλλάζει την τιμή εισόδου μίας συνάρτησης για το πόσο γρήγορα μπορεί το drone να πάρει ύψος.
- Η τελευταία ομάδα ονομάζεται **rp** από το **roll**, **pitch** και είναι αρμόδια για την οριζόντια κίνηση του UAV. Τα κέρδη για τους ελεγκτές $\boldsymbol{PID_{roll}}$ και $\boldsymbol{PID_{pitch}}$ δέχονται τις ίδιες τιμές σαν εισόδους στο σύστημα ($\boldsymbol{k_{P\_rp}, k_{I\_rp}, k_{D\_rp}}$), με την παράμετρο "**aggressiveness**" (επιθετικότητα) να διαμορφώνει αναλογικά τις κινήσεις του αεροσκάφους σε επιθετικές ή ομαλές. Έτσι, οι συναρτήσεις μεταφοράς για την κάθε κίνηση ξεχωριστά είναι οι εξής:

$$\boldsymbol{PID_{roll(S)} = 0,6 + 0,35s} \qquad \textbf{(2.9)}$$

$$\boldsymbol{PID_{pitch(S)} = 0,5 + 0,35s} \qquad \textbf{(2.10)}$$

Οι παράμετροι αυτοί μαζί με τις τιμές των κερδών των PIDs εκπέμπονται με συχνότητα 100Hz προς το drone. Αξίζει να σημειωθεί σε αυτό το σημείο, ότι κύριο ρόλο στην ευστάθεια του συστήματος έχει η υλοποίηση και ενσωμάτωση του **Extended Kalman Filter (EKF)** [17], όπως και άλλων αντίστοιχων αλγορίθμων, τα οποία και θα επεξηγηθούν στις επόμενες ενότητες αυτού του κεφαλαίου. Τέλος, η ρύθμιση των ελεγκτών θα αναλυθεί περαιτέρω στο Κεφάλαιο 3: Πρακτική Εφαρμογή, όπου θα γίνει και παράθεση πηγαίου κώδικα για επιπλέον επεξήγηση.

2.4.2. Φίλτρο Bayes – Bayes Filter

Το **Bayes Filter**, το οποίο είναι γνωστό και ως **Recursive Bayesian Estimation** (Αναδρομική Εκτίμηση Bayes), είναι μία μέθοδος εκτίμησης καταστάσεων, η οποία υπολογίζει την συνάρτηση πυκνότητας πιθανοτήτων **(Probability Density Function – PDF)** σε μια χρονική περίοδο παίρνοντας ως δεδομένα μετρήσεις από το σύστημα και επεξεργάζοντας τα μέσω μαθηματικών μοντέλων. Πιο συγκεκριμένα, μία συνάρτηση PDF αναλύει τις πιθανότητες που υπάρχουν για μία τυχαία μεταβλητή στο να δεχθεί ως όρισμα μία δεδομένη τιμή. Έτσι, ο αλγόριθμος αυτός χρησιμοποιείται με σκοπό να υπολογίζει πολλαπλές πιθανότητες **(state estimation)** σε διάφορα πιθανά σύνολα δεδομένων κατάστασης **(beliefs)**, με αποτέλεσμα την πρόβλεψη κινήσεων ενός drone και ακόμη περισσότερο τον συνεχή εντοπισμό της τοποθεσίας του.

Εκτίμηση Κατάστασης – State Estimation

Για να γίνει πιο κατανοητός ο αλγόριθμος Bayes, είναι σημαντικό να αναλυθεί το κομμάτι της θεωρίας πιθανοτήτων και εκτίμησης της επόμενης κατάστασης ενός μη γραμμικού συστήματος, δηλαδή ενός ρομπότ ή UAV.

Σε ένα προηγμένο αυτόνομο σύστημα, όπως είναι ένα ρομπότ, για να περιγραφεί η κατάσταση του χρησιμοποιείται η λέξη **state**, το οποίο ορίζεται στις συναρτήσεις ως $\mathbf{x_t}$. Η μεταβλητή αυτή μπορεί να έχει "στατική" **(static)** ιδιότητα, δηλαδή να μην αλλάζει κατάσταση, ή "δυναμική" **(dynamic)**. Σε αυτή τη μεταβλητή συνήθως εμπεριέχεται η θέση **(pose)** του ρομπότ μέσα στο χώρο, η ταχύτητα του **(velocity)** όπως και η επιτάχυνση του **(acceleration)** ανα χρονική στιγμή. Επίσης, λόγω του ότι τα δεδομένα από τους αισθητήρες του ρομπότ μπορεί να εμπεριέχουν θόρυβο **(noisy data)** υπολογίζεται μια κατανομή πιθανότητων με βάση τα states έτσι ώστε η εκτίμηση των καταστάσεων, δηλαδή το state estimation, να είναι πιο ακριβές. Το state estimation αυτών των δεδομένων ονομάζεται **"belief"** και ουσιαστικά είναι η εκτίμηση του ρομπότ όσον αφορά το περιβάλλον του και την κατάσταση του.

Ένα ρομπότ ονομάζεται δυναμικό σύστημα λόγω του ότι μπορεί να επηρεάσει άμεσα τα states μέσα από εντολές ελέγχου **(control actions)**, αλλά και να λάβει δεδομένα από τους αισθητήρες του **(measurements data)** τα οποία είναι παράμετροι για τις συναρτήσεις ελέγχου του [18]. Στους συγκεκριμένους state estimation αλγόριθμους, τα *control* και *measurement data* ορίζονται όπως φαίνεται παρακάτω.

Τα δεδομένα μετρήσεων **(measurement data)** περιγράφονται ως εξής:

$$\boldsymbol{Z_{t_1:t_2} = Z_{t_1}, Z_{t_1+1}, Z_{t_1+2}, \dots, Z_{t_2}} \qquad \textbf{(2.11)}$$

Όπου: $\boldsymbol{Z_{t_1:t_2}}$ είναι τα **measurement data** που λήφθηκαν από τη χρονική στιγμή $\mathbf{t_1}$ έως τη χρονική στιγμή $\mathbf{t_2}$, με $\mathbf{t_1 \leq t_2}$.

Επίσης, παρόμοια εξίσωση γενικού τύπου ισχύει και για τα δεδομένα ελέγχου **(control data)**:

$$\boldsymbol{U_{t_1:t_2} = U_{t_1}, U_{t_1+1}, U_{t_1+2}, \dots, U_{t_2}} \qquad \textbf{(2.12)}$$

Όπου: $\boldsymbol{U_{t_1:t_2}}$ είναι τα **control data** που λήφθηκαν από τη χρονική στιγμή $\mathbf{t_1}$ έως τη χρονική στιγμή $\mathbf{t_2}$, με $\mathbf{t_1 \leq t_2}$.

Πιο συγκεκριμένα, τα **measurement data** παρέχουν πληροφορίες για το περιβάλλον του ρομπότ και την κατάσταση του μέσα από τις μετρήσεις των αισθητήρων του, ενώ τα **control data** υποδηλώνουν την αλλαγή κατάστασης του ρομπότ όσον αφορά της θέση του. Για παράδειγμα, αν σταλεί στο drone ένα control action το οποίο να του λέει να προχωρήσει *2 m/sec* μπροστά, τότε οι πιθανές θέσεις του μετά από 2 δευτερόλεπτα (εκτός από αυτή στα 4 μέτρα) θα είναι *3.9* και *4.1* μέτρα, λόγω θορύβου που υπάρχει γενικά μέσα στο σύστημα. Έτσι, για να γίνεται σωστά το state estimation των κινήσεων ενός αυτόνομου συστήματος και να αποφεύγεται ο θόρυβος των μετρήσεων, πρέπει η πορεία των states να γίνεται μέσω πιθανοθεωρητικών νόμων **(probabilistic laws)**. Ένας από τους κυριότερους νόμους πιθανοτήτων είναι η *Υπόθεση Markov* **(Markov Assumption)** η οποία ορίζει ότι: *"αν μία κατάσταση* $\boldsymbol{x}$ *είναι ολοκληρωμένη και σωστή, τότε η ίδια είναι και μία κατάλληλη ένδειξη για τις παρελθοντικές και μελλοντικές καταστάσεις του ίδιου του ρομπότ*"**.** Όμως, τα δεδομένα για κάθε state επηρεάζονται άμεσα μόνο από την προηγούμενη κατάσταση, καθιστώντας έτσι τα δεδομένα του παρελθόντος και του μέλλοντος τελείως ανεξάρτητα μεταξύ τους. Αξίζει να σημειωθεί ότι αυτά τα δεδομένα καταστάσεων χωρίζονται σε τρείς βασικές κατηγορίες κατάστασης:

- **Λείανση (Smoothing)**: Παρελθοντική εκτίμηση κατάστασης
- **Φιλτράρισμα (Filtering)**: Παροντική εκτίμηση κατάστασης
- **Πρόβλεψη (Prediction)**: Μελλοντική εκτίμηση κατάστασης

Η γενική εξίσωση για το *Markov Assumption* είναι η εξής:

$$\boldsymbol{p(x_t|x_{0:t-1}, Z_{1:t-1}, U_{1:t}) = p(x_t|x_{t-1}, U_t)} \qquad \textbf{(2.13)}$$

Όπου: το $\boldsymbol{p(x_t|x_{t-1}, U_t)}$ ονομάζεται "πιθανότητα μεταβατικής κατάστασης" **(state transition probability)** και υποδηλώνει την εξέλιξη της κατάστασης περιβάλλοντος **(environmental state)** του drone ανά χρονική στιγμή με βάση τα control data $\boldsymbol{U_t}$.

Σημαντικό επίσης είναι να οριστεί η εξίσωση για την εκτίμηση των δεδομένων, ή αλλιώς **belief**, όπως και η **prediction belief** συνάρτηση.

Έτσι η εκτίμηση με βάση τις μεταβλητές κατάστασης είναι:

$$\boldsymbol{bel(x_t) = p(x_t|Z_{1:t}, U_{1:t})} \qquad \textbf{(2.14)}$$

Ενώ, η πρόβλεψη της εκτίμησης για τη χρονική στιγμή **t** μπορεί να γίνει χωρίς τα δεδομένα μετρήσεων του παρόντος ως εξής:

$$\boldsymbol{\overline{bel}(x_t) = p(x_t|Z_{1:t-1}, U_{1:t})} \qquad \textbf{(2.15)}$$

Ως αποτέλεσμα, η συνάρτηση αυτή προβλέπει την επόμενη κατάσταση βάσει υπολογισμών με δεδομένα μετρήσεων από την προηγούμενη κατάσταση.

Αλγόριθμος Bayes – Bayes Filter

Ο αλγόριθμος υπολογισμού των beliefs ενός συστήματος, δέχεται ως εισόδους δεδομένων την αρχική εντύπωση **(initial belief)** *bel* σε προηγούμενο χρόνο *t-1*, τα measurement data $\boldsymbol{Z_t}$ και το τελευταίο control data $\boldsymbol{U_t}$. Σύμφωνα με αυτά τα δεδομένα, το Bayes Filter χωρίζεται σε δύο βήματα:

a) **Βήμα Πρόβλεψης – Prediction Step**
b) **Βήμα Ενημέρωσης – Update Step**

Στο πρώτο βήμα **(prediction step)** γίνεται ο υπολογισμός μίας νέας εκτίμησης **(belief)** για την επόμενη χρονική στιγμή του ρομπότ με βάση τα παραπάνω δεδομένα και των εξής συναρτήσεων **(motion model → 2.17)**:

$$\boldsymbol{p(x_{t-1}|Z_{1:t-1}) \rightarrow\ p(x_t|Z_{1:t-1})} \qquad \textbf{(2.16)}$$

$$\boldsymbol{\overline{bel}(x_t) = \int p(x_t|U_t\,, x_{t-1})\, bel(x_{t-1}) dx_{t-1}} \qquad \textbf{(2.17)}$$

Έπειτα, το state estimation σύστημα περνάει στο επόμενο βήμα, αυτό του **update step**. Εκεί με τη χρήση των παρακάτω συναρτήσεων **(sensor model → 2.19)**, γίνεται ενημέρωση **(update)** του αρχικού belief χρησιμοποιώντας τα νέα δεδομένα που δέχεται εκείνη τη χρονική στιγμή το ρομπότ από τους αισθητήρες του.

$$\boldsymbol{p(x_t|Z_{1:t-1}) \rightarrow\ p(x_t|Z_{1:t})} \qquad \textbf{(2.18)}$$

$$\boldsymbol{bel(x_t) = \eta\, p(Z_t|x_t)\overline{bel}(x_t)} \qquad \textbf{(2.19)}$$

Όπου: **η** είναι μία μεταβλητή ομαλοποίησης για να την διαβεβαίωση ότι το belief είναι σωστό.

Τέλος, αυτά τα δύο βήματα εφαρμόζονται διαρκώς κατά τη διάρκεια εκτέλεσης του συστήματος, έτσι ώστε το ρομπότ να έχει πάντα ένα belief του που βρίσκεται (correct state estimation) και άρα να εκτελείται και πιο σωστά το navigation του.

Υποσημείωση: Το ρομπότ δόθηκε σαν ένα πιο γενικό παράδειγμα το οποίο όμως αντιπροσωπεύει πλήρως και την πλοήγηση ενός μη επανδρωμένου αεροσκάφους.

2.4.3. Φίλτρο Kalman – Kalman Filter

Ένας από τους βασικότερους αλγόριθμους όσον αφορά την πρόβλεψη καταστάσεων σε ένα σύστημα, είναι το **Kalman Filter** [14]. Αναλυτικότερα, το Kalman Filter υπολογίζει την κατάσταση **(state)** μίας ελεγχόμενης διαδικασίας διακριτού χρόνου, η οποία διέπεται από την παρακάτω γραμμική διαφορική εξίσωση **(2.20)** και τις γραμμικές μετρήσεις κατάστασης από τους αισθητήρες **(2.21)**:

$$\boldsymbol{x_t = Ax_{t-1} + BU_t + \varepsilon_t} \qquad \textbf{(2.20)}$$

$$\boldsymbol{Z_t = (x_t + \delta_t)}\ , \mu\varepsilon\ \boldsymbol{\delta_t \sim N(0, R)}\ \text{και}\ \boldsymbol{\varepsilon_t \sim N(0, Q)} \qquad \textbf{(2.21)}$$

Θέτοντας το διαφορετικά, το **Kalman Filter** είναι μία μέθοδος η οποία υλοποιεί τη θεωρία εκτίμησης καταστάσεων του Bayes στην πράξη. Χρησιμοποιείται περισσότερο από οποιαδήποτε άλλη, λόγω της ευκολίας που προσφέρει στην υλοποίηση, και εφαρμόζεται σε διάφορους επιστημονικούς κλάδους. Στον χώρο της αυτόνομης πλοήγησης, το Kalman Filter βρίσκει εφαρμογή όσον αφορά την βελτίωση πλοήγησης του drone σε ένα περιβάλλον, έχοντας ζωτικό ρόλο στην δομή της λειτουργίας και της πτήσης του αεροσκάφους.

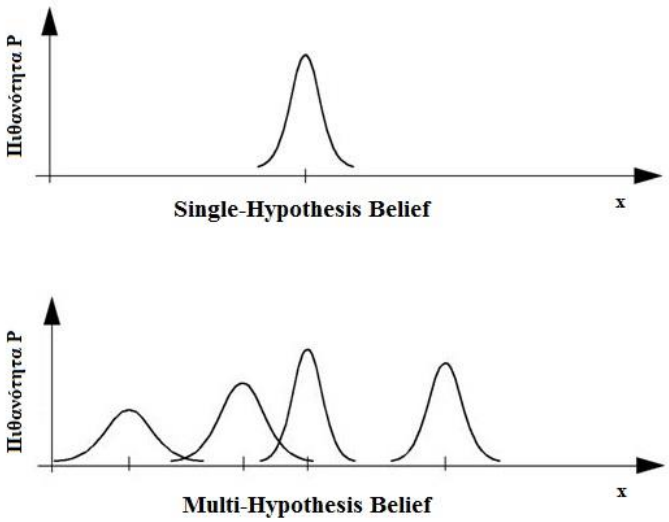
Πιο συγκεκριμένα, το Kalman Filter έχει ως σκοπό την απαλοιφή μετρήσεων που χαρακτηρίζονται από φυσικές διαταραχές, όπως για παράδειγμα αναταράξεις, έτσι ώστε η νέα κατάσταση που θα ληφθεί από το σύστημα ελέγχου να μην περιέχει θορύβους. Ακόμα, το φίλτρο αυτό έχει αρκετά κοινά στοιχεία με το Bayes Filter (κάτι το οποίο είναι λογικό αφού βασίζεται πάνω στη θεωρία του Bayes) όσον αφορά την εκτίμηση παρελθοντικών **(Smoothing)** και μελλοντικών καταστάσεων **(Prediction)**, αλλά και την υποστήριξη αυτών όταν το σύστημα δεν είναι γνωστό. Βασική προϋπόθεση για τη σωστή εφαρμογή του αλγόριθμου, είναι η γραμμικότητα του συστήματος **(Linear System)** όπως και η επιρροή του θορύβου Gauss **(Gaussian noise)** στο μοντέλο κίνησης και μοντέλο μετρήσεων. Ο θόρυβος Gauss ή αλλιώς στατιστικός θόρυβος, υπάρχει όταν η συνάρτηση PDF ισούται ή ακολουθεί την κανονική κατανομή **(κατανομή Gauss)**.

Ένα από τα κυριότερα προβλήματα αυτόνομης πλοήγησης που ονομάζεται **SLAM**, ακολουθεί την τεχνική του φίλτρου Kalman και των αντίστοιχών του (Bayes Filter, Particle Filter, EKF). Το SLAM θα αναλυθεί στην ενότητα (*2.5)*, λόγω της σημαντικότητας του όσον αφορά την αυτόνομη πλοήγηση ενός drone· παρ' όλα αυτά είναι σημαντικό να γίνουν κατανοητές βασικές έννοιες, οι οποίες αναφέρονται τόσο στους αλγόριθμους αυτούς όσο και στο SLAM, αλλά και η λειτουργία τέτοιων φίλτρων.

Για την επίλυση προβλημάτων με τη χρήση πεποιθήσεων **(belief)**, χρησιμοποιούνται δύο βασικές κατηγορίες:

- Απλής Υπόθεσης Πεποίθηση – **Single-Hypothesis Belief**
- Πολλαπλής Υπόθεσης Πεποίθηση – **Multi-Hypothesis Belief**

Το φίλτρο **Kalman** αποτελεί εφαρμογή μίας λύσης **Single-Hypothesis belief**, η οποία βασίζεται σε μία απλή γενική εκτίμηση **(single world state)** και επηρεάζεται άμεσα από τη διακύμανση **(variance)** αυτής· επιτρέποντας έτσι στο σύστημα να διευρύνει ή να περιορίζει την εκτιμώμενη περιοχή στο χώρο κατάστασης. Το κύριο πλεονέκτημα της Single-Hypothesis είναι ότι λόγω της μονής διακύμανσης το ρομπότ μπορεί να σχηματίσει εύκολα απόφαση για την θέση του, βέβαια με τη δυσκολία μοντελοποίησης των ασαφειών. Η διαφορά με τις **Multi-Hypothesis** λύσεις είναι ότι βασίζεται σε πολλαπλές και αναρίθμητες εκτιμήσεις **(states),** δίνοντας έτσι στο ρομπότ τη δυνατότητα επιλογής της πιο κατάλληλης θέσης μέσα από πολλαπλές επιλογές αλλά και τη δυνατότητα απόρριψης πολλών υποθέσεων.



Εικόνα 18: Απλή & Πολλαπλή Υπόθεση Πεποιθήσεων

Ως αποτέλεσμα, το Kalman Filter χρησιμοποιεί Single-hypothesis belief αναπαράσταση έτσι ώστε να μπορεί να την εφαρμόζει σε real time και χωρίς πολλαπλή επεξεργασία μεγάλου όγκου πληροφοριών.

