ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών
Προηγμένες Τεχνολογίες Υπολογιστικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μελέτη και κατασκευή μη επανδρωμένου αεροσκάφους (Drone) με δυνατότητα υψηλής ακρίβειας προσγείωσης με χρήση Computer Vision

Ιωάννης Κοτσιφάκης
A.M. mscacs22011

Εισηγητής: Δρ. Στυλιανός Βουτσινάς, Καθηγητής

(Κενό φύλλο)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μελέτη και κατασκευή μη επανδρωμένου αεροσκάφους (Drone) με δυνατότητα υψηλής ακρίβειας προσγείωσης με χρήση Computer Vision

Ιωάννης Κοτσιφάκης
A.M. mscacs22011

Εισηγητής:

Εισηγητής: Δρ. Στυλιανός Βουτσινάς, Καθηγητής

Εξεταστική Επιτροπή:

Δρ. Στυλιανός Βουτσινάς, Καθηγητής	Digitally signed by Stylianos Voutsinas Date: 2024.03.29 19:49:39 +02'00'
Δρ. Ιωάννης Βογιατζής, Καθηγητής	Ioannis Voyiatzis Digitally signed by Ioannis Voyiatzis Date: 2024.04.23 20:00:49 +03'00'
Δρ. Σταύρος Φατούρος, Αναπληρωτής Καθηγητής	Stavros Fatouros Digitally signed by Stavros Fatouros Date: 2024.05.13 13:55:30 +03'00'

Ημερομηνία εξέτασης:

(Κενό φύλλο)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Ιωάννης Κοτσιφάκης του Σταματίου, με αριθμό μητρώου mscacs22011 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Kotsifakis I

(Κενό φύλλο)

ΕΥΧΑΡΙΣΤΙΕΣ

Με την παρούσα διπλωματική εργασία ολοκληρώνονται οι σπουδές μου στο μεταπτυχιακό πρόγραμμα σπουδών **«Προηγμένες Τεχνολογίες Υπολογιστικών Συστημάτων»** του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών του Πανεπιστημίου Δυτικής Αττικής

Θα ήθελα να ευχαριστήσω τους καθηγητές μου για την καθοριστική συνεισφορά τους στα γνωστικά αντικείμενα που παρακολούθησα.

Ιδιαίτερα ευχαριστώ τον καθηγητή μου και επιβλέπων στην παρούσα διπλωματική εργασία, Δρ. Στυλιανό Βουτσινά, για την πολύτιμη καθοδήγηση και συμβολή του κατά τη διάρκεια της διεκπεραίωσης της.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για την συμπαράσταση της στην ολοκλήρωση των σπουδών μου.

(Κενό φύλλο)

ΠΕΡΙΛΗΨΗ

Η ανάπτυξη της τεχνολογίας των μη επανδρωμένων αεροσκαφών (UAV) έχει αποκτήσει σημαντική δυναμική τα τελευταία χρόνια, με ιδιαίτερη έμφαση στην ενίσχυση των δυνατοτήτων για ακριβείς και ελεγχόμενες προσγειώσεις. Η παρούσα μεταπτυχιακή διατριβή ασχολείται με την έρευνα και την κατασκευή ενός μη επανδρωμένου αεροσκάφους με εξειδικευμένη έμφαση στην επίτευξη προσγειώσεων υψηλής ακρίβειας χρησιμοποιώντας τεχνολογίες υπολογιστικής όρασης. Η μελέτη διερευνά το OpenCV, τους δείκτες ArUco και το Raspberry Pi 5 ως συνοδευτικό υπολογιστή για τον ελεγκτή πτήσης.

Το κίνητρο πίσω από αυτή την έρευνα πηγάζει από την αυξανόμενη ζήτηση για UAV σε διάφορες εφαρμογές, όπως η επιτήρηση τομέα, η αναγνώριση και οι υπηρεσίες παράδοσης. Η επίτευξη προσγειώσεων υψηλής ακρίβειας είναι ζωτικής σημασίας για τη βελτιστοποίηση αυτών των εφαρμογών, καθώς επιτρέπει την ασφαλή και ακριβή προσγείωση σε περιορισμένα ή δυναμικά περιβάλλοντα. Η διατριβή στοχεύει να συμβάλει στο πεδίο με το σχεδιασμό και την εφαρμογή ενός ισχυρού συστήματος που αξιοποιεί τεχνικές υπολογιστικής όρασης για την ενίσχυση της ακριβούς προσγείωσης των UAV.

Η θεμελιώδης τεχνολογία που διερευνάται σε αυτή τη διατριβή είναι το OpenCV, μια ευρέως χρησιμοποιούμενη βιβλιοθήκη όρασης υπολογιστή ανοιχτού κώδικα. Το OpenCV παρέχει ένα ευέλικτο σύνολο αλγορίθμων για την επεξεργασία εικόνας, την ανίχνευση αντικειμένων και την παρακολούθηση χαρακτηριστικών, βασικά στοιχεία για την ανάπτυξη ενός συστήματος προσγείωσης υψηλής ακρίβειας. Το σύστημα ArUco Marker αξιοποιείται από την υπολογιστική όραση και αποτελεί κεντρικό στοιχείο της προτεινόμενης μεθοδολογίας.

Η αρχιτεκτονική υλικού που επιλέχθηκε για αυτή την έρευνα περιλαμβάνει ένα Raspberry Pi 5, που χρησιμεύει ως συνοδευτικός υπολογιστής του ελεγκτή πτήσης. Το Raspberry Pi 5 παρέχει την υπολογιστική ισχύ που απαιτείται για την επεξεργασία εικόνας σε πραγματικό χρόνο και τη λήψη αποφάσεων κατά τη φάση της προσγείωσης. Η ενσωμάτωση ενός συνοδευτικού υπολογιστή ενισχύει τις αυτόνομες λειτουργίες του UAV, επιτρέποντάς του να εκτελεί πολύπλοκους αλγόριθμους χωρίς να επιβαρύνει υπερβολικά τον ελεγκτή πτήσης.

Σημαντική πτυχή της διατριβής είναι η ανάπτυξη κώδικα Python που υλοποιεί την ακριβή προσγείωση του drone σε βάση αποτελούμενη από δείκτες ArUco. Ο κώδικας ενσωματώνει αλγόριθμους επεξεργασίας εικόνας από το OpenCV για την ανίχνευση και ανάλυση των δεικτών ArUco σε πραγματικό χρόνο. Με την αξιοποίηση των δεικτών ArUco, ο αλγόριθμος επιτρέπει στο drone να υπολογίσει τη θέση και τον προσανατολισμό του σε σχέση με τους δείκτες, διευκολύνοντας μια ελεγχόμενη κάθοδο και προσγείωση.

Η μεθοδολογία της έρευνας περιλαμβάνει συνδυασμό θεωρητικής ανάλυσης, ανάπτυξης αλγορίθμων και πρακτικής εφαρμογής. Οι θεωρητικές πτυχές περιλαμβάνουν μια ολοκληρωμένη ανασκόπηση της υπάρχουσας βιβλιογραφίας σχετικά με τα συστήματα προσγείωσης UAV και τις λειτουργίες του OpenCV. Η

ανάπτυξη αλγορίθμων περιλαμβάνει το σχεδιασμό και την κωδικοποίηση αλγορίθμων επεξεργασίας και ελέγχου εικόνας προσαρμοσμένων για προσγειώσεις υψηλής ακρίβειας και εντολές για τον χειρισμό του UAV. Η πρακτική εφαρμογή περιλαμβάνει την κατασκευή του drone , την ανάπτυξη του κώδικα Python και εκτεταμένες δοκιμές σε πραγματικά σενάρια.

Συμπερασματικά, η διατριβή αυτή διερευνά τη διασταύρωση της τεχνολογίας UAV, της όρασης υπολογιστών και των προσγειώσεων υψηλής ακρίβειας. Με την ενσωμάτωση του OpenCV, των δεικτών ArUco και του Raspberry Pi 5, η έρευνα στοχεύει στην ανάπτυξη ενός αξιόπιστου και αποτελεσματικού συστήματος για την ενίσχυση των δυνατοτήτων προσγείωσης μη επανδρωμένων αεροσκαφών. Τα αποτελέσματα αυτής της μελέτης έχουν τη δυνατότητα να επηρεάσουν τις εξελίξεις στις εφαρμογές UAV, ιδιαίτερα σε τομείς που απαιτούν ακριβείς και ελεγχόμενες προσγειώσεις.

ABSTRACT

The development of Unmanned Aerial Vehicle (UAV) technology has gained considerable momentum in recent years, with a particular focus on enhancing capabilities for precise and controlled landings. This master's thesis deals with the research and construction of an unmanned aircraft with a specialized emphasis on achieving high-precision landings using computer vision technologies. The study investigates OpenCV, ArUco markers and the Raspberry Pi 5 as a companion computer for the flight controller.

The motivation behind this research stems from the growing demand for UAVs in various applications, such as domain surveillance, identification, and delivery services. Achieving high-precision landings is crucial to optimizing these applications, as it allows for safe and accurate landing in restricted or dynamic environments. The thesis aims to contribute to the field by designing and implementing a robust system that leverages computer vision techniques to enhance accurate landing of UAVs.

The fundamental technology explored in this thesis is OpenCV, a widely used open-source computer vision library. OpenCV provides a flexible set of algorithms for image processing, object detection and feature tracking – key elements for developing a high-precision landing system. The ArUco Marker system is utilized by computer vision and is a central element of the proposed methodology.

The hardware architecture chosen for this research includes a Raspberry Pi 5, serving as the flight controller's companion computer. The Raspberry Pi 5 provides the computing power needed for real-time image processing and decision-making during the landing phase. The integration of a companion computer enhances the autonomous functions of the UAV, allowing it to run complex algorithms without overloading the flight controller.

An important aspect of the thesis is the development of Python code that implements the precise landing of the drone on a basis consisting of ArUco markers. The code incorporates image processing algorithms from OpenCV to detect and analyze ArUco markers in real time. By utilizing ArUco indicators, the algorithm allows the drone to calculate its position and orientation relative to the indicators, facilitating a controlled descent and landing.

The research methodology involves a combination of theoretical analysis, algorithm development and practical application. Theoretical aspects include a comprehensive review of existing literature on UAV landing systems and OpenCV operations. Algorithm development involves the design and coding of image processing and control algorithms adapted for high-precision landings and commands to manipulate the UAV. Practical implementation includes building the drone, developing the Python code, and extensive testing in real-world scenarios.

In conclusion, this thesis explores the intersection of UAV technology, computer vision, and high-precision landings. By integrating OpenCV, ArUco indicators and Raspberry Pi 5, the research aims to develop a reliable and effective system to enhance drone landing capabilities. The results of this study have the potential to influence developments in UAV applications, particularly in areas that require precise and controlled landings.

KEY WORDS: Unmanned Aerial Vehicle (UAV), Precision Landing, Computer Vision, ArUco Marker System, Camera Calibration, High-Accuracy Positioning, Real-time Processing, Drone Applications, Unmanned Aerial Vehicle (UAV) Navigation

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μη επανδρωμένα αεροσκάφη, Προσγειώσεις υψηλής ακρίβειας, μηχανική όραση, Σύστημα δεικτών ArUco, Βαθμονόμηση κάμερας, Ακριβής θέση, Επεξεργασία σε πραγματικό χρόνο, Εφαρμογές drone, Πλοήγηση μη επανδρωμένων αεροσκαφών

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

AR Augmented reality

CV Computer Vision

ESC Electronic Speed Controller

GPS Global Positioning System

IMU Inertial Measurement Units

LiDAR Light Detection and Ranging

PDB Power Distribution Board

RTK Real-Time Kinematic positioning

UAV Unmanned Aerial Vehicle

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1: Παρουσίαση προβλήματος

Οι ακριβείς προσγειώσεις αντιπροσωπεύουν μια κρίσιμη πτυχή των μη επανδρωμένων αεροσκαφών (UAV) και η αντιμετώπιση των σχετικών προκλήσεων είναι υψίστης σημασίας. Η επίτευξη ακρίβειας στις προσγειώσεις με μη επανδρωμένα αεροσκάφη είναι αναγκαία για διάφορες εφαρμογές, συμπεριλαμβανομένης της επιτήρησης, της παράδοσης δεμάτων, της έρευνας και διάσωσης και της επιθεώρησης εγκαταστάσεων. Οι προκλήσεις και η σημασία των ακριβών προσγειώσεων μπορούν να εξεταστούν μέσω διαφόρων βασικών εκτιμήσεων.

Μία από τις σημαντικότερες προκλήσεις στις προσγειώσεις των μη επανδρωμένων αεροσκαφών είναι η διασφάλιση της ασφάλειας. Οι ακριβείς προσγειώσεις είναι ιδιαίτερα σημαντικές για την αποφυγή συγκρούσεων με εμπόδια, κτήρια ή ανθρώπους. Οι ρυθμιστικοί φορείς επιβάλλουν αυστηρές κατευθυντήριες γραμμές για τις λειτουργίες των μη επανδρωμένων αεροσκαφών και η τήρηση αυτών των κανονισμών απαιτεί ακριβή έλεγχο κατά την προσγείωση για την ελαχιστοποίηση των κινδύνων και τη συμμόρφωση με τους κανονισμούς του εναέριου χώρου.

Τα αεροσκάφη που χρησιμοποιούνται για την παράδοση ωφέλιμου φορτίου, όπως η μεταφορά ιατρικών προμηθειών ή πακέτων, απαιτούν ακριβείς προσγειώσεις για να εξασφαλιστεί η ασφαλής παράδοση αγαθών. Η επίτευξη ακρίβειας είναι απαραίτητη για την αποφυγή ζημιών στο ωφέλιμο φορτίο και για την παροχή αποτελεσματικών και αξιόπιστων λειτουργιών εφοδιαστικής αλυσίδας.

Οι δυσμενείς καιρικές συνθήκες, όπως ισχυροί άνεμοι, βροχή ή χαμηλή ορατότητα, παρουσιάζουν προκλήσεις για τις επιχειρήσεις των μη επανδρωμένων αεροσκαφών, συμπεριλαμβανομένων των προσγειώσεων. Η εξασφάλιση προσγειώσεων υπό διαφορετικές περιβαλλοντικές συνθήκες είναι καθοριστικής σημασίας για τη διατήρηση των επιχειρησιακών δυνατοτήτων και της αξιοπιστίας.

Η εφαρμογή αυτόνομων συστημάτων προσγείωσης απαιτεί εξελιγμένους αλγόριθμους πλοήγησης και ελέγχου. Η πρόκληση έγκειται στην ανάπτυξη αλγορίθμων που μπορούν να προσαρμοστούν στις μεταβαλλόμενες συνθήκες, να εκτιμήσουν με ακρίβεια τη θέση του drone και να εκτελέσουν ακριβείς προσγειώσεις δίχως ανθρώπινη παρέμβαση. Η παρουσία εμποδίων στη ζώνη προσγείωσης αποτελεί πρόβλημα για τα αεροσκάφη. Η εξασφάλιση ακριβών προσγειώσεων απαιτεί αποτελεσματικούς μηχανισμούς ανίχνευσης και αποφυγής εμποδίων για την πρόληψη συγκρούσεων κατά την κάθοδο, ειδικά σε δυναμικά περιβάλλοντα.

Σε εφαρμογές όπως η αντιμετώπιση έκτακτης ανάγκης και επιχειρήσεις έρευνας και διάσωσης, τα αεροσκάφη πρέπει να προσγειώνονται ακριβώς για να αναπτύξουν εξοπλισμό ή και αισθητήρες σε συγκεκριμένες τοποθεσίες. Η σημασία των ακριβών προσγειώσεων σε κρίσιμα σενάρια ευαίσθητα στο χρόνο δεν μπορεί να παραληφθεί.

Η ενσωμάτωση συστημάτων όρασης υπολογιστή, όπως συστήματα προσγείωσης που χρησιμοποιούν τεχνολογίες δεικτών ArUco, εισάγει προκλήσεις

που σχετίζονται με την ακριβή ανίχνευση των δεικτών, την επεξεργασία σε πραγματικό χρόνο και την αξιοπιστία του συστήματος. Η υπέρβαση αυτών των προκλήσεων είναι ιδιαίτερα σημαντική για την επίτευξη ακριβών και ισχυρών προσγειώσεων μη επανδρωμένων αεροσκαφών.

Η αντιμετώπιση αυτών των προκλήσεων είναι απαραίτητη για την απελευθέρωση του πλήρους δυναμικού των αεροσκαφών σε διάφορους τομείς. Επηρεάζει άμεσα την ασφάλεια, αξιοπιστία, και αποτελεσματικότητα των αποστολών UAV, καθιστώντας τη συνεχή έρευνα και ανάπτυξη σε αυτόν τον τομέα αναγκαία για την πρόοδο της τεχνολογίας των UAV και την ευρύτερη ενσωμάτωσή αυτής σε πραγματικές εφαρμογές.

1.2: Αναμενόμενα αποτελέσματα

Τα αναμενόμενα αποτελέσματα αυτής της έρευνας είναι η κατασκευή ενός πλήρως λειτουργικού UAV που να αναδεικνύει την αποτελεσματικότητα της προσγείωσης ακριβείας με την χρήση μηχανικής όρασης. Κύριος στόχος είναι η επίτευξη σημαντικής βελτίωσης της ακρίβειας και της αξιοπιστίας των προσγειώσεων UAV μέσω της ενσωμάτωσης τεχνολογιών υπολογιστικής όρασης. Η επιτυχής εφαρμογή αλγορίθμων OpenCV θα επιτρέπει στο UAV να προσγειώνεται επιδεικνύοντας βελτιωμένη ακρίβεια και σταθερότητα κατά τη φάση της καθόδου.

Η ενσωμάτωση των δεικτών ArUco στο OpenCV απαιτεί μια απρόσκοπτη και ταχεία εκτέλεση αλγορίθμων επεξεργασίας εικόνας για να εξασφαλιστεί η έγκαιρη λήψη αποφάσεων κατά τη διάρκεια της προσγείωσης. Το αναμενόμενο αποτέλεσμα είναι ένα σύστημα που λειτουργεί με χαμηλή καθυστέρηση, επιτρέποντας στο UAV να προβαίνει σε γρήγορες προσαρμογές με βάση την ανατροφοδότηση εικόνας που λαμβάνει σε πραγματικό χρόνο. Τα κριτήρια αξιολόγησης περιλαμβάνουν την ταχύτητα επεξεργασίας της ανάλυσης εικόνας, την ανταπόκριση στις περιβαλλοντικές αλλαγές και την ικανότητα του συστήματος να προσαρμόζεται στις δυναμικές συνθήκες προσγείωσης.

Η επιτυχής επίδειξη των επιδόσεων σε πραγματικό χρόνο θα επαληθεύσει την πρακτική εφαρμογή του προτεινόμενου συστήματος προσγείωσης σε διάφορα επιχειρησιακά σενάρια. Οι αλγόριθμοι που βασίζονται στην όραση του υπολογιστή πρέπει να επιδεικνύουν ανθεκτικότητα στις διακυμάνσεις των περιβαλλοντικών συνθηκών, του φωτισμού και της ορατότητας των δεικτών. Το αναμενόμενο αποτέλεσμα είναι ένα σύστημα προσγείωσης που λειτουργεί αξιόπιστα υπό διαφορετικές συνθήκες, εξασφαλίζοντας συνεπή απόδοση σε μια σειρά σεναρίων. Συμπερασματικά, τα αναμενόμενα αποτελέσματα αυτού του ερευνητικού έργου περιλαμβάνουν την επίτευξη αυξημένου επιπέδου ακρίβειας στις προσγειώσεις UAV μέσω της ενσωμάτωσης τεχνολογιών υπολογιστικής όρασης. Η εστίαση στην ακρίβεια και την ανταπόκριση σε πραγματικό χρόνο στοχεύει να συμβάλει στην πρόοδο των δυνατοτήτων των UAV, ιδιαίτερα σε σενάρια όπου οι ακριβείς προσγειώσεις είναι πρωταρχικής σημασίας.

Τα αποτελέσματα αυτής της έρευνας έχουν πιθανές επιπτώσεις σε ποικίλες εφαρμογές, από την επιτήρηση και την αναγνώριση έως τις υπηρεσίες παράδοσης, όπου οι προσγειώσεις υψηλής ακρίβειας διαδραματίζουν κρίσιμο ρόλο στη βελτιστοποίηση της επιχειρησιακής αποτελεσματικότητας και ασφάλειας.

