ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Πειραματική Υλοποίηση Σχήματος Ελέγχου Κίνησης με Οπτική Ανατροφοδότηση για Υποβρύχιο Ρομποτικό Όχημα

ΜΠΡΑΤΣΗΣ ΠΕΤΡΟΣ ΜΑΡΙΝΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Καρράς Γεώργιος
Αναπληρωτής Καθηγητής

Λαμία 15 Οκτωβρίου έτος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Πειραματική Υλοποίηση Σχήματος Ελέγχου Κίνησης με Οπτική Ανατροφοδότηση για Υποβρύχιο Ρομποτικό Όχημα

ΜΠΡΑΤΣΗΣ ΠΕΤΡΟΣ ΜΑΡΙΝΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Καρράς Γεώργιος
Αναπληρωτής Καθηγητής

Λαμία 15 Οκτωβρίου έτος 2024

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

Experimental Implementation of a Motion Control Scheme with Visual Feedback for an Underwater Robotic Vehicle

BRATSIS PETROS MARINOS

FINAL THESIS

ADVISOR

Karras Georgios
Associate Professor

Lamia 15 October year 2024

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

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

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

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

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

Ημερομηνία: 15/10/2024

Ο - Η Δηλ.

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

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία αναλύεται και υλοποιείται ο τρόπος με τον οποίο ένα υποβρύχιο ρομποτικό σύστημα μέσω της οπτικής ανατροφοδότησης και την χρήση αλγορίθμου ελέγχου κίνησης, κάνει ελεγχόμενη κίνηση σε χαρτογραφημένο χώρο. Αρχικά, υπάρχει μία εισαγωγή στην ρομποτική και συγκεκριμένα την υποβρύχια ρομποτική. Στην συνέχεια, υπάρχει εισαγωγικό κείμενο που αναλύει το ROS (ROBOT OPERATING SYSTEM) καθώς και τα πακέτα που χρησιμοποιήθηκαν σε αυτό. Επίσης, αναφέρονται πακέτα και αλγόριθμοι που είχαν συνεισφορά στην υποβρύχια χαρτογράφηση με την βοήθεια της οπτικής ανατροφοδότησης (κάμερας). Μετέπειτα, υπάρχει μία επεξήγηση του αλγορίθμου ελέγχου κίνησης. Τέλος , γίνεται ανάλυση των προβλημάτων και των λύσεων τους, τα οποία εμφανίστηκαν κατά την προσπάθεια διεξαγωγής του πειράματος.

ABSTRACT

In this thesis, the method by which an underwater robotic system utilizes visual feedback and a motion control algorithm to perform controlled movement in a mapped environment is analyzed and implemented. Initially, there is an introduction to robotics, with a specific focus on underwater robotics. Subsequently, an introductory text analyzes ROS (Robot Operating System) as well as the packages used within it. Additionally, the thesis discusses packages and algorithms that contributed to underwater mapping using visual feedback (camera). Next, there is an explanation of the motion control algorithm. Finally, the problems encountered during the experiment and their solutions are analyzed.

Table of Contents

ΠΕΡΙΛΗΨΗ I
ABSTRACT III

ΚΕΦΑΛΑΙΟ 1 ΕΙΣΑΓΩΓΗ 2

(ΥΠΟΚΕΦΑΛΑΙΟ 1.1)ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΡΟΜΠΟΤΙΚΗ 2
(ΥΠΟΚΕΦΑΛΑΙΟ 1.2)ΥΠΟΒΡΥΧΙΑ ΡΟΜΠΟΤΙΚΑ ΟΧΗΜΑΤΑ(ROVS) 2
(ΥΠΟΚΕΦΑΛΑΙΟ 1.3)ΧΡΗΣΗ ΚΑΙ ΕΦΑΡΜΟΓΕΣ ΤΩΝ ROVS 3
(ΥΠΟΚΕΦΑΛΑΙΟ 1.4)ΤΙ ΕΠΙΦΥΛΑΣΣΕΙ ΤΟ ΜΕΛΛΟΝ 4