Μαθηματική Εφαρμογή του Kalman Filter

Ο αλγόριθμος Kalman χωρίζεται σε δύο βασικά στάδια, παρόμοια με τον αλγόριθμο του Bayes. Αυτά τα στάδια ονομάζονται **Πρόβλεψης (Prediction)** και **Ενημέρωσης (Update)**. Η πεποίθηση **(belief)** σε χρόνο **t** περιγράφεται μέσα από τις εξισώσεις **μέσης τιμής** $\boldsymbol{\mu_t}$ και **συνδιακύμανσης** $\boldsymbol{\Sigma_t}$, όπως αυτές φαίνονται παρακάτω, με τις εισόδους **(inputs)** στο σύστημα να είναι οι $\boldsymbol{\mu_{t-1}}$, $\boldsymbol{\Sigma_{t-1}}$, $\boldsymbol{U_t}$ και $\boldsymbol{Z_t}$. Σε αυτό το σημείο είναι σημαντικό να διευκρινιστεί ότι το $\boldsymbol{x_t}$ που περιγράφηκε στον αλγόριθμο του Bayes από τις συναρτήσεις **(2.17)** και **(2.19)**, έχει αντικατασταθεί από τις συναρτήσεις $\boldsymbol{\mu_t}$ και $\boldsymbol{\Sigma_t}$.

- Έτσι, οι συναρτήσεις που εκφράζουν το βήμα πρόβλεψης **(prediction step)** είναι οι εξής:

Εξίσωση **Κατάστασης Μέσης Τιμής**: $\overline{\boldsymbol{\mu_t}} = \boldsymbol{A_t \mu_{t-1} + B_t U_t}$ **(2.22)**

Εξίσωση **Συνδιακύμανσης**: $\overline{\boldsymbol{\Sigma_t}} = \boldsymbol{A_t \Sigma_{t-1} A_t^T + R_t}$ **(2.23)**

Όπου:

- **A** → Συνάρτηση η οποία εκφράζει τη μεταβατική κατάσταση του συστήματος
- **B** → Το μοντέλο κίνησης **(Motion model)**
- **C** → Το μοντέλο των αισθητήρων **(Sensor model)**

Και οι τρείς παραπάνω παράμετροι αποτελούν πίνακες **(matrix operations)**

- Ενώ στο βήμα ενημέρωσης **(update step)** οι εξίσωσης του συστήματος είναι:

Κέρδος Kalman: $\boldsymbol{K_t = \overline{\Sigma_t} C_t^T (C_t \overline{\Sigma_t} C_t^T + Q_t)^{-1}}$ **(2.24)**

Εξίσωση **Κατάστασης (updated)**: $\boldsymbol{\mu_t = \overline{\mu_t} + K_t (Z_t - C_t \overline{\mu_t})}$ **(2.25)**

Εξίσωση **Συνδιακύμανσης (updated)**: $\boldsymbol{\Sigma_t = (I - K_t C_t) \overline{\Sigma_t}}$ **(2.26)**

Όπου:

- **K** → Το κέρδος του αλγόριθμου Kalman
- **Q** → Το μοντέλο θορύβου του Gauss (Gaussian noise)
- $\mathbf{Z_t}$ → Το μοντέλο δεδομένων μετρήσεων (measurement data)

Αναλυτικότερα, ο αλγόριθμος Kalman εκφράζεται μέσω της πεποίθησης $\boldsymbol{bel(x_t)}$, η οποία στην προκειμένη περίπτωση έχει αντικατασταθεί από τις εξισώσεις $\boldsymbol{\mu_t}$ και $\boldsymbol{\Sigma_t}$. Η βασική λογική πίσω από τον αλγόριθμο Kalman είναι στο **βήμα πρόβλεψης** να λαμβάνει το σύστημα σήμα μετρήσεων από τους αισθητήρες και από το μοντέλο κίνησης, και αυτό βάσει των προηγούμενων δεδομένων ελέγχου, της προηγούμενης πεποίθησης $\boldsymbol{bel(x_{t-1})}$ και του μοντέλου κίνησης και μετάβασης κατάστασης, να εκτιμά το νέο belief $\boldsymbol{bel(x_t)}$· το οποίο αντιστοιχεί στη μέση πιθανότητα $\boldsymbol{\overline{\mu_t}}$ και συνδιακύμανσης $\boldsymbol{\overline{\Sigma_t}}$. Στη συνέχεια, στο **βήμα ενημέρωσης**, οι μετρήσεις που λαμβάνονται από την αρχή ενσωματώνονται στην γενική εξίσωση του update step έτσι ώστε η αρχική πεποίθηση $(\boldsymbol{\overline{\mu_t}, \overline{\Sigma_t}})$ να μετατραπεί σε επιθυμητή πεποίθηση $(\boldsymbol{\mu_t, \Sigma_t})$. Στο βήμα αυτό, σημαντικό ρόλο στην μετατροπή της πεποίθησης έχουν η παράμετρος $\boldsymbol{K_t}$ **(κέρδος Kalman)**, η οποία καθορίζει τον βαθμό επιρροής του $\boldsymbol{Z_t}$ στην εκτίμηση της νέας κατάστασης, αλλά και το πηλίκο $\boldsymbol{C_t\overline{\mu_t}}$ το οποίο αντικαθιστά το $\boldsymbol{Z_t}$ λόγω αναξιοπιστίας.

Από το Bayes Filter στο Kalman Filter

Λόγω της άμεσης συσχέτισης των δύο αυτών αλγορίθμων αλλά και της επιρροής του φίλτρου Kalman από το φίλτρο Bayes, είναι εύκολο μέσα από απλές αντικαταστάσεις να δημιουργηθούν οι εξισώσεις του αλγόριθμου Kalman βάσει του Bayes.

- Εφαρμογή των τύπων στο μοντέλο κίνησης της εξίσωσης **(2.17)**:

$$\boldsymbol{\overline{bel}(x_t) = \int p(x_t|U_t, x_{t-1})\, bel(x_{t-1}) dx_{t-1}}$$

Αντικατάσταση:

$$\boldsymbol{p(x_t|U_t, x_{t-1}) = N(x_t;\, Ax_{t-1} + BU_t, Q_t)} \qquad \textbf{(2.27)}$$

$$\boldsymbol{bel(x_{t-1}) = N(x_{t-1};\, \mu_{t-1}, \Sigma_{t-1})} \qquad \textbf{(2.28)}$$

Οπότε η εξίσωση **(2.17)** μετατρέπεται μέσω των **(2.27)** και **(2.28)** στην εξής:

$$\boldsymbol{\overline{bel}(x_t) = N(x_t;\, A\mu_{t-1} + BU_t, A\Sigma A^T + Q_t)}$$

$$\boldsymbol{\rightarrow\ \overline{bel}(x_t) = N(x_t;\, \overline{\mu_t}, \overline{\Sigma_t})} \qquad \textbf{(2.29)}$$

- Εφαρμογή των τύπων στο μοντέλο αισθητήρων της εξίσωσης **(2.19)**:

$$\boldsymbol{bel}(\boldsymbol{x_t}) = \boldsymbol{\eta}\, \boldsymbol{p}(\boldsymbol{Z_t}|\boldsymbol{x_t})\overline{\boldsymbol{bel}}(\boldsymbol{x_t})$$

Αντικατάσταση:

$$\boldsymbol{p}(\boldsymbol{Z_t}|\boldsymbol{x_t}) = \boldsymbol{N}(\boldsymbol{Z_t};\ \boldsymbol{C}\boldsymbol{x_t}, \boldsymbol{Q_t}) \qquad \textbf{(2.30)}$$

$$\overline{\boldsymbol{bel}}(\boldsymbol{x_t}) = \boldsymbol{N}(\boldsymbol{x_t};\ \overline{\boldsymbol{\mu_t}}, \overline{\boldsymbol{\Sigma_t}}) \qquad \textbf{(2.29)}$$

Οπότε η εξίσωση **(2.19)** μετατρέπεται μέσω των **(2.29)** και **(2.30)** στην εξής:

$$\boldsymbol{bel}(\boldsymbol{x_t}) = \boldsymbol{N}(\boldsymbol{x_t}; \overline{\boldsymbol{\mu_t}} + \boldsymbol{K_t}(\boldsymbol{Z_t} - \boldsymbol{C}\overline{\boldsymbol{\mu_t}}), (\boldsymbol{I} - \boldsymbol{K_t}\boldsymbol{C})\overline{\boldsymbol{\Sigma_t}})$$

$$\rightarrow\ \boldsymbol{bel}(\boldsymbol{x_t}) = (\boldsymbol{x_t};\ \boldsymbol{\mu_t}, \boldsymbol{\Sigma_t}) \qquad \textbf{(2.31)}$$

Όπου:

$\boldsymbol{K_t} = \overline{\boldsymbol{\Sigma_t}}\boldsymbol{C^T}(\boldsymbol{C}\overline{\boldsymbol{\Sigma_t}}\boldsymbol{C^T} + \boldsymbol{Q})^{-1}$ είναι το **κέρδος Kalman**

Extended Kalman Filter

Η εφαρμογή του φίλτρου Kalman, όπως προαναφέρθηκε, γίνεται μόνο σε συστήματα γραμμικά **(linear systems)** κάτι το οποίο περιορίζει τις επιλογές όταν ένα σύστημα είναι μη-γραμμικό **(non-linear system)**, όπως για παράδειγμα ένα σύστημα πλοήγησης ενός UAV. Έτσι, μία παραλλαγή του αλγόριθμου αυτού είναι το **Extended Kalman Filter – EKF** [17], το οποίο μετατρέπει τα μη-γραμμικά *motion* και *sensor* models σε γραμμικά. Στην περίπτωση αυτή, οι εξισώσεις μετάβασης κατάστασης και μετρήσεων έχουν την μορφή των *g* και *h* αντίστοιχα και αντικαθιστούνε τα ***A***, ***B*** και ***C*** του φίλτρου Kalman:

$$\boldsymbol{x_t} = (\boldsymbol{U_t}, \boldsymbol{x_{t-1}}) \qquad \textbf{(2.32)}$$

$$\boldsymbol{Z_t} = \boldsymbol{h}(\boldsymbol{x_t}) + \boldsymbol{\delta_t} \qquad \textbf{(2.33)}$$

Αντίστοιχα με το απλό Kalman filter, το EKF ακολουθεί τα **prediction** και **update** βήματα με βασικό στόχο την γραμμικοποίηση των συναρτήσεων.

- Επομένως, στο **prediction step** οι συναρτήσεις που χαρακτηρίζουν τον αλγόριθμο του **EKF** είναι οι εξής:

Εξίσωση **Κατάστασης Μέσης Τιμής**: $\overline{\boldsymbol{\mu_t}} = \boldsymbol{g(U_t, \mu_{t-1})}$ **(2.34)**

Εξίσωση **Συνδιακύμανσης**: $\overline{\boldsymbol{\Sigma_t}} = \boldsymbol{G_t \Sigma_{t-1} G_t^T + R_t}$ **(2.35)**

- Ενώ στο **update step** το κέρδος του φίλτρου Kalman όπως και οι συναρτήσεις **(2.34)** και **(2.35)** γίνονται:

Κέρδος Kalman: $\boldsymbol{K_t = \overline{\Sigma_t} H_t^T (H_t \overline{\Sigma_t} H_t^T + Q_t)^{-1}}$ **(2.36)**

Εξίσωση **Κατάστασης (updated)**: $\boldsymbol{\mu_t = \overline{\mu_t} + K_t(Z_t - h(\overline{\mu_t}))}$ **(2.37)**

Εξίσωση **Συνδιακύμανσης (updated)**: $\boldsymbol{\Sigma_t = (I - K_t H_t)\overline{\Sigma_t}}$ **(2.38)**

Όπου τα $\boldsymbol{G_t}$ και $\boldsymbol{H_t}$ αντικαθιστούν τις παραμέτρους/πίνακες $\boldsymbol{A_t}$ και $\boldsymbol{C_t}$ αντίστοιχα.

Τέλος, όπως προαναφέρθηκε, το **Extended Kalman Filter** γραμμικοποιεί τις μη-γραμμικές συναρτήσεις πιθανοτήτων έτσι ώστε να είναι εφικτό μετά να εφαρμοστεί στο σύστημα το κανονικό Kalman Filter. Στα αεροσκάφη αυτόνομης πλοήγησης, το φίλτρο αυτό χρησιμοποιείται κατά κόρον λόγω της σημαντικότητας του όσον αφορά την σταθερότητα και την τοποθέτηση του drone στο χώρο, αλλά και επειδή τα συστήματα τους είναι μη γραμμικά και άρα είναι αδύνατη η εφαρμογή ενός απλού φίλτρου Kalman.

2.5. Μέθοδοι και Αλγόριθμοι για Αυτόνομη Πλοήγηση

2.5.1. Simultaneous Localization and Mapping – SLAM

Στο χώρο της αυτόνομης ρομποτικής, ένα από τα συνηθέστερα και πιο σημαντικά προβλήματα, είναι αυτό του συνεχόμενου εντοπισμού θέσης και χαρτογράφησης άγνωστου περιβάλλοντος **(Simultaneous Localization and Mapping – SLAM)**. Με άλλα λόγια, η κύρια λειτουργία του SLAM είναι ο στιγμιαίος εντοπισμός νέων σημείων στο χώρο όπως και της θέσης του ρομπότ κατά τη διάρκεια της χαρτογράφησης. Όπως είναι εμφανές, είναι μία διαδικασία η οποία προϋποθέτει ποικίλες παραμέτρους αλλά και αυξημένη υπολογιστική ισχύει από το σύστημα. Ο όρος ***mapping*** αναφέρεται στη διαδικασία δημιουργίας ενός χάρτη από το ρομπότ, ο οποίος εμπεριέχει σημαντικά σημεία τα οποία μπορεί να αναγνωρίσει το ρομπότ μέσω των αισθητήρων του. Ακόμα, ο όρος ***localization*** αναφέρεται στο κομμάτι του συστήματος που χρησιμοποιεί τα σημεία του χάρτη, έτσι ώστε να υπολογίσει και να καθορίσει την τοποθεσία του ρομπότ στον χώρο. Το σημαντικότερο πρόβλημα που αντιμετωπίζεται σε αυτές τις διαδικασίες, είναι η ύπαρξη θορύβου στις μετρήσεις· κάτι το οποίο δυσκολεύει κατά πολύ το SLAM και συνεπώς, την πλοήγηση του ρομπότ.

Ιστορικά, η πρώτη αναφορά στο SLAM έγινε από τους (J. J. Leonard και H. F. Durrant-Whyte) [19] οι οποίοι συνδύασαν δύο ξεχωριστές διατριβές τους. Η πρώτη αναφερόταν στον αλγόριθμο που ήταν υπεύθυνος για τη λειτουργία του **mapping** και η δεύτερη ήταν για το **localization** ενός ρομπότ. Ως αποτέλεσμα, ανέπτυξαν έναν αλγόριθμο πλοήγησης με υποβρύχιο ραντάρ **(sonar-based navigation)** ο οποίος βασιζόταν σε ένα μοντέλο περιβάλλοντος σύμφωνα με το οποίο γινόταν εκτίμηση της τοποθεσίας του. Ο αλγόριθμος αυτός **(localization algorithm)** αναπτύχθηκε με βάση το extended Kalman Filter **(EKF)** [17], το οποίο μπορούσε να εκτιμήσει με μεγάλη ταχύτητα τις θέσεις του συστήματος μέσω διαφόρων δεδομένων από το sonar. Έπειτα από αυτή την έρευνα, το SLAM επεκτάθηκε από τον εσωτερικό χώρο ενός κτιρίου **(indoors)** στον έξω χώρο **(outdoors)** μέσω διαφόρων άλλων ερευνών, αλλά και σε πιο περίπλοκα τρισδιάστατα περιβάλλοντα π.χ. υποθαλάσσια ή εναέρια.

Τεχνικές SLAM – SLAM Techniques:

Οι κύριες τεχνικές για την εφαρμογή του SLAM σε ένα ρομποτικό αυτόνομο σύστημα, όπως είναι ένα UAV, είναι δύο: η προσέγγιση με μεθόδους πιθανοτήτων, όπως είναι οι βασικοί αλγόριθμοι πρόβλεψης που αναλύθηκαν στις ενότητες *2.4.2*, *2.4.3*, ή με τη μέθοδο σάρωσης και αντιστοίχισης του χώρου **(Scan Matching)**, η οποία αναλύει το χώρο μέσω των αισθητήρων και αντιστοιχίζει τα χαρακτηριστικά μέσα από ειδικές μεθόδους.

Στην πρώτη προσέγγιση, τα στάδια αλγορίθμων του SLAM, δηλαδή της *γραμμικοποίησης* ***(linearization)*** – αν είναι απαραίτητο – και της *πρόβλεψης* ***(prediction)*** και *ενημέρωσης* ***(update)*** του συστήματος, γίνονται με τη χρήση των φίλτρων **Bayes** (σε απλά συστήματα) και **Kalman Filter**, **EKF** (σε πιο περίπλοκα συστήματα).

Στη δεύτερη προσέγγιση, οι μέθοδοι αντιστοίχισης των χαρακτηριστικών είναι τέσσερις και διαφέρουν ως προς το είδος των χαρακτηριστικών, όπως φαίνεται και στον *Πίνακας 2* [20].

	Μέθοδος	**Συντομογραφία**
1	Χαρακτηριστικό σε Χαρακτηριστικό **(Feature to Feature)**	**F2F**
2	Σημείο σε Χαρακτηριστικό **(Point to Feature)**	**P2F**
3	Σημείο σε Σημείο **(Point to Point)**	**P2P**
4	Συνδυασμός όλων	**Com**

Πίνακας 2: Μέθοδοι Scan Matching

Αξίζει να σημειωθεί ότι, **features** ονομάζονται χαρακτηριστικά όπως γραμμές, γωνίες και ακμές **(PoI)**, κύκλοι και άλλα γεωμετρικά σχήματα. Η πρώτη μέθοδος, **F2F**, παρέχει το πλεονέκτημα στο σύστημα του ότι συρρικνώνει κατά πολύ τον όγκο των δεδομένων, ή δεν τον χρησιμοποιεί όλο, και άρα μειώνει και τη διάρκεια εκτέλεσης και δημιουργίας ενός χάρτη, το οποίο κατασκευάζει επιτυχώς. Βέβαια, η μέθοδος αυτή απαιτεί την εξαγωγή πιο έντονων στοιχείων, όπως είναι τα **features**, εν αντιθέσει με τα **points** τα οποία είναι πιο συχνό να ανιχνευθούν σε εσωτερικούς χώρους. Σε αντίθεση με την πρώτη μέθοδο, η προσέγγιση **P2P** χρησιμοποιεί όλο τον όγκο των δεδομένων από τα points που έχουν ανιχνευτεί, κάτι το οποίο ωθεί τη μέθοδο αυτή στην αποφυγή ανακριβειών και μειωμένης ποιότητας απόδοση. Ένας από

τους πιο διαδεδομένους αλγόριθμους **P2P** είναι ο αλγόριθμος επαναληπτικής αναγνώρισης κοντινότερου σημείου **(Iterative Closest Point – ICP)** ο οποίος παρουσιάστηκε για πρώτη φορά από τους (Y. Chen και G. Medioni) [21]. Η λογική που ακολουθεί αυτός ο αλγόριθμος, είναι ότι εφαρμόζει μία μέθοδο εντοπισμού κοντινότερου σημείου από την αρχική πεποίθηση και έτσι δημιουργεί αντιστοιχίες μεταξύ των σημείων, λύνοντας έτσι το πρόβλημα ελαχίστων τετραγώνων P2P και έχοντας ως αποτέλεσμα τον υπολογισμό της σχετικής θέσης του ρομπότ στον χάρτη. Το μόνο μειονέκτημα που έχει η μέθοδος P2P είναι η ανάγκη ύπαρξης ακριβής αρχικής πεποίθησης.