ΚΕΦΑΛΑΙΟ 2:

ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1: Drones

2.1.1: Precision Landing Techniques

Η αποτελεσματικότητα ενός συστήματος προσγείωσης σε μη επανδρωμένα αεροσκάφη είναι ζωτικής σημασίας για εφαρμογές όπως η επιτήρηση τομέα, η παράδοση αντικειμένων, η ανίχνευση και πολλά άλλα.

Ο συνδυασμός πολλαπλών τεχνολογιών ενισχύει την αξιοπιστία και την ασφάλεια των λειτουργιών προσγείωσης μη επανδρωμένων αεροσκαφών. Η επιλογή συγκεκριμένων συστημάτων προσγείωσης εξαρτάται από την εφαρμογή, τις περιβαλλοντικές συνθήκες και το απαιτούμενο επίπεδο ακρίβειας. [1]

Τα βασικά στοιχεία και προσεγγίσεις που χρησιμοποιούνται συνήθως σε συστήματα προσγείωσης μη επανδρωμένων αεροσκαφών είναι τα εξής:

- Συστήματα βασισμένα σε GPS:

Το Global Positioning System (GPS) χρησιμοποιείται ευρέως για την εύρεση της τοποθεσίας και κατεύθυνσης των μη επανδρωμένων αεροσκαφών. Το GPS παρέχει πληροφορίες σχετικά με την τοποθεσία, το υψόμετρο και την ταχύτητα του drone, βοηθώντας στη διαδικασία προσγείωσης. Ωστόσο, το GPS από μόνο του δύναται να μην παρέχει επαρκή ακρίβεια για ορισμένες εφαρμογές.

Τα συμβατικά GPS έχουν μέγιστη ακρίβεια τα 0,3-1 μ. Τα RTK (Real-time kinematic positioning) GPS έχουν ακρίβεια 0,01-0,2 μέτρα με το ανάλογο κόστος και με βασικό μειονέκτημα ότι χρειάζονται σταθμό βάσης για να λειτουργήσουν.

Εικόνα 1 RTK F9P GPS με Base station

[1] Keller, A., & Ben-Moshe, B. (2022). A Robust and Accurate Landing Methodology for Drones on Moving Targets. Drones, 6(4), 98. https://doi.org/10.3390/drones6040098

- Computer Vision (CV):

Το Computer Vision έχει σημαντική συμβολή στα συστήματα προσγείωσης σε μη επανδρωμένα αεροσκάφη, ενισχύοντας την ακρίβεια και την αυτονομία. Οι κάμερες και οι αισθητήρες καταγράφουν εικόνες σε πραγματικό χρόνο του περιβάλλοντος χώρου και οι αλγόριθμοι επεξεργάζονται αυτά τα δεδομένα για την πλοήγηση και την προσγείωση. Το OpenCV είναι μια δημοφιλής βιβλιοθήκη όρασης υπολογιστή που χρησιμοποιείται συχνά για την επεξεργασία εικόνας και την αναγνώριση χαρακτηριστικών.

- Συστήματα LiDAR και ραντάρ:

Τα συστήματα ανίχνευσης και εμβέλειας φωτός (Light Detection and Ranging - LiDAR) και ραντάρ χρησιμοποιούν λέιζερ ή ραδιοκύματα για τη μέτρηση αποστάσεων και την ανίχνευση εμποδίων.

Αυτά τα συστήματα παρέχουν πρόσθετες πληροφορίες σχετικά με το περιβάλλον που βρίσκεται το μη επανδρωμένο αεροσκάφος, βοηθώντας το στο να αποφύγει εμπόδια κατά την προσγείωση.

Εικόνα 2 Lidar Sensor

- Αδρανειακές Μονάδες Μέτρησης (IMU):

Τα IMU (Inertial Measurement Units): αποτελούνται από επιταχυνσιόμετρα και γυροσκόπια που μετρούν την επιτάχυνση και την περιστροφή του drone. Η ενσωμάτωση με άλλους αισθητήρες συμβάλλει στη βελτίωση της ακρίβειας της τοποθέτησης και της πλοήγησης κατά την προσγείωση.

Τα flight controllers έχουν ενσωματωμένα γυροσκόπια.

Εικόνα 3 IMU Sensor

- Διάφοροι αισθητήρες

Οι αισθητήρες υπερήχων, λέιζερ και υπέρυθρων χρησιμοποιούνται συχνά για τη μέτρηση του υψομέτρου και της απόστασης του drone από το έδαφος κατά την κάθοδο. Αυτοί οι αισθητήρες συνεισφέρουν στον ακριβή έλεγχο του υψομέτρου και στην ασφαλή προσγείωση. Το μειονέκτημα τους είναι ότι η εμβέλεια των συμβατικών αισθητήρων, πολλές φορές δεν ξεπερνά τα 20 μέτρα.

Εικόνα 4 Ultrasonic Sensor

Εικόνα 5 Infrared Sensor

- Ασύρματα Συστήματα Επικοινωνίας:

Η επικοινωνία με επίγειους σταθμούς ή άλλα αεροσκάφη μπορεί να ενισχύσει την ακρίβεια προσγείωσης. Οι ενημερώσεις σε πραγματικό χρόνο και οι οδηγίες από εξωτερικές πηγές μπορούν να βοηθήσουν το drone να προσαρμοστεί στις μεταβαλλόμενες συνθήκες κατά την προσγείωση.

- Αυτόνομοι Αλγόριθμοι Προσγείωσης:

Διάφοροι αλγόριθμοι έχουν σχεδιαστεί για να λαμβάνουν αποφάσεις με βάση τις εισόδους των αισθητήρων, εξασφαλίζοντας μια ομαλή και ασφαλή προσγείωση. Αυτοί οι αλγόριθμοι μπορεί να λαμβάνουν υπόψη παράγοντες, όπως οι συνθήκες ανέμου, το έδαφος και τα πιθανά εμπόδια.

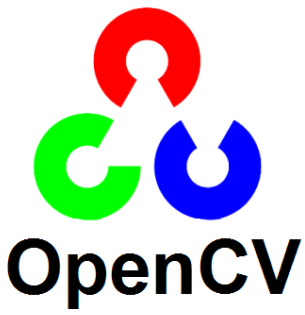
2.2: Computer Vision

2.2.1: Τι είναι το OpenCV

Το OpenCV[2], ή Open Source Computer Vision Library, είναι μια ολοκληρωμένη βιβλιοθήκη λογισμικού ανοιχτού κώδικα σχεδιασμένη για εφαρμογές υπολογιστικής όρασης και επεξεργασίας εικόνας. Αρχικά αναπτύχθηκε από την Intel το 1999, το OpenCV από τότε διατηρείται και επεκτείνεται ενεργά από την κοινότητα προγραμματιστών. Η φύση ανοιχτού κώδικα της βιβλιοθήκης, διευκολύνει τη συνεργασία και τη συνεχή του βελτίωση. Ο ανοιχτός κώδικας του OpenCV επιτρέπει στους χρήστες να τον τροποποιούν και να τον διανέμουν ελεύθερα, προωθώντας μια κουλτούρα καινοτομίας και κοινής γνώσης. Η βιβλιοθήκη είναι universal, υποστηρίζοντας διάφορα λειτουργικά συστήματα όπως Windows, Linux, macOS, Android και iOS, καθιστώντας την προσαρμόσιμη σε ένα ευρύ φάσμα εφαρμογών και συσκευών. [3]

Το OpenCV διαθέτει μια εκτεταμένη συλλογή αλγορίθμων που καλύπτουν την επεξεργασία εικόνας, την υπολογιστική όραση και τη μηχανική μάθηση. Αυτοί οι αλγόριθμοι επιτρέπουν εργασίες όπως φιλτράρισμα εικόνας, ανίχνευση χαρακτηριστικών και αναγνώριση αντικειμένων. Η βιβλιοθήκη ενσωματώνει αποτελεσματικές δομές δεδομένων όπως η κλάση Mat για χειρισμό και επεξεργασία δεδομένων εικόνας, συμβάλλοντας στη βελτιστοποιημένη απόδοση. Το OpenCV είναι γνωστό για τις δυνατότητες επεξεργασίας σε πραγματικό χρόνο, καθιστώντας το καθοριστικό σε εφαρμογές όπως η ρομποτική, η επιτήρηση, η εικονική πραγματικότητα και τα αυτόνομα οχήματα.

Η σπουδαιότητα του OpenCV διευρύνεται σε διάφορους τομείς, συμπεριλαμβανομένης της αναγνώρισης αντικειμένων, της αναγνώρισης προσώπου, της ανάλυσης ιατρικής εικόνας και των αυτόνομων οχημάτων. Η ενσωμάτωση της μηχανικής μάθησης επιτρέπει στους χρήστες να εκπαιδεύουν μοντέλα για εργασίες όπως για παράδειγμα η ανίχνευση και η ταξινόμηση αντικειμένων. Η ευελιξία της βιβλιοθήκης και η ενεργή υποστήριξη της κοινότητας την καθιστούν θεμελιώδες εργαλείο για ερευνητές, προγραμματιστές και μηχανικούς που εργάζονται στον τομέα του Computer Vision και σε συναφείς τομείς.



Εικόνα 6 OpenCV library logo

[2] OpenCV. (2000). The OpenCV Library. OpenCV. https://opencv.org/about/
[3] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools, 25(11), 120-126.

2.2.2: Εφαρμογές του OpenCV στην καθημερινότητα

Μια ανασκόπηση των εφαρμογών Computer vision και OpenCV στην καθημερινότητα παρουσιάζει την ευελιξία και την αποτελεσματικότητα αυτών των τεχνολογιών σε διάφορα τμήματα. Το OpenCV, παρέχει ένα πλούσιο σύνολο εργαλείων και αλγορίθμων που επιτρέπουν στους προγραμματιστές και τους ερευνητές να εφαρμόσουν λύσεις βασισμένες στην όραση μέσω του υπολογιστή . Η συνεχής ανάπτυξη αλγορίθμων υπολογιστικής όρασης και η προσβασιμότητα τους έχουν τροφοδοτήσει την καινοτομία σε διάφορους κλάδους. Αυτές οι εφαρμογές αποδεικνύουν την προσαρμοστικότητα της όρασης του υπολογιστή στην επίλυση προκλήσεων στον πραγματικό κόσμο και τη βελτίωση της αποτελεσματικότητας σε διάφορους τομείς.

Το OpenCV αξιοποιείται ευρέως για την αναγνώριση και την παρακολούθηση αντικειμένων σε εφαρμογές όπως τα αυτόνομα οχήματα και η ρομποτική. Οι αλγόριθμοι ανίχνευσης αντικειμένων, όπως το Haar-cascade[4] και τα μοντέλα που βασίζονται σε deep learning, επιτρέπουν την αναγνώριση και την παρακολούθηση αντικειμένων σε πραγματικό χρόνο. Το OpenCV παρέχει εργαλεία για την ανίχνευση και την αναγνώριση προσώπου[5], καθιστώντας το πολύτιμο για την ασφάλεια, τον έλεγχο πρόσβασης και την αλληλεπίδραση ανθρώπου-υπολογιστή. Τα συστήματα αναγνώρισης προσώπου χρησιμοποιούνται για τον εντοπισμό και την επαλήθευση ατόμων με βάση τα χαρακτηριστικά του προσώπου.

Η οπτική ικανότητα του υπολογιστή, χρησιμοποιείται σε συστήματα αναγνώρισης χειρονομιών[6]. Αυτή η τεχνολογία βρίσκει εφαρμογές σε παιχνίδια, εικονική πραγματικότητα και αλληλεπίδραση ανθρώπου-υπολογιστή, επιτρέποντας στους χρήστες να ελέγχουν συσκευές μέσω χειρονομιών. Το OpenCV συμβάλλει στις εφαρμογές AR επιτρέποντας την αναγνώριση αντικειμένων στο φυσικό περιβάλλον. Η AR επικαλύπτει ψηφιακές πληροφορίες στον πραγματικό κόσμο, ενισχύοντας τις εμπειρίες των χρηστών σε τομείς όπως η εκπαίδευση και η ψυχαγωγία.

Το OpenCV αποτελεί βασικό στοιχείο στην ανάπτυξη αυτόνομων οχημάτων και βοηθά σε εργασίες όπως η ανίχνευση λωρίδας και η αποφυγή εμποδίων, συμβάλλοντας στην ασφάλεια και την πλοήγηση των αυτόνομων οχημάτων. Το OpenCV χρησιμοποιείται εκτενώς στη ρομποτική για εργασίες όπως η πλοήγηση ρομπότ, ο χειρισμός αντικειμένων και η κατανόηση χώρου. Η όραση του υπολογιστή επιτρέπει στα ρομπότ να αντιλαμβάνονται και να αλληλοεπιδρούν με το περιβάλλον τους, καθιστώντας τα πιο προσαρμόσιμα και ευέλικτα.

Τα συστήματα όρασης υπολογιστών, χρησιμοποιούνται στον βιομηχανικό αυτοματισμό για τον έλεγχο ποιότητας, την ταξινόμηση αντικειμένων και τη βελτιστοποίηση διαδικασιών. Επιπρόσθετα το OpenCV εντοπίζεται σε εφαρμογές

[4] OpenCV. (2015). Cascade Classifier. OpenCV Documentation. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

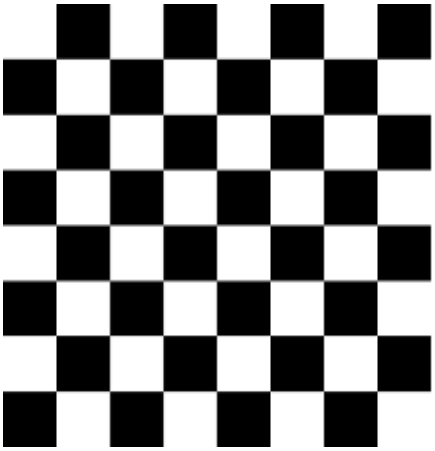
[5] Viola, P., & Jones, M. J. (2004). Robust real-time face detection. International Journal of Computer Vision, 57(2), 137–154.

[6] AIMIND. (2019). Real-Time Hand Gesture Recognition Using OpenCV: A Step-by-Step Guide. AIMIND Blog. https://pub.aimind.so/real-time-hand-gesture-recognition-using-opencv-a-step-by-step-guide-2820618caa08

στη γεωργία[7] για εργασίες όπως η παρακολούθηση των καλλιεργειών και η ανίχνευση ασθενειών. Η μηχανική όραση βοηθά στην ανάλυση αερο-φωτογραφιών για τεκμηριωμένη λήψη αποφάσεων στη γεωργία. Τέλος, η όραση του υπολογιστή χρησιμοποιείται στα αθλήματα[8] για την παρακολούθηση των παικτών, την αναγνώριση δράσης και την ανάλυση της απόδοσης. Πιο συγκεκριμένα, βοηθά στην εξαγωγή πολύτιμων πληροφοριών από αθλητικά βίντεο, συμβάλλοντας στην προπόνηση και την ανάπτυξη των παικτών.

2.2.3: Camera lens distortion

Η παραμόρφωση του φακού της κάμερας[9] είναι ένα φαινόμενο που συμβαίνει, όταν η εικόνα που συλλαμβάνεται από μια κάμερα αποκλίνει από την πραγματικότητα λόγω ατελειών στον φακό της κάμερας. Αυτές οι στρεβλώσεις μπορούν να εκδηλωθούν ως ακτινικές ή εφαπτόμενες στρεβλώσεις, επηρεάζοντας την ακρίβεια των εικόνων, ειδικά σε εφαρμογές όρασης υπολογιστή και επεξεργασίας εικόνας.[10]



Εικόνα 7 No distortion

Η ακτινική παραμόρφωση (Radial Distortion) συμβαίνει λόγω της καμπυλότητας του φακού της κάμερας, οδηγώντας σε μεγέθυνση ή συρρίκνωση των σημείων εικόνας καθώς απομακρύνονται από το κέντρο της εικόνας. Αυτή η παραμόρφωση είναι πιο αισθητή στις άκρες μιας εικόνας και χαρακτηρίζεται από ένα φαινόμενο βαρελιού ή μαξιλαριού.

Η Barrel distortion προκαλεί την εικόνα να διογκωθεί προς τα έξω, ενώ η Pincushion distortion έχει ως αποτέλεσμα μια προς τα μέσα κάμψη εμφάνιση. Οι ακτινικές παραμορφώσεις προκαλούν καμπύλες ευθείες γραμμές, ιδιαίτερα προς τις άκρες της εικόνας.

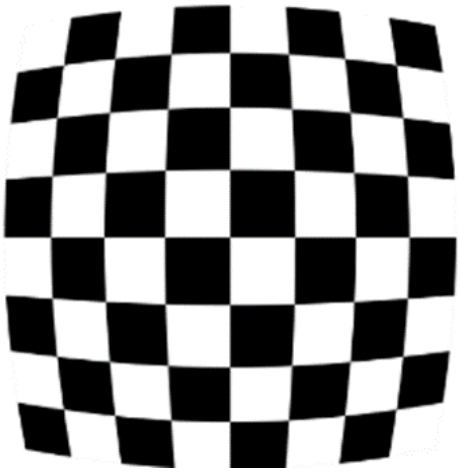
---

[7] OpenCV. (2023). Computer Vision in Agriculture: Challenges & Solutions. OpenCV Blog. https://www.opencv.ai/blog/computer-vision-in-agriculture-challenges-solutions

[8] Seth, M. (2020). Athlete Pose Detection Using OpenCV in Deep Learning. Medium. https://medium.com/swlh/athlete-pose-detection-3d1b93f2d82e

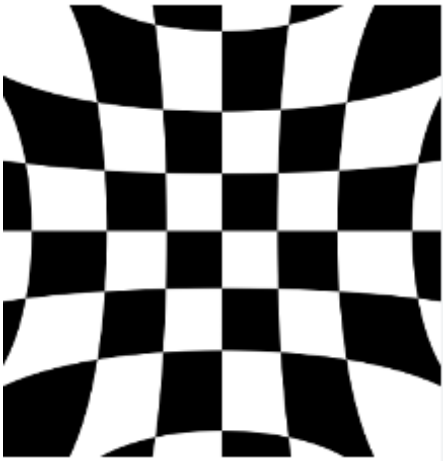
[9] Hartley RI, Kang SB (2007) Parameter-free radial distortion correction with center of distortion estimation. IEEE Trans Pattern Anal Mach Intell 29.

[10] Howse, J., Minichino, J., & Packt Publishing. (2020). Learning OpenCV 4 Computer Vision with Python 3: Get to grips with tools, techniques, and algorithms for computer vision and machine learning, 3rd Edition (3rd ed.), Packt Publishing.



Εικόνα 8 Positive Radial Distortion (Barrel distortion)

Από την άλλη πλευρά, οι εφαπτόμενες παραμορφώσεις συμβαίνουν, όταν ο φακός δεν είναι απόλυτα ευθυγραμμισμένος με τον αισθητήρα εικόνας, οδηγώντας σε μια λοξή εμφάνιση. Αυτή η κακή ευθυγράμμιση εισάγει σφάλματα στην τοποθέτηση των σημείων εικόνας, προκαλώντας την μετατόπιση τους εφαπτομενικά. Η εφαπτομένη παραμόρφωση συνήθως εκδηλώνεται ως ένα στρεβλό αποτέλεσμα στην εικόνα.



Εικόνα 9 Negative Radial Distortion (Pincushion distortion)

Η παραμόρφωση της κάμερας έχει σημαντικές επιπτώσεις για τις εργασίες υπολογιστικής όρασης, καθώς μπορεί να εισαγάγει ανακρίβειες στις χωρικές σχέσεις μεταξύ αντικειμένων στο πεδίο παρατήρησης. Ως εκ τούτου, η βαθμονόμηση της κάμερας, η οποία περιλαμβάνει την εκτίμηση των παραμέτρων παραμόρφωσης και τη διόρθωση αυτών των στρεβλώσεων[11], είναι ένα κρίσιμο βήμα για τη διασφάλιση της ακρίβειας της ανάλυσης εικόνας και των εφαρμογών Computer Vision.

Με την κατανόηση και την αντιστάθμιση των επιπτώσεων παραμόρφωσης, η χαρτογράφηση μεταξύ του πραγματικού κόσμου και των εικόνων που έχουν ληφθεί μπορεί να βελτιωθεί, οδηγώντας σε πιο αξιόπιστα και ακριβή αποτελέσματα σε διάφορες εφαρμογές που βασίζονται στην όραση μέσω του υπολογιστή.