ΚΕΦΑΛΑΙΟ 2 ΠΕΡΙΓΡΑΦΗ ΕΡΓΑΛΕΙΩΝ 5

(ΥΠΟΚΕΦΑΛΑΙΟ 2.1)BLUEROV2 5
(ΕΝΟΤΗΤΑ 2.1.Α)ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ 5
(ΕΝΟΤΗΤΑ 2.1.Β)ΕΙΚΟΝΕΣ 6
(ΥΠΟΚΕΦΑΛΑΙΟ 2.2)ΚΑΜΕΡΑ(SONY PS3) 8
(ΕΝΟΤΗΤΑ 2.2.Α)ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΚΑΜΕΡΑΣ 8
(ΕΝΟΤΗΤΑ 2.2.Β)ΟΠΤΙΚΗ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΚΑΜΕΡΑΣ 8
(ΥΠΟΚΕΦΑΛΑΙΟ 2.3)ΠΙΣΙΝΑ 11
(ΕΝΟΤΗΤΑ 2.3.Α)ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΠΙΣΙΝΑΣ 11
(ΕΝΟΤΗΤΑ 2.3.Β)ΟΠΤΙΚΗ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΠΙΣΙΝΑΣ 11
(ΥΠΟΚΕΦΑΛΑΙΟ 2.4)ROS(ROBOT OPERATING SYSTEM) 14
(ΕΝΟΤΗΤΑ 2.4.Α)ΕΙΣΑΓΩΓΗ ΣΤΟ ROS 14
(ΕΝΟΤΗΤΑ 2.4.Β)ΕΙΣΑΓΩΓΗ ΣΤΑ TRANSFORMATIONS 15
(ΕΝΟΤΗΤΑ 2.4.Γ) ΕΙΣΑΓΩΓΗ ΣΤΟ RQT 17
(ΕΝΟΤΗΤΑ 2.4.Δ) ΕΙΣΑΓΩΓΗ ΣΤΟ RVIZ 18
(ΥΠΟΚΕΦΑΛΑΙΟ 2.5)ROS ΚΑΙ BLUEROV2 20
(ΕΝΟΤΗΤΑ 2.5.Α)ΣΥΝΔΕΣΗ ΜΕΤΑΞΥ ROS ΚΑΙ BLUEROV2 20
(ΕΝΟΤΗΤΑ 2.5.Β)ΕΠΕΞΗΓΗΣΗ ΤΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΤΟΥ BLUEROV_ROS_PLAYGROUND 20
(ΕΝΟΤΗΤΑ 2.5.Γ)ΕΠΕΞΗΓΗΣΗ ΤΗΣ ΧΡΗΣΗΣ ΤΟΥ BLUEROV_ROS_PLAYGROUND 21

ΚΕΦΑΛΑΙΟ 3 LOCALIZATION,MAPPING AND MAVROS…… 23

(ΥΠΟΚΕΦΑΛΑΙΟ 3.1)VISUAL SLAM 23
(ΥΠΟΚΕΦΑΛΑΙΟ 3.2)FIDUCIAL SLAM 24
(ΕΝΟΤΗΤΑ 3.2.Α)ΕΠΕΞΗΓΗΣΗ ΤΩΝ FIDUCIAL MARKERS 24
(ΕΝΟΤΗΤΑ 3.2.Β)ΠΑΚΕΤΟ FIDUCIAL ΣΤΟ ROS 25
(ΕΝΟΤΗΤΑ 3.2.Γ) ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ ΤΟ ΠΑΚΕΤΟ FIDUCIALS 25
(ΕΝΟΤΗΤΑ 3.2.Δ)ΠΩΣ ΓΙΝΕΤΑΙ Η ΧΡΗΣΗ ΤΟΥ ΠΑΚΕΤΟΥ FIDUCIALS 26
(ΕΝΟΤΗΤΑ 3.2.Ε)ΚΩΔΙΚΑΣ ΤΟΥ ΠΑΚΕΤΟΥ FIDUCIALS 27

ΚΕΦΑΛΑΙΟ 4 ΑΛΓΟΡΙΘΜΟΣ ΕΛΕΓΧΟΥ ΚΙΝΗΣΗΣ…… 31

ΚΕΦΑΛΑΙΟ 5 ΣΥΜΠΕΡΑΣΜΑΤΑ..... 33

ΒΙΒΛΙΟΓΡΑΦΙΑ 37

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

(Υποκεφάλαιο 1.1) ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΡΟΜΠΟΤΙΚΗ

Ένας κλάδος της μηχανοηλεκτρονικής επιστήμης που επικεντρώνεται στον προγραμματισμό, την ανάπτυξη, τη λειτουργία και την αλληλεπίδραση των ρομποτικών μηχανών είναι η ρομποτική [1]. Ένα ρομπότ αποτελεί ένα μηχανικό σύστημα ικανό να εκτελεί καθήκοντα ή εργασίες αυτόνομα ή με ελάχιστη ανθρώπινη συμμετοχή. Η προαναφερθείσα επιστήμη ενσωματώνει ποικίλους τομείς, όπως η μηχανολογία, η ηλεκτρονική, η επιστήμη των υπολογιστών, η τεχνητή νοημοσύνη και ο αυτοματισμός. Οι χρήσεις αυτής εκτείνονται σε πολλούς τομείς, όπως η βιομηχανία, η ιατρική, η εξερεύνηση του διαστήματος, η αυτόνομη οδήγηση, η αεροδιαστημική και ποικίλοι άλλοι. Οι τεχνολογίες αυτές εφαρμόζονται για τη δημιουργία μηχανών που μπορούν να αναλάβουν ρόλο ως αντικαταστάτες του ανθρώπου.

Η έννοια της κατασκευής μηχανών που μπορούν να λειτουργούν αυτόνομα και να παίρνουν αποφάσεις χωρίς την ανθρώπινη παρουσία υπάρχει εδώ