Αλγόριθμοι Εξαγωγής Χαρακτηριστικών – Feature Extraction

Έχοντας δει τις μεθόδους αντιστοίχισης των δεδομένων οι οποίοι συμβάλλουν στον υπολογισμό της σχετικής θέσης ενός ρομπότ, είναι σημαντικό να κατανοηθούν οι αλγόριθμοι εξαγωγής των σημαντικών αυτών χαρακτηριστικών ή σημείων **(Feature Extraction Algorithms)**. Πιο αναλυτικά, το **Feature Extraction** είναι μία λογική κατά την οποία το ρομπότ αντλεί πληροφορίες μέσω των αισθητήρων αναγνώρισης και ξεχωρίζει σημαντικά χαρακτηριστικά στην εικόνα μέσω των παραπάνω μεθόδων, τα οποία βοηθούν το ρομπότ στο localization του. Τα χαρακτηριστικά αυτά **(features ή points)** διαφέρουν αναλόγως με το σχήμα τους και το πόσο καθαρά φαίνονται. Με άλλα λόγια, οι feature extraction αλγόριθμοι διαφέρουν αναλόγως με το αν τα χαρακτηριστικά ή σημεία, που έχουν ανιχνευτεί, έχουν συγκεκριμένο σχήμα και είναι κατάλληλα προς αναγνώριση. Τα σημαντικότερα σχήματα ως προς την ευκολία αναγνώρισης αλλά και χαρτογράφησης του χώρου, είναι οι ακμές **(edges)** και οι γωνίες **(corners)**.

- Για την αναγνώριση ακμών **(edges)** σε μία εικόνα, οι αλγόριθμοι που χρησιμοποιούνται φαίνονται ονομαστικά στον *Πίνακας 3*:

	Αλγόριθμοι	**Εφευρέτες**
1	Canny (edge detector)	John F. Canny [22]
2	Deriche (edge detector) *Βασισμένος στον Canny*	Rachid Deriche [23]
3	Differential (edge detection)	
4	Sobel filter	Irwin Sobel & Gary Feldman [24]
5	Prewitt operator	Judith M. S. Prewitt [25]

Πίνακας 3: Αλγόριθμοι αναγνώρισης ακμών

Ο σημαντικότερος αλγόριθμος από αυτούς του *Πίνακας 3* είναι ο πρώτος **(Canny edge detector)** [22], στον οποίο βασίστηκαν και οι επόμενοι. Περιληπτικά, η διαδικασία αναγνώρισης των ακμών ξεκινάει με την **εφαρμογή φίλτρου Gauss**, έτσι ώστε να αφαιρεθεί ο θόρυβος στην εικόνα, και έπειτα βρίσκονται οι **διαβαθμίσεις έντασης** στην εικόνα. Αφότου έχουν ολοκληρωθεί τα δύο πρώτα βήματα, γίνεται εφαρμογή ενός **διπλού ορίου (double threshold)** για τον καθορισμό των πιθανών ακμών και, εν τέλει, γίνεται **αναγνώριση των ακμών** καταστέλωντας τις ακμές που έχουν αδύναμη ύπαρξη και κακή σύνδεση με άλλες ακμές.

- Για την αναγνώριση γωνιών **(corners)** σε μία εικόνα, χρησιμοποιούνται διαφορετικοί αλγόριθμοι από αυτούς για τις ακμές, οι οποίοι φαίνονται στον *Πίνακας 4*.

	Αλγόριθμοι	**Εφευρέτες**
1	Moravec corner detection	Hans Moravec [26]
2	The Harris & Stephens corner detection	Chris Harris & Mike Stephens [27]
3	Shi–Tomasi corner detection	J. Shi & C. Tomasi [28]
4	Level Curve Curvature Approach	L. Kitchen & A. Rosenfeld [29]
5	SUSAN	S. M. Smith & J. M. Brady [30]
6	FAST	E. Rosten & T. Drummond [31]

Πίνακας 4: Αλγόριθμοι αναγνώρισης γωνιών

Ο πιο διαδεδομένος από τους παραπάνω είναι ο αλγόριθμος **Harris & Stephens** [27], ο οποίος είναι μία βελτιωμένη έκδοση του αλγόριθμου του **Moravec** [26]. Η βασική λογική του αλγόριθμου Moravec, αναλύει την εικόνα pixel με pixel και συγκρίνει τις λεπτομέρειες έτσι ώστε να βρει μία γωνία σε ένα pixel. Η σύγκριση είναι ανάμεσα σε γειτονικά pixel, ψάχνοντας τη διαφορά στην πυκνότητα τους. Η έξοδος αυτού του αλγόριθμου είναι μία τιμή που ονομάζεται **"ομοιότητα" (similarity)** και βρίσκεται από το άθροισμα των διαφορών στο τετράγωνο **(SSD)** μεταξύ αντίστοιχων pixel· όσο μικρότερη είναι η έξοδος αυτή, τόσο πιο πιθανό είναι να βρέθηκε γωνία.

Ο αλγόριθμος **Harris & Stephens** [27], βελτίωσε τον τρόπο εντοπισμού γωνιών αλλάζοντας τον τρόπο υπολογισμού της **ομοιότητας**. Αντί να αθροίζονται τα σχετικά pixels μεταξύ τους, ο αλγόριθμος χρησιμοποιεί το διαφορικό **(differential)** των σχετικών pixels, ή όπως αλλιώς ονομάζεται **"αυτοσυσχέτιση" (autocorrelation)**, λαμβάνοντας παράλληλα υπόψιν και την κατεύθυνση που τείνουν.

Αλγόριθμοι Περιγραφής Χαρακτηριστικών – Feature Description

Για την περιγραφή των χαρακτηριστικών ενός αντικειμένου μίας εικόνας στο σύστημα, χρησιμοποιούνται διαφορετικοί αλγόριθμοι από αυτούς στο feature extraction κομμάτι. Οι αλγόριθμοι αυτοί, οι οποίοι φαίνονται στον *Πίνακας 5*, αποτελούν πιο προηγμένους αλγόριθμους λόγω του ότι όχι μόνο περιγράφουν τα χαρακτηριστικά στο σύστημα, αλλά τα μπορούν και να τα αναγνωρίζουν.

	Αλγόριθμοι	**Εφευρέτες**
1	SIFT – *Scale-invariant Feature Transform*	David Lowe [32]
2	SURF – *Speeded Up Robust Features*	Herbert Bay, Andreas Ess, Tinne Tuytelaars, & Luc Van Gool [33]
3	GLOH - *Gradient Location and Orientation Histogram*	Krystian Mikolajczyk & Cordelia Schmid [34]
4	HOG – *Histogram of Oriented Gradients*	Navneet Dalal & Bill Triggs [35]

Πίνακας 5: Αλγόριθμοι Περιγραφής Χαρακτηριστικών

Οι αλγόριθμοι αυτοί χρησιμοποιούν τα **Points of Interest – PoI** ως μέσω για να καθορίζουν και να χαρακτηρίζουν τα features των αντικειμένων σε μία εικόνα. Η λογική ως προς τη λειτουργία τους είναι αρκετά παρόμοια και στους τέσσερις αλγόριθμους, ωστόσο αξίζει να περιγραφούν συνοπτικά [36].

- **SIFT – Scale-Invariant Feature Transform** [32] → Ο αλγόριθμος αυτός μεταφράζεται ως *"Μετασχηματισμός Χαρακτηριστικών Αμετάβλητης Κλιμάκωσης"* και χρησιμοποιείται, εκτός της αναγνώρισης αντικειμένων, στη δημιουργία χαρτών πλοήγησης και τρισδιάτης μοντελοποίησης. Ένα από τα συνηθέστερα προβλήματα που αντιμετωπίζουν αυτοί οι αλγόριθμοι είναι η λανθασμένη αντιστοίχιση των features από εικόνα σε εικόνα, κάτι το οποίο δημιουργεί σύγχυση στο σύστημα και αδυναμία εντοπισμού θέσης. Ο **SIFT** αλγόριθμος, έχει το πλεονέκτημα της ποιοτικής αναγνώρισης αντικειμένων, λόγω του ότι ο *"feature descriptor"* του παραμένει πλήρως αναλλοίωτος στην κλιμάκωση των αντικειμένων ή στην αλλαγή προσανατολισμού, και μερικώς αναλλοίωτος σε διαφοροποιήσεις φωτισμού. Η **βασική** του **λειτουργία** ξεκινάει με τον **εντοπισμό σημείων ενδιαφέροντος** μέσα σε ένα σύνολο εικόνων. Αυτά τα σημεία **(keypoints)** αποθηκεύονται σε μία **βάση δεδομένων (database)**, έτσι ώστε να αποτελούν τα βασικά δεδομένα για τον εντοπισμό αντικειμένων μέσω

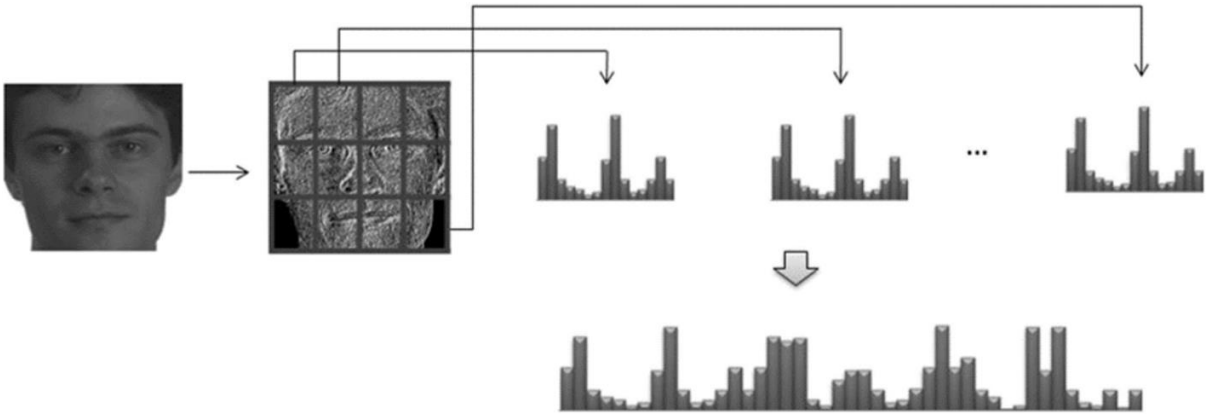
σύγκρισης με καινούργιες εικόνες. Ένα **καινούργιο αντικείμενο**, θα **εντοπιστεί** σε μία νέα εικόνα, όταν τα keypoints που υπάρχουν ήδη στη βάση δεδομένων συμπίπτουν με τα νέα features, βάσει της Ευκλείδειας απόστασης τους. Τα νέα features κατηγοριοποιούνται ως **"good matches"** αναλόγως με το πόσο συμπίπτουν με τα προϋπάρχοντα, δηλαδή με την κλίμακα μεγέθους τους, τον προσανατολισμό τους και την θέση τους. Στο τελικό βήμα, τα good matches κατηγοριοποιούνται σε ομάδες **(clusters)** των τριών και πάνω, με τη βοήθεια του γενικευμένου μετασχηματισμού Hough **(Hough transform)** [37], και ξεδιαλύνονται από ακραίες τιμές.

- **SURF – Speeded Up Robust Features** [33] → Ο αλγόριθμος αυτός χρησιμοποιείται εκτός από τον εντοπισμό αντικειμένων και σαν μέθοδος για να δημιουργείται ένα σύστημα συντεταγμένων μέσα από σύνολα δεδομένων εικόνας **(Image registration).** Είναι βασισμένος στον αλγόριθμο SIFT, με την μόνη διαφορά ότι είναι αρκετά πιο γρήγορος και πιο σταθερός στην απόδοση του. Επίσης, διαφέρει στο ότι χρησιμοποιεί μεθόδους αναγνώρισης άμορφης μάζας **(blob detectors)** και όχι αναγνώρισης γωνιών ή ακμών. Η μέθοδος του βασίζεται στον αλγόριθμο **DoH – Determinant of Hessian** και συγκρίνει την εικόνα ως προς τις αλλαγές πυκνότητας και χρωμάτων μέσω μαθηματικών υπολογισμών. Η βασική τεχνική του SURF είναι ο μετασχηματισμός των **PoI** μίας εικόνας σε συντεταγμένες, κάνοντας χρήση της τεχνικής **"Multi-resolution Pyramid"** [38]. Ως αποτέλεσμα, δημιουργείται ένα αντίγραφο της εικόνας, στο οποίο όμως έχουν εφαρμοστεί τεχνικές όπως **Laplacian Pyramid** και **Pyramidal Gaussian** [39], έτσι ώστε η νέα εικόνα να έχει μειωμένο εύρος ζώνης, άλλα ίδιο μέγεθος. Τέλος, το αποτέλεσμα αυτού του αλγόριθμου είναι ότι έχει εφαρμοστεί στην εικόνα ένα ειδικό φίλτρο, το οποίο ονομάζεται **"scale-space"**, και τα points of interest δεν επηρεάζουν την σύγκριση features σε ότι αναφορά το μέγεθος.

- **HOG –Histogram of Oriented Gradients** [35] → Είναι ένας από τους πιο διαδεδομένους αλγόριθμους, ο οποίος χρησιμοποιείται για τον παρόμοιο λόγω της αναγνώρισης αντικειμένων, αλλά και την αναγνώρισης προσώπων. Η λογική του βασίζεται στον διαχωρισμό της εικόνας, προς αναγνώριση, σε πολλές μικρές περιοχές που ονομάζονται **κελιά (cells)**. Τα σχήματα που υπάρχουν μέσα στην εικόνα όπως και η εμφάνιση τους, περιγράφονται στο σύστημα ως **διαβαθμίσεις έντασης (intensity gradients)** και **εναλλαγές ακμών**. Έτσι, σε κάθε κελί εμπεριέχεται ένα μέρος της εικόνας, όπου το σύνολο των pixels του κάθε κελιού σχηματίζουν ένα **ιστόγραμμα**

(histogram) από διαβαθμίσεις εντάσεων. Το σύνολο όλων των ιστογραμμάτων μίας εικόνας αποτελεί τον **περιγραφέα (descriptor)** του αλγόριθμου. Πέρα από τη βασική λειτουργία και έχοντας ως στόχο την βελτίωση της ποιότητας κατά την αναγνώριση, η μέθοδος αυτή εφαρμόζει μία **εξομάλυνση αντίθεσης**, υπολογίζοντας τον όγκο έντασης σε μία μεγαλύτερη περιοχή που ονομάζεται **block**. Έτσι, η εκτιμώμενη τιμή όγκου τίθεται ως **σταθεροποιητής** και εφαρμόζεται σε κάθε ιστόγραμμα που βρίσκεται μέσα στο block αυτό. Αυτή η διαδικασία έχει ως απώτερο σκοπό την εξομάλυνση έντονων αλλαγών φωτισμού **(illumination)** και σκιάσεων **(shadowing)**. Τέλος, τα πλεονεκτήματα που διαθέτει ο παρόν αλγόριθμος σε σύγκριση με τους υπόλοιπους είναι τα εξής:

- Λόγω της ανάλυσης ανα κελιού που γίνεται στην εικόνα, ο αλγόριθμος αυτός δεν επηρεάζεται από γεωμετρικές ή φωτομετρικές διαμορφώσεις.
- Λόγω του ότι μπορεί να εντοπίσει με υψηλή ακρίβεια τον προσανατολισμό και να ομαλοποιήσει τις φωτομετρικές διαβαθμίσεις, ο αλγόριθμος αυτός αποτελεί ιδανική μέθοδος για εντοπισμό προσώπων **(face detection)**.



Εικόνα 19: Εφαρμογή του αλγόριθμου HoG σε φωτογραφία προσώπου *[40]*

Όπως προαναφέρθηκε, για την βέλτιστη κατανόηση του χώρου μέσω των παραπάνω αλγορίθμων, ο καταλληλότερος αισθητήρας είναι η κάμερα βάθους ή ο laser αισθητήρας, διότι δίνει τη δυνατότητα στο ρομπότ να καταλαβαίνει το βάθος του σκηνικού. Όμως, στην πρακτική εφαρμογή της παρούσας πτυχιακής εργασίας, το **Parrot AR Drone 2.0.** [7] ήταν εξοπλισμένο μόνο με **monocular camera** ωθώντας έτσι την πλοήγηση του drone, και άρα το SLAM, σε **monocular SLAM algorithms**.

2.5.2. Αλγόριθμοι SLAM με Κάμερα Μονού Οφθαλμού – Monocular SLAM

Όπως έχει προαναφερθεί, ένα από τα σημαντικότερα προβλήματα στο χώρο της αυτόνομης πλοήγησης, είναι η ασάφεια και η δυσκολία στην εκτίμηση της κλίμακας του περιβάλλοντος **(scale estimation)** στο οποίο βρίσκεται το ρομπότ. Το πρόβλημα αυτό εμφανίζεται σε πιο έντονο βαθμό, όταν οι αισθητήρες αναγνώρισης περιβάλλοντος ενός ρομπότ δεν είναι παρά μόνο μία κάμερα μονού οφθαλμού. Τότε, το σύστημα πρέπει να εφαρμόσει **monocular SLAM** [41] [42] για να μπορεί να εντοπίζει την θέση του στον χώρο **(real-time positioning)**, όπως και για να δημιουργήσει ένα χάρτη του περιβάλλοντος του. Η μέθοδος αυτή, ξεκίνησε από τον *Prof. Andrew Davison* το 2003, ενώ η επιστημονική του έρευνα εκδόθηκε το 2007 [43]. Το μειονέκτημα που υπάρχει στην περίπτωση της κάμερας μονού οφθαλμού, είναι η έλλειψη δυνατότητας εντοπισμού του βάθους του περιβάλλοντος μόνο από τον αισθητήρα. Ως αποτέλεσμα, ο αλγόριθμος του monocular SLAM μοιάζει με τους παραπάνω όσο αφορά την αναγνώριση εικόνας, αλλά χρειάζεται και επιπρόσθετες λειτουργίες έτσι ώστε να μπορεί να εκτιμά και το βάθος στο περιβάλλον του ρομπότ.

Οι μέθοδοι που εφαρμόζουν monocular SLAM χρησιμοποιούν σαν βάση, για την εξαγωγή και περιγραφή χαρακτηριστικών, τους βασικούς αλγόριθμους που αναλύθηκαν προηγουμένως και έπειτα, αναπαριστούν τα χαρακτηριστικά αυτά σε τρισδιάστο μοντέλο μέσα στο χάρτη. Σε αυτό το σημείο, αξίζει να σημειωθεί ότι το monocular SLAM χωρίζεται σε δύο κύριες κατηγορίες: (1) την κανονική μέθοδο SLAM **(feature-based SLAM)** και (2) την άμεση μέθοδο SLAM **(direct SLAM)**. Η διαφορά τους είναι, ότι η *feature-based SLAM* μέθοδος βασίζεται πάνω στους αλγόριθμους που αναλύθηκαν στην προηγούμενη υποενότητα (**SIFT**, **SURF**, …**)**, ενώ η *direct SLAM* μέθοδος προσπερνάει το βήμα του feature extraction και description και αναλύει ολόκληρη την εικόνα pixel ανα pixel. Ο αλγόριθμος που εφαρμόζει αυτή τη μέθοδο ονομάζεται **LCD-SLAM (Large-Scale Direct Monocular SLAM)** [44] και έχει αναπτυχθεί και σε πακέτο ROS, με συμβατότητα στο AR Drone 2.0.