[11] OpenCV. (2019). Camera Calibration and 3D Reconstruction. OpenCV Documentation. https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

2.2.4: Ρύθμιση κάμερας για διόρθωση στρεβλώσεων (Calibration)

Η ρύθμιση της κάμερας είναι μια θεμελιώδης διαδικασία στη μηχανική όραση που εξετάζει την παραμόρφωση του φακού για να βελτιώσει την ακρίβεια της ανάλυσης εικόνας.[12] Πρώτον, μια σειρά εικόνων με γνωστό μοτίβο διακρίβωσης, όπως μια σκακιέρα ή κύκλοι, συλλαμβάνεται από διάφορες οπτικές γωνίες.[13] Αυτές οι εικόνες χρησιμεύουν ως βάση για την κατανόηση και τη διόρθωση των στρεβλώσεων που απεικονίζονται από τον φακό της κάμερας. Στη συνέχεια, εξελιγμένοι αλγόριθμοι μηχανικής όρασης, που συχνά εφαρμόζονται μέσω βιβλιοθηκών, όπως το OpenCV, χρησιμοποιούνται για την ανίχνευση και τον ακριβή εντοπισμό βασικών σημείων εντός του μοτίβου διακρίβωσης σε κάθε εικόνα. Αυτό το βήμα είναι πολύ σημαντικό για τον ακριβή εντοπισμό της παραμόρφωσης που υπάρχει στις εικόνες.

Η επόμενη φάση επικεντρώνεται στον υπολογισμό των εγγενών παραμέτρων της κάμερας χρησιμοποιώντας μαθηματικά μοντέλα. Αυτές οι παράμετροι περιλαμβάνουν συντελεστές παραμόρφωσης, εστιακή απόσταση και το κύριο σημείο. Το OpenCV παρέχει ειδικές συναρτήσεις που αξιοποιούν αυτά τα μαθηματικά μοντέλα για να αντλήσουν τις εγγενείς παραμέτρους με υψηλή ακρίβεια.

Η κάμερα πρέπει να είναι σωστά τοποθετημένη σε σταθερή επιφάνεια ή τρίποδο για να αποφευχθεί η κίνηση κατά τη διαδικασία λήψης εικόνας. Αυτή η σταθερότητα είναι μείζονος σημασίας για την ακριβή προσαρμογή της κάμερας και για τα βέλτιστα επιδιωκόμενα αποτελέσματα. Προς τούτοις, απαραίτητη είναι η χρήση του κατάλληλου και διάχυτου φωτισμού, για να γίνεται αντιληπτό με ακρίβεια το μοτίβο. Σκιές ή ανισομερής φωτισμός, θα επηρεαστεί την ανίχνευση των χαρακτηριστικών του μοτίβου.

Αναγκαία κρίνεται η δοκιμή φωτογράφισης μιας σειράς εικόνων, διασφαλίζοντας ότι το μοτίβο είναι ορατό σε κάθε πλαίσιο. Ενδείκνυται η εναλλαγή των οπτικών γωνιών της κάμερας, αλλάζοντας γωνίες, αποστάσεις και προσανατολισμούς, καθώς βοηθά στην επίτευξη μίας καλύτερης απεικόνισης του μοτίβου. Οφείλει κανείς να βεβαιωθεί το περιεχόμενο μιας εικόνας εκτείνεται και στο περιεχόμενο της επόμενης εικόνας. Αυτό διασφαλίζει ότι υπάρχουν αναγνωρίσιμα σημεία ή χαρακτηριστικά (όπως γωνίες ενός μοτίβου σκακιέρας) που είναι ορατά σε πολλές εικόνες.

Τα κοινά χαρακτηριστικά σε επικαλυπτόμενες περιοχές συμβάλλουν στη δημιουργία αντιστοιχιών μεταξύ διαφορετικών εικόνων, καθιστώντας την βαθμονόμηση πιο ισχυρή και αξιόπιστη. Επιπροσθέτως, η σταθερή διατήρηση των ρυθμίσεων ζουμ και εστίασης σε όλες τις εικόνες είναι θεμελιώδης για να επικρατεί

[12] OpenCV. (2019). Camera Calibration. OpenCV Documentation. https://docs.opencv.org/3.4/dc/dbb/tutorial_py_calibration.html
[13] MATLAB. (2019, July 12). Camera Calibration with MATLAB [Video]. YouTube. https://www.youtube.com/watch?v=x6YIwoQBBxA

η ομοιομορφία. Η αλλαγή αυτών των ρυθμίσεων μπορεί να επηρεάσει το εμφανές μέγεθος και τη σαφήνεια του μοτίβου βαθμονόμησης. Ακολούθως, η μεταβλητότητα στις παραμέτρους της κάμερας, όπως ο χρόνος έκθεσης και η ισορροπία λευκού, συνδράμει να ληφθούν υπόψη οι διαφορετικές συνθήκες που μπορεί να αντιμετωπιστούν σε σενάρια πραγματικού κόσμου.

΄Υστερα, η λήψη εικόνων από διαφορετικές αποστάσεις όχι μόνον από κοντά, αλλά και μακρινά πλάνα, θα έχει ως αποτέλεσμα αυτή η ποικιλομορφία στη διαδικασία βαθμονόμησης να φιλοξενήσει διαφορετικά σενάρια. Στην ουσία, καταγράφοντας εικόνες σε διάφορες αποστάσεις, εξετάζεται πώς ο τρισδιάστατος χώρος (βάθος) επηρεάζει την παραμόρφωση που παρατηρείται στις δισδιάστατες εικόνες. Η συμπερίληψη τόσο των κοντινών, όσο και των μακρινών λήψεων παρέχει ένα πιο ολοκληρωμένο σύνολο δεδομένων για τον αλγόριθμο βαθμονόμησης για την ακριβή μοντελοποίηση και διόρθωση στρεβλώσεων που μπορεί να προκύψουν σε διαφορετικές καταστάσεις. [14]

Στόχος είναι οι εικόνες υψηλής ανάλυσης για τη λήψη λεπτών λεπτομερειών του μοτίβου βαθμονόμησης. Αυτό επιτυγχάνεται ακολουθώντας ορισμένα βήματα. Πρώτα, γίνεται μία ρύθμιση ή διαμόρφωση της κάμερας που να επιτρέπει τη λήψη εικόνων με υψηλό επίπεδο λεπτομέρειας. Υψηλότερη ανάλυση σημαίνει ότι η εικόνα έχει περισσότερα pixels, παρέχοντας μια σαφέστερη και πιο λεπτομερή αναπαράσταση του σκηνικού. Έπειτα, έπεται η καταγραφή λεπτών λεπτομερειών. Το μοτίβο βαθμονόμησης πιθανότατα έχει περίπλοκες λεπτομέρειες, όπως γωνίες ή συγκεκριμένα χαρακτηριστικά. Η καταγραφή των λεπτών λεπτομερειών σημαίνει ότι διασφαλίζεται ότι αυτά τα μικρότερα στοιχεία είναι ορθά αποτυπωμένα στην εικόνα. Ένα σημαντικό βήμα αποτελεί και η ακριβής ανίχνευση χαρακτηριστικών. Πιο ειδικά, η ανίχνευση χαρακτηριστικών περιλαμβάνει τον εντοπισμό συγκεκριμένων σημείων ή χαρακτηριστικών σε μια εικόνα, όπως γωνίες ενός μοτίβου σκακιέρας.

Οι εικόνες υψηλότερης ανάλυσης συμβάλλουν στην ακριβέστερη ανίχνευση χαρακτηριστικών, επειδή υπάρχουν περισσότερες διαθέσιμες πληροφορίες για την ανάλυση του αλγορίθμου. Άρα, το ιδανικό σενάριο είναι να υπάρχει στην κατοχή μια κάμερα ικανή να καταγράφει εικόνες με υψηλό επίπεδο λεπτομέρειας. Και αυτό διότι, όσο πιο λεπτομερείς είναι οι εικόνες, τόσο καλύτερα ο αλγόριθμος βαθμονόμησης μπορεί να εντοπίσει και να χρησιμοποιήσει βασικά σημεία για ακριβή διόρθωση παραμόρφωσης. Ακόμη, είναι δηλωτικός ο επαρκής αριθμός εικόνων, συνήθως κατά προσέγγιση 20-30 καρέ, για να αποκτήσει ο αλγόριθμος άφθονα δεδομένα για βαθμονόμηση. Όσο πιο ποικίλες και καλά κατανεμημένες είναι οι εικόνες, τόσο καλύτερα είναι τα αποτελέσματα βαθμονόμησης.

Με τις εγγενείς παραμέτρους που καθορίζονται, το τελικό βήμα είναι η εφαρμογή αυτών των πληροφοριών για την αφαίρεση των εικόνων που έχουν ληφθεί και δεν

[14] Zhengyou Zhang. (2000). A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334.

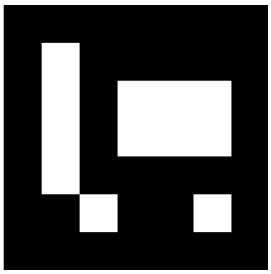
ανταποκρίνονται στις παραμέτρους. Με τη διόρθωση των στρεβλώσεων, οι εικόνες που προκύπτουν παρέχουν μια πιο ακριβή αναπαράσταση της σκηνής, διευκολύνοντας την αξιόπιστη ανάλυση της μηχανικής όρασης.

2.3: ArUco marker system

2.3.1: Τι είναι οι δείκτες ArUco

Ένας δείκτης ArUco[15] είναι ένας συνθετικός τετράγωνος δείκτης που αποτελείται από ένα ευρύ μαύρο περίγραμμα και μια εσωτερική δυαδική μήτρα που καθορίζει το αναγνωριστικό του. Οι δείκτες ArUco είναι 2D δυαδικά κωδικοποιημένα συμβατικά μοτίβα που έχουν σχεδιαστεί για να εντοπίζονται γρήγορα από οπτικά συστήματα στους υπολογιστές. Χρησιμοποιούνται για την εκτίμηση της στάσης της κάμερας και η ανίχνευσή τους είναι ισχυρή, γρήγορη και απλή.[16] Ο προσδιορισμός της στάσης της κάμερας μέσα από εικόνες περιλαμβάνει την καθιέρωση αντιστοιχιών μεταξύ αναγνωρίσιμων σημείων στο περιβάλλον και των προβολών τους στην κάμερα. Ενώ, ορισμένες μέθοδοι επικεντρώνονται στον εντοπισμό φυσικών χαρακτηριστικών, όπως βασικά σημεία (key points), η χρήση δεικτών είναι πολύ εύχρηστη λόγω της ευκολίας ανίχνευσής τους, επιτρέποντας την εκτίμηση με υψηλή ταχύτητα και την αναγνώριση της ακριβούς τοποθεσίας και στάσης της κάμερας.

Το πρότζεκτ ArUco αναπτύχθηκε από τους Rafael Munoz και Sergio Garrido.[17]Η αυτοματοποιημένη μονάδα βασίζεται στη βιβλιοθήκη ArUco, μια δημοφιλή βιβλιοθήκη για την ανίχνευση των δεικτών. Οι διάφορες λειτουργίες της βιβλιοθήκης επιστρέφουν αναγνωριστικά μοτίβου και θέτουν πληροφορίες από μία σαρωμένη εικόνα. Το κύριο όφελος αυτών των δεικτών είναι ότι ένας μόνο δείκτης παρέχει αρκετές αντιστοιχίες και η εσωτερική δυαδική κωδικοποίηση τους καθιστά ιδιαίτερα ισχυρούς, επιτρέποντας τη δυνατότητα εφαρμογής τεχνικών ανίχνευσης και διόρθωσης σφαλμάτων.



Εικόνα 10 ArUco marker (Id: 82)

[15] OpenCV. (2019). Detection of ArUco Markers. OpenCV Documentation. https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html#:~:text=An%20ArUco%20marker%20is%20a,determines%20its%20identifier%20(id)

[16] Muñoz-Salinas, R., & Garrido-Jurado, S. (2014). Fast detection of fiducial markers. Journal of Real-Time Image Processing, 10(4), 599-609. https://doi.org/10.1007/s11554-014-0447-x

[17] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition, 47(6), 2280-2292.

2.3.2: Δείκτες ArUco και Computer Vision

Το σύστημα σήμανσης ArUco, αναπόσπαστο συστατικό στοιχείο των εφαρμογών όρασης υπολογιστή, αντιπροσωπεύει μια κατηγορία εμπιστευτικών σημάτων που χαρακτηρίζονται από διακριτικά τετράγωνα μοτίβα. Αναπτύχθηκε ως μέρος της βιβλιοθήκης OpenCV, οι δείκτες ArUco έχουν σχεδιαστεί για να χρησιμεύουν ως ισχυρά σημεία αναφοράς για προσανατολισμό και εντοπισμό σε περιβάλλοντα πραγματικού κόσμου. Η ουσία του συστήματος ArUco έγκειται στη μοναδική διάταξη ασπρόμαυρων τετραγώνων, σχηματίζοντας εύκολα ανιχνεύσιμα μοτίβα. Κάθε δείκτης διαθέτει ένα συγκεκριμένο αναγνωριστικό κωδικοποιημένο εντός του προτύπου του, επιτρέποντας την απλή διαφοροποίηση μεταξύ των δεικτών. Αυτή η διαδικασία αναγνώρισης, που βασίζεται στις αρχές ανίχνευσης και διόρθωσης σφαλμάτων, εξασφαλίζει ανθεκτικότητα σε προκλήσεις όπως είναι οι διακυμάνσεις των συνθηκών φωτισμού. Ένα καθοριστικό χαρακτηριστικό των δεικτών ArUco είναι η συμπερίληψη ενός μαύρου περιγράμματος, η ενίσχυση της ανιχνευσιμότητας τους και η απλοποίηση της διαδικασίας ταυτοποίησης.[18]

Η απλότητα και η αποτελεσματικότητα των δεικτών ArUco τα καθιστούν ευέλικτα σε διάφορες εφαρμογές. Στον τομέα της βαθμονόμησης της κάμερας, αυτοί οι δείκτες είναι καθοριστικοί για τον προσδιορισμό των εγγενών παραμέτρων της κάμερας όπως η εστιακή απόσταση και οι συντελεστές παραμόρφωσης. Πέρα από τη βαθμονόμηση, οι δείκτες ArUco βρίσκουν εφαρμογή στην εκτίμηση της θέσης αντικειμένων, ενεργώντας ως κρίσιμα σημεία αναφοράς για την εκτίμηση της θέσης και του προσανατολισμού των αντικειμένων μέσα σε μια σκηνή. Η επαυξημένη πραγματικότητα επωφελείται από τους δείκτες ArUco ως σημεία “αγκύρωσης”, διευκολύνοντας την ακριβή τοποθέτηση και ευθυγράμμιση εικονικών αντικειμένων μέσα σε ένα πραγματικό περιβάλλον. Στο πλαίσιο του εντοπισμού drone, οι δείκτες ArUco διαδραματίζουν καθοριστικό ρόλο παρέχοντας σημεία αναφοράς για τη θέση και τον προσανατολισμό του drone κατά τη διάρκεια της πτήσης. Αυτή η ικανότητα είναι ιδιαίτερα πολύτιμη για εργασίες όπως η προσγείωση ακριβείας, η πλοήγηση και η αποφυγή εμποδίων.

Ένα βασικό πλεονέκτημα των δεικτών ArUco είναι η προσαρμοστικότητά τους σε εφαρμογές σε πραγματικό χρόνο, χάρη στο χαμηλό υπολογιστικό κόστος και τους αλγόριθμους γρήγορης ανίχνευσης. Οι δείκτες συνδέονται στρατηγικά με επιφάνειες ή αντικείμενα μέσα στο περιβάλλον, επιτρέποντας σε ένα σύστημα εξοπλισμένο με κάμερα να τους ανιχνεύει και να τους αναγνωρίζει γρήγορα. Το μαύρο περίγραμμα, ένα διακριτικό χαρακτηριστικό των δεικτών ArUco, βοηθά στη δημιουργία ενός σαφούς ορίου για ακριβή ανίχνευση. Αυτή η προσαρμοστικότητα και η αποτελεσματικότητα καθιστούν τους δείκτες ArUco μια πρακτική επιλογή για ενσωμάτωση σε συστήματα όρασης υπολογιστή, όπου η απλότητά τους δεν θέτει σε κίνδυνο την ακρίβεια.

Στην ουσία, το σύστημα σήμανσης ArUco αντιπροσωπεύει μια εξελιγμένη αλλά προσβάσιμη λύση για την ενσωμάτωση εμπιστευτικών δεικτών σε εφαρμογές όρασης υπολογιστή. Οι αρχές του έχουν τις ρίζες τους στην παροχή αξιόπιστων σημείων αναφοράς για τον χωρικό προσανατολισμό και τον εντοπισμό,

[18] OpenCV. (2019). ArUco Marker Detection (aruco module). OpenCV Documentation. https://docs.opencv.org/4.x/d9/d6d/tutorial_table_of_content_aruco.html

εξασφαλίζοντας την ευρωστία του συστήματος ενόψει των ποικίλων περιβαλλοντικών συνθηκών. Τα διακριτικά χαρακτηριστικά των δεικτών ArUco, συμπεριλαμβανομένων των μοναδικών μοτίβων τους, των μηχανισμών ανίχνευσης σφαλμάτων και των αποτελεσματικών αλγορίθμων ανίχνευσης, συμβάλλουν στην ευρεία υιοθέτησή τους σε εφαρμογές που κυμαίνονται από τη βαθμονόμηση της κάμερας έως την επαυξημένη πραγματικότητα και τον εντοπισμό drone.

2.3.3: ArUco markers και Augmented reality (AR)

Η εικονική πραγματικότητα (AR) εντάσσεται στην πρώτη γραμμή των μετασχηματιστικών τεχνολογιών, αναδιαμορφώνοντας τον τρόπο που αντιλαμβανόμαστε και αλληλοεπιδρούμε με τον κόσμο γύρω μας. Σε αντίθεση με την εικονική πραγματικότητα, η οποία εισάγει τους χρήστες σε ένα εντελώς τεχνητό περιβάλλον, η εικονική πραγματικότητα ενσωματώνει στοιχεία παραγόμενα από τον υπολογιστή που στον πραγματικό κόσμο. Μέσω της ενσωμάτωσης των ψηφιακών πληροφοριών στο πραγματικό περιβάλλον, η AR ενισχύει τις αισθητηριακές μας εμπειρίες, δημιουργώντας μια δυναμική αλληλεπίδραση μεταξύ του εικονικού και του απτού περιβάλλοντος.

Οι εφαρμογές της εικονικής πραγματικότητας καλύπτουν μια πληθώρα βιομηχανιών, από τα παιχνίδια και την ψυχαγωγία έως την υγειονομική περίθαλψη, την εκπαίδευση και τις κατασκευές. Αυτή η δυναμική τεχνολογία έχει εξελιχθεί πέρα από τον αρχικό σκοπό που ήταν με γνώμονα την ψυχαγωγία, καθιστώντας την ένα ανεκτίμητο εργαλείο για την ενίσχυση της παραγωγικότητας, τη βελτίωση των μαθησιακών εμπειριών και την επανάσταση σε διάφορους επαγγελματικούς τομείς.

Κεντρικό στοιχείο της λειτουργικότητας της εικονικής πραγματικότητας είναι τα συστήματα σήμανσης. Μεταξύ αυτών των συστημάτων σήμανσης, το σύστημα σήμανσης ArUco αναδεικνύεται ως βασικό στοιχείο, προσφέροντας ένα μοναδικό σύνολο χαρακτηριστικών και δυνατοτήτων που συμβάλλουν στη συνεχή ενσωμάτωση εικονικών στοιχείων στην καθημερινή μας ζωή.

Καθώς ερευνούμε ενδελεχώς το σύστημα δεικτών ArUco, είναι αναγκαίο να εμβαθύνουμε στα θεμέλια της εικονικής πραγματικότητας, να κατανοήσουμε το ρόλο των συστημάτων δεικτών και να αναγνωρίσουμε το μετασχηματιστικό δυναμικό που φέρνουν οι δείκτες ArUco σε αυτό το ταχέως εξελισσόμενο τεχνολογικό περιβάλλον. Μέσω αυτής της έρευνας, στοχεύουμε να αναλύσουμε την πολυπλοκότητα των δεικτών ArUco, να εξετάσουμε τις εφαρμογές τους σε διάφορους κλάδους και να προβλέψουμε τις μελλοντικές τάσεις που θα διαμορφώσουν την εξέλιξη της εικονικής πραγματικότητας. [19]

Το ArUco είναι πολύ σημαντικό για εφαρμογές AR που βασίζονται αποκλειστικά στο OpenCV (Open Source Computer Vision Library) διότι βασίζεται

[19] Fuentes-Pacheco, J., Ruiz-Suarez, J. C., & Sucar, L. E. (2014). "A Simple and Robust Approach for Marker-Based Augmented Reality."