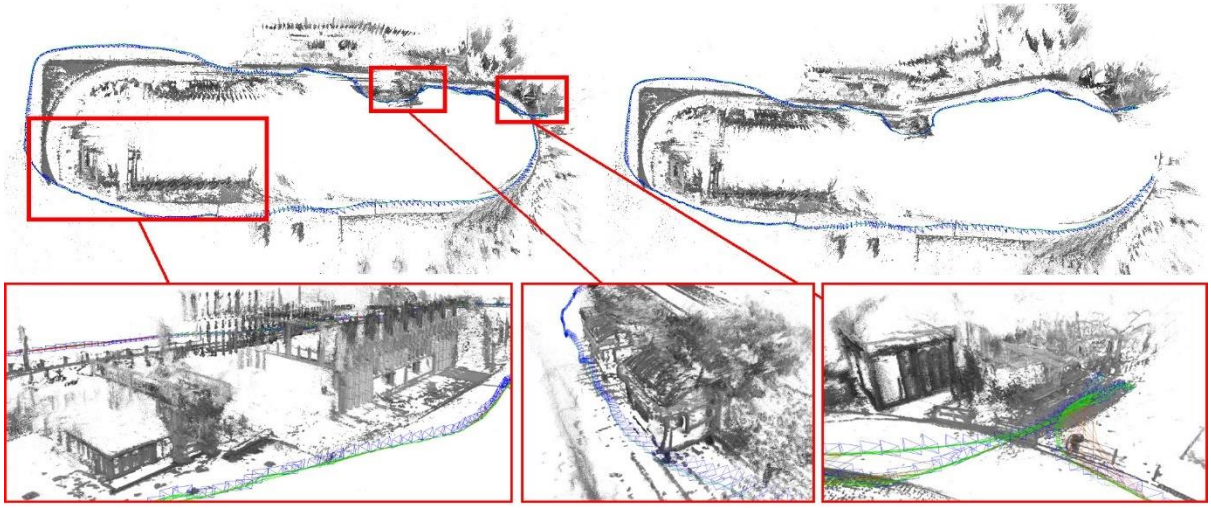
Feature-Based SLAM Λογική

Γενικώς, οι **feature-based** αλγόριθμοι **SLAM** ακολουθούν δύο κύρια βήματα, έτσι ώστε να μπορέσουν να εντοπίσουν το βάθος στην εικόνα και να δημιουργήσουν ένα τρισδιάστατο μοντέλο. Το πρώτο βήμα είναι η εφαρμογή των αλγορίθμων SIFT, SURF, … έτσι ώστε να εξαχθούν σύνολα χαρακτηριστικών από την εικόνα και να τα περιγραφούν στο σύστημα ως παρατηρήσεις. Το δεύτερο και πιο σημαντικό βήμα, είναι ο υπολογισμός την τοποθεσίας της κάμερας αλλά και της γεωμετρίας του περιβάλλοντος, συναρτήσει των

παρατηρήσεων που δημιουργήθηκαν από το πρώτο βήμα. Το βήμα αυτό γίνεται με τη χρήση μίας αντίστροφης παραμετροποίησης βάθους **(Inverse Depth Parametrization)** [45]. Ως αποτέλεσμα, έπειτα από μαθηματικούς υπολογισμούς, οι οποίοι διαφέρουν από μέθοδο σε μέθοδο, δημιουργείται ένας 3D χάρτης στον οποίο εμπεριέχεται και το βάθος του χώρου.

Direct SLAM Λογική – LSD-SLAM

Εν αντιθέσει, η μέθοδος **LSD-SLAM** [44] για τον εντοπισμό θέσης κάνει ευθυγράμμιση εικόνας-σε-εικόνα **(image-to-image alignment)** μέσω ενός αλγόριθμου **coarse-to-fine**, κάνοντας παράλληλα χρήση της στατιστικής συνάρτησης **Huber loss.** Πιο αναλυτικά, ένας αλγόριθμος *coarse-to-fine* ξεκινάει να αναλύει με τριγωνισμό **(triangulation)** ένα μικρότερο σύνολο δεδομένων και κάθε φορά που καταλήγει σε ένα αποτέλεσμα προσθέτει επιπλέον δεδομένα από το σύνολο. Η διαδικασία αυτή γίνεται έως ότου το σφάλμα μεταξύ του τελευταίου τριγωνικού αποτελέσματος και του ολικού, να βρίσκεται μέσα σε ένα επιτρεπτό όριο. Επίσης, με σκοπό ο αλγόριθμος να είναι ακριβής γίνεται χρήση της συνάρτησης *Huber loss*, η οποία είναι πιο ελαστική στις ακραίες τιμές από ότι είναι το τετραγωνικό σφάλμα. Η εκτίμηση του βάθους του χώρου γίνεται με την αντίστροφη παραμετροποίηση βάθους, χρησιμοποιώντας ένα σχετικό αριθμό από ζεύγη εικόνας. Η βασική διαφορά του αλγόριθμου αυτού είναι ότι δεν στηρίζεται στα χαρακτηριστικά της εικόνας, αλλά στη σύσταση του χώρου· εκτελεί δηλαδή **"εντοπισμό σύστασης" (texture tracking).** Έτσι, η συνολική χαρτογράφηση γίνεται σε real-time μέσω ενός **pose graph**, το οποίο δημιουργείται από την διαδρομή του ρομπότ μέσα στο χώρο.



Εικόνα 20: Pose Graph βάσει της διαδρομής του ρομπότ *[46]*

Τέλος, ο αλγόριθμος αυτός χρησιμοποιεί τη λογική δημιουργίας χάρτη βάθους **semi-dense**, διότι μπορεί να εντοπίσει αποτελεσματικά το βάθος του χώρου μέσω των pixels μόνο στα όρια της εικόνας. Αντιπροσωπευτικό στοιχείο της έρευνας αυτής αποτελεί το παρακάτω βίντεο:

https://youtu.be/GnuQzP3gty4 [46]

Συγκρίνοντας τις μεθόδους **feature-based SLAM** και **Direct SLAM**, προκύπτει ότι οι αλγόριθμοι που βασίζονται στα χαρακτηριστικά και τις παρατηρήσεις είναι πιο γρήγοροι στην εκτέλεση, πιο σταθεροί σε ασυνέπειες του συστήματος ή του μοντέλου και δεν χρειάζονται κάποια αρχικοποίηση παραμέτρων. Εν αντιθέσει, το direct SLAM έχει τη δυνατότητα να χρησιμοποιήσει ολόκληρη την εικόνα ως πληροφορία και όχι μόνο τις γωνίες και τις ακμές, δημιουργώντας έτσι πιο ακριβές χάρτη· κάτι το οποίο σημαίνει ότι οι αποφάσεις του συστήματος είναι πιο ολοκληρωμένες λόγω πληρότητας δεδομένων. Τέλος, η ακρίβεια του αλγόριθμου το καθιστά ιδανικό για την εφαρμογή του σε αυτόνομη πλοήγηση UAV.

2.5.3. PTAM – Parallel Tracking & Mapping

Η μέθοδος αυτή, του παράλληλου εντοπισμού σημείων και της χαρτογράφησης **(Parallel Tracking and Mapping – PTAM)** [47], εξυπηρετεί στην εκτίμηση της θέσης του ρομπότ σε πραγματικό χρόνο μέσα στο χώρο. Πιο αναλυτικά, ο αλγόριθμος αυτός βασίζεται στην κάμερα μονού οφθαλμού και μέσα από real-time υπολογισμούς, εκτιμά τη θέση της (άρα και του ρομπότ) στον τρισδιάστατο χώρο **(tracking)**. Επιπλέον, χαρτογραφεί **(mapping)** το περιβάλλον μέσω των θέσεων διαφόρων ορατών σημείων, που εντοπίζει χρησιμοποιώντας feature-based αλγόριθμους και επεξεργάζοντας τις πληροφορίες αυτές σε πραγματικό χρόνο. Επομένως, ο αλγόριθμος αυτός χωρίζεται σε δύο βασικά κομμάτια: **(1)** την εκτίμηση θέσης ή αλλιώς **tracking**, και **(2)** την χαρτογράφηση του περιβάλλοντος ή αλλιώς **mapping.** Η δομή του αλγόριθμου αυτού, βασίζεται στην παράλληλη επεξεργασία και εκτέλεση των δύο αυτών ενεργειών, διαμοιράζοντάς αυτές σε ξεχωριστά threads ενός multi-core επεξεργαστή με σκοπό την σταθερή απόδοση του tracking, αλλά και τον υψηλής ακρίβειας points-based χάρτη.

1) **<u>Tracking:</u>** Λόγω του ότι το ρομπότ μπορεί και κινείτε στο χώρο, είναι δυνατό να υπολογιστεί η θέση του με τη βασική μέθοδο του SLAM. Με άλλα λόγια, η θέση εκτιμάται μέσω τριγωνικού υπολογισμού **(triangulation)** των παρατηρήσεων που έχει λάβει το σύστημα από τους feature extraction και description αλγόριθμους, αλλά και με τεχνικές στερεοφωνικής αρχικοποίησης **(stereo initialization)**· μιάς και το

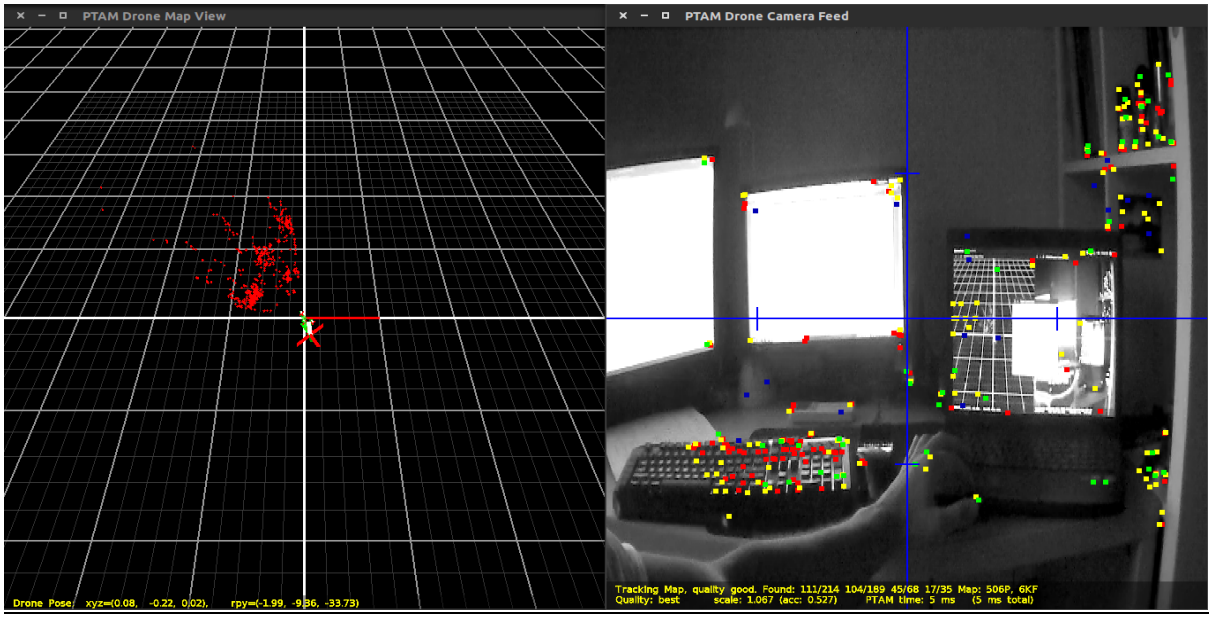
ρομπότ μπορεί να παρατηρήσει τον χώρο μέσα από διαφορετικές οπτικές γωνίες. Με τον τρόπο αυτό, επιτυγχάνεται η εκτίμηση της σχετικής θέσης **(relative position)** του ρομπότ σε πραγματικό χρόνο, με βάση τα αντικείμενα και γενικώς τα χαρακτηριστικά που βρίσκονται στο χώρο και έχουν παρατηρηθεί από το σύστημα. Αξίζει να σημειωθεί ότι το σύστημα ακολουθεί την τεχνική **"πυραμίδα"** (η οποία αναφέρθηκε στην ανάλυση του αλγόριθμου SURF), δηλαδή δημιουργεί μία πυραμίδα πολλαπλών επιπέδων στην οποία εμπεριέχονται διάφορα frames από το real-time βίντεο σε διαφορετικές αναλύσεις, με τις καλύτερες από αυτές να βρίσκονται στην κορυφή. Με αυτό τον τρόπο, το σύστημα πετυχαίνει υψηλή ευστάθεια όσον αφορά πιθανά σφάλματα θολούρας ή λανθασμένων κινήσεων, και αντιστοιχίζει τα χαρακτηριστικά σε πολλαπλές αποστάσεις και αναλύσεις· πετυχαίνοντας έτσι υψηλότερη απόδοση στον εντοπισμό της θέσης του, όπως και στην ανάλυση του χώρου. Ακόμα, η αρχικοποίηση **(stereo initialization)** παρέχει στο σύστημα την πρώτη εκτίμηση θέσης μέσα από ένα μικρό αριθμό σημείων που έχουν ανιχνευτεί. Βάσει αυτής της θέσης, το σύστημα μπορεί να αντιστοιχίσει τα features με αυτά από ένα υψηλό επίπεδο της πυραμίδας, έχοντας έτσι πολύ καλύτερη ανάλυση. Ως αποτέλεσμα, η δεύτερη εκτίμηση θέσης του ρομπότ γίνεται με αισθητά μεγαλύτερη ακρίβεια, αλλά και οι επόμενες ενημερώσεις θέσεων με εκθετική διαφορά. Τέλος, εν αντιθέσει με τον αλγόριθμο **MonoSLAM** [43], ο αλγόριθμος PTAM εξοικονομεί χρόνο και αποθηκευτικό χώρο διότι αποθηκεύει τις μετρήσεις και την θέση του ρομπότ, μόνο στην τελική θέση. Δυστυχώς, αυτή η ενέργεια έχει το αρνητικό ότι αν το σύστημα χάσει την επαφή του με τον χώρο, δηλαδή αν το tracking χαθεί, ο αλγόριθμος ξεκινάει την αρχικοποίηση πάλι από την αρχή.

2) **<u>Mapping:</u>** Για τη σωστή χαρτογράφηση του χώρου, το tracking και το mapping πρέπει να συνδέονται άμεσα μεταξύ τους. Έτσι, οι μετρήσεις περιγράφονται στο σύστημα με κοινούς όρους. Με άλλα λόγια, η θέση της κάμερας (δηλαδή του ρομπότ) εκφράζεται με βάση τη σχετική απόσταση από κάποια σταθερά σημεία στο χώρο, η οποία είναι απαραίτητη πληροφορία για τη χαρτογράφηση νέων σημείων.

   Το κομμάτι του mapping, ξεκινάει αμέσως μετά την ολοκλήρωση της αρχικοποίησης και έχει ως στόχο την συλλογή όσο περισσότερων σημείων από το περιβάλλον. Αξίζει να σημειωθεί ότι η μέθοδος αυτή λειτουργεί με βάση **"keyframes"** και όχι με κάθε frame που έρχεται από την κάμερα. Αυτή η λογική πλεονεκτεί στο ότι, παρά τη μη real-time επεξεργασία που μπορεί να συμβαίνει, το σύστημα είναι αρκετά σταθερό στον υπολογισμό σφαλμάτων, αλλά και τα επιλεγμένα frames έχουν καλύτερη

ποιότητα σε σχέση με τα υπόλοιπα· κάτι το οποίο βελτιώνει και την ποιότητα του τελικού χάρτη. Ακόμα, λόγω ότι τα **keyframes** είναι επιλεγμένα frames από το σύστημα, δίνεται η δυνατότητα επανεξέτασης αυτών με στόχο την βελτίωση είδη υπάρχοντων χαρτών. Δηλαδή, όταν το σύστημα αντιλαμβάνεται ότι η παρούσα περιοχή έχει ξανά αναλυθεί, τότε το thread που εκτελεί στο παρασκήνιο επανεξετάζει τα προηγούμενα keyframes με στόχο την βελτίωση της ποιότητας του χάρτη, όπως και τη λήψη επιπλέον σημείων.

Η επιλογή των keyframes γίνεται με κριτήρια μίας προκαθορισμένης απόστασης μεταξύ σημείων, αλλά και ανα 20 frames ενός βίντεο. Οι λόγοι μίας τόσο φιλτραρισμένης επιλογής είναι **(1)** η αποφυγή πολλαπλών ίδιων σημείων μέσα σε δύο συνεχόμενα keyframes, όπως και **(2)** η εξάλειψη πιθανής διαφθοράς του χάρτη **(map corruption)**.



Εικόνα 21: Παράδειγμα εφαρμογής PTAM από την κάμερα του AR Drone

Το συμπέρασμα που βγαίνει από την ανάλυση του αλγόριθμου PTAM, είναι ότι η λογική λειτουργίας του το ωθεί διαρκώς να βελτιώνει την ποιότητα της ανάλυσης που κάνει, όπως και την ανάλυση του χάρτη. Στην πρακτική εφαρμογή αυτής της πτυχιακής εργασίας, το PTAM χρησιμοποιήθηκε ως βασικό εργαλείο για την χαρτογράφηση του περιβάλλοντος στο οποίο κινούταν το drone. Αξίζει να σημειωθεί, ότι ο αλγόριθμος PTAM είναι ενσωματωμένος στο πακέτο λογισμικού **“tum_ardrone”** [4] [48] [49] [50].

3. <u>Κεφάλαιο 3:</u> Πρακτική Εφαρμογή

3.1. Σύνοψη Κεφαλαίου

Στις ενότητες του κεφαλαίου αυτού, θα αναλυθεί η πρακτική εφαρμογή που μελετήθηκε σε διάρκεια 6 μηνών και φέρει τον τίτλο *"Ανάλυση τεχνικής πραγματοποιησιμότητας στη χρήση αυτόνομης πλοήγησης μη επανδρωμένων αεροσκαφών, για έλεγχο και επιθεώρηση σε multi-megawatt πτέρυγες ανεμογεννητριών"*. Με άλλα λόγια, θα περιγραφεί το βασικό υπόβαθρο όσον αφορά την επιθεώρηση πτερύγων με τη χρήση αυτόνομης πλοήγησης, αλλά θα γίνει και παράθεση των στόχων αυτής της εφαρμογής παράλληλα με την επεξήγηση των δυσκολιών που συναντήθηκαν. Ακόμα, θα αναλυθούν τα πακέτα λογισμικού ROS [3] **(1) ardrone_autonomy** [5] και **(2) tum_ardrone** [4]**,** τα οποία χρησιμοποιήθηκαν για την ανάπτυξη της ερευνητικής εφαρμογής, καθώς θα γίνει και επεξήγηση του λογισμικού κώδικα που αναπτύχθηκε για την αυτόνομη κίνηση του **Parrot AR Drone** [7].

Εικόνα 22: Παράδειγμα Επιθεώρησης Ανεμμογεννήτριας με UAV *[51]*

3.2. Στόχοι & Αντιμετώπιση Δυσκολιών της Πρακτικής Εφαρμογής

Ο βασικός στόχος αυτής της πρακτικής εφαρμογής, ήταν η ολοκληρωτική θεωρητική και εν μέρη πρακτική μελέτη της τεχνικής πραγματοποιησιμότητας για επιθεώρηση πτέρυγας ανεμογεννήτριας, με τη χρήση αυτόνομου μη επανδρωμένου αεροσκάφους. Βέβαια, λόγω της αυξημένης δυσκολίας στην έρευνα αλλά και του υψηλού κόστους, δεν ήταν δυνατό να αποκτηθεί ένα μη επανδρωμένο αεροσκάφος κατάλληλο για μία τόσο προηγμένη εφαρμογή. Έτσι, η πρακτική εφαρμογή περιορίστηκε στην προσομοίωση των δυσκολιών αλλά και τη δημιουργία πηγαίου κώδικα, ο οποίος εκτελείται από το UAV **Parrot AR Drone 2.0.** [7] κάνοντας μία βαθμωτή κυκλική κίνηση επιθεώρησης γύρω από ένα αντικείμενο.

Αυτόνομη Επιθεώρηση μίας Πτέρυγας Ανεμογεννήτριας με χρήση UAV

Η ***επιθεώρηση μίας πτέρυγας ανεμογεννήτριας*** με τη χρήση αυτόνομης πλοήγησης από ένα UAV έχει μία δύσκολη και ιδιαίτερα προηγμένη ***διαδικασία***. Το αεροσκάφος κατά την αρχική του θέση **(initial state)**, δηλαδή προσγειωμένο στη βάση του, θα πρέπει να δέχεται τις συντεταγμένες της εκάστοτε πτέρυγας προς επιθεώρηση από τον χρήστη. Έπειτα, μέσω ενός βέλτιστα ρυθμισμένου συστήματος ελέγχου, το drone θα εκτελεί κίνηση με πορεία προς την πτέρυγα, αντιμετωπίζοντας παράλληλα όλες τις αναταράξεις. Το δύσκολο κομμάτι της εφαρμογής έρχεται όταν το drone φτάσει την εκάστοτε πτέρυγα, σε κάποια κοντινή απόσταση. Εκεί, λόγω την αυξημένης ακρίβειας που απαιτεί η εφαρμογή, αλλά και της μικρής απόστασης παράλληλα με τις πιθανές αναταράξεις, η δυσκολία αυξάνει εκθετικά διότι το drone θα πρέπει να είναι τόσο ακριβές όσο και γρήγορο. Έχοντας σταθεροποιηθεί, η επόμενη διαδικασία είναι η αναγνώριση των ορίων της πτέρυγας μέσω feature-based αλγόριθμων [52], και η λήψη φωτογραφιών με σκοπό τον εντοπισμό πιθανών ζημιών. Μία από τις κυριότερες δυσκολίες είναι, ολοκληρώνοντας τη διαδικασία εντοπισμού βλαβών της μίας πλευράς, η μετακίνηση στην αντίθετη πλευρά. Εκεί, λόγω δημιουργίας χαοτικού μοντέλου ριπών ανέμου, το drone έχει αυξημένες πιθανότητες μεγάλων και επικίνδυνων διαταραχών. Τέλος, έχοντας αναλύσει την πτέρυγα όσον αφορά πιθανές ζημιές, το UAV θα πρέπει να αξιολογήσει την κατανάλωση της μπαταρίας του και αναλόγως με το αποτέλεσμα, να προχωρήσει σε επόμενη πτέρυγα ή να γυρίσει στη βάση του· αντιμετωπίζοντας πάλι με ακρίβεια τυχόν αναταράξεις. Στις δύο επόμενες ενότητες **(3.3.1, 3.3.2)**, αναλύονται με περισσότερες λεπτομέρειες τα προβλήματα, όπως παραθέτονται και πιθανές λύσεις, με σκοπό την αναλυτική περιγραφή της πρακτικής εφαρμογής.

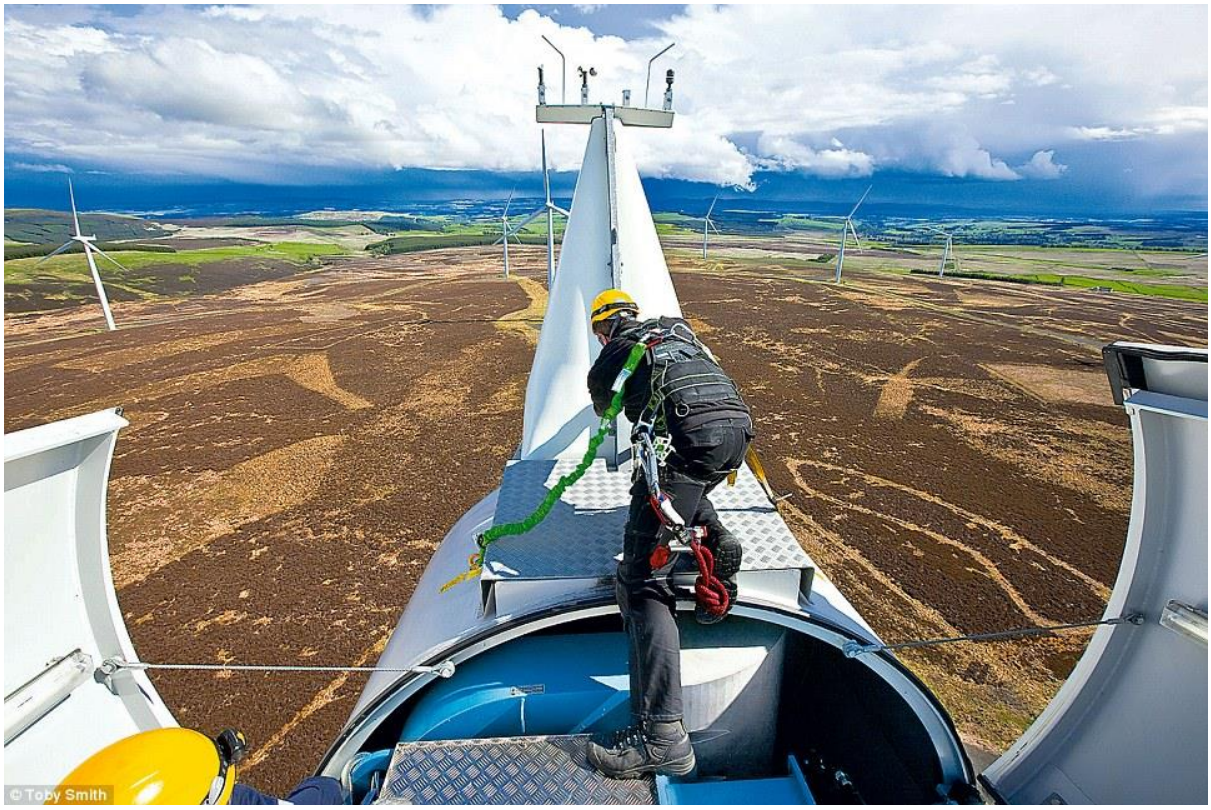
Στόχοι και Δυσκολίες κατά την Ανάπτυξη του Πηγαίου Κώδικα

Ο πηγαίος κώδικας που αναπτύχθηκε, και παρατίθεται στην ενότητα (3.4.2) της πτυχιακής εργασίας, έχει ως στόχο την πλοήγηση του **Parrot AR Drone 2.0.** [7] γύρω από ένα αντικείμενο. Η κίνηση αυτή, προσπαθεί να αντιγράψει όσο γίνεται περισσότερο την κίνηση ενός drone κατά την αυτόνομη επιθεώρηση μίας πτέρυγας ανεμογεννήτριας. Επίσης, γίνεται εφαρμογή και προσομοίωση του κώδικα μέσω του open-source εργαλείου **Gazebo** [6] έτσι ώστε να φαίνεται η κίνηση του drone με βέλτιστη απόδοση. Ο πηγαίος κώδικας όπως έχει προαναφερθεί, βασίστηκε σε προϋπάρχοντα πακέτα με σκοπό την αποφυγή επιπρόσθετης έρευνας για την ανάπτυξη αντίστοιχων. Έτσι, τα πακέτα **tum_ardrone** [4] και **ardrone_autonomy** [5] χρειάστηκε να αναλυθούν ολοκληρωτικά, έτσι ώστε να κατανοηθεί η δομή και η λειτουργία τους σε όλα τα επίπεδα. Δυσκολίες κατά τη διάρκεια ανάπτυξης του πηγαίου κώδικα υπήρξαν πολλές· κάποιες πιο σημαντικές, όπως η σωστή ανάπτυξη μεθόδων εύρεσης συντεταγμένων, αλλά και συνηθισμένες για έναν προγραμματιστή, όπως τα συχνά λάθη στον πηγαίο κώδικα. Παρ' όλα αυτά, η κυριότερη δυσκολία που υπήρξε και κόστισε αρκετό χρόνο, ήταν η σωστή επικοινωνία όλων των μεθόδων του πηγαίου κώδικα με τα πακέτα που χρησιμοποιήθηκαν. Πιο συγκεκριμένα, ο πηγαίος κώδικας χρειάστηκε να αλλαχτεί ολοκληρωτικά τρείς φορές, μέχρι να φτάσει στο τελικό στάδιο. Οι συναρτήσεις του πηγαίου κώδικα επικοινωνούν με τα άλλα πακέτα μέσω **topics** και κάποιων **nodes**, με σκοπό τη γρήγορη αλληλεπίδραση αυτών αλλά και την χαμηλή απαιτητικότητα στο θέμα ισχύς του ηλεκτρονικού υπολογιστή. Στην ενότητα **(3.4)** περιγράφεται αναλυτικά η δομή των πακέτων που χρησιμοποιήθηκαν, όπως και οι πιο σημαντικές μέθοδοι που απαρτίζουν τον πηγαίο κώδικα.

3.3. Βασικά Στοιχεία για την Αυτόνομη Επιθεώρηση Ανεμογεννητριών με UAV – Basics of Wind Turbine Blade Inspection with UAV

3.3.1. Βασικό Υπόβραθρο και Εισαγωγή – Basics and Introduction

Λόγω της υψηλής έκθεσης στο περιβάλλον, οι πτέρυγες των ανεμογεννητριών επιδέχονται σημαντικές ζημιές δεδομένου των ακραίων καιρικών συνθηκών. Επίσης, ζημιές στις πτέρυγες μπορεί να προκληθούν και από άλλες αιτίες, όπως είναι τα πτηνά, οι κεραυνοί αλλά και η διάβρωση. Το αποτέλεσμα τέτοιων ζημιών αλλά και πιο απλών διαταραχών, όπως είναι η βρωμιά, είναι η απώλεια παραγωγικότητας· κάτι το οποίο σε τόσο μεγάλες κατασκευές έχει ως αποτέλεσμα και μεγάλη ζημιά σε χρηματικό κόστος στην ιδιοκτήτρια εταιρεία. Ως εκ τούτου, με σκοπό την αποφυγή πιθανών προβλημάτων ή ακόμα πιο σοβαρών ζημιών, είναι σημαντικό να εντοπίζονται οι πιθανές βλαβες των πτερύγων έτσι ώστε να μειωθεί και το κόστος ζημιάς αλλά και ο χρόνος διακοπής της λειτουργίας της. Αυτό καταλήγει στην επιθεώρηση **(inspection)** των ανεμογεννητριών, κάτι το οποίο στην παρούσα χρονική στιγμή γίνεται μέσω επαγγελματιών τεχνικών **(rope access technicians)**.



Εικόνα 23: Rope Access Technician *[53]*

Η διαδικασία της επιθεώρησης, όπως προαναφέρθηκε, εκτελείται από ειδικευμένους εναερίτες οι οποίοι αναρριχώνται στην ανεμογεννήτρια, καθώς έχει διακοπεί η λειτουργία της, με σκοπό να εξετάσουν πιθανές βλάβες και αν είναι δυνατό, να τις διορθώσουν. Δυστυχώς, ο υψηλός βαθμός ρίσκου στο να τραυματιστούν αλλά και οι υψηλοί μισθοί, καθιστούν το επάγγελμα τους σε μειονεκτική θέση απέναντι στις εταιρείες. Τη λύση σε αυτό το θέμα, έρχεται να δώσει η επιθεώρηση με τη χρήση μη επανδρωμένων αυτόνομων αεροσκαφών. Με αυτόν τον τρόπο, οι εταιρίες όχι μόνο θα μειώσουν το κόστος αλλά, ταυτόχρονα, θα αυξήσουν την ποιότητα ελέγχου, την αποδοτικότητα και την ασφάλεια κατά τη διάρκεια της επιθεώρησης.

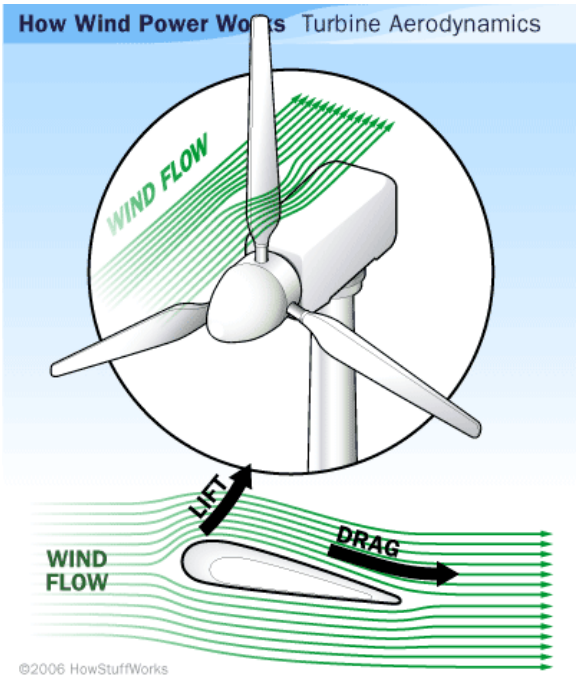
Λόγω της συνεχής βελτίωσης και ανάπτυξης της τεχνολογίας των drones, πολλά νέα και καινοτόμα χαρακτηριστικά προστίθενται διαρκώς. Επι των ημερών μας, τα **πλεονεκτήματα** που υπάρχουν για την επιθεώρηση ανεμογεννητριών με drones, σε αντίθεση με τον τυπικό μέχρι τώρα τρόπο της επιθεώρησης με τεχνικούς, είναι πολυάριθμα. Για παράδειγμα, καθαρότερη λήψη φωτογραφιών (λόγω μικρής απόστασης), ακρίβεια στην πλοήγηση του drone (σε αρκετά κοντινή απόσταση από την πτέρυγα), μειωμένος χρόνος επιθεώρησης και μειωμένο κόστος λόγω της έλλειψης προσωπικού. Βέβαια, όλα αυτά τα πλεονεκτήματα δεν έχουν γίνει πράξη ακόμα, λόγω της έλλειψης έρευνας πάνω στο συγκεκριμένο θέμα, αλλά και της αυξημένης δυσκολίας. Πιο συγκεκριμένα, μέχρι στιγμής η μόνη εξέλιξη που έχει γίνει στο χώρο της επιθεώρησης, είναι η χρήση UAV για επιθεώρηση πτερύγων βάσει πιλότου. Για τον λόγω αυτό, παρακάτω θα αναλυθούν τα προβλήματα και οι δυσκολίες που πρέπει να αντιμετωπιστούν αλλά και πιθανές λύσεις αυτών.

Αεροδυναμική & Αιολική Ενέργεια – Aerodynamics & Wind Power

Οι αιολικές μηχανές λειτουργούν σε περιβάλλον που χαρακτηρίζεται από συνεχόμενα εναλλασσόμενη ταχύτητα ανέμου, αλλά και κατεύθυνση. Ο λόγος για τον οποίο η πτέρυγα της ανεμογεννήτριας **(rotor blade)** και ο κεντρικός άξονας **(hub)** είναι τοποθετημένοι προσήνεμα **(upwind)** της ατράκτου και του κεντρικού πύργου, είναι επειδή ο άνεμος που βρίσκεται πίσω από την ανεμογεννήτρια δημιουργεί υψηλές αναταράξεις, οι οποίες προκαλούνε ανακυκλική φόρτιση κόπωσης στην υπήμενη διάταξη [54]. Βάσει αυτού του παραδείγματος, οι συνεχόμενες και μη προβλέψιμες αλλαγές κατεύθυνσης και ταχύτητας ανέμου είναι ένα από τα κύρια και πιο περίπλοκα προβλήματα κατά τη διάρκεια μίας πτήσης ενός drone. Πιο αναλυτικά, οι μη προβλέψιμες αλλαγές ανέμου μπορεί να προκαλέσουν στο drone σημαντική διαταραχή στο σύστημα ευστάθειας του, με αποτέλεσμα τον αποπροσανατολισμό του ή ακόμα χειρότερα της σύγκρουση του με την ανεμογεννήτρια. Επομένως, οι στιγμιαίες αλλαγές πίεσης

στην ατμόσφαιρα είναι ένα από τα κυριότερα προβλήματα που πρέπει να αντιμετωπίσει ένα σύστημα ευστάθειας ενός drone.

Πιο συγκεκριμένα, η διαφοροποίηση της πίεσης της ατμόσφαιρας, ασκεί στην πτέρυγα μία ανοδική δύναμη **(lift)** η οποία είναι κάθετη στην κατεύθυνση του ανέμου **(wind flow)** και βάσει αυτού έτσι δημιουργείται αντίσταση στο άνεμο **(incident flow)**. Για την περιστροφή ενός πτερυγίου μίας ανεμογεννήτριας γύρω από τον άξονα, το **incident flow** είναι *το "αποτέλεσμα της γεωμετρικής πρόσθεσης της γραμμικής ταχύτητας του ανέμου και της περιστροφικής ταχύτητας"*, το οποίο αυξάνεται γραμμικά βάσει του μήκους της πτέρυγας. Με άλλα λόγια, η άνωση που ασκείται στην πτέρυγα μίας ανεμογεννήτριας δεν είναι μόνο το αποτέλεσμα της γραμμικής ταχύτητας του ανέμου, αλλά ως επι το πλείστο από την ίδια την περιστροφή της. Βέβαια, καθ' όλη τη διάρκεια της επιθεώρησης η πτέρυγα είναι στατική· παρ' όλα αυτά, λόγω της εκτροπής της άκρης που υπάρχει, η πτέρυγα μπορεί να κινηθεί περίπου 10% της συνολικής περιστροφής της υπό συνθήκες έντασης ανέμου 2 m/sec με 14 m/sec (οι οποίες είναι οι τυπικές συνθήκες εργασίας **O&M – Operation & Maintenance).** Ως αποτέλεσμα με βάση τα παραπάνω χαρακτηριστικά, η ευστάθεια ενός κατά τη διάρκεια πτήσης του μπορεί να επηρεαστεί σημαντικά και να προκαλέσει πιθανά προβλήματα στο σύστημα ελέγχου του.



Εικόνα 24: Η κατεύθυνση του ανέμου σε σχέση με την άνωση που προκαλείται *[57]*

Με τη χρήση του παραγωγικού συλλογισμού, η αιολική δύναμη επηρεάζει αρνητικά όχι μόνο τη σταθερότητα ενός μη επανδρωμένου αεροσκάφους, αλλά και την συνολική του απόδοση. Οι ριπές ανέμου όπως και οι αναταράξεις κατά τη διάρκεια πτήσης ενός drone, είναι πραγματικές προκλήσεις για το σύστημα ελέγχου του. Επομένως, ένα από τα μεγαλύτερα προβλήματα που χρειάζονται λύση, είναι η ανάπτυξη συστήματος ελέγχου ικανού να αντιστέκεται επικίνδυνες και μη προβλέψιμες ριπές ανέμου.

3.3.2. Προκλήσεις & Λύσεις – Challenges & Solutions

Ένα από τα κύρια θέματα προς έρευνα, όσον αφορά την επίτευξη βέλτιστης πτήσης κάτω από ασυνήθιστες καιρικές συνθήκες, είναι η αυτόνομη πλοήγηση όπως και η ακρίβεια στον προσδιορισμό θέσης ενός drone κατά τη διάρκεια μίας αυτόνομης επιθεώρησης. Μερικές από τις κυριότερες προκλήσεις που απαιτούν και την περισσότερη προσοχή, αναλύονται παρακάτω.

Πλοήγηση από και προς την περιοχή της πτέρυγας

Κάτω από τις τυπικές συνθήκες επιθεώρησης για ένα UAV, οι οποίες προσδιορίζουν ως επιθυμητή ταχύτητα ανέμου τα 25 m/sec, το εκάστοτε drone θα πρέπει να διατηρεί την σταθερή πορεία του όπως και την ευστάθεια του, ανεξαρτήτως πιθανών διαταραχών. Κατά τη διαδικασία προσέγγισης της πτέρυγας, όταν η επιθεώρηση ξεκινάει, ή απομάκρυνσης από αυτή, όταν η διαδικασία έχει ολοκληρωθεί, το αεροσκάφος πρέπει να είναι τόσο γρήγορο στις κινήσεις του όσο και ακριβές έτσι ώστε να αυξάνει τη διάρκεια λειτουργίας της μπαταρίας του κατά τη διάρκεια της κύριας διαδικασίας. Οι προκλήσεις αυτού του μέρους είναι αρκετά σημαντικές.