σε ασπρόμαυρους δείκτες με κωδικούς που ανιχνεύονται αξιοποιώντας μία μόνο λειτουργία.[20]

Η ανίχνευση των δεικτών ArUco, και των υπολοίπων δεικτών, επηρεάζονται από τον θόρυβο, και τη μη διαύγεια (θολούρα) παρά τη σχετική ανθεκτικότητα τους στις διακυμάνσεις του φωτός.[21]

2.3.4: Χρήσεις των ArUco marker στον πραγματικό κόσμο (real life)

Οι δείκτες ArUco χρησιμοποιούνται ευρέως στη ρομποτική για διεργασίες, όπως είναι ο εντοπισμός και η πλοήγηση ρομπότ. Ο εντοπισμός και η πλοήγηση ρομπότ αναφέρονται στις διαδικασίες μέσω των οποίων ένα ρομπότ καθορίζει τη δική του θέση και προσανατολισμό μέσα σε ένα περιβάλλον (εντοπισμός) και σχεδιάζει μια διαδρομή για να μετακινηθεί από τη μια θέση στην άλλη (πλοήγηση). Αυτές οι διαδικασίες είναι απαραίτητες για να λειτουργούν αποτελεσματικά τα αυτόνομα ρομπότ σε διάφορα περιβάλλοντα, όπως βιομηχανικές εγκαταστάσεις, αποθήκες και δημόσιους χώρους. Ο εντοπισμός συχνά περιλαμβάνει τη χρήση αισθητήρων, όπως IMU και οδομετρία, για την εκτίμηση της θέσης του ρομπότ σε σχέση με το περιβάλλον του, ενώ η πλοήγηση περιλαμβάνει τη δημιουργία σχεδίων διαδρομής και την εκτέλεση εντολών ελέγχου για τη μετακίνηση του ρομπότ στην επιθυμητή θέση του. Τα ρομπότ που είναι εξοπλισμένα με κάμερες μπορούν να αναγνωρίσουν τους δείκτες ArUco στο περιβάλλον τους, επιτρέποντάς τους να προσδιορίσουν με ακρίβεια τη θέση τους. [22]

Ταυτόχρονα, οι δείκτες ArUco χρησιμοποιούνται για τη ρύθμιση και προσαρμογή της κάμερας, ένα ουσιαστικό στάδιο στον τομέα της μηχανικής όρασης. Με τη λήψη εικόνων γνωστών δεικτών ArUco από διαφορετικές γωνίες, οι εγγενείς παράμετροι μιας κάμερας μπορούν να προσδιοριστούν με ακρίβεια.[23]

Συγχρόνως, έχει την ευχέρεια κανείς να αξιοποιήσει τους δείκτες ArUco και να τους χρησιμοποιήσει σε εκπαιδευτικά περιβάλλοντα για πρακτικές μαθησιακές εμπειρίες. Οι εκπαιδευόμενοι δύνανται να πειραματιστούν με τον εντοπισμό με βάση το δείκτη και την εικονική πραγματικότητα, αποκτώντας πρακτικές γνώσεις σχετικά με τις έννοιες της μηχανικής όρασης.[24] Το εικονικό περιεχόμενο, όπως τρισδιάστατα μοντέλα μπορεί να συσχετιστεί με συγκεκριμένους δείκτες ArUco, δημιουργώντας διαδραστικές μαθησιακές εμπειρίες.

---

[20] Nayak, S. (2020). Augmented Reality using ArUco Markers in OpenCV (C++ / Python). Learn OpenCV. https://learnopencv.com/augmented-reality-using-aruco-markers-in-opencv-c-python/

[21] Fiala, M. ARTag, a fiducial marker system using digital techniques. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 2, pp. 590–596.

[22] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). "Automatic generation and detection of highly reliable fiducial markers under occlusion."

[23] Bouguet, J. Y. (2008). "Camera Calibration Toolbox for Matlab."

[24] Smisek, J., Stava, O., & Polok, L. (2013). "OpenCV and ARToolKit comparison for natural feature tracking."

Επιπρόσθετα, οι δείκτες ArUco στρατηγικά τοποθετημένοι σε εσωτερικούς ή εξωτερικούς χώρους χρησιμεύουν ως βοηθήματα στις εφαρμογές πλοήγησης AR. Οι χρήστες λαμβάνουν καθοδήγηση μέσω εικονικών ενδείξεων που εμφανίζονται στις οθόνες της συσκευής τους, κατευθύνοντάς τους σε συγκεκριμένες τοποθεσίες αναγνωρίζοντας τους δείκτες ArUco. Αντί να βασίζονται σε παραδοσιακούς χάρτες ή απλές κατευθύνσεις, αυτά τα συστήματα αξιοποιούν τους δείκτες ArUco για να παρέχουν στους χρήστες διαισθητικές και οπτικά καθοδηγούμενες εμπειρίες πλοήγησης. Όταν οι χρήστες ενεργοποιούν την εφαρμογή πλοήγησης AR, η κάμερα της συσκευής τους σαρώνει το περιβάλλον για αναγνωρίσιμους δείκτες ArUco. Μόλις εντοπιστούν, οι δείκτες λειτουργούν ως σημεία αναφοράς που ενεργοποιούν εικονικές πληροφορίες, ενσωματώνοντας αυτούσια το πραγματικό περιβάλλον όπως αυτό συλλαμβάνεται από την κάμερα. Το σύστημα AR αναγνωρίζει και παρακολουθεί συνεχώς τους δείκτες ArUco. Τα εικονικά βέλη προσδιορίζουν τον προσανατολισμό και την τοποθέτησή τους στην οθόνη της συσκευής, καθοδηγώντας τους χρήστες κατά μήκος προς την επιθυμητή διαδρομή. Αυτή η προσέγγιση προσφέρει μια διαδραστική και συναρπαστική εμπειρία πλοήγησης.

2.3.5: Χρήσεις των ArUco marker σε μη επανδρωμένα αεροσκάφη

Οι δείκτες ArUco βοηθούν τα drones με διάφορους τρόπους, αξιοποιώντας τεχνικές όρασης υπολογιστή για βελτιωμένη πλοήγηση, εντοπισμό και έλεγχο. Αρχικά, βοηθούν στην τοπικοποίηση (localization), αφού οι δείκτες ArUco λειτουργούν ως οπτικά σημεία αναφοράς που είναι εύκολα ανιχνεύσιμα από την ενσωματωμένη κάμερα του drone.[25] Τα μοναδικά μοτίβα στους δείκτες ArUco επιτρέπουν γρήγορη και ακριβή αναγνώριση, επιτρέποντας στο drone να καθορίσει τη θέση του σε σχέση με τους δείκτες. Ακόμη, συμβάλλουν στη θέση εκτίμησης, αναγνωρίζοντας πολλαπλούς δείκτες ArUco στο περιβάλλον, η μηχανική όραση του drone μπορεί να εκτιμήσει τη στάση του, η οποία περιλαμβάνει πληροφορίες σχετικά με τη θέση και τον προσανατολισμό του. Αυτή η εκτίμηση θέσης είναι ιδιαιτέρως σημαντική για την πλοήγηση[26], καθώς παρέχει στο drone μια σαφή κατανόηση της χωρικής σχέσης του με τους δείκτες. Το πιο αξιοσημείωτο που προσφέρουν οι δείκτες αυτοί είναι προσγείωση με ακρίβεια. Το drone χρησιμοποιεί αυτούς τους δείκτες για να καθοδηγήσει τον εαυτό του κατά την κάθοδο.

[25] Mráz, E., Rodina, J., & Babinec, A. (2020). Using fiducial markers to improve localization of a drone. In 2020 23rd International Symposium on Measurement and Control in Robotics (ISMCR) (pp. 1-6). IEEE. DOI: 10.1109/ISMCR51255.2020.9263754

[26] Wang, G., Liu, Z., & Wang, X. (October 2019). UAV Autonomous Landing using Visual Servo Control based on Aerostack. In the 3rd International Conference. DOI: 10.1145/3331453.3361667

Εικόνα 11 ArUco marker with drone

Οι δείκτες βοηθούν στην προσαρμογή της πλοήγησης του μη επανδρωμένου αεροσκάφους, διασφαλίζοντας ότι ευθυγραμμίζεται με ακρίβεια με τον στόχο προσγείωσης για ελεγχόμενη και ακριβή προσγείωση. Εκτός από αυτό, οι δείκτες ArUco μπορούν να χρησιμοποιηθούν για τον καθορισμό μιας προκαθορισμένης διαδρομής ή σημείων για να ακολουθήσει το drone. Η μηχανική όραση του drone ανιχνεύει και παρακολουθεί τους δείκτες κατά μήκος της διαδρομής του, βοηθώντας στην πλοήγηση και παρέχοντας ενδείξεις για διόρθωση πορείας. Χάρη στους δείκτες ArUco, όταν τοποθετούνται στρατηγικά, μπορούν να βοηθήσουν τα αεροσκάφη να εντοπίσουν εμπόδια στο περιβάλλον τους, ώστε να επιτύχουν τα προαναφερόμενα.

Πιο συγκεκριμένα, μπορούν να αναγνωρίσουν αποκλίσεις από τις αναμενόμενες διαμορφώσεις δεικτών, επιτρέποντας στο UAV να λαμβάνει αποφάσεις σε πραγματικό χρόνο για να αποφύγει εμπόδια. Πέραν τούτου, τα drones που είναι εξοπλισμένα με κάμερες μπορούν να χρησιμοποιήσουν δείκτες ArUco για να δημιουργήσουν χάρτες του περιβάλλοντός τους. Καθώς το drone κινείται μέσα από μια περιοχή, αναγνωρίζει και καταγράφει τις θέσεις των δεικτών ArUco, συμβάλλοντας στη δημιουργία ενός οπτικού χάρτη που μπορεί να χρησιμοποιηθεί για περαιτέρω εξερεύνηση. [27]

Εν συνεχεία, οι δείκτες ArUco χρησιμοποιούνται ορισμένες φορές σε εφαρμογές, όπου τα drones αλληλοεπιδρούν με ανθρώπους ή αντικείμενα. Για παράδειγμα, οι δείκτες σε αντικείμενα μπορούν να διευκολύνουν την ικανότητα του drone να κατανοήσει ή να αλληλοεπιδράσει μαζί τους με προκαθορισμένο τρόπο. Συνοπτικά, οι δείκτες ArUco ενισχύουν τις δυνατότητες των drone παρέχοντας οπτικές ενδείξεις που βοηθούν στον εντοπισμό, την πλοήγηση και συγκεκριμένες εργασίες, όπως η προσγείωση ακριβείας ή η αποφυγή εμποδίων. Η απλότητα, η ευκολία ανίχνευσης και τα μοναδικά πρότυπα ταυτοποίησης τους καθιστούν πολύτιμα εργαλεία στον τομέα της τεχνολογίας drone.

[27] Marut, A., Wojtowicz, K., & Falkowski, K. (2019). ArUco markers pose estimation in UAV landing aid system. IEEE Xplore. https://ieeexplore.ieee.org/document/8869572

ΚΕΦΑΛΑΙΟ 3

HARDWARE

3.1: Drone Components

- Frame

Χρησιμοποιήθηκε πλαίσιο από Carbon Fiber[28] μεγέθους 500mm. Οι ίνες άνθρακα, είναι γνώστες για την ανθεκτικότητα τους καθώς και το χαμηλό τους βάρος. Με την ελαχιστοποίηση του συνολικού βάρους του μη επανδρωμένου αεροσκάφους, ενισχύεται η αποδοτικότητα της πτήσης και η εξοικονόμηση ενέργειας. Η δύναμη των ίνας άνθρακα διαδραματίζει έναν κρίσιμο ρόλο στην ενίσχυση της δομικής ακεραιότητας του drone. Η ανθεκτικότητα που προσφέρει το Carbon Fiber πλαίσιο είναι απαραίτητη για την αντοχή στις πιέσεις των επαναλαμβανόμενων προσγειώσεων, απογειώσεων καθώς και πιθανών πτώσεων, διατηρώντας ακέραια τα κρίσιμα εξαρτήματα όπως ο ελεγκτής πτήσης, το Raspberry Pi 5 και το σύστημα της κάμερας. Το σχετικά μεγάλο μέγεθος πλαισίου προσφέρει ευελιξία στην τοποθέτηση των απαιτούμενων εξαρτημάτων και αισθητήρων χωρίς να υπάρχει μεγάλο αντίκτυπο στο βάρος και την απόδοση. Αξιοποιεί τα πλεονεκτήματα της ευκινησίας, της σταθερότητας και της ανθεκτικότητας σε περιβαλλοντικούς παράγοντες.

Εικόνα 12 500mm Carbon fiber Frame

- Electronic Speed Controller (ESC)

Το ESC λαμβάνει σήματα από το flight controller και τα μεταφράζει σε κίνηση των motors. Η τροφοδοσία ρεύματος που παρέχεται στους κινητήρες διέρχεται πρώτα από το ESC και ως συνέπεια η ταχύτητα και η κατεύθυνση περιστροφής μπορεί να ρυθμιστεί για τον κάθε κινητήρα ξεχωριστά. Η επιλογή τεσσάρων ESC με ισχύ 30 Ampere αποδεικνύεται βέλτιστη για το συγκεκριμένο σύστημα.

[28] FPVKing. (2019). FPVKing 500 X4 Quadcopter Frame Kit Upgrade Tall Landing Skid Gear. Amazon. https://www.amazon.com/FPVKing-500-X4-Quadcopter-Upgrade-Landing/dp/B087LT81C8

Τα τεχνικά χαρακτηριστικά τους είναι:

-Continuous Current: 30A
-Burst current(10S): 40A
-Li-xx battery(cell): 2-4
-Dimension L*W*H(mm): 52x26x7
-BEC Mode: Linear
-BEC Output: 2A/5V
-Programmable: yes
-Weight: 28g / pc

Εικόνα 13 BLHeli 30A ESC

- Motors

Επιλέχθηκε ένα σετ από 2212 brushless motor[29] 920 Kv. Είναι κατάλληλοι για αυτό το μέγεθος πλαισίου και προσφέρουν την απαιτούμενη ισχύ (throttle) που χρειάζεται. Τα βασικά πλεονεκτήματα των brushless έναντι των brushed motor είναι η υψηλότερη αποδοτικότητα, η μακρύτερη διάρκεια ζωής και οι μειωμένες απαιτήσεις συντήρησης.

Τα brushless motors είναι γνωστά για την παροχή καλύτερων αναλογιών ισχύος-βάρους, αυξημένη ροπή και βελτιωμένη συνολική απόδοση. Το τμήμα που περιστρέφεται περιέχει μόνιμους μαγνήτες ενώ το σταθερό τμήμα αποτελείται από πηνία.

Το ESC διαχειρίζεται με ακρίβεια το χρονισμό και την κατεύθυνση της ροής ρεύματος στα πηνία, δημιουργώντας ένα περιστρεφόμενο μαγνητικό πεδίο που αλληλοεπιδρά με τους μόνιμους μαγνήτες για να παράγει μηχανική περιστροφή.

Τα τεχνικά χαρακτηριστικά τους είναι:

-KV: 920
- Max Efficiency: 80%
- Max Efficiency Current: 4-10A (>75%)
- Current Capacity: 12A/60s
- No Load Current @ 10V: 0.5A
- No. Of Cells: 2-3 Li-Poly
- Motor Dimensions: Φ27.5 x 30mm

[29] EMAX. (2021). 2212 920KV CW/CCW Brushless Motor for DJI Phantom 1, Phantom 2, F450. EMAX Model. https://emaxmodel.com/products/2212-920kv-cw-ccw-brushless-motor-for-dji-phantom-1-phantom-2-f450

- Shaft Diameter: Φ3.17mm
- Weight: 47g

Εικόνα 14 2212 motors

- Propellers

Οι προπέλες 1045[30] έχουν διάμετρο 10 inches (25.4 cm) και Slope: 4.5 inches (11.43 cm). Είναι κατασκευασμένες από Nylon/ABS.

Εικόνα 15 1045 propellers

- Flight Controller

Το πιο σημαντικό component του συστήματος είναι ο ελεγκτής πτήσεως. Επιλέχθηκε το Pixhawk 2.4.8[31] το οποίο είναι συμβατό με ardupilot. Το κύριο πλεονέκτημα του είναι το γεγονός ότι μπορεί να λειτουργήσει σε συνδυασμό με ένα companion pc όπως ένα Raspberry pi. Αυτό προσφέρει στον χρήστη άπειρες επιλογές διαμόρφωσης μιας πτήσης. Το Flight Controller πρόκειται για μια εξειδικευμένη ηλεκτρονική συσκευή που διαχειρίζεται και ελέγχει τις πτητικές λειτουργίες του drone. Ο ελεγκτής πτήσης ερμηνεύει την είσοδο από διάφορους αισθητήρες, όπως επιταχυνσιόμετρα και γυροσκόπια, για να αξιολογήσει τον

[30] Aries RC. (2019). 1 Pair New Upgraded 1045 Propeller CW/CCW Blade for 2212/2216 Motor Self-locking Multicopter Drone Spare Parts. Aries RC. https://www.ariesrc.gr/en/propellers/18834-1-pair-new-upgraded-1045-propeller-cw-ccw-blade-for-22122216-motor-self-locking-multicopter-drone-spare-parts.html

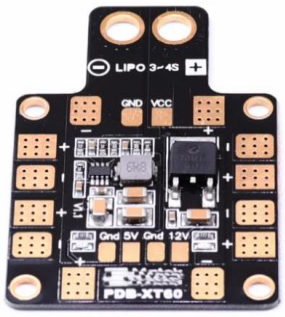
[31] ArduPilot Development Team. (2021). Pixhawk Overview. ArduPilot Documentation. https://ardupilot.org/copter/docs/common-pixhawk-overview.html

προσανατολισμό και την κίνηση του drone. Χρησιμοποιώντας αυτές τις πληροφορίες, ο ελεγκτής πτήσης ρυθμίζει την ταχύτητα των κινητήρων του drone για να διατηρήσει τη σταθερότητα και να εκτελέσει της επιθυμητές κινήσεις. Επιπλέον, οι ελεγκτές πτήσης διαθέτουν συχνά λογισμικό που επιτρέπει στους χρήστες να ορίζουν παραμέτρους πτήσης, σημεία πορείας και άλλες λειτουργικές ρυθμίσεις, επιτρέποντας αυτόνομες ή ημιαυτόνομες δυνατότητες πτήσης.

Εικόνα 16 PixHawk 2.4.8

- Power Distribution Board (PDB)

Το PDB είναι ένα κρίσιμο στοιχείο που χρησιμεύει για τη διανομή ηλεκτρικής ενέργειας από την κύρια μπαταρία σε διάφορα ηλεκτρονικά εξαρτήματα εντός του συστήματος. Το PDB συνδέεται συνήθως με την κύρια μπαταρία και παρέχει ρυθμιζόμενη ισχύ σε εξαρτήματα όπως ελεγκτές πτήσης, ηλεκτρονικούς ελεγκτές ταχύτητας (ESC), κάμερες και άλλα αξεσουάρ. Συχνά περιλαμβάνει χαρακτηριστικά όπως ρυθμιστές τάσης, αισθητήρες ρεύματος και μερικές φορές συνδέσμους για εύκολη ενσωμάτωση στο συνολικό ηλεκτρικό σύστημα του drone. Η κύρια λειτουργία ενός PDB είναι να διανέμει αποτελεσματικά την ισχύ, εξασφαλίζοντας παράλληλα ότι κάθε στοιχείο λαμβάνει μια σταθερή και κατάλληλη τάση. Βοηθά στην οργάνωση της καλωδίωσης και των συνδέσεων μέσα στο σύστημα, καθιστώντας την ηλεκτρική ρύθμιση πιο εύχρηστη και αξιόπιστη. Στο συγκεκριμένο σύστημα επιλέχθηκε ένα απλό PDB[32] το οποίο είναι συμβατό με 3-4S lithium - polymer battery (LiPo).



Εικόνα 17 PDB

[32] Matek Systems. (2019). PDB-XT60 Manual. Matek Systems. http://www.mateksys.com/downloads/PDB-XT60_Manual_EN.pdf

- Global Positioning System (GPS) module

Το GPS Module επιτρέπει στα μη επανδρωμένα αεροσκάφη να καθορίσουν την ακριβή τους θέση μέσω σημάτων που λαμβάνονται από τους δορυφόρους προσδιορισμού θέσεως. Υπολογίζει το γεωγραφικό πλάτος, το γεωγραφικό μήκος, το υψόμετρο. Τα περισσότερα συστήματα GPS για drone έχουν ενσωματωμένη πυξίδα. Επιλέχθηκε το M8N GPS Module[33].

Εικόνα 18 M8N GPS Module

- Lithium-Polymer (LiPo) Battery

Η LiPo είναι μια επαναφορτιζόμενη μπαταρία που έχει ως κύρια χαρακτηριστικά την υψηλή απόδοση και το χαμηλό βάρος. Αξιοποιώντας την τεχνολογία ιόντων λιθίου, οι μπαταρίες αυτές προσφέρουν σταθερή τάση εξόδου κατά την εκφόρτιση, καθιστώντας τις κατάλληλες για εφαρμογές που απαιτούν σταθερή ισχύ. Μπορούν να παρέχουν υψηλά ποσοστά ενέργειας και είναι η κατάλληλη επιλογή για drone, καθώς αυτά μερικές φορές απαιτούν εκρήξεις ισχύος. Μια μπαταρία 3S1P LiPo έχει τρία cells σε σειρά και αυτά τα cells δεν συνδέονται παράλληλα με κανένα άλλο σύνολο κελιών. Επιλέχθηκε μια 5000 mAh[34] μπαταρία με έξοδο 11,1V και μέγιστη εκφόρτιση 50C.

Εικόνα 19 Gens ace 5000mAh 11.1V 50C 3S1P Battery

[33] Holybro. (2024). M8N GPS. Holybro. https://holybro.com/products/m8n-gps
[34] Top Electronics. (2023). Gens Ace 5000mAh 11.1V 50C 3S1P Short Size LiPo with XT60 Plug. Top Electronics. https://topelectronics.gr/rc-and-drones/batteries/lipo/3s-11.1v/gens-ace-5000mah-11.1v-50c-3s1p-short-size-lipo-with-xt60-plug/

- Transmitter:

Ο πομπός είναι η συσκευή που παράγει και στέλνει ραδιοσήματα που μεταφέρουν πληροφορίες ή δεδομένα. Μετατρέπει ηλεκτρικά σήματα (όπως φωνητικά ή ψηφιακά δεδομένα) σε ραδιοκύματα κατάλληλα για ασύρματη μετάδοση. Ο πομπός στέλνει σήματα ελέγχου (όπως throttle, steering κλπ.), και ο δέκτης ερμηνεύει αυτά τα σήματα για να ελέγξει τις αντίστοιχες ενέργειες της συσκευής.

Η αποτελεσματικότητα του συστήματος πομπού-δέκτη είναι αναγκαία για την αξιόπιστη επικοινωνία, ειδικά σε εφαρμογές όπου ο έλεγχος σε πραγματικό χρόνο και η ανταπόκριση είναι ζωτικής σημασίας. Χρησιμοποιήθηκε ο πομπός FS – i6.[35]

Τα τεχνικά χαρακτηριστικά του είναι:

-Control Range: 500m
-Channels: 6
-Model Type: Glider/Heli/Airplane
-RF Range: 2.40-2.48GHz
-Bandwidth: 500KHz
-band: 142
-RF Power: Less Than 20dBm
-2.4ghz System: AFHDS 2A and AFHDS
-Code Type: GFSK
-Sensitivity: 1024

Εικόνα 20 FlySky FS - i6 TC

- Reciever

Ο δέκτης είναι υπεύθυνος για τη λήψη και την ερμηνεία των μεταδιδόμενων ραδιοσημάτων.
Ανιχνεύει και μετατρέπει τα λαμβανόμενα ραδιοκύματα πίσω σε ηλεκτρικά σήματα που μπορούν να υποστούν επεξεργασία ή να χρησιμοποιηθούν από τη συσκευή λήψης.
Χρησιμοποιήθηκε ο δέκτης FS – iA6[36].

[35] Flysky. (2019). FS-i6. Flysky. https://www.flysky-cn.com/fsi6
[36] Aigaior RC. (2023). Dektis Flysky FS-iA6 2.4G 6CH AFHDS Receiver. Aigaior RC.

Τα τεχνικά χαρακτηριστικά του είναι:

-Frequency Range :2.4-2.48 GHz
-Number of Band: 140
-Transmitting Power: not more than 20dBm
-Receiver Sensitivity:-105dBm
-2.4G modes: Automatic frequency second generation digital systems Encoding: GFSK
-Weight: 6.4g

Εικόνα 21 FS - i6A Reciever

- Companion Computer

Η σύνδεση ενός Companion computer όπως ένα Raspberry pi 5 με τον ελεγκτή πτήσης Pixhawk 2.4.8 επεκτείνει σημαντικά τις δυνατότητες του drone. Μέσω αυτόνομων αλγορίθμων πλοήγησης, το drone αποκτά την ικανότητα να λαμβάνει έξυπνες αποφάσεις με βάση τα δεδομένα περιβάλλοντος, επιτρέποντάς του να πλοηγείτε ανεξάρτητα σε σύνθετα σενάρια.

Η ενσωμάτωση του Computer Vision στο Raspberry Pi ενισχύει περαιτέρω την αντίληψη του drone, επιτρέποντας εργασίες όπως η ανίχνευση και η παρακολούθηση αντικειμένων. Επιπλέον, το Raspberry pi μπορεί να επεξεργαστεί δεδομένα από πρόσθετους αισθητήρες πέρα από εκείνους που είναι ενσωματωμένοι στο Pixhawk. Επιλέχθηκε το Raspberry pi 5[37] για την συγκεκριμένη εφαρμογή.

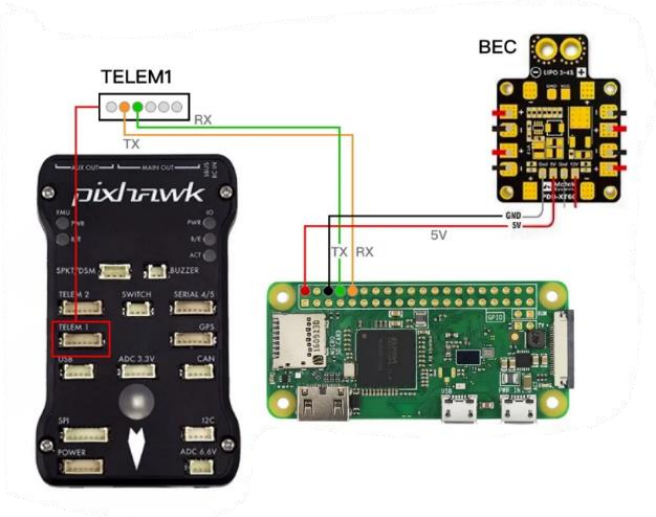
https://www.aigaiorcb.gr/product/dektis-flysky-fs-ia6-ia6-2-4g-6ch-afhds/

[37] Raspberry Pi Foundation. (2023). Raspberry Pi 5. Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-5/

Εικόνα 22 Raspberry Pi 5

- Cables

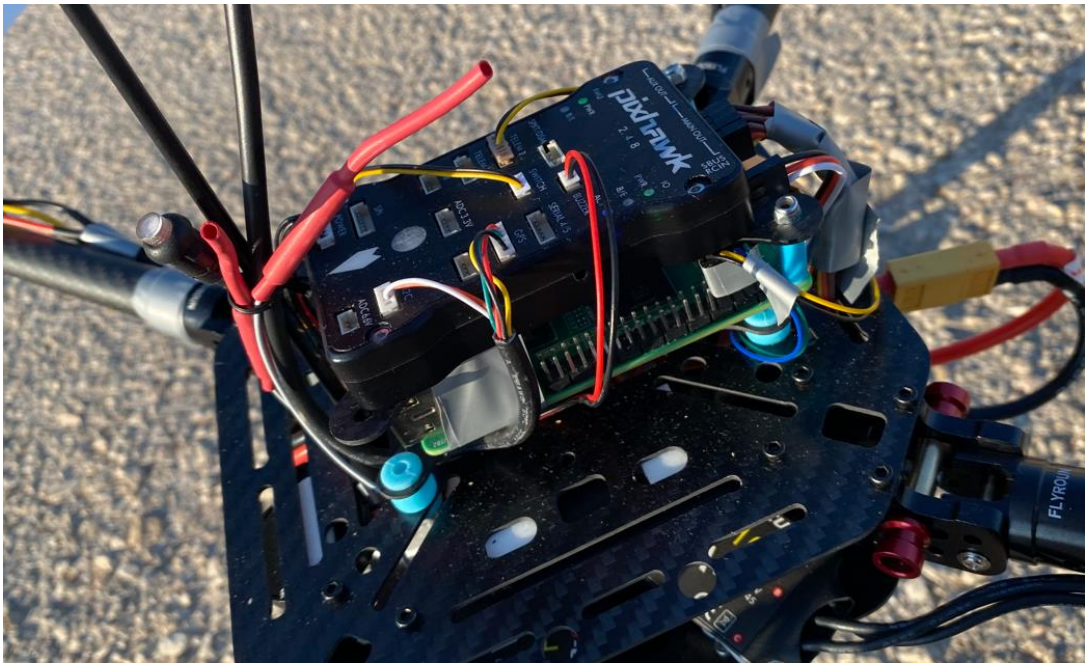
Το καλώδιο DF13 σε 6 pin dupont[38] είναι αναγκαίο για την σύνδεση του flight controller με το Raspberry pi 5.



Εικόνα 23 Raspberry pi to Pixhawk connection

Εικόνα 24 DF13 to 6 Pin dupont cable

[38] AliExpress. (2023). DF13 JST GH 1.25mm to 1pin dupont cable DF13 Series. AliExpress. https://www.aliexpress.com/item/1005002901216924.html



Εικόνα 25 Raspberry pi and Pixhawk

- Camera Module

Επιλέχθηκε η Raspberry pi camera module 3.[39]

Τα τεχνικά χαρακτηριστικά της είναι:

-Diagonal field of view: 75 degrees
-Resolution: 11.9 megapixels
-Common video modes: 1080p50, 720p100, 480p120
-Back-illuminated, stacked CMOS 12-megapixel Sony IMX708 image sensor
-High signal-to-noise ratio (SNR)
-Built-in 2D Dynamic Defect Pixel Correction (DPC)
-Phase Detection Autofocus (PDAF) for rapid autofocus
-Weight: 4g

Εικόνα 26 Raspberry pi camera Module 3

[39] Raspberry Pi Foundation. (2023.). Camera Module V2. Raspberry Pi. https://www.raspberrypi.com/documentation/accessories/camera.html

3.2 Τελική κατασκευή

Εικόνα 27 Drone + RC + Aruco Marker Landing

Εικόνα 28 Τελική κατασκευή

3.3: Θεωρητικά τεχνικά χαρακτηριστικά της κατασκευής

Πριν από την τελική υλοποίηση της κατασκευής του drone είναι σημαντική η ανάλυση του συνδυασμού των εξαρτημάτων. Οι αριθμητικοί υπολογισμοί μπορούν να γίνουν αυτόματα μέσω εφαρμογών όπως το eCalc[40]. Αυτή η εξελιγμένη πλατφόρμα προσφέρει μια πληθώρα λειτουργιών που βελτιστοποιούν την διαδικασία κατασκευής drone, βοηθώντας τον κατασκευαστή να λάβει τεκμηριωμένες αποφάσεις σχετικά με διάφορα στοιχεία και παραμέτρους.

Battery	
Load:	9.77 C
Voltage:	11.27 V
Rated Voltage:	11.10 V
Energy:	55.5 Wh
Total Capacity:	5000 mAh
Used Capacity:	4250 mAh
min. Flight Time:	5.2 min
Mixed Flight Time:	12.4 min
Hover Flight Time:	17.3 min
Weight:	378 g
	13.3 oz

Motor @ Optimum Efficiency	
Current:	4.51 A
Voltage:	11.55 V
Revolutions*:	9929 rpm
electric Power:	52.0 W
mech. Power:	45.5 W
Efficiency:	87.4 %

Motor @ Maximum	
Current:	12.22 A
Voltage:	11.17 V
Revolutions*:	8397 rpm
electric Power:	136.5 W
mech. Power:	109.1 W
Power-Weight:	404.3 W/kg
	183.4 W/lb
Efficiency:	79.9 %
est. Temperature:	33 °C
	91 °F
Wattmeter readings	
Current:	48.88 A
Voltage:	11.27 V
Power:	550.9 W

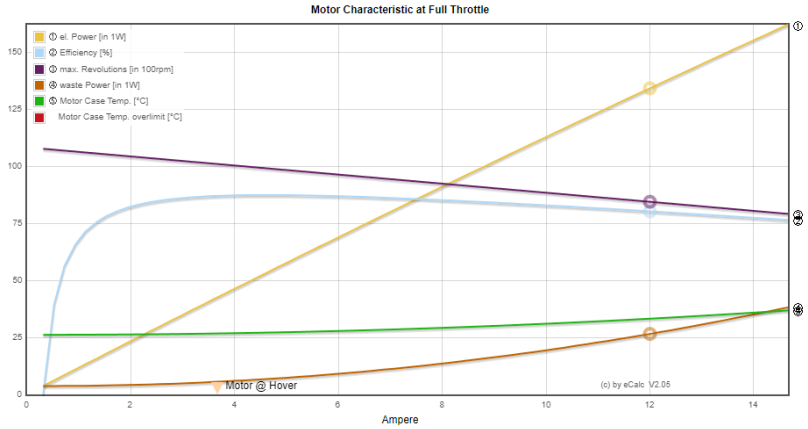
Motor @ Hover	
Current:	3.68 A
Voltage:	11.59 V
Revolutions*:	5100 rpm
Throttle (log):	43 %
Throttle (linear):	57 %
electric Power:	42.6 W
mech. Power:	34.5 W
Power-Weight:	128.3 W/kg
	58.2 W/lb
Efficiency:	80.9 %
est. Temperature:	28 °C
	82 °F
specific Thrust:	7.92 g/W

Total Drive	
Drive Weight:	843 g
	29.7 oz
Thrust-Weight:	2.1 : 1
Current @ Hover:	14.72 A
P(in) @ Hover:	173.2 W
P(out) @ Hover:	137.9 W
Efficiency @ Hover:	79.6 %
Current @ max:	48.86 A
P(in) @ max:	574.9 W
P(out) @ max:	436.3 W
Efficiency @ max:	75.9 %

Multicopter	
All-up Weight:	1350 g
	47.6 oz
add. Payload:	1154 g
	40.7 oz
max Tilt:	57 °
max. Speed:	59 km/h
	36.6 mph
est. Range:	- m
	- mi
est. rate of climb:	6.9 m/s
	1358 ft/min
Total Disc Area:	20.27 dm²
	314.19 in²

Εικόνα 29 Θεωρητικά τεχνικά χαρακτηριστικά (eCalc).

Μία από τις κύριες δυνατότητες της eCalc είναι να βοηθά τους χρήστες στην επιλογή κινητήρων και προπέλας, μια κρίσιμη πτυχή του σχεδιασμού των drone. Ο υπολογιστής εξετάζει ένα πλήθος παραγόντων, συμπεριλαμβανομένης της επιθυμητής ώσης, της απόδοσης και της κατανάλωσης ενέργειας, παρέχοντας συστάσεις για συνδυασμούς κινητήρων και προπέλας που ευθυγραμμίζονται με τις συγκεκριμένες απαιτήσεις του χρήστη.



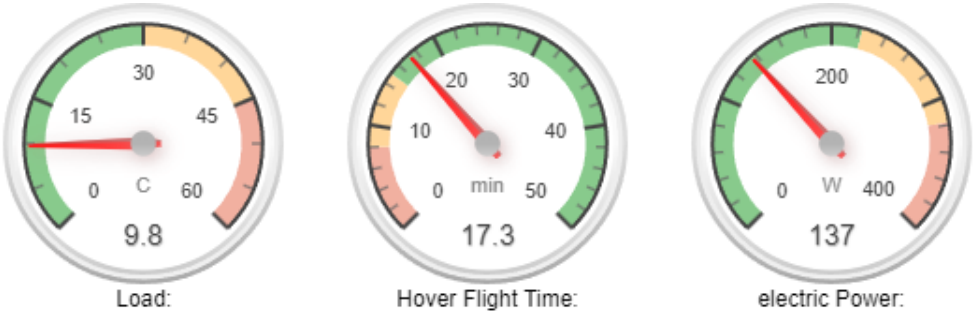
Εικόνα 30 Θεωρητικά χαρακτηριστικά κινητήρων (eCalc).

[40] eCalc. (2023). Multicopter Calculator. eCalc. https://www.ecalc.ch/xcoptercalc.php

Αυτό το χαρακτηριστικό όχι μόνο απλοποιεί τη διαδικασία λήψης αποφάσεων, αλλά διασφαλίζει επίσης ότι τα επιλεγμένα εξαρτήματα λειτουργούν αρμονικά για να επιτύχουν τη βέλτιστη απόδοση.

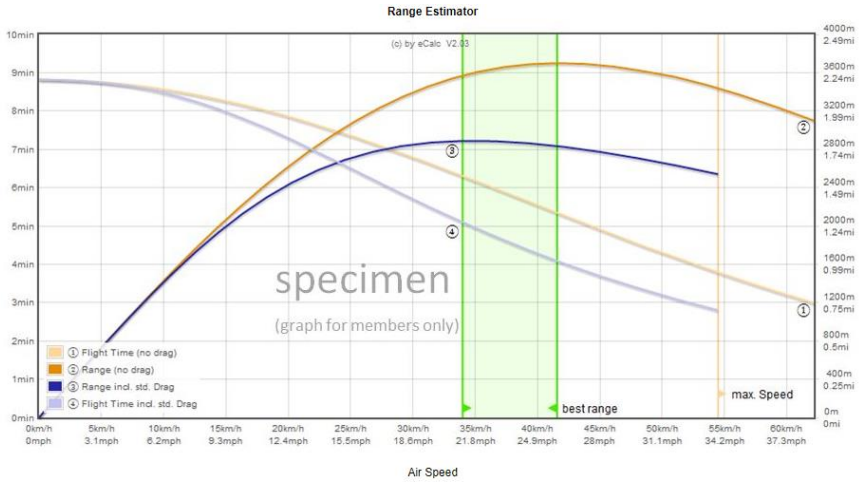
Εικόνα 31 Θερμοκρασία – Thrust (eCalc).

Το μέγεθος της μπαταρίας είναι μια άλλη κρίσιμη πτυχή που αντιμετωπίζεται από την eCalc. Η επιλογή της σωστής μπαταρίας είναι απαραίτητη για την επίτευξη του επιθυμητού χρόνου πτήσης και την ικανοποίηση των απαιτήσεων ισχύος. Η αριθμομηχανή λαμβάνει υπόψη παράγοντες όπως η χωρητικότητα της μπαταρίας, η τάση και ο ρυθμός εκφόρτισης για να προτείνει τη βέλτιστη διαμόρφωση της μπαταρίας. Αυτό εξασφαλίζει ότι η πηγή ισχύος του drones ευθυγραμμίζεται με το γενικό σχέδιό, συμβάλλοντας στην αποδοτική και αξιόπιστη απόδοση.



Εικόνα 32 Απόδοση μπαταρίας (eCalc).

Τέλος, ο χρήστης έχει την δυνατότητα να εκτιμήσει την μέγιστη εμβέλεια της κατασκευής του.



Εικόνα 33 Θεωρητική σχέση Εμβέλειας-Ταχύτητας (eCalc).