Το πρώτο πρόβλημα που φέρει λύση, είναι η ***αντιμετώπιση του ανέμου*****.** Το εκάστοτε UAV πρέπει να μπορεί να φτάσει το στόχο, που στην προκειμένη περίπτωση είναι η πτέρυγα, σε πολύ κοντινή απόσταση (περίπου 1.5 μέτρο), μέσα σε ένα χρονικό όριο και με μεγάλη ακρίβεια σχετικά με τις συντεταγμένες που του έχουν δοθεί. Βάσει αυτού, αξίζει να σημειωθεί ότι στην αυτόνομη πλοήγηση ενός UAV η μόνη πληροφορία που λαμβάνει το drone από τον χρήστη, είναι οι συντεταγμένες της πτέρυγας της ανεμογεννήτριας όταν αυτό είναι προσγειωμένο **(initial state).** Έτσι, το drone πρέπει να έχει πρόσβαση είτε σε τεχνολογία **GPS** με σκοπό τον ακριβή εντοπισμό της αρχικής του θέσης, είτε σε ένα σύστημα από **laser beacons** τα οποία θα είναι εγκατεστημένα πάνω στην εκάστοτε πτέρυγα, για ακόμα καλύτερο αποτέλεσμα. Για την ***αντιμετώπιση του θέματος της πλοήγησης***, από και προς την πτέρυγα, το drone πρέπει να έχει ένα βέλτιστα ρυθμισμένο σύστημα ελέγχου **(fine-tuned control system)**, από την άποψη των PIDs του και των φίλτρων πρόβλεψης. Πιο συγκεκριμένα, τα κέρδη του κάθε PID στο σύστημα **(inner loop)** πρέπει να είναι ρυθμισμένα έτσι, ώστε το drone να είναι τόσο ευαίσθητο όσο χρειάζεται για να διατηρεί τη σταθερότητα του κατά τη διάρκεια πτήσης με τυπικές ή και πιο δύσκολες καιρικές συνθήκες. Παράλληλα, το σύστημα ελέγχου πρέπει να μπορεί να μειώνει την υπεραντιστάθμιση **(overcompensation)** και την ασταθή πτήση σε περίπτωση μικρών αναταραχών. Με άλλα λόγια, το σύστημα ελέγχου του θα πρέπει

να είναι τόσο ευαίσθητο έτσι ώστε να ανταποκρίνεται στις δυνατές ριπές ανέμου, αλλά και τόσο μη-ευαίσθητο έτσι ώστε να έχει σταθερή πτήση και στις μικρές αναταραχές. Εκτός από το control system, το drone πρέπει να είναι εξοπλισμένο από δυνατούς κινητήρες στις έλικες του, έτσι ώστε να κρατιέται σταθερό απέναντι σε στιγμιαίες ριπές ανέμου. Στις περισσότερες των περιπτώσεων η πλοήγηση σε εξωτερικό χώρο **(outdoor navigation)** απαιτεί μία αρχιτεκτονική από πολλαπλά επίπεδα συστήματος και αλγορίθμων ελέγχου **(layered control architecture)**, από τα οποία σχεδιάζεται ένας υψηλού επιπέδου αλγόριθμος ελέγχου που περιγράφει την κίνηση του αεροσκάφους, βάσει των κινηματικών διαφορικών εξισώσεων του συστήματος. Ακόμα, σημαντικό ρόλο στην ποιότητα πλοήγησης του drone παίζει το σχέδιο πτήσης, το οποίο επιτρέπει στο χρήστη να οργανώσει το πλάνο επιθεώρησης όπως και να καθορίζει συντεταγμένες σχετικά με την τοποθεσία των πτερύγων τις ανεμογεννήτριας. Ως αποτέλεσμα, μία εφαρμογή δημιουργίας πλάνου πτήσης **(flight planner)** σε συνεργασία με ένα καλά ρυθμισμένο σύστημα ελέγχου, το οποίο μπορεί να εντοπίζει τα προβλήματα μέσω φίλτρων πρόβλεψης (βλ. 2.4), είναι οι πιθανές λύσεις για μία ασφαλή και αποδοτική πτήση από και προς την ανεμογεννήτρια.

Πλοήγηση γύρω από το σημείο ενδιαφέροντος με αυξημένη ακρίβεια

Το επόμενο βήμα αφότου το drone έχει φτάσει στην ανεμογεννήτρια, είναι και το πιο σημαντικό. Η διαδικασία που πρέπει να εκτελέσει, είναι να ***κλειδώσει την θέση του σε κοντινή απόσταση από την πτέρυγα***, έτσι ώστε η κάμερα να μπορεί να εστιάσει σε καλό βαθμό και να εντοπίσει πιθανές βλάβες, όπως και ***να πλοηγείται γύρω από αυτήν σε κοντινή σταθερή απόσταση***. Όπως είναι λογικό, οι δυσκολίες στην εκτέλεση τέτοιων διαδικασιών είναι αρκετές.

Αρχικά, το drone πρέπει να είναι εξαιρετικά ακριβές στον προσδιορισμό θέσης του, λόγω της μικρής απόστασης από την πτέρυγα. Σε οποιαδήποτε άλλη περίπτωση, το drone είτε μπορεί να συντριβεί πάνω στην πτέρυγα, είτε να τραβήξει θολές φωτογραφίες· ως εκ τούτου, το drone πρέπει να διαθέτει ένα σύστημα από αισθητήρες και αλγόριθμους πρόβλεψης αρκετά ακριβές, έτσι ώστε να του παρέχει την απαραίτητη ασφάλεια. Στην παρούσα πρακτική εφαρμογή, όπως προαναφέρθηκε, το αεροσκάφος διαθέτει μόνο κάμερα μονού οφθαλμού, η οποία δεν αρκεί για να υπάρχει μεγάλη ακρίβεια θέσης. Ο ιδανικότερος συνδυασμός σε αισθητήρες, είναι να υπάρχει μία **κάμερα** έτσι ώστε να εφαρμόζεται απλά το SLAM [19] και να είναι εξοπλισμένο, επίσης, με **αποστασιόμετρα laser** και **GPS** τα οποία μπορούν να αποδώσουν με πολύ μεγάλη ακρίβεια. Στην περίπτωση του συνδυασμού αυτού, ο αλγόριθμος **EKF** [17] είναι ο ιδανικός για την πρόβλεψη της τοποθεσίας του drone όπως και για το

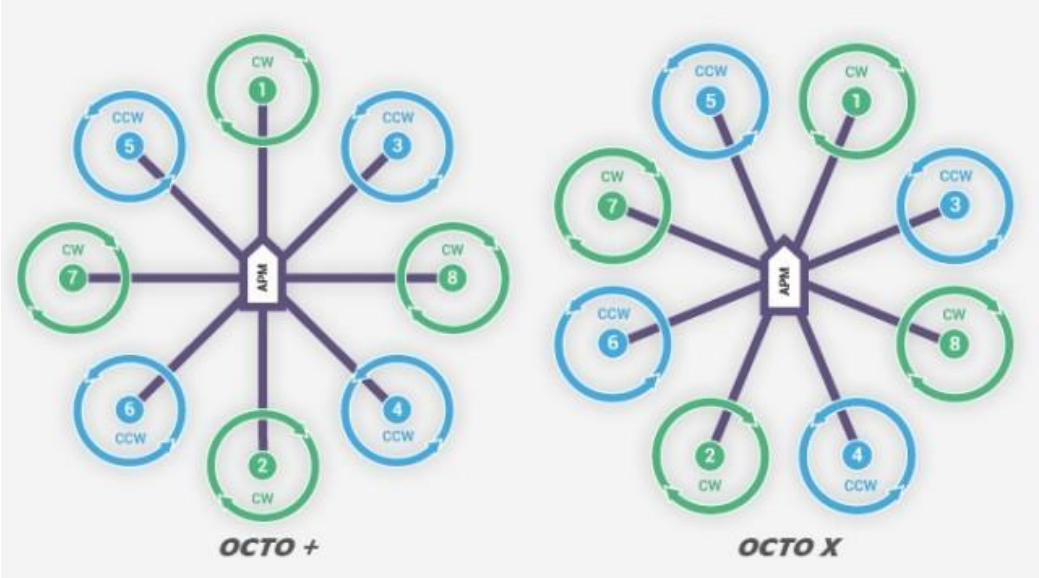
σύστημα ελέγχου, αφού δέχεται σαν εισόδους τις τιμές ανάδρασης των αισθητήρων. Επίσης, υποστηρίζοντας το σύστημα μέσω του λειτουργικού συστήματος **ROS** [3] και χρησιμοποιώντας πακέτα αναγνώρισης αντικειμένων μέσω κάμερας, όπως είναι το **tum_ardrone** [4] που χρησιμοποιεί τη μέθοδο **PTAM** [47], το αεροσκάφος θα έχει τη δυνατότητα να εντοπίσει γραμμές με απώτερο σκοπό τον εντοπισμό της απόστασης του από την πτέρυγα, αλλά και τη δημιουργία real-time χάρτη του περιβάλλοντος. Ακόμα, χρησιμοποιώντας αλγόριθμους feature-based (βλ. 2.5.1) το drone θα είναι ικανό να εντοπίζει τα αντικείμενα γύρω του (φτερωτή, ρότορα, κεντρικό άξονα, κ.α.) με στόχο την αποφυγή τους, ή και την κίνηση γύρω από αυτά. Τέλος, ένα ακόμα μέτρο ασφάλειας για την προφύλαξη του UAV, είναι η εφαρμογή μίας μεθόδου επείγουσας κατάστασης **(emergency mode)** κατά την οποία όταν ενεργοποιείται, το αεροσκάφος θα πρέπει να επιστρέφει στη βάση του ασχέτως άλλων ενεργειών.

Στιγμιαία Απόκριση του UAV σε Ριπές Ανέμου και Αναταραχές

Ένα από τα σημαντικότερα θέματα στην επιθεώρηση ανεμογεννητριών με τη χρήση αυτόνομων UAVs, είναι ο άνεμος. Ως αποτέλεσμα, το πιο δυνατό σημείο στο σύστημα ενός τέτοιου drone πρέπει να είναι η αντιμετώπιση στιγμιαίων και δυνατών ανέμων. Οι λύσεις στο θέμα αυτό διαφέρουν είτε σε φύση λογισμικού **(software)**, είτε σε υλικού **(hardware)**.

Γενικώς, είναι σημαντικό το εκάστοτε αεροσκάφος να διαθέτει, όπως προαναφέρθηκε δυνατούς κινητήρες, έτσι ώστε να είναι ικανό να πιάνει μεγάλες ταχύτητες, και ιδίως ταχύτητα κίνησης με κλίση **(pitch velocity)**. Επομένως, μία πιθανή λύση στην αντιμετώπιση ριπών ανέμου, είναι ο σχεδιασμός ενός drone που θα διαθέτει **pitch speed** διπλάσια από την επιτρεπόμενη ταχύτητα ανέμου. Με αυτόν τον τρόπο, το UAV θα είναι πιο ικανό σε διαφόρων ειδών ελιγμούς αλλά και στην ευθεία κίνηση του. Βέβαια, κυριότερο ρόλο στην απόδοση της πλοήγησης έχει η **αεροδυναμική σχεδίαση** του εκάστοτε UAV. Αναλυτικότερα, ακόμα ένα πρόβλημα αποτελεί η αστάθεια στην πτήση που δημιουργείται λόγω μικρών μεταβολών στον άνεμο· φαινόμενο το οποίο στο επιστημονικό χώρο της αεροδυναμικής ονομάζεται **“stall”**. Όταν ένα drone επιδέχεται τέτοια διαταραχή **(stalling)** γίνεται αρκετά ασταθές στην πτήση του, με αποτέλεσμα το σύστημα ελέγχου να εξαναγκάζει την αλλαγή των παραμέτρων του PID για μεγαλύτερη ευαισθησία. Όπως φαίνεται, ένα έξυπνο σύστημα ελέγχου μπορεί να αντιμετωπίσει αποτελεσματικά το φαινόμενο **stall**, μόνο που στην προκειμένη περίπτωση οι στιγμιαίοι άνεμοι που προκαλούν τη διαταραχή αυτή στο drone, είναι απρόβλεπτοι και ελάχιστης διάρκειας. Ως αποτέλεσμα, το αεροσκάφος ναι μεν θα γίνει πιο ευαίσθητο για να

αντιμετωπίσει την πιθανή ριπή ανέμου, αλλά την επόμενη χρονική στιγμή το drone θα είναι υπερβολικά ευαίσθητο και ασταθές, λόγω των λανθασμένων (για εκείνη τη χρονική στιγμή) παραμέτρων του PID. Το λόγω στην σωστή και αποτελεσματική καταπολέμηση του **stall**, είναι ο αεροδυναμικός σχεδιασμός του drone με σκοπό την αυξημένη δύναμη στην άνωση. Επίσης, σημαντικό ρόλο έχει και το μέγεθος του, όπως και το βάρος ενός UAV. Οι μακρύτερες αεροδυναμικές βάσεις, βελτιώνουν την ευστάθεια στην πτήση όπως και την σταθερότητα απέναντι σε ριπές ανέμων. Ένα πλαίσιο ενός drone το οποίο έχει το κέντρο βάρους του τοποθετημένο στο κέντρο, μπορεί να αποδώσει εξαιρετικά καλύτερα στην αντίσταση δυνατών ανέμων. Πιο αναλυτικά, ένα πολυκόπτερο έχει τις διαβαθμίσεις των τεσσάρων ελίκων, έξι ελίκων και 8+ ελίκων. Είναι κοινώς γνωστό, ότι όσους περισσότερους ρότορες, τόση περισσότερη άνωση μπορεί να παράξει ένα UAV. Τα πλαίσια που μπορούν να αποδώσουν καλύτερα απέναντι σε ριπές ανέμων και αναταράξεις, ονομάζονται **Octo X** και **Octo +** και φαίνονται στην Εικόνα.



Εικόνα 25: Πλαίσια Οκτακόπερου σε σχήμα "X" και "+" *[55]*

Στην παρούσα πτυχιακή εργασία, το **Parrot AR Drone 2.0.** [7] που χρησιμοποιήθηκε για την προσομοίωση της εφαρμογής έχει πλαίσιο σχήματος **X4 (Quadcopter)** και βάρος 420g, καθιστώντας το αρκετά ελαφρύ και αδύναμο για την αντιμετώπιση μεγάλων ριπών ανέμου. Βέβαια, σε δοκιμές με ριπές ανέμου ανάλογες με το μέγεθος του, το ardrone μπορούσε να ανταποκριθεί αποτελεσματικά διατηρώντας τη θέση του αλλά και την πορεία πτήσης του.

3.4. Περιγραφή Υλοποίησης της Εφαρμογής

3.4.1. Ανάλυση Πακέτων

Πακέτο ardrone autonomy

Το πιο βασικό open-source πακέτο που χρησιμοποιήθηκε στην εφαρμογή της παρούσας πτυχιακής εργασίας, ονομάζεται **ardrone_autonomy** [5] [56] και αποτελεί τη βάση για την σύνδεση του λογισμικού **ROS** [3] με το **Parrot AR Drone 2.0** [7]. Με άλλα λόγια, περιέχει nodes ειδικά για την επικοινωνία του ηλεκτρονικού υπολογιστή με το ardrone. Το συγκεκριμένο πακέτο, λόγω του ότι είναι ανοιχτού κώδικα, διατίθεται σε διαδικτυακή μορφή και γίνεται εύκολα εγκατάσταση στον ηλεκτρονικό υπολογιστή και το ROS, ακολουθώντας τις οδηγίες που παρέχουν οι δημιουργοί [56].

Πιο αναλυτικά, το πακέτο αυτό είναι ανεπτυγμένο σε προγραμματιστικό περιβάλλον **C++** και έχει ως στόχο τη διευκόλυνση στην επικοινωνία μεταξύ του συστήματος και του ardrone. Αποτελείται από αρχεία με πολλαπλές λειτουργίες, τα οποία είναι κατηγοριοποιημένα αναλόγως με τη συνολική τους λειτουργία. Όπως προαναφέρθηκε και στην ενότητα (2.2.2), ένα πακέτο ROS απαρτίζεται από συγκεκριμένους φακέλους και αρχεία, τα οποία αναγνωρίζονται από το σύστημα και με τον τρόπο αυτό γίνεται σωστή κατανομή ισχύς. Το συγκεκριμένο πακέτο απαρτίζεται από τους εξής σημαντικούς φακέλους:

- ***data/camera_info*** → Ο φάκελος αυτός εμπεριέχει αρχεία για την βαθμονόμηση της κάμερας του drone. Πιο συγκεκριμένα, περιέχει δείγματα τα οποία αυτοματοποιούν τη διαδικασία βαθμονόμησης, κάθε φορά που εκτελείται το σύστημα
- ***include/ardrone_autonomy*** → Στο συγκεκριμένο φάκελο βρίσκονται τα **header files** του λογισμικού προγράμματος, τα οποία καθορίζουν τις ιδιότητες των κλάσεων και συναρτήσεων του πηγαίου κώδικα.
- ***launch*** → Ο φάκελος launch περιέχει **launch files**, ή αλλιώς εκτελέσιμα αρχεία, τα οποία εμπεριέχουν εντολές εκκίνησης/τερματισμού κάποιων nodes, όπως και αρχικοποιήσεις παραμέτρων. Στο συγκεκριμένο φάκελο, βρίσκεται το αρχείο που εκτελεί ο χρήστης, με όνομα **ardrone.launch**, έτσι ώστε να συνδεθεί με το ardrone.
- ***msg*** → Ο φάκελος αυτός, όπως έχει περιγραφεί και στην ενότητα (2.2.2), περιέχει τους τύπους μηνυμάτων που μεταδίδονται μεταξύ των διαφόρων διεργασιών. Το συγκεκριμένο πακέτο έχει 32 διαφορετικούς τύπους μηνυμάτων, οι οποίοι σχετίζονται με τα δεδομένα πλοήγησης του drone **(navigation data – navdata)**.

- ***src*** → Ο φάκελος αυτός αποτελεί και τον σημαντικότερο, αφού περιέχει αρχεία πηγαίου κώδικα τα οποία είναι ο πυρήνας του λογισμικού προγράμματος. Πιο συγκεκριμένα, περιέχει το αρχείο **ardriver.cpp** το οποίο είναι το κεντρικό αρχείο driver του συστήματος. Επίσης, περιέχει το αρχείο **ardrone_sdk.cpp** στο οποίο υλοποιούνται οι μέθοδοι που παρέχονται από τους κατασκευαστές του ardrone [7] και με τις οποίες επικοινωνεί το drone με το ROS. Τέλος, υπάρχουν και τα αρχεία **teleop_twist.cpp** και **video.cpp** που είναι υπεύθυνα για τη λειτουργία της κάμερας του drone.
- ***srv*** → Στο φάκελο αυτό, εμπεριέχονται τα αρχεία **services** τα οποία αλληλοεπιδρούν με το drone άμεσα σε κάποια λειτουργία του (π.χ. επιλογή κάμερας).

Βάσει όλων αυτών των φακέλων, το **ardrone_autonomy** αποτελεί το πλέον σημαντικό πακέτο για τη λειτουργία του **Parrot AR Drone** με το **ROS**· γι' αυτό και χρησιμοποιήθηκε σαν βασικό driver της πρακτικής εφαρμογής, το οποίο παρείχε διαρκώς χρήσιμα δεδομένα για την πλοήγηση του ardrone που κατέληγαν σαν εισόδους στις συναρτήσεις του πηγαίου κώδικα.

Πακέτο tum ardrone

Το δεύτερο πακέτο που χρησιμοποιήθηκε για την ανάπτυξη της πρακτικής εφαρμογής είναι το **tum_ardrone** [4] [16], στο οποίο γίνεται υλοποίηση του αλγορίθμου **PTAM** [47] όντας βασισμένο στο **ardrone**. Με άλλα λόγια, ο αλγόριθμος PTAM χρησιμοποιείται έτσι ώστε το ardrone να μπορεί μέσω της κάμερας του, να εντοπίσει τη θέση του και να πλοηγηθεί στο χώρο. Ακόμα, παρέχει στο σύστημα πληροφορίες για το εκάστοτε περιβάλλον του drone, αλλά και τρισδιάστο χάρτη σημείων αυτού μέσα από ένα γραφικό περιβάλλον.