ΚΕΦΑΛΑΙΟ 4

SOFTWARE

4.1: Εγκατάσταση OpenCV

Το OpenCV χρησιμοποιεί βιβλιοθήκες λογισμικού τρίτων. Πρέπει αρχικά να γίνει εγκατάσταση αυτών πριν εγκατασταθεί το OpenCV. Η παρακάτω διαδικασία εγκαθιστά τα πιο πρόσφατα πακέτα.[41]

```
# check for updates
$ sudo apt-get update
$ sudo apt-get upgrade
# dependencies
$ sudo apt-get install build-essential cmake git unzip pkg-config
$ sudo apt-get install libjpeg-dev libpng-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
$ sudo apt-get install libgtk2.0-dev libcanberra-gtk* libgtk-3-dev
$ sudo apt-get install libgstreamer1.0-dev gstreamer1.0-gtk3
$ sudo apt-get install libgstreamer-plugins-base1.0-dev gstreamer1.0-gl
$ sudo apt-get install libxvidcore-dev libx264-dev
$ sudo apt-get install python3-dev python3-numpy python3-pip
$ sudo apt-get install libtbb2 libtbb-dev libdc1394-22-dev
$ sudo apt-get install libv4l-dev v4l-utils
$ sudo apt-get install libopenblas-dev libatlas-base-dev libblas-dev
$ sudo apt-get install liblapack-dev gfortran libhdf5-dev
$ sudo apt-get install libprotobuf-dev libgoogle-glog-dev libgflags-dev
$ sudo apt-get install protobuf-compiler
```

Στην συνέχεια γίνεται clone το git του OpenCV και ξεκινάει η εγκατάσταση με την εντολή cmake και make -j4. Η όλη διαδικασία απαιτεί περίπου 1 ώρα για να ολοκληρωθεί επιτυχώς.

```
$ git clone --depth=1 https://github.com/opencv/opencv.git
$ git clone --depth=1 https://github.com/opencv/opencv_contrib.git
```

```
$ cd ~/opencv
$ mkdir build
$ cd build
```

```
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
        -D CMAKE_INSTALL_PREFIX=/usr/local \
        -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
        -D ENABLE_NEON=ON \
        -D WITH_OPENMP=ON \
```

[41] Qengineering. (2019). Install OpenCV on Raspberry Pi. qengineering.eu. https://qengineering.eu/install-opencv-on-raspberry-pi.html

```
        -D WITH_OPENCL=OFF \
        -D BUILD_TIFF=ON \
        -D WITH_FFMPEG=ON \
        -D WITH_TBB=ON \
        -D BUILD_TBB=ON \
        -D WITH_GSTREAMER=ON \
        -D BUILD_TESTS=OFF \
        -D WITH_EIGEN=OFF \
        -D WITH_V4L=ON \
        -D WITH_LIBV4L=ON \
        -D WITH_VTK=OFF \
        -D WITH_QT=OFF \
        -D WITH_PROTOBUF=ON \
        -D OPENCV_ENABLE_NONFREE=ON \
        -D INSTALL_C_EXAMPLES=OFF \
        -D INSTALL_PYTHON_EXAMPLES=OFF \
        -D OPENCV_FORCE_LIBATOMIC_COMPILER_CHECK=1 \
        -D PYTHON3_PACKAGES_PATH=/usr/lib/python3/dist-packages \
        -D OPENCV_GENERATE_PKGCONFIG=ON \
        -D BUILD_EXAMPLES=OFF ..
```

```
$ make -j4
```

```
[ 99%] Building CXX object modules/gapi/CMakeFiles/opencv_perf_gapi.dir/perf/gp
u/gapi_core_perf_tests_gpu.cpp.o
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_perf_gapi.dir/perf/gp
u/gapi_imgproc_perf_tests_gpu.cpp.o
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_perf_gapi.dir/perf/in
ternal/gapi_compiler_perf_tests.cpp.o
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_perf_gapi.dir/perf/pe
rf_bench.cpp.o
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_perf_gapi.dir/perf/pe
rf_main.cpp.o
[100%] Building CXX object modules/gapi/CMakeFiles/opencv_perf_gapi.dir/perf/re
nder/gapi_render_perf_tests_ocv.cpp.o
[100%] Linking CXX executable ../../bin/opencv_perf_gapi
[100%] Built target opencv_perf_gapi
[100%] Linking CXX shared module ../../lib/cv2.so
[100%] Built target opencv_python2
[100%] Linking CXX shared module ../../lib/python3/cv2.cpython-37m-aarch64-linu
x-gnu.so
[100%] Built target opencv_python3
pi@raspberrypi:~/opencv/build $
```

Εικόνα 34 make -j4 command

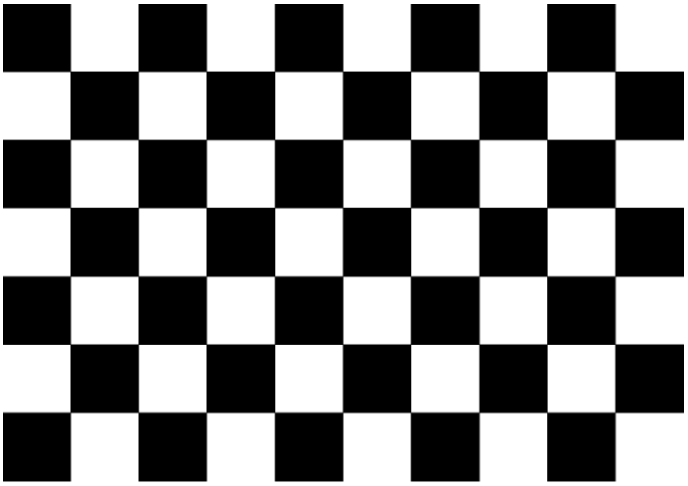
```
$ sudo make install
$ sudo ldconfig
$ make clean
$ sudo apt-get update
```

4.2: Camera calibration

4.2.1: Save Snapshots Script

Για την αντιμετώπιση της παραμόρφωσης της κάμερας, όπως αναφέρθηκε σε προηγούμενο κεφάλαιο χρειάζεται μεγάλος αριθμός από φωτογραφίες του πίνακα που φαίνεται παρακάτω. Οι φωτογραφίες πρέπει να είναι τραβηγμένες από διαφορετικές γωνίες και κατευθύνσεις και να είναι ευκρινείς.

.



Εικόνα 35 Camera Calibration Chessboard

Το παρακάτω python script SaveSnapshots.py αποθηκεύει τις φωτογραφίες πατώντας το πλήκτρο ‘space’, σε φάκελο που θα επιλέξει ο χρήστης. Για να είναι επιτυχή η διαδικασία, χρειάζονται 10-50 φωτογραφίες.

```
"""
Saves a series of snapshots with the current camera as
snapshot_<width>_<height>_<nnn>.jpg

Arguments:
   --f <output folder>     default: current folder
   --n <file name>         default: snapshot
   --w <width px>          default: none
   --h <height px>         default: none

Buttons:
   q          - quit
   space bar   - save the snapshot
"""
import cv2
import time
import sys
import argparse
import os
from picamera2 import Picamera2
```

```
#Preparing camera parameters
picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration(main={"format":
'XRGB8888', "size": (640, 480)}))
picam2.start()

def save_snaps(width=0, height=0, name="snapshot", folder=".", raspi=False):

    if raspi:
        os.system('sudo modprobe bcm2835-v4l2')

    if width > 0 and height > 0:
        print("Setting the custom Width and Height")

    try:
        if not os.path.exists(folder):
            os.makedirs(folder)
            # ---------- CREATE THE FOLDER -----------------
            folder = os.path.dirname(folder)
            try:
                os.stat(folder)
            except:
                os.mkdir(folder)
    except:
        pass

    nSnap   = 0
    w       =+ 1
    h       =+ 1

    fileName    = "%s/%s_%d_%d_" %(folder, name, w, h)
    while True:

        cap = picam2.capture_array()
        cv2.imshow('camera', cap)

        key = cv2.waitKey(1) & 0xFF
        if key == ord('q'):
            break
        if key == ord(' '):
            print("Saving image ", nSnap)
            cv2.imwrite("%s%d.jpg"%(fileName, nSnap), cap)
            nSnap += 1

    cap.release()
    cv2.destroyAllWindows()

def main():
```

```
    # ---- DEFAULT VALUES ---
    SAVE_FOLDER = "New"
    FILE_NAME = "snapshot"
    FRAME_WIDTH = 640
    FRAME_HEIGHT = 480

    # ---------- PARSE THE INPUTS ----------------
    parser = argparse.ArgumentParser(
        description="Saves snapshot from the camera. \n q to quit \n spacebar to
save the snapshot")
    parser.add_argument("--folder", default=SAVE_FOLDER, help="Path to the
save folder (default: current)")
    parser.add_argument("--name", default=FILE_NAME, help="Picture file name
(default: snapshot)")
    parser.add_argument("--dwidth", default=FRAME_WIDTH, type=int,
help="<width> px (default the camera output)")
    parser.add_argument("--dheight", default=FRAME_HEIGHT, type=int,
help="<height> px (default the camera output)")
    parser.add_argument("--raspi", default=True, type=bool, help="<bool> True if
using a raspberry Pi")
    args = parser.parse_args()

    SAVE_FOLDER = args.folder
    FILE_NAME = args.name
    FRAME_WIDTH = args.dwidth
    FRAME_HEIGHT = args.dheight


    save_snaps(width=args.dwidth, height=args.dheight, name=args.name,
folder=args.folder, raspi=args.raspi)

    print("Files saved")

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

4.2.2: Calibration Script

Οι φωτογραφίες που αποθηκεύτηκαν, χρησιμοποιούνται από το παρακάτω script CameraCalibration.py ώστε να υπολογιστεί η παραμόρφωση της κάμερας.

Εικόνα 36 Camera Calibration Process

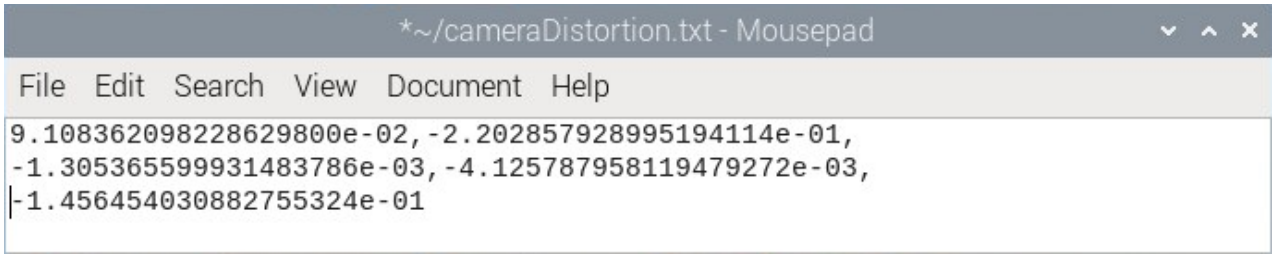
Αρχικά, το πρόγραμμα θα εμφανίσει τις φωτογραφίες από τον επιλεγμένο φάκελο και ο χρήστης πρέπει να επιλέξει τις κατάλληλες (ευκρινείς) με το πλήκτρο ‘Enter’ και της ακατάλληλες με το πλήκτρο ‘Esc’. Τα αποτελέσματα θα αποθηκευτούν ως αρχεία Text με όνομα cameraMatrix.txt και cameraDistortion.txt.

```
import numpy as np
import cv2
import glob
import sys
import argparse

#-- Set the parameters
nRows = 9
nCols = 6
dimension = 25 #- mm

#-- Input the image folder
workingFolder   = "./New"
imageType       = 'jpg'
#-----------------------------------

#-- Termination criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,
dimension, 0.001)
```

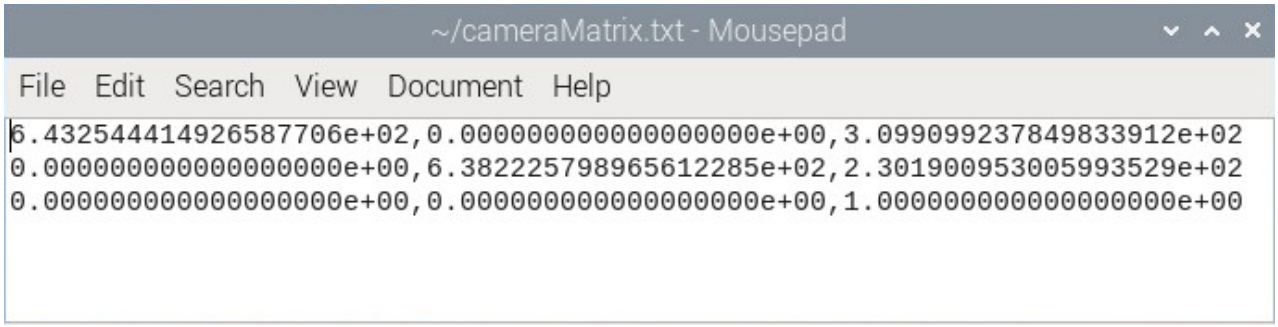
```
if len(sys.argv) < 6:
        print("\n Not enough inputs are provided. Using the default values.\n\n" \
              " type -h for help")
else:
    workingFolder   = sys.argv[1]
    imageType       = sys.argv[2]
    nRows           = int(sys.argv[3])
    nCols           = int(sys.argv[4])
    dimension       = float(sys.argv[5])

#-- Prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((nRows*nCols,3), np.float32)
objp[:,:2] = np.mgrid[0:nCols,0:nRows].T.reshape(-1,2)

#-- Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.

if '-h' in sys.argv or '--h' in sys.argv:
    print("\n IMAGE CALIBRATION GIVEN A SET OF IMAGES")
    print(" call: python cameracalib.py <folder> <image type> <num rows (9)>
<num cols (6)> <cell dimension (25)>")
    print("\n The script will look for every image in the provided folder and will show
the pattern found." \
          " User can skip the image pressing ESC or accepting the image with
RETURN. " \
          " At the end the end the following files are created:" \
          "  - cameraDistortion.txt" \
          "  - cameraMatrix.txt \n\n")

    sys.exit()

# Find the images files
filename    = workingFolder + "/*." + imageType
images      = glob.glob(filename)

print(len(images))
if len(images) < 9:
    print("Not enough images were found: at least 9 shall be provided!!!")
    sys.exit()

else:
    nPatternFound = 0
    imgNotGood = images[1]

    for fname in images:
        if 'calibresult' in fname: continue
        #-- Read the file and convert in greyscale
```

```
        img     = cv2.imread(fname)
        gray    = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

        print("Reading image ", fname)

        #-- Find the chess board corners
        ret, corners = cv2.findChessboardCorners(gray, (nCols,nRows),None)

        #-- If found, add object points, image points (after refining them)
        if ret == True:
            print("Pattern found! Press ESC to skip or ENTER to accept")
            #-- Sometimes, Harris cornes fails with crappy pictures, so
            corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)

            #-- Draw and display the corners
            cv2.drawChessboardCorners(img, (nCols,nRows), corners2,ret)
            cv2.imshow('img',img)

            k = cv2.waitKey(0) & 0xFF
            if k == 27: #-- ESC Button
                print("Image Skipped")
                imgNotGood = fname
                continue

            print("Image accepted")
            nPatternFound += 1
            objpoints.append(objp)
            imgpoints.append(corners2)

        else:
            imgNotGood = fname


cv2.destroyAllWindows()

if (nPatternFound > 1):
    print("Found %d good images" % (nPatternFound))
    ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints,
gray.shape[::-1],None,None)

    # Undistort an image
    img = cv2.imread(imgNotGood)
    h,  w = img.shape[:2]
    print("Image to undistort: ", imgNotGood)
    newcameramtx, roi=cv2.getOptimalNewCameraMatrix(mtx,dist,(w,h),1,(w,h))

    # undistort
    mapx,mapy =
cv2.initUndistortRectifyMap(mtx,dist,None,newcameramtx,(w,h),5)
```

```
    dst = cv2.remap(img,mapx,mapy,cv2.INTER_LINEAR)

    # crop the image
    x,y,w,h = roi
    dst = dst[y:y+h, x:x+w]
    print("ROI: ", x, y, w, h)

    cv2.imwrite(workingFolder + "/calibresult.png",dst)
    print("Calibrated picture saved as calibresult.png")
    print("Calibration Matrix: ")
    print(mtx)
    print("Disortion: ", dist)

    #--------- Save result
    filename = workingFolder + "/cameraMatrix.txt"
    np.savetxt(filename, mtx, delimiter=',')
    filename = workingFolder + "/cameraDistortion.txt"
    np.savetxt(filename, dist, delimiter=',')

    mean_error = 0
    for i in range(len(objpoints)):
        imgpoints2, _ = cv2.projectPoints(objpoints[i], rvecs[i], tvecs[i], mtx, dist)
        error = cv2.norm(imgpoints[i],imgpoints2, cv2.NORM_L2)/len(imgpoints2)
        mean_error += error

    print("total error: ", mean_error/len(objpoints))

else:
    print("In order to calibrate you need at least 9 good pictures... try again")
```



Εικόνα 37 CameraDistortion.txt



Εικόνα 38 CameraMatrix.txt

4.3: Main precision landing script

Ο κύριος αλγόριθμος για την ακριβή προσγείωση βασίζεται σε 10 βήματα:

1. Ο δείκτης ArUco αναγνωρίζεται από το OpenCV.
2. Η θέση του δείκτη ως προς την camera frame γίνεται γνωστή.
3. Υπολογίζεται η θέση του μη επανδρωμένου αεροσκάφους σε σχέση με την θέση του ArUco marker.
4. Η παραπάνω θέση του drone μετατρέπεται σε συντεταγμένες σε πραγματικό χρόνο. Οι συντεταγμένες αφορούν την θέση της κάμερας, δηλαδή το κέντρο του drone. (uav_location.lat, uav_location.lon).
5. Η θέση του ArUco marker μετατρέπεται σε συντεταγμένες (location_marker.lat, location_marker.lon).
6. Το όχημα μεταβαίνει στις συντεταγμένες του δείκτη μέσω της εντολής vehicle.simple_goto.
7. Το μη επανδρωμένο αεροσκάφος παραμένει πάνω από τον δείκτη μέχρι το σφάλμα να είναι αρκετά χαμηλό.
8. Δίνεται η εντολή για descent.
9. Όταν το υψόμετρο είναι αρκετά χαμηλό, δίνεται εντολή για προσγείωση μέσω του vehicle.mode = "LAND".
10. Η προσγείωση ολοκληρώνεται.

Το παρακάτω python script PrecisionLanding.py έχει ρυθμιστεί να ξεκινάει την λειτουργία του με το bootup του Raspberri pi 5, χωρίς να χρειάζεται καμία ενέργεια του χρήστη. Αυτό σημαίνει ότι δεν χρειάζεται κάποια τηλεμετρία μεταξύ του companion pc και του χρήστη. Ο χειριστής μπορεί να ελέγχει κανονικά το μη επανδρωμένο αεροσκάφος. Όταν αποφασίσει να προσγειώσει το όχημα, τοποθετεί το mode 'ALT_HOLD' (if vehicle.mode == 'ALT_HOLD': TRYLANDING = True) με το τηλεχειριστήριο και ο αλγόριθμός ξεκινάει την διαδικασία της προσγείωσης.

```
import numpy as np
import cv2
import cv2.aruco as aruco
import sys, time, math
import time
import math
import argparse
from picamera2 import Picamera2
from os import sys, path

sys.path.append(path.dirname(path.dirname(path.abspath(__file__))))
from dronekit import connect, VehicleMode, LocationGlobalRelative, Command,
LocationGlobal
from pymavlink import mavutil

#-- Prepare the camera parameters
picam2 = Picamera2()
```

```
picam2.configure(picam2.create_preview_configuration(main={"format":
'XRGB8888', "size": (640, 480)}))
picam2.start()

#-- Other parameters
rad_2_deg   = 180.0/math.pi
deg_2_rad   = 1.0/rad_2_deg

#-- Landing Marker
#-- Define Tag
id_to_find      = 72
marker_size     = 10 #- [cm]
freq_send       = 1 #- Hz

land_alt_cm         = 50.0
angle_descend       = 20*deg_2_rad
land_speed_cms      = 30.0

#-- [UartEndpoint to_fc] for raspi5
Raspi5 = '/dev/ttyAMA0'

#-- Aruco tracker class
class SingleArucoTracker():
    def __init__(self,
                 id_to_find,
                 marker_size,
                 camera_matrix,
                 camera_distortion,
                 camera_size=[640,480],
                 show_video=True
                 ):


        self.id_to_find     = id_to_find
        self.marker_size    = marker_size
        self._show_video    = show_video

        self._camera_matrix = camera_matrix
        self._camera_distortion = camera_distortion

        self.is_detected    = False
        self._kill          = False

        #-- 180 deg rotation matrix around the x axis
        self._R_flip      = np.zeros((3,3), dtype=np.float32)
        self._R_flip[0,0] = 1.0
        self._R_flip[1,1] =-1.0
        self._R_flip[2,2] =-1.0
```

```
        #-- Define the aruco dictionary
        self._aruco_dict  =
aruco.getPredefinedDictionary(aruco.DICT_ARUCO_ORIGINAL)
        self._parameters  = aruco.DetectorParameters_create()

        #-- Font for the text in the image
        self.font = cv2.FONT_HERSHEY_PLAIN

        self._t_read      = time.time()
        self._t_detect    = self._t_read
        self.fps_read     = 0.0
        self.fps_detect   = 0.0

    def _rotationMatrixToEulerAngles(self,R):
    # Calculates rotation matrix to euler angles
    # The result is the same as MATLAB except the order
    # of the euler angles ( x and z are swapped ).

        def isRotationMatrix(R):
            Rt = np.transpose(R)
            shouldBeIdentity = np.dot(Rt, R)
            I = np.identity(3, dtype=R.dtype)
            n = np.linalg.norm(I - shouldBeIdentity)
            return n < 1e-6
        assert (isRotationMatrix(R))

        sy = math.sqrt(R[0, 0] * R[0, 0] + R[1, 0] * R[1, 0])

        singular = sy < 1e-6

        if not singular:
            x = math.atan2(R[2, 1], R[2, 2])
            y = math.atan2(-R[2, 0], sy)
            z = math.atan2(R[1, 0], R[0, 0])
        else:
            x = math.atan2(-R[1, 2], R[1, 1])
            y = math.atan2(-R[2, 0], sy)
            z = 0

        return np.array([x, y, z])

    def _update_fps_read(self):
        t          = time.time()
        self.fps_read    = 1.0/(t - self._t_read)
        self._t_read     = t

    def _update_fps_detect(self):
```

```
        t          = time.time()
        self.fps_detect  = 1.0/(t - self._t_detect)
        self._t_detect      = t

    def stop(self):
        self._kill = True

    def track(self, loop=True, verbose=False, show_video=None):

        self._kill = False
        if show_video is None: show_video = self._show_video

        marker_found = False
        x = y = z = 0

        while not self._kill:

            #--- Capture the videocamera
            #-- Read the camera frame
            frame = picam2.capture_array()

            self._update_fps_read()

            #-- Convert in gray scale
            gray    = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #-- remember,
OpenCV stores color images in Blue, Green, Red

            #-- Find all the aruco markers in the image
            corners, ids, rejected = aruco.detectMarkers(image=gray,
dictionary=self._aruco_dict,
                        parameters=self._parameters,)

            if not ids is None and self.id_to_find in ids[0]:
                marker_found = True
                self._update_fps_detect()
                #-- ret = [rvec, tvec, ?]
                #-- array of rotation and position of each marker in camera frame
                #-- rvec = [[rvec_1], [rvec_2], ...]    attitude of the marker respect to
camera frame
                #-- tvec = [[tvec_1], [tvec_2], ...]    position of the marker in camera
frame
                ret = aruco.estimatePoseSingleMarkers(corners, self.marker_size,
self._camera_matrix, self._camera_distortion)

                #-- Unpack the output, get only the first
                rvec, tvec = ret[0][0,0,:], ret[1][0,0,:]

                x = tvec[0]
```

```
            y = tvec[1]
            z = tvec[2]

            #-- Draw the detected marker and put a reference frame over it
            aruco.drawDetectedMarkers(gray, corners)
            cv2.drawFrameAxes(frame, self._camera_matrix,
self._camera_distortion, rvec, tvec, 10)

            #-- Obtain the rotation matrix tag->camera
            R_ct    = np.matrix(cv2.Rodrigues(rvec)[0])
            R_tc    = R_ct.T

            #-- Get the attitude in terms of euler 321 (Needs to be flipped first)
            roll_marker, pitch_marker, yaw_marker =
self._rotationMatrixToEulerAngles(self._R_flip*R_tc)

            #-- Now get Position and attitude f the camera respect to the marker
            pos_camera = -R_tc*np.matrix(tvec).T

            # print "Camera X = %.1f  Y = %.1f  Z = %.1f  - fps =
%.0f"%(pos_camera[0], pos_camera[1], pos_camera[2],fps_detect)
            if verbose: print ("Marker X = %.1f  Y = %.1f  Z = %.1f  - fps =
%.0f"%(tvec[0], tvec[1], tvec[2],self.fps_detect))

            if show_video:

                #-- Print the tag position in camera frame
                str_position = "MARKER Position
x=%4.0f  y=%4.0f  z=%4.0f"%(tvec[0], tvec[1], tvec[2])
                cv2.putText(frame, str_position, (0, 100), self.font, 1, (0, 255, 0), 2,
cv2.LINE_AA)

                #-- Print the marker's attitude respect to camera frame
                str_attitude = "MARKER Attitude
r=%4.0f  p=%4.0f  y=%4.0f"%(math.degrees(roll_marker),math.degrees(pitch_mar
ker),
                                math.degrees(yaw_marker))
                cv2.putText(frame, str_attitude, (0, 150), self.font, 1, (0, 255, 0), 2,
cv2.LINE_AA)

                str_position = "CAMERA Position
x=%4.0f  y=%4.0f  z=%4.0f"%(pos_camera[0], pos_camera[1], pos_camera[2])
                cv2.putText(frame, str_position, (0, 200), self.font, 1, (0, 255, 0), 2,
cv2.LINE_AA)

                #-- Get the attitude of the camera respect to the frame
                roll_camera, pitch_camera, yaw_camera =
self._rotationMatrixToEulerAngles(self._R_flip*R_tc)
```

```
                str_attitude = "CAMERA Attitude
r=%4.0f  p=%4.0f  y=%4.0f"%(math.degrees(roll_camera),math.degrees(pitch_ca
mera),
                                math.degrees(yaw_camera))
                cv2.putText(frame, str_attitude, (0, 250), self.font, 1, (0, 255, 0), 2,
cv2.LINE_AA)


            else:
                if verbose: print ("Nothing detected - fps = %.0f"%self.fps_read)

            if show_video:
                #--- Display the frame
                cv2.imshow('frame', frame)

                #--- use 'q' to quit
                key = cv2.waitKey(1) & 0xFF
                if key == ord('q'):
                    self._cap.release()
                    cv2.destroyAllWindows()
                    break

            if not loop: return(marker_found, x, y, z)

#-- Functions

def get_location_metres(original_location, dNorth, dEast):
    #-- Source: https://gis.stackexchange.com/questions/2951/algorithm-for-
offsetting-a-latitude-longitude-by-some-amount-of-meters
    """
    Returns a Location object containing the latitude/longitude `dNorth` and `dEast`
metres from the
    specified `original_location`. The returned Location has the same `alt and
`is_relative` values
    as `original_location`.

    The function is useful when you want to move the vehicle around specifying
locations relative to
    the current vehicle position.
    The algorithm is relatively accurate over small distances (10m within 1km)
except close to the poles.
    """
    #--Radius of earth
    earth_radius=6378137.0
    #--Coordinate offsets in radians
    dLat = dNorth/earth_radius
    dLon = dEast/(earth_radius*math.cos(math.pi*original_location.lat/180))

    print ("dlat, dlon", dLat, dLon)
```

```
    #--New position in decimal degrees
    newlat = original_location.lat + (dLat * 180/math.pi)
    newlon = original_location.lon + (dLon * 180/math.pi)
    return(newlat, newlon)

def marker_position_to_angle(x, y, z):

    angle_x = math.atan2(x,z)
    angle_y = math.atan2(y,z)

    return (angle_x, angle_y)

def camera_to_uav(x_cam, y_cam):
    x_uav =-y_cam
    y_uav = x_cam
    return(x_uav, y_uav)

def uav_to_ne(x_uav, y_uav, yaw_rad):
    c       = math.cos(yaw_rad)
    s       = math.sin(yaw_rad)

    north   = x_uav*c - y_uav*s
    east    = x_uav*s + y_uav*c
    return(north, east)

def check_angle_descend(angle_x, angle_y, angle_desc):
    return(math.sqrt(angle_x**2 + angle_y**2) <= angle_desc)

#-----------------------------------------------
#-------------- CONNECTION
#-----------------------------------------------
#-- Connect to the vehicle

print ('Connecting to vehicle on: %s' % Raspi5)
vehicle = connect(Raspi5, baud=57600, wait_ready=True)
print ('Success')
print ('Vehicle is connected on: %s' % Raspi5)


#--- Get the camera calibration path
# Find full directory path of this script, used for loading config and other files
calib_path          = ""
camera_matrix       = np.loadtxt(calib_path+'cameraMatrix.txt', delimiter=',')
camera_distortion   = np.loadtxt(calib_path+'cameraDistortion.txt',
delimiter=',')
aruco_tracker       = SingleArucoTracker(id_to_find=id_to_find,
marker_size=marker_size, show_video=True,
              camera_matrix=camera_matrix, camera_distortion=camera_distortion)
```

```
time_0 = time.time()

#checks if the user is trying to land
TRYLANDING = False

while True:
    marker_found, x_cm, y_cm, z_cm = aruco_tracker.track(loop=False)
    print(marker_found)
    time.sleep(0.1)

    #-- ALT_HOLD is a mode I selected to give drone permission to start the
proccess of landing
    if vehicle.mode == 'ALT_HOLD':
        print ("CHANGING MODE TO GUIDED")
        print (" Attempting to find the Aruco marker and land")
        TRYLANDING = True
        vehicle.mode = 'GUIDED'
        print (vehicle.mode)

    if marker_found:

        print('Marker found')
        x_cm, y_cm          = camera_to_uav(x_cm, y_cm)
        uav_location        = vehicle.location.global_relative_frame

        #-- If high altitude, use baro rather than visual
        if uav_location.alt >= 5.0:
            z_cm = uav_location.alt*100.0
            print('z_cm')

        angle_x, angle_y    = marker_position_to_angle(x_cm, y_cm, z_cm)


        if time.time() >= time_0 + 1.0/freq_send:
            time_0 = time.time()
            # print ""
            print (" ")
            print ("Altitude = %.0fcm"%z_cm)
            print ("Marker found x = %5.0f cm  y = %5.0f cm -> angle_x =
%5f  angle_y = %5f"%(x_cm, y_cm, angle_x*rad_2_deg, angle_y*rad_2_deg))

            north, east             = uav_to_ne(x_cm, y_cm, vehicle.attitude.yaw)
            print ("Marker N = %5.0f cm   E = %5.0f cm    Yaw = %.0f deg"%(north,
east, vehicle.attitude.yaw*rad_2_deg))
```

```
            marker_lat, marker_lon  = get_location_metres(uav_location, north*0.01,
east*0.01)
            #-- If angle is good, descend

            if check_angle_descend(angle_x, angle_y, angle_descend):
                print ("Low error: descending")
                location_marker         = LocationGlobalRelative(marker_lat, marker_lon,
uav_location.alt-(land_speed_cms*0.01/freq_send))
            else:
                location_marker         = LocationGlobalRelative(marker_lat, marker_lon,
uav_location.alt)

            #-- Going to the location of the marker
            if TRYLANDING == True and vehicle.mode == "GUIDED":
                vehicle.simple_goto(location_marker)
                print('Ordering Vehicle to go to the location of the aruco  marker')

            print ("UAV Location    Lat = %.7f  Lon = %.7f"%(uav_location.lat,
uav_location.lon))
            print ("Commanding to   Lat = %.7f  Lon = %.7f"%(location_marker.lat,
location_marker.lon))