Η ανάπτυξη αυτού του πακέτου έχει γίνει σε προγραμματιστικό περιβάλλον **C++** και περιέχει συναρτήσεις και μεθόδους για την αποτελεσματική εφαρμογή του PTAM, αλλά και την πλοήγηση του drone σε εσωτερικούς χώρους. Στην περίπτωση αυτού του πακέτου, ο σημαντικότερος φάκελος που περιέχει τις κύριες συναρτήσεις λειτουργίας, είναι ο ***src***. Μέσα σε αυτόν, υπάρχουν υποφάκελοι με τα ονόματα **UINode**, **autopilot** και **stateestimation**, οι οποίοι εκτελούν διαφορετικές διεργασίες κατά την εκτέλεση του προγράμματος.

- **UINode** → Το UINode προκύπτει από το **User Interface Node**, το οποίο είναι υπεύθυνο για το γραφικό περιβάλλον του προγράμματος. Στον φάκελο αυτό εμπεριέχονται **header files** αλλά και **implementation files**, με λειτουργίες όπως τον

σχεδιασμό και την λειτουργικότητα του γραφικού περιβάλλοντος, ή ακόμα και την επικοινωνία αυτού με το ROS και το ardrone.

- **autopilot →** Ο "αυτόματος πιλότος", όπως ονομάζεται αυτός ο υποφάκελος, είναι ένα σύνολο από nodes και λειτουργίες τα οποία είναι υπεύθυνα για την αυτόνομη πλοήγηση του ardrone με βάση συντεταγμένες ορισμένες από το χρήση, μέσω του UI. Τα πιο σημαντικά αρχεία, είναι το **ControlNode.cpp**, το **DroneController.cpp** αλλά και ο φάκελος **KI.** Στον φάκελο **KI**, βρίσκονται τα αρχεία που περιέχουν τις βασικές συναρτήσεις για την λειτουργία και πτήση του ardrone. Αρχεία τα οποία στέλνουν εντολές στο ardrone, για πτήση προς κάποιο σημείο στο χάρτη, γύρω από κάποιο σημείο ή και απλά για απογείωση και προσγείωση. Το αρχείο **ControlNode.cpp** από την άλλη, είναι υπεύθυνο για την λήψη δεδομένων μέσα από διάφορα topics, με στόχο την αλληλοσύνδεση των εντολών με τις μεθόδους που εμπεριέχονται στο **KI**. Με άλλα λόγια, καθορίζει και τις κινήσεις του drone δημοσιεύοντας σε publishers τις αντίστοιχες εντολές, έτσι ώστε να ενεργοποιούνται οι μέθοδοι κίνησης και αλληλεπίδρασης με το ardrone. Τέλος, στο αρχείο **DroneController.cpp** εμπεριέχεται ο πηγαίος κώδικας ο οποίος ρυθμίζει αναλόγως τις παραμέτρους των PIDs αλλά και του αλγόριθμου EKF.
- **stateestimation →** Στο φάκελο αυτό, εμπεριέχονται οι λειτουργίες που συνθέτουν τους αλγόριθμους **PTAM** και **EKF**, αλλά και τον σχεδιασμό και τη λειτουργία της χαρτογράφησης **(mapping)**. Η ονομασία του, προέρχεται από το **State Estimation** που, όπως έχει προαναφερθεί, είναι η λειτουργία εκτίμησης της παρούσας κατάστασης σε ένα αυτόνομο σύστημα. Πιο συγκεκριμένα, στον υποφάκελο **PTAM** βρίσκονται όλα τα αρχεία **(header files** και **implementation files)** που εφαρμόζουν τον αλγόριθμο PTAM κατά τη διάρκεια εκτέλεσης μίας αυτόνομης πλοήγησης. Ακόμα, αρχεία όπως το **DroneKalmanFilter.cpp** και το **EstimationNode.cpp**, είναι αυτά που προβλέπουν την τοποθεσία του drone κατά τη διάρκεια πτήσης του, άλλα και εφαρμόζουν τον αλγόριθμο EKF. Τέλος, αρχεία όπως το **PTAMWrapper.cpp** και το **MapView.cpp**, είναι υπεύθυνα για την εφαρμογή του mapping αλλά και την γραφική απεικόνιση αυτού.

Ως αποτέλεσμα, το πακέτο **tum_ardrone** αποτέλεσε σημαντικό κομμάτι της πρακτικής εφαρμογής του drone, αφού παρείχε αρκετά δεδομένα και αρκετές μεθόδους για την βέλτιστη πλοήγηση του ardrone. Τέλος, αξίζει να σημειωθεί ότι τα nodes που χρησιμοποιήθηκαν κατά κόρον, ήταν το **autopilot** και το **stateestimation**.

3.4.2. Επεξήγηση Πηγαίου Κώδικα Πλοήγησης

Ο πηγαίος κώδικας της πρακτικής εφαρμογής, αναπτύχθηκε σε προγραμματιστικό περιβάλλον γλώσσας **Python** και εφαρμόζει μία κυκλική κίνηση προσομοίωσης γύρω από ένα αντικείμενο, σε προκαθορισμένες συντεταγμένες. Όπως έχει προαναφερθεί, ο κώδικας αυτός έχει βασιστεί στα open-source πακέτα **tum_ardrone** [4] και **ardrone_autonomy** [5] και εκτελεί μία επιπρόσθετη λειτουργία πτήσης. Στο σημείο αυτό, αξίζει να περιγραφούν οι πιο βασικοί μέθοδοι **(functions)** που απαρτίζουν τον πηγαίο κώδικα, όπως φαίνεται παρακάτω.

Οι συναρτήσεις του πηγαίου κώδικα εμπεριέχονται μέσα στην κεντρική κλάση **DroneNavigation**, ενώ πριν από τον ορισμό της γίνονται οι απαραίτητες αρχικοποιήσεις και εισαγωγές.

Μέθοδος: __init__()

Στη πρώτη μέθοδο του πηγαίου κώδικα, γίνεται αρχικοποίηση των *publisher* και των *subscriber* που χρησιμοποιούνται στην εκτέλεση της εφαρμογής, αλλά και κάποιων σημαντικών τιμών.

```
def __init__( self ):

        print "initialized"

        self.land_pub = rospy.Publisher( "ardrone/land", Empty, queue_size=10 )
        self.takeoff_pub = rospy.Publisher( "ardrone/takeoff", Empty, queue_size=10 )
        self.reset_pub = rospy.Publisher( "ardrone/reset", Empty, queue_size=10 )

        self.command_pub = rospy.Publisher( "/tum_ardrone/com", String, queue_size=10 )

        self.predictedPose = rospy.Subscriber( "/ardrone/predictedPose", filter_state,
self.position_update )

        self.pose = Point( 0, 0, 0 )
        self.pose_yaw = 0.0

        self.target = Point( 0, 0, 0 )
        self.target_yaw = 0.0

        self.target_found = 0

        self.current_cmd = Twist()

        self.speed = 0.3
        self.yaw_speed = 0.3


        self.main_logic()
```

Οι *publishers* και οι *subscribers* συνδέονται με τα άλλα πακέτα, έτσι ώστε να δέχονται και να στέλνουν δεδομένα για την πλοήγηση του drone. Πιο συγκεκριμένα, οι πρώτοι τρείς *publishers* **(land_pub**, **takeoff_pub**, **reset_pub)** είναι υπεύθυνοι για την προσγείωση, απογείωση και

επανεκκίνηση του drone, αντίστοιχα. Ακόμα, ο *publisher* **command_pub** και ο *subscriber* **predictedPose**, είναι αυτοί που στέλνουν εντολές στο σύστημα του **tum_ardrone** και λαμβάνουν μετρήσεις θέσης από το αυτό.

Μέθοδοι Βασικών Εντολών

Οι μέθοδοι αυτοί, αφορούν την προσγείωση, απογείωση, επανεκκίνηση και το hover του αεροσκάφους.

```
def takeoff( self ):
        self.takeoff_pub.publish( Empty() )

def land( self ):
        self.land_pub.publish( Empty() )

def reset( self ):
        self.reset_pub.publish( Empty() )

def hover( self ):
        self.command_pub.publish( "c goto 0 0 0 0" )
```

Ουσιαστικά, οι μέθοδοι αυτοί είναι εντολές οι οποίες στέλνονται μέσω των *publishers* που αναφέρθηκαν πιο πάνω, στο σύστημα του **tum_ardrone**.

Μέθοδος: relative_coordinates()

Η μέθοδος αυτή, είναι υπεύθυνη για τον υπολογισμό και την ενημέρωση των συντεταγμένων του στόχου. Η τεχνική που ακολουθεί είναι να περιστρέφει τις συντεταγμένες στις οποίες βρίσκεται ο στόχος του drone, έτσι ώστε να προσανατολίζονται σωστά με αυτές του drone.

```
def relative_coordinates( self ):

        ## Coordinate Z
        rel_z = self.target.z - self.pose.z

        ## Coordinate Yaw
        rel_yaw = self.target_yaw - self.pose_yaw
        if rel_yaw > 180:
                rel_yaw -= 360
        if rel_yaw < -180:
                rel_yaw += 360

        ## Coordinate X, Y
        rel_x = self.target.x - self.pose.x
        rel_y = self.target.y - self.pose.y

        ## Rotating dest_vector (difference between reference and pose yaw)
        theta = np.deg2rad( self.pose_yaw )
        rotated_x = ( rel_x * np.cos( theta ) ) - ( rel_y * np.sin(theta ) )
        rotated_y = ( -rel_x * np.sin( theta ) ) - ( rel_y * np.cos(theta ) )

        return ( rotated_x,rotated_y,rel_z,rel_yaw )
```

Με άλλα λόγια, η μέθοδος αυτή δημιουργεί ένα κοινό σύστημα συντεταγμένων μεταξύ της τοποθεσίας του στόχου, στον οποίο το drone πρέπει να φτάσει, όπως και της εκάστοτε τοποθεσίας του drone. Ο τρόπος με τον οποίο γίνεται αυτό, είναι η σύγκριση των δύο συντεταγμένων ανά άξονα και ο υπολογισμός της διαφοράς τους. Για τον άξονα του **yaw**, βέβαια, ο υπολογισμός γίνεται αναλόγως με την γωνία που βρίσκεται το drone εκείνη τη χρονική στιγμή. Ακόμα, υπολογίζεται η γωνία **theta** που συμβολίζει τη σχετική γωνιακή διαφορά των δύο συντεταγμένων, και χρησιμοποιείται για τον υπολογισμό των τελικών γενικών συντεταγμένων του στόχου. Τέλος, μέσω του μαθηματικής τριγωνομετρικής συνάρτησης, που φαίνεται στην παράθεση του κώδικα, μετατρέπονται οι τελικές συντεταγμένες, και καταχωρούνται στις τιμές **rotated_x**, **rotated_y**, **rel_yaw**, **rel_z**.

Μέθοδος: distance_to_target()

Στη μέθοδο αυτή, γίνεται ο υπολογισμός των σημείων στο χάρτη στα οποία το drone πρέπει να πετάξει, έτσι ώστε να εκτελέσει μία βαθμωτή κίνηση πολυγώνου.

```
def distance_to_target( self ):

        relative = Point ( 0, 0, 0 )

        relative.x = self.target.x - self.pose.x
        relative.y = self.target.y - self.pose.y
        relative.z = self.target.z - self.pose.z
        dist_array = np.array( ( relative.x, relative.y, relative.z ) )

        return np.sqrt( np.sum( dist_array**2 ) )
```

Πιο συγκεκριμένα, γίνεται εφαρμογή του **Πυθαγόρειου Θεωρήματος** σε κάθε μία από τις συντεταγμένες ***x, y*** και ***z***, και με τον τρόπο αυτό υπολογίζεται κάθε φορά το επόμενο σύνολο συντεταγμένων στο οποίο θα πρέπει να πετάξει το drone. Σημαντική λεπτομέρεια στην μέθοδο αυτή, είναι ότι η τελικές συντεταγμένες καταχωρούνται σε μία ειδική μεταβλητή **Point**, η οποία αναγνωρίζεται από το σύστημα του **tum_ardrone** κατευθείαν.

Μέθοδος: angle_to_target()

Παρόμοια με τη αμέσως προηγούμενη μέθοδο, η ***angle_to_target*** υπολογίζει τη γωνία **(yaw)** που θα πρέπει να περιστραφεί το drone στην επόμενη θέση.

```
def angle_to_target( self ):

        angle = np.absolute( self.pose_yaw - self.target_yaw )
        if angle > 180:
                angle = np.absolute( angle - 360 )

        return angle
```

Ο υπολογισμός της γωνίας είναι απλός, αφού βρίσκει την απόλυτη διαφορά μεταξύ της παρούσας θέσης του drone και της θέσης στόχου και την καταχωρεί στην τιμή **angle**, αφού ελέγξει την γωνία στο αν ξεπερνάει τις 180 μοίρες (έτσι ώστε να μην κάνει περιττούς κύκλους).

Μέθοδος: position_update()

Η μέθοδος ***position_update*** είναι μία από τις πιο σημαντικές μεθόδους του κώδικα, αφού υπολογίζει την εκάστοτε θέση του drone κάθε χρονική στιγμή.

```
def position_update( self, data ):

        x = float( data.x )
        y = float( data.y )
        z = float( data.z )
        self.pose_yaw = float( data.yaw )
        self.pose = Point ( x, y, z )
```

Η συνάρτηση δέχεται δεδομένα από το **topic** με όνομα *"/ardrone/predictedPose"*, που εκπέμπεται από το πακέτο **tum_ardrone**, και καταχωρεί τις τιμές του κάθε άξονα σε διαφορετικές μεταβλητές, οι οποίες στο τέλος καταλήγουν σε μία μεταβλητή **Point**.

Μέθοδος: moveto()

Η σημαντικότερη μέθοδος για την κίνηση του drone στο χώρο, είναι η ***moveto***. Μέσω εντολών που καταχωρούνται σε έναν *publisher*, η συνάρτηση ωθεί το drone στο να προχωρήσει σε συγκεκριμένες συντεταγμένες, αφού παράλληλα παρουσιάζεται σε real-time η εξέλιξη κίνησης του.

```
def moveto( self, x, y, z, yaw, range ):

        self.target = Point( x,y,z )
        self.target_yaw = yaw
        print self.target

        self.command_pub.publish( "c clearCommands" )

        self.command_pub.publish( "c goto %f %f %f %f"%( self.target.x,self.target.y,self.target.z,
self.target_yaw ) )

        while ( self.distance_to_target() > range or self.angle_to_target() > 6 ):
                print "\nWithin distance %f"%( self.distance_to_target() )
                print " I am at: x:%f , y:%f , z:%f ,yaw:%f"%( self.pose.x, self.pose.y, self.pose.z,
self.pose_yaw )
                print "Going to: x:%f , y:%f , z:%f ,yaw:%f\n"%(
self.target.x,self.target.y,self.target.z, self.target_yaw )
                rospy.sleep(0.1)

        return True
```

Πιο συγκεκριμένα, η συνάρτηση δέχεται σαν δεδομένα τις συντεταγμένες στόχου του drone και εκπέμπει μέσω του **command_pub** τις εντολές *"c clearCommands"* και *"c goto"*· οι οποίες επικοινωνούν με το πακέτο **tum_ardrone** και κινούν το drone στα επιθυμητά σημεία. Επίσης, μέσα στη συνθήκη **while()** το πρόγραμμα εκτυπώνει στην οθόνη του χρήστη την υπολειπόμενη απόσταση του drone από τον στόχο **(remaining distance)**, την εκάστοτε θέση

του **(current position)** και την θέση του στόχου **(destination position)** (για λόγους περισσότερο debugging).

Μέθοδος: arc_coordinates()

Η παρούσα μέθοδος αποτελεί κύρια συνάρτηση για τον εντοπισμό των κινήσεων του drone με βάση τον αριθμών γωνιών στο πολύγωνο που πρέπει να εκτελέσει.

```
def arc_coordinates( self, distance, angle ):

        arc_length = 2 * np.sin(angle / 2) * distance
        phi = (np.pi - angle) / 2
        new_x = arc_length * np.sin(phi)
        new_y = arc_length * np.cos(phi)
        move_coordinates = (new_x,new_y)

        return move_coordinates
```

Πιο συγκεκριμένα, η μέθοδος αυτή μετατρέπει την απόσταση στο τόξο που δημιουργείται ανάμεσα στην παρούσα τοποθεσία του drone και τις συντεταγμένες στόχου, και στον αριθμό κινήσεων που πρέπει να εκτελέσει το drone, και επιστρέφει ως αποτέλεσμα τις συντεταγμένες των σημείων που θα πρέπει να κινηθεί το drone.

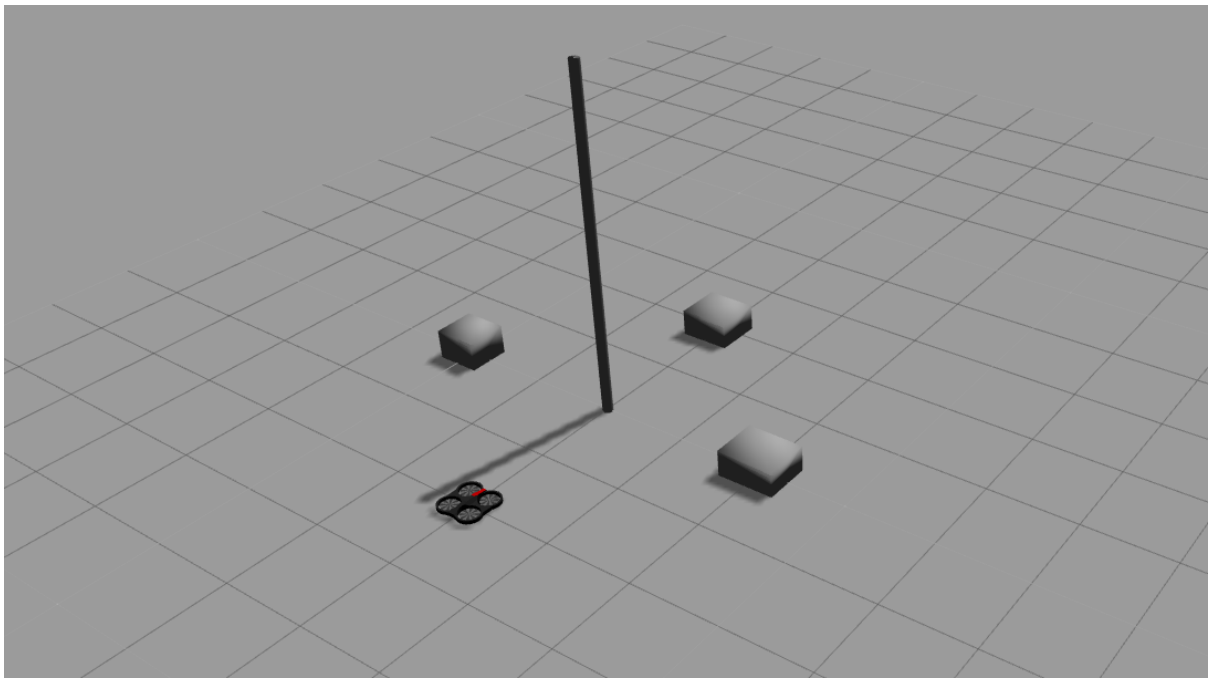
Μέθοδος: main_logic()

Αποτελεί τη βασική μέθοδο, στην οποία εκτελούνται οι εντολές που ωθούν το drone στο να κάνει την προκαθορισμένη αλληλουχία κινήσεων. Η μέθοδος ξεκινάει με τον καθορισμό των συντεταγμένων που βρίσκεται ο στόχος, και στο οποίο το drone θα πρέπει να χαράξει πολυγωνική τροχιά· αναλόγως με το πόσες γωνίες αλλά και την απόσταση από το αντικείμενο που του έχει ορίσει ο χρήστης. Έπειτα, το drone δέχεται την εντολή απογείωσης αλλά και αναμονής, μέχρι να εκτελεστεί πλήρως ο αλγόριθμος PTAM από το πακέτο **tum_ardrone.** Αμέσως μετά, το drone ελέγχει αν έχουν οριστεί σωστά οι συντεταγμένες του στόχου και ξεκινάει τη διαδικασία προσέγγισης του πρώτου σημείου. Τέλος, το drone αφού ολοκληρώσει όλες τις γραμμικές αλλά και περιστροφικές κινήσεις, επιστρέφει στο σημείο εκκίνησης όπου και προσγειώνεται, ολοκληρώνοντας την διαδικασία επιθεώρησης.