        #--- COmmand to land
        if z_cm <= land_alt_cm:
            if vehicle.mode == "GUIDED":
                print (" -->>COMMANDING TO LAND<<")
                vehicle.mode = "LAND"
                #-- TRYLANDING back to false to get ready for the next flight
                TRYLANDING = False
```

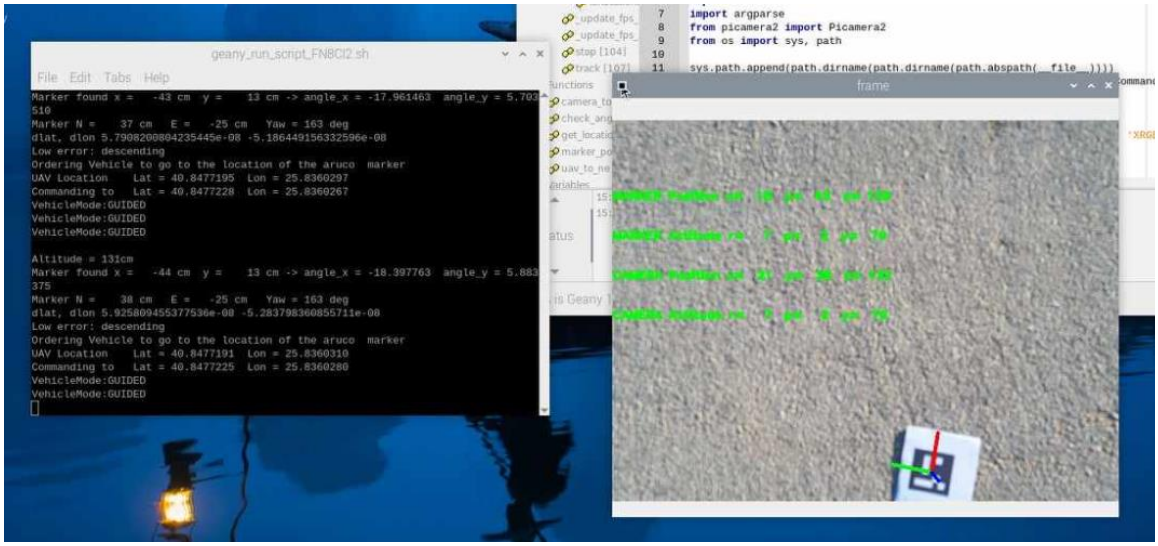
ΚΕΦΑΛΑΙΟ 5

ΠΡΑΚΤΙΚΗ ΕΦΑΡΜΟΓΗ

5.1: Εφαρμογή του αλγορίθμου ακριβούς προσγείωσης

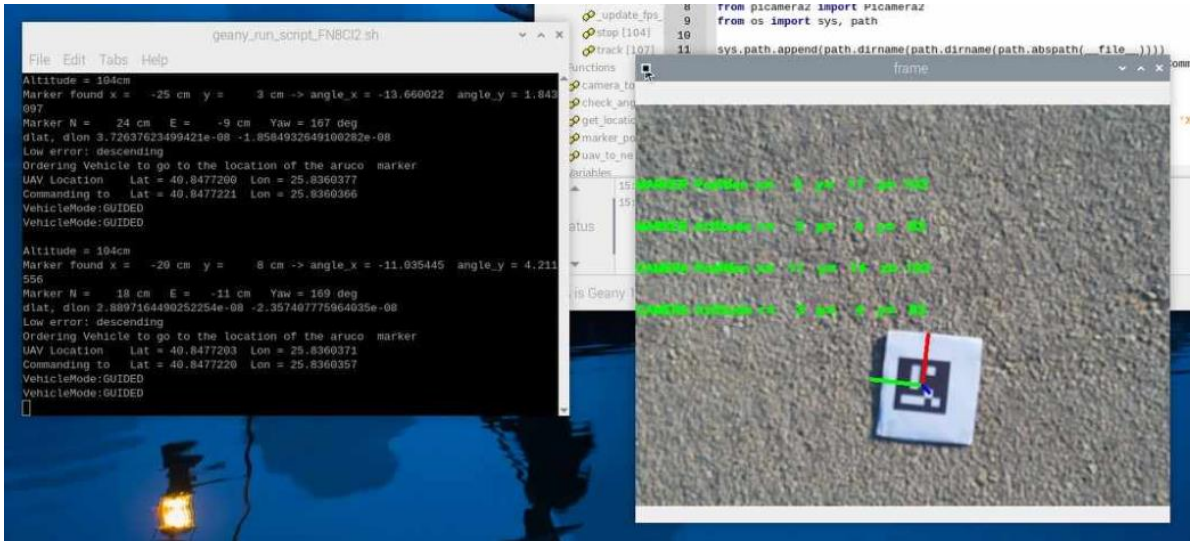
Εικόνα 39 Preparing to Land

Κατά τη διαδικασία των δοκιμών, ο αλγόριθμος πραγματοποίησε 5 ακριβείς προσγειώσεις του UAV χωρίς ανθρώπινη παρέμβαση από τα 5-8 μέτρα ύψος. Το drone είχε την δυνατότητα να λαμβάνει εντολές από το τηλεχειριστήριο καθ' όλη την διάρκεια της πτήσης. Όταν ο χειριστής επέλεγε την προσγείωση, το Raspberry pi σε συνδυασμό με τον flight controller αναλάμβανε εξολοκλήρου την διαδικασία.



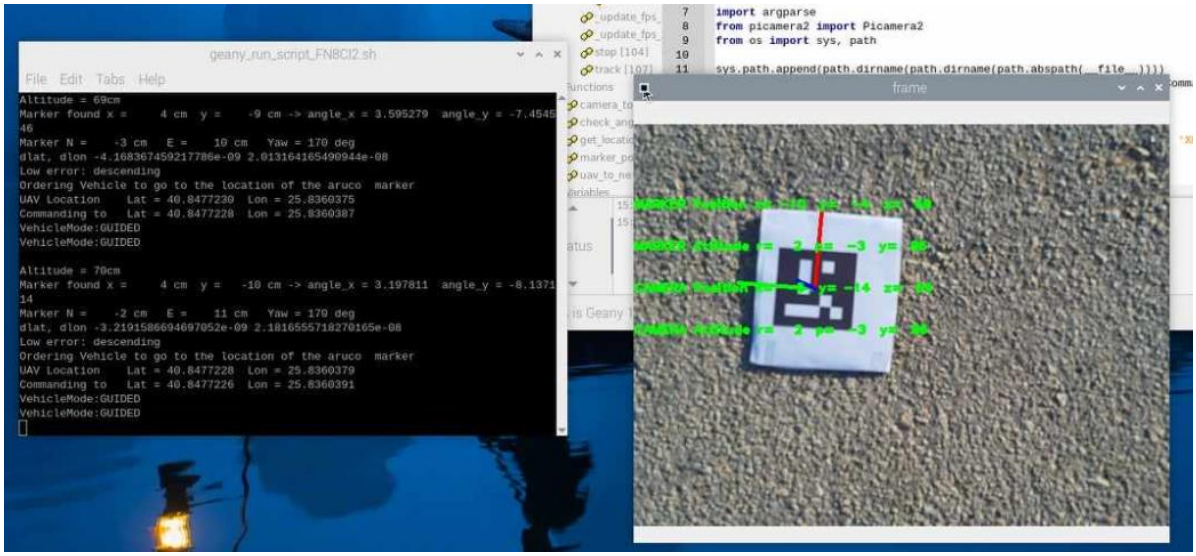
Εικόνα 40 131cm for LAND

Στην εικόνα 40,41 και 42 διακρίνεται η οθόνη του companion computer. Στην δεξιά πλευρά (frame) φαίνεται η οπτική γωνία της pi Module v3 camera και πάνω από τον ArUco marker εμφανίζονται οι δείκτες κατευθύνσεως του. Στην αριστερή πλευρά βρίσκεται το terminal του scipt και παρέχει πληροφορίες για τις συντεταγμένες του ArUco marker, τις συντεταγμένες του drone καθώς και στοιχεία σχετικά με το υψόμετρο, το τρέχων vehicle mode και τα yaw, pitch και roll.



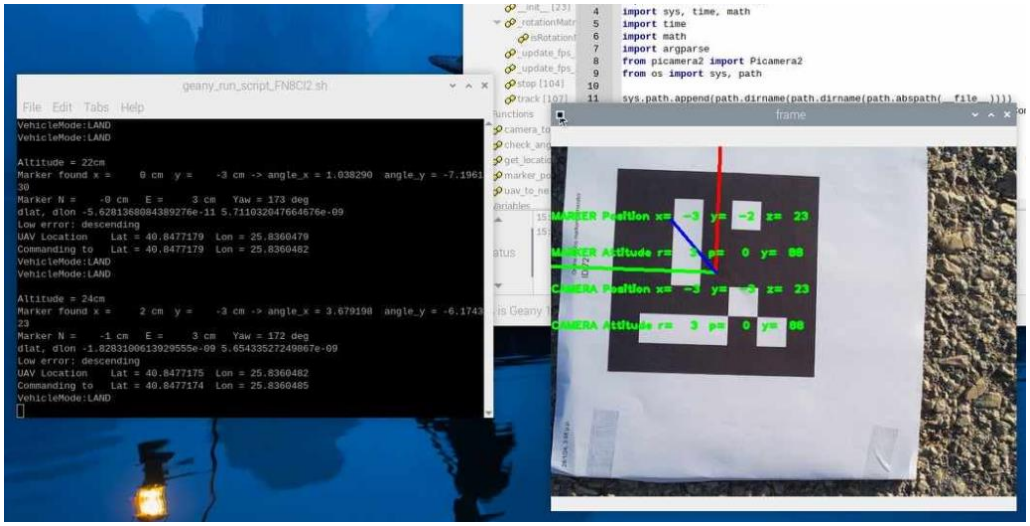
Εικόνα 41 104cm for LAND

Το vehicle mode σε αυτή την περίπτωση είναι ‘GUIDED’ και ο αλγόριθμος κατευθύνει σε πραγματικό χρόνο το μη επανδρωμένο αεροσκάφος κάθετα από τον μπλε τεχνητό δείκτη, ενώ ταυτόχρονα μειώνει την ισχύ των κινητήρων για αργή και ομαλή κάθοδο.



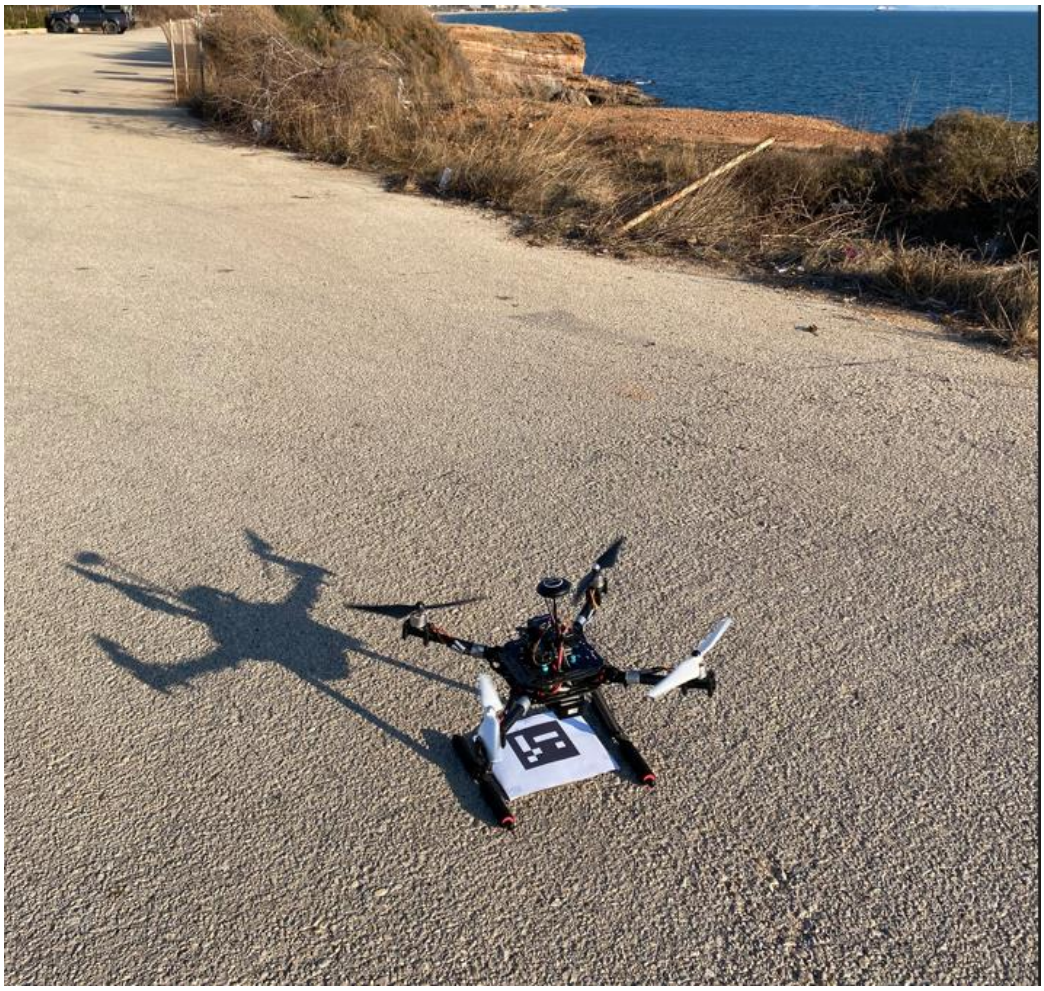
Εικόνα 42 70cm for LAND

Εικόνα 43 Preparing for landing using aruco marker and openCV



Εικόνα 44 24cm for LAND

Στην εικόνα 44 το UAV έχει φτάσει σε υψόμετρο 24 cm, το error είναι αρκετά χαμηλό και έχει ήδη διαταχθεί από το raspberry pi να προσγειωθεί. Το vehicle mode έχει αλλάξει σε 'LAND'. Το drone προσγειώνεται με πολύ μικρή απόκλιση στην βάση του.



Εικόνα 45 Landing using aruco marker and OpenCV

5.2: Συμβατικό Return to Home (RTH) mode

Την ίδια μέρα, έγιναν 5 δοκιμές και με το RTH mode χωρίς την χρήση του Raspberry pi 5 και κατ' επέκταση του αλγορίθμου ακριβούς προσγείωσης. Το drone απογειωνόταν από το σταθμό του και μετά από μια σύντομη πτήση διατασσόταν μέσω του τηλεχειριστήριου να επιστρέψει στο έδαφος.

Εικόνα 46 Preparing for landing using simple RTH

Το UAV, αν και ήταν αρκετά σταθερό κατά της διαδικασία της προσγείωσης, δεν κατάφερε να αποφέρει τα επιθυμητά αποτελέσματα.

Εικόνα 47 Landing using simple RTH

5.3: Αποτελέσματα

Τα αποτελέσματα της δοκιμής δεν θα πρέπει να είναι αντιπροσωπευτικά, διότι επηρεάζονται από τις συνθήκες περιβάλλοντος την δεδομένη χρονική στιγμή. Επίσης, διαφέρουν ανάλογα το είδος του μη επανδρωμένου αεροσκάφους και του hardware που φέρει. Παρά το γεγονός αυτό, αποτελούν μια αρκετά καλή εκτίμηση για την αποδοτικότητα της υλοποίησης και του αλγορίθμου σε σχέση με τις συμβατικές μεθόδους προσγείωσης.

Η μέτρηση απόκλισης προσγείωσης λήφθηκε από το κέντρο του ArUco marker και από το κέντρο του UAV (τοποθεσία κάμερας).

Στον πίνακα 1 αναγράφονται τα τελικά αποτελέσματα της παρούσας κατασκευής με την χρήση του Python κώδικα και του ArUco marker, ενώ στον πίνακα 2 τα αποτελέσματα από την κοινή προσγείωση με χρήση του συμβατικού GPS και Return to Home (RTH) mode.

Πίνακας 1 Αποτελέσματα υλοποίησης αλγορίθμου ακριβούς προσγείωσης

Αλγόριθμος ακριβούς προσγείωσης με OpenCV		
Αρ. Προσγείωσης	Απόκλιση από το κέντρο (cm)	Χρόνος προσγείωσης (s)
1	4,5	25
2	3	23
3	6,5	28
4	8	19
5	2	24
M.O	**4,8**	**23,8**

Τα 2 είδη δοκιμών έγιναν την ίδια μέρα με κοινές συνθήκες περιβάλλοντος και χωρίς καμία παρέμβαση στο hardware του συστήματος.

Πίνακας 2 Αποτελέσματα συμβατικού τρόπου προσγείωσης

Return to Home (RTH) mode		
Αρ. Προσγείωσης	Απόκλιση από το κέντρο (cm)	Χρόνος προσγείωσης (s)
1	130	12
2	224	15
3	201	13
4	55	9
5	162	12
M.O:	**154,4**	**12,2**

Η εφαρμογή του αλγορίθμου ακριβούς προσγείωσης με χρήση OpenCV στο UAV έδειξε αξιοσημείωτη ακρίβεια και αποτελεσματικότητα σε σύγκριση με την συμβατική μέθοδο RTH. Παρουσίασε ένα εντυπωσιακά χαμηλό σφάλμα προσγείωσης μόλις 4,8 cm, με χρόνο προσγείωσης 23,8 δευτερόλεπτα. Αντίθετα, το RTH mode που βασίζεται στο GPS module παρουσίασε σημαντικά υψηλότερο σφάλμα προσγείωσης 154,4 cm και χρόνο προσγείωσης 12,2 δευτερολέπτων. Τα ευρήματα αυτά υπογραμμίζουν την αποτελεσματικότητα και την υπεροχή της προσγείωσης ακριβείας με βάση την όραση του υπολογιστή σε σχέση με τις παραδοσιακές προσεγγίσεις που εξαρτώνται αποκλειστικά και μόνο από το GPS για την επίτευξη προσγειώσεων.

ΚΕΦΑΛΑΙΟ 6

ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα τελευταία χρόνια, έχει σημειωθεί σημαντική αύξηση της χρήσης μη επανδρωμένων αεροσκαφών (UAV), σε μεγάλο φάσμα εφαρμογών. Μία από τις κρίσιμες προκλήσεις στις επιχειρήσεις UAV είναι η επίτευξη ακριβών ελιγμών προσγείωσης, ειδικά σε δυναμικά και απρόβλεπτα περιβάλλοντα. Οι παραδοσιακές μέθοδοι προσγείωσης, δεν παρέχουν τα προσδοκώμενα αποτελέσματα, ιδιαίτερα σε σενάρια όπου τα σήματα GPS είναι αναξιόπιστα και προσδίδουν μικρή ακρίβεια. Για να αντιμετωπιστεί αυτή η πρόκληση, η τεχνολογία οδηγεί σε προηγμένες τεχνολογίες, όπως η μηχανική όραση, για να επιτραπούν δυνατότητες προσγείωσης υψηλής ακρίβειας για μη επανδρωμένα αεροσκάφη.

Το Computer Vision, ένα υπό-πεδίο της τεχνητής νοημοσύνης, επικεντρώνεται στο να επιτρέπει στις μηχανές να ερμηνεύουν και να κατανοούν οπτικές πληροφορίες. Τα UAV μπορούν να αναλύσουν οπτικά δεδομένα σε πραγματικό χρόνο, να ανιχνεύσουν εμπόδια και να εκτελέσουν ακριβείς ελιγμούς προσγείωσης με μεγάλη ακρίβεια. Ένα από τα βασικά συστατικά των συστημάτων προσγείωσης με βάση την Όραση Υπολογιστών είναι η χρήση δεικτών ArUco, οι οποίοι χρησιμεύουν ως σημεία αναφοράς για το UAV, για τον εντοπισμό και την πλοήγηση κατά τη διάρκεια της διαδικασίας προσγείωσης. Αυτοί οι δείκτες είναι εύκολα ανιχνεύσιμοι από τις ενσωματωμένες κάμερες και παρέχουν αξιόπιστες πληροφορίες, ακόμη και σε δύσκολες συνθήκες φωτισμού.

Η ενσωμάτωση της όρασης υπολογιστή σε συστήματα UAV προσφέρει πολλά πλεονεκτήματα σε σχέση με τις υπάρχουσες μεθόδους προσγείωσης. Τα συστήματα προσγείωσης με βάση την τεχνητή όραση μπορούν να επιτύχουν ακρίβεια κάτω των 5 εκατοστών, επιτρέποντας στα UAV να προσγειώνονται σε καθορισμένους στόχους με ελάχιστη απόκλιση. Αυτό το επίπεδο ακρίβειας είναι ζωτικής σημασίας για μεγάλο πλήθος εφαρμογών. Επιπλέον, τα συστήματα προσγείωσης με βάση την όραση υπολογιστή είναι πιο ανθεκτικά σε περιβαλλοντικούς παράγοντες που μπορούν να επηρεάσουν τα σήματα GPS. Αυτή η ανθεκτικότητα εξασφαλίζει αξιόπιστη απόδοση σε διαφορετικές συνθήκες λειτουργίας, συμπεριλαμβανομένων των αστικών περιβαλλόντων, εσωτερικών χώρων ή περιοχών με περιορισμένη κάλυψη GPS.

Με την ενσωμάτωση συνοδευτικών υπολογιστών όπως το Raspberry Pi 5 στον υπολογιστή πτήσης, τα συστήματα UAV μπορούν να αναπτύξουν εξελιγμένους αλγόριθμους όρασης υπολογιστή για αυτόνομη πλοήγηση, ανίχνευση εμποδίων και προσγείωσης ακριβείας. Αυτοί οι συνοδευτικοί υπολογιστές προσφέρουν επαρκή υπολογιστική ισχύ για την εκτέλεση εργασιών επεξεργασίας εικόνας, την εκτέλεση μοντέλων μηχανικής μάθησης και τη λήψη αποφάσεων σε πραγματικό χρόνο, με βάση τα οπτικά δεδομένα που συλλαμβάνονται από τις ενσωματωμένες κάμερες.

Καθώς τα UAV συνεχίζουν να διαδραματίζουν ολοένα και πιο ζωτικό ρόλο σε διάφορους κλάδους και εφαρμογές, η ικανότητα εκτέλεσης ακριβών και αξιόπιστων προσγειώσεων ανοίγει νέες δυνατότητες για μη επανδρωμένες αεροπορικές αποστολές. Τα UAV που είναι εξοπλισμένα με δυνατότητες προσγείωσης υψηλής

ακρίβειας μπορούν να αναπτυχθούν για κρίσιμες επιχειρήσεις όπως αποστολές έρευνας και διάσωσης, αντιμετώπιση καταστροφών ή παράδοση ιατρικών προμηθειών σε απομακρυσμένες ή απρόσιτες περιοχές, χωρίς ανθρώπινη παρέμβαση. Εν κατακλείδι, η ενσωμάτωση της υπολογιστικής όρασης σε συστήματα UAV ανοίγει το δρόμο για αυξημένη αυτονομία, ασφάλεια και αποτελεσματικότητα στις εναέριες λειτουργίες, συμβάλλοντας στη συνεχή πρόοδο των μη επανδρωμένων εναέριων τεχνολογιών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] ArduPilot Development Team. (2021). Pixhawk Overview. ArduPilot Documentation. https://ardupilot.org/copter/docs/common-pixhawk-overview.html

[2] AIMIND. (n.d.). Real-Time Hand Gesture Recognition Using OpenCV: A Step-by-Step Guide. AIMIND Blog. https://pub.aimind.so/real-time-hand-gesture-recognition-using-opencv-a-step-by-step-guide-2820618caa08

[3] Billinghurst, M., Clark, A., & Lee, G. (2015). A Survey of Augmented Reality.

[4] Bouguet, J. Y. (2008). Camera Calibration Toolbox for Matlab.

[5] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools, 25(11), 120-126.

[6] Fiala, M. (2005). ARTag, a fiducial marker system using digital techniques. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 2, pp. 590–596.

[7] Fuentes-Pacheco, J., Ruiz-Suarez, J. C., & Sucar, L. E. (2014). A Simple and Robust Approach for Marker-Based Augmented Reality.

[8] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition, 47(6), 2280-2292.

[9] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Fast detection of fiducial markers. Journal of Real-Time Image Processing, 10(4), 599-609. https://doi.org/10.1007/s11554-014-0447-x

[10] Hartley RI, Kang SB (2007) Parameter-free radial distortion correction with center of distortion estimation. IEEE Trans Pattern Anal Mach Intell 29.

[11] Howse, J., Minichino, J., & Packt Publishing. (2020). Learning OpenCV 4 Computer Vision with Python 3: Get to grips with tools, techniques, and algorithms for computer vision and machine learning, 3rd Edition (3rd ed.), Packt Publishing.

[12] Keller, A., & Ben-Moshe, B. (2022). A Robust and Accurate Landing Methodology for Drones on Moving Targets. Drones, 6(4), 98. https://doi.org/10.3390/drones6040098

[13] Khazetdinov, A., Zakiev, A., Tsoy, T., Svinin, M., & Magid, E. (2021). Embedded ArUco: A novel approach for high precision UAV landing. In 2021 International Siberian Conference on Control and Communications (SIBCON) (pp. 1–6).

[14] Marut, A., Wojtowicz, K., & Falkowski, K. (2019). ArUco markers pose estimation in UAV landing aid system. IEEE Xplore. https://ieeexplore.ieee.org/document/8869572

[15] Mráz, E., Rodina, J., & Babinec, A. (2020). Using fiducial markers to improve localization of a drone. In 2020 23rd International Symposium on Measurement and Control in Robotics (ISMCR) (pp. 1-6). IEEE. DOI: 10.1109/ISMCR51255.2020.9263754

[16] Muñoz-Salinas, R., & Garrido-Jurado, S. (2014). Fast detection of fiducial markers. Journal of Real-Time Image Processing, 10(4), 599-609. https://doi.org/10.1007/s11554-014-0447-x

[17] Nayak, S. (2020). Augmented Reality using ArUco Markers in OpenCV (C++ / Python). Learn OpenCV.https://learnopencv.com/augmented-reality-using-aruco-markers-in-opencv-c-python/

[18] Nguyen, X. M., Lee, J. Y., Lee, S. T., & Hong, S. K. (2021). Target state estimation for UAV, target tracking and precision landing control: Algorithm and verification system. In 2021 21st International Conference on Control, Automation and Systems (ICCAS) (pp. 173–177).

[19] OpenCV. (2000). The OpenCV Library. OpenCV. https://opencv.org/about/

[20] OpenCV. (2015). Cascade Classifier. OpenCV Documentation. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[21] OpenCV. (2019). Camera Calibration. OpenCV Documentation. https://docs.opencv.org/3.4/dc/dbb/tutorial_py_calibration.html

[22] OpenCV. (2023). Computer Vision in Agriculture: Challenges & Solutions. OpenCV Blog. https://www.opencv.ai/blog/computer-vision-in-agriculture-challenges-solutions

[23] OpenCV. (2019). ArUco Marker Detection (aruco module). OpenCV Documentation. https://docs.opencv.org/4.x/d9/d6d/tutorial_table_of_content_aruco.html

[24] OpenCV. (2019). Detection of ArUco Markers. OpenCV Documentation. https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html#:~:text=An%20ArUco%20marker%20is%20a,determines%20its%20identifier%20(id)

[25] OpenCV. (2019). Camera Calibration and 3D Reconstruction. OpenCV Documentation. https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

[26] OpenCV. (2019). Detection of ArUco Markers. OpenCV Documentation. https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

[27] Qengineering. (2022). Install OpenCV on Raspberry Pi. qengineering.eu.

https://qengineering.eu/install-opencv-on-raspberry-pi.html

[28] Raspberry Pi Foundation. (2023). Raspberry Pi 5. Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-5/

[29] Raspberry Pi Foundation. (2023.). Camera Module V2. Raspberry Pi. https://www.raspberrypi.com/documentation/accessories/camera.html

[30] Rong Liu, Jianjun Yi, Yajun Zhang, Bo Zhou, Wenlong Zheng, Hailei Wu, Shuqing Cao, & Jinzhen Mu. (2020). Vision-guided autonomous landing of multirotor UAV on fixed landing marker. In 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA) (pp. 455–458).

[31] Sampathkrishna, A. (2022, August 19). ArUco Marker-based Localization and Node Graph Approach to Mapping. arXiv:2208.09355v1 [cs.RO].

[32] Seth, M. (2020). Athlete Pose Detection Using OpenCV in Deep Learning. Medium. https://medium.com/swlh/athlete-pose-detection-3d1b93f2d82e

[33] Smisek, J., Stava, O., & Polok, L. (2013). OpenCV and ARToolKit comparison for natural feature tracking.

[34] Viola, P., & Jones, M. J. (2004). Robust real-time face detection. International Journal of Computer Vision, 57(2), 137–154.

[35] Wang, G., Liu, Z., & Wang, X. (October 2019). UAV Autonomous Landing using Visual Servo Control based on Aerostack. In the 3rd International Conference. DOI: 10.1145/3331453.3361667

[36] Zhengyou Zhang. (2000). A flexible new technique for camera calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22(11):1330–1334.

[37] Zhou, J., Wang, X.-M., & Wang, X.-Y. (2012). Automatic landing control of UAV based on optical guidance. In 2012 International Conference on Industrial Control and Electronics Engineering (pp. 152–155).