3.5. Αποτελέσματα Εφαρμογής

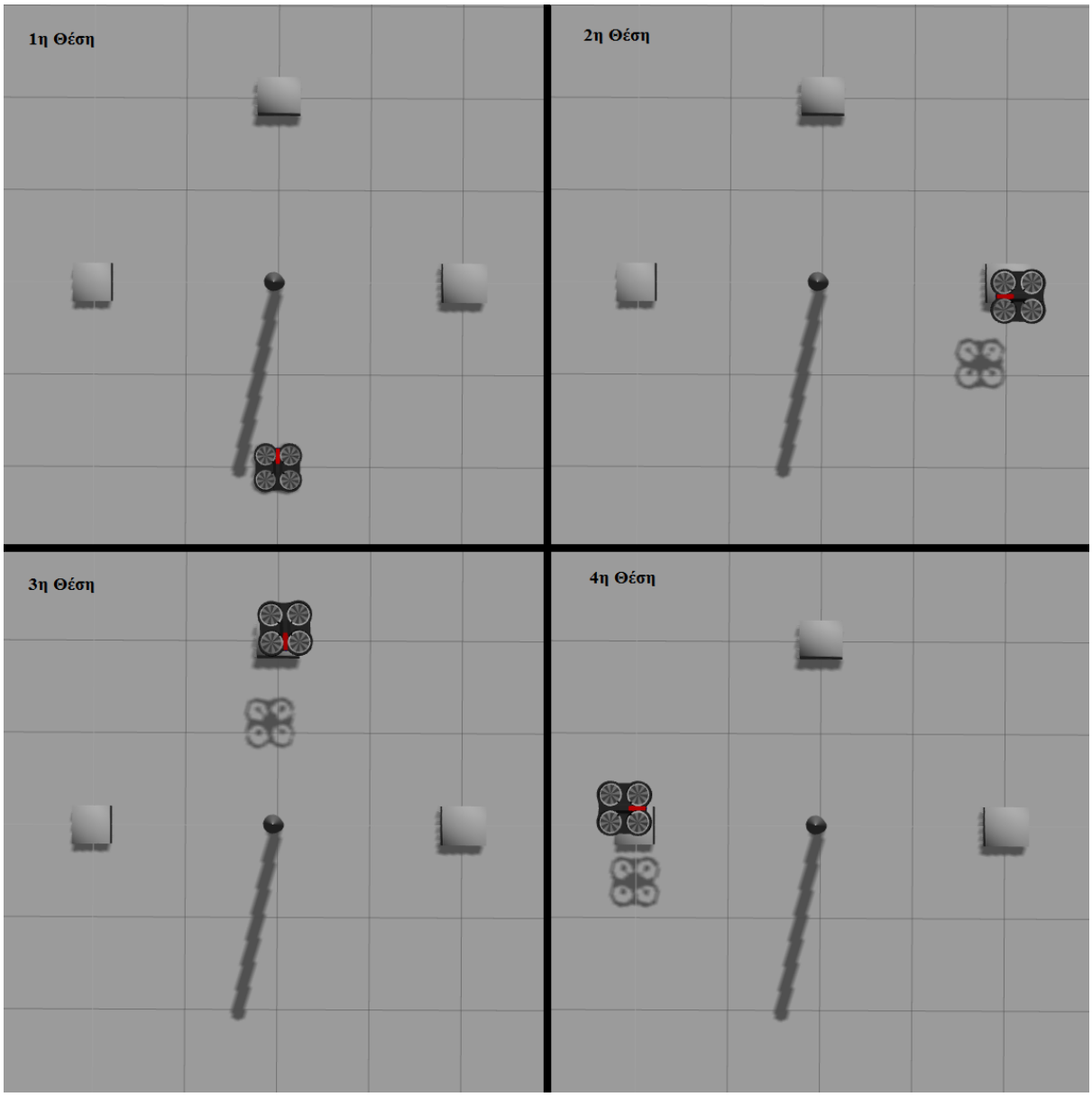
Το αποτέλεσμα της εφαρμογής αυτής, ήταν αρχικά η σύνδεση ενός πηγαίου κώδικα με προϋπάρχοντα πακέτα αυτόνομης πλοήγησης, η εφαρμογή πλήρως αυτόνομης κίνησης ενός drone για την επιθεώρηση ενός αντικειμένου, αλλά και η προσομοίωση αυτού στο λογισμικό

Gazebo. Οι παρακάτω εικόνες παρουσιάζουν στιγμιότυπα από την εκτέλεση της παρούσας εφαρμογής στο λογισμικό προσομοίωσης Gazebo.



Εικόνα 26: Αρχική θέση κατά την εκτέλεση της εφαρμογής στον προσομοιωτή Gazebo

Στην *Εικόνα 26* απεικονίζεται το drone στην αρχική του θέση από πλάγια γωνία θέασης, καθώς έχει ξεκινήσει η εκτέλεση του του **Gazebo** [6] αλλά και των πακέτων **tum_ardrone** [4] και **ardrone_autonomy** [5]. Ακόμα, στην *Εικόνα 27* φαίνεται ένα χρονοδιάγραμμα **(timeframe)** κατά τη διάρκεια της εκτέλεσης της πρακτικής εφαρμογής, στο οποίο παρουσιάζεται το drone σε κάθε μία από τις τέσσερις θέσης που έπρεπε να πάρει, όπως του είχε προκαθοριστεί, έτσι ώστε να κάνει μία πλήρη επιθεώρηση γύρω από το αντικείμενο.



Εικόνα 27: Εικόνα χρονοδιάγραμμα μίας κίνησης τεσσάρων σημείων στο Gazebo

4. Παρατηρήσεις & Συμπεράσματα

4.1. Συμπεράσματα

Τα συμπεράσματα που προκύπτουν από τη συγκεκριμένη πτυχιακή εργασία, όπως και οι παρατηρήσεις, ποικίλουν. Για την ανάπτυξη μίας τόσο προηγμένης εφαρμογής, για αυτόνομη επιθεώρηση σε πτέρυγες ανεμογεννητριών με τη χρήση UAVs, η έρευνα που απαιτείται είναι πέρα του ενός χρόνου, όπως και πέρα του ενός ατόμου. Οι δυσκολίες που περιγράφηκαν στα προηγούμενα κεφάλαια, αλλά και οι πιθανές λύσεις για αυτές, είναι μία μικρή ένδειξη για το ξεκίνημα έρευνας ενός τόσο σοβαρού τεχνολογικού θέματος. Επίσης, η θεωρητική προσέγγιση που έγινε κατά τη διάρκεια της παρούσας πτυχιακής εργασίας, αποτελεί ένα βασικό μέρος για μία ολοκληρωμένη προσέγγιση όσον αφορά το θέμα της αυτόνομης πλοήγησης ενός μη επανδρωμένου αεροσκάφους. Γενικώς, η αυτόνομη πλοήγηση προσφέρεται για έρευνα και ανάπτυξη εφαρμογών, λόγω της σχετικά πρόσφατης ανάπτυξης και ανάδειξης της στον ευρύ επιστημονικό χώρο. Ως αποτέλεσμα, η παρούσα πτυχιακή εργασία προσέφερε εξαιρετική θεωρητική γνώση πάνω στο αντικείμενο των αυτόνομων μη επανδρωμένων οχημάτων, αλλά και πρακτική γνώση καθώς αναπτύχθηκα αποτελεσματικά μία εφαρμογή αυτόνομη πλοήγησης.

4.2. Μελλοντική Εργασία

Στην παρούσα πτυχιακή εργασία, αναλύθηκαν σε θεωρητικό επίπεδο τα βασικά χαρακτηριστικά και προβλήματα της αυτόνομης πλοήγησης, αλλά παρουσιάστηκε και μία πρακτική εφαρμογή πάνω στο ίδιο αντικείμενο.

Στο μέλλον, θα γίνει προσπάθεια έρευνας και ανάπτυξης μίας ολοκληρωμένης εφαρμογής με σκοπό την πλήρη αυτόνομη επιθεώρηση πτερύγων μίας ανεμογεννήτριας με τη χρήση μη επανδρωμένων αυτόνομων αεροσκαφών. Στόχος είναι η συγκρότηση μίας ομάδας από ερευνητές από διάφορα αξιόλογα πανεπιστήμια, με σκοπό την έρευνα και την ολοκληρωμένη ανάπτυξη της εφαρμογής, αλλά και την σύνδεση και συνεργασία των πανεπιστημίων. Μία τέτοια εφαρμογή, όχι μόνο θα μειώσει το ρίσκο τραυματισμού των εκάστοτε εναεριτών, αλλά θα προσφέρει μείωση στο χρόνο και τα κόστη επιθεώρησης και συντήρησης μίας ανεμογεννήτριας για μία εταιρεία. Τέλος, η έρευνα που θα πραγματοποιηθεί θα συντελέσει σε μεγάλο κομμάτι στη βέλτιστη πτήση ενός UAV, όσον αφορά τον έλεγχο ενός μη επανδρωμένου αεροσκάφους αλλά και την αεροδυναμική του.

Βιβλιογραφικές Αναφορές

[1] «The UAV,» [Ηλεκτρονικό]. Available: http://www.theuav.com/. [Πρόσβαση 2016].

[2] BAA Training, «BAA Training,» 2015. [Ηλεκτρονικό]. Available: https://www.baatraining.com/uav-types-how-to-choose-yours/. [Πρόσβαση 2016].

[3] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler και A. Ng, «ROS: an open-source Robot Operating System,» σε *ICRA Workshop on Open Source Software*, CA, 2009.

[4] J. Engel, «Computer Vision Group,» Informatik IX, Computer Vision Group, [Ηλεκτρονικό]. Available: http://vision.in.tum.de/data/software/tum_ardrone.

[5] M. Monajjemi , «ardrone_autonomy,» Autonomy Lab, Simon Fraser University, [Ηλεκτρονικό]. Available: http://ardrone-autonomy.readthedocs.io/en/latest/index.html#.

[6] N. Koenig και A. Howard, «Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator,» σε *Proceedings 01 2004 IEEE!RSJ International Conference on Intelligent Robots and Systems*, Los Angeles, 2004.

[7] «Parrot,» Parrot SA, [Ηλεκτρονικό]. Available: http://www.parrot.com/uk/products/ardrone-2/.

[8] P.-J. Bristeau, F. Callou, D. Vissière και N. Petit, «The Navigation and Control technology inside the AR.Drone micro UAV,» Paris, FRANCE, 2011.

[9] «RobotShop,» RobotShop Inc., [Ηλεκτρονικό]. Available: http://www.robotshop.com/en/brushless-motor-kit-ardrone.html.

[10] «Wikipedia,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Abbe_error.

[11] B. D. Lucas και T. Kanade, «An iterative image registration technique with an application to stereo vision,» σε *Proceedings of Imaging Understanding Workshop*, Pennsylvania, 1981, pp. 121-130.

[12] B. G. Schunck και B. K. Horn, «"Determining optical flow": a retrospective,» σε *Artificial Intelligence*, USA, 1993, pp. 81-87.

[13] M. Trajkovic και M. Hedley, «Fast corner detection,» σε *Image and Vision Computing*, 1998, p. 16:75–87.

[14] R. E. KALMAN, «A New Approach to Linear Filtering and Prediction Problems,» *Transactions of the ASME – Journal of Basic Engineering*, αρ. 82, pp. 35-45, 1960.

[15] A. KOSZEWNIK, «THE PARROT UAV CONTROLLED BY PID CONTROLLERS,» Bialystok, 2014.

[16] «GitHub,» TUM Computer Vision Group, [Ηλεκτρονικό]. Available: https://github.com/tum-vision/tum_ardrone.

[17] S. J. JULIER και J. K. UHLMANN, «Unscented Filtering and Nonlinear Estimation,» *PROCEEDINGS OF THE IEEE,* τόμ. 92, αρ. 3, pp. 401-422, MARCH 2004.

[18] N. Dijkshoorn, «Simultaneous localization and mapping with the AR.Drone,» 2012.

[19] J. J. Leonard και H. F. Durrant-Whyte, «Simultaneous map building and localization for an autonomous mobile robot,» *Workshop on Intelligent Robots and Systems, vol 3,* τόμ. 3, pp. 1442-47, 1991.

[20] C.-C. Wang και C. Thorpe, «Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects,» Robotics Institute, Carnegie Mellon University, Pittsburgh.

[21] Y. Chen και G. Medioni, «Object Modeling by Registration of Multiple Range Images,» *IEEE Int. Conf. On Robotics and Automation,* 1991.

[22] J. Canny, «A Computational Approach To Edge Detection,» σε *Pattern Analysis and Machine Intelligence,* IEEE Trans, 1986, p. 679–698.

[23] R. Deriche, «Using Canny's criteria to derive a recursively implemented optimal edge detector,» σε *Computer Vision,* Int. J., 1987, p. 167–187.

[24] I. Sobel, σε *History and Definition of the Sobel Operator,* 2014.

[25] J. Prewitt , «Object Enhancement and Extraction" in "Picture processing and Psychopictorics,» *Academic Press,* 1970.

[26] H. P. Moravec, «Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover,» Pennsylvani, 1980.

[27] C. Harris και M. Stephens, «A combined corner and edge detector,» σε *Proceedings of the 4th Alvey Vision Conference,* United Kingdom, 1988, p. 147–151.

[28] J. Shi και C. Tomasi , «Good Features to Track,» 1994.

[29] L. Kitchen και A. Rosenfeld , «Gray-level corner detection,» σε *Pattern Recognition Letters,* 1982, pp. 95-102.

[30] S. M. Smith και J. M. Brady , « SUSAN – a new approach to low level image processing,» σε *International Journal of Computer Vision,* 1997, p. 45–78.

[31] E. Rosten και T. Drummond, «Machine learning for high-speed corner detection,» 2006.

[32] D. Lowe, «Object recognition from local scale-invariant features,» σε *Proceedings of the International Conference on Computer Vision,* 1999, p. 1150–1157.

[33] H. Bay, A. Ess, T. Tuytelaars και L. Van Gool, «Speeded Up Robust Features (SURF),» ETH Zurich, Belgium.

[34] K. Mikolajczyk και C. Schmid, «A performance evaluation of local descriptors,» σε *IEEE Transactions on Pattern Analysis \& Machine Intelligence*, 2005, pp. 1615--1630.

[35] N. Dalal και B. Triggs, «Histograms of Oriented Gradients for Human Detection,» Conference on Computer Vision and Pattern Recognition, France, 2005.

[36] P. J. Burt, «Fast filter transforms for image processing,» σε *Computer Graphics and Image Processing*, τόμ. 16, Elsevier Inc., 1981, pp. 20-51.

[37] R. O. Duda και P. E. Hart, «Use of the Hough Transformation to Detect Lines and Curves in Pictures,» *Comm. ACM, Vol. 15,* p. 11–15, January 1972.

[38] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt και J. M. Ogden, «Pyramid Methods in Image Processing,» RCA Engineer, 1984.

[39] K. G. Derpanis, «The Gaussian Pyramid,» 2005.

[40] R. Sharma και A. Savakis, «SPIE,» 16 October 2015. [Ηλεκτρονικό]. Available: http://electronicimaging.spiedigitallibrary.org/article.aspx?articleid=2473553.

[41] S. Weiss, D. Scaramuzza και R. Siegwart, «Monocular-SLAM–Based Navigation for Autonomous Micro Helicopters in GPS-Denied Environments,» *Journal of Field Robotics,* τόμ. 28, αρ. 6, pp. 854-874, 2011.

[42] A. J. Davison, «Real-Time Simultaneous Localisation and Mapping with a Single Camera».

[43] A. J. Davison, I. D. Reid, N. D. Molton και O. Stasse, «MonoSLAM: Real-Time Single Camera SLAM,» *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,* τόμ. VOL. 29, αρ. 6, JUNE 2007.

[44] J. Engel, T. Schops και D. Cremers, «LSD-SLAM: Large-Scale Direct Monocular SLAM,» Technical University Munich, Munich, 2014.

[45] J. Civera, A. J. Davison και J. M. Martinez Montiel, «Inverse Depth Parametrization for Monocular SLAM,» *IEEE TRANSACTIONS ON ROBOTICS,* τόμ. 24, αρ. 5, October 2008.

[46] Computer Vision Group, «Computer Vision Group,» Informatik IX, [Ηλεκτρονικό]. Available: http://vision.in.tum.de/research/vslam/lsdslam. [Πρόσβαση July 2016].

[47] G. Klein και D. Murray, «Parallel Tracking and Mapping for Small AR Workspaces,» σε *6th IEEE and ACM International Symposium on. IEEE,* 2007.

[48] J. Engel, J. Sturm και D. Cremers, «Scale-Aware Navigation of a Low-Cost Quadrocopter with a Monocular Camera,» Munich, 2013.

[49] J. Engel, J. Sturm και D. Cremers, «Camera-Based Navigation of a Low-Cost Quadrocopter,» Munich, 2013.

[50] J. Engel, J. Sturm και D. Cremers, «Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing,» Munich, 2013.

[51] «MIAT,» MIAT College of Technology, [Ηλεκτρονικό]. Available: https://plus.google.com/+MIATCollegeofTechnologyCanton/posts.

[52] P. Smith, I. Reid και A. Davison, «Real-Time Monocular SLAM with Straight Lines,» UK, 2006.

[53] «MAIN PUMP,» [Ηλεκτρονικό]. Available: http://www.mainpump.ru/news/lifting/5473.htm.

[54] P. Dvorak, «Windpower Engineering & Development,» 5 May 2015 . [Ηλεκτρονικό]. Available: http://www.windpowerengineering.com/design/how-turbulent-wind-abuse-wind-turbine-drivetrains/. [Πρόσβαση 2016].

[55] O. Liang , «scarLiang.com,» 25 October 2013. [Ηλεκτρονικό]. Available: https://oscarliang.com/types-of-multicopter/. [Πρόσβαση July 2016].

[56] «GitHub,» Autonomy Lab at SFU, [Ηλεκτρονικό]. Available: https://github.com/AutonomyLab/ardrone_autonomy.

[57] «dianliwenmi,» [Ηλεκτρονικό]. Available: http://www.dianliwenmi.com/postimg_4532273_11.html.

[58] «The Electric Energy,» [Ηλεκτρονικό]. Available: http://theelectricenergy.com/what-is-a-brushless-dc-electric-motor/.

[59] C. Hill, «Equipment World,» 22 December 2015. [Ηλεκτρονικό]. Available: http://www.equipmentworld.com/n-carolina-dot-issues-its-own-drone-guidelines-for-recreational-operation/. [Πρόσβαση 2016].

[60] «Alpha Unmanned Systems,» Alpha Unmanned Systems, [Ηλεκτρονικό]. Available: http://www.alphaunmannedsystems.com/gallery.

[61] «Arrishobby.com,» ARRIS HOBBY, [Ηλεκτρονικό]. Available: http://www.arrishobby.com/dji-newly-released-flagship-multirotor-controller-a2-p-995ARRIS H.

[62] Clearpath Robotics, «Robohub.org,» 29 January 2014. [Ηλεκτρονικό]. Available: http://robohub.org/ros-101-intro-to-the-robot-operating-system/.

[63] P. Bupe, «http://diydrones.com/,» 14 April 2015. [Ηλεκτρονικό]. Available: http://diydrones.com/profiles/blogs/autonomous-uav-clustering-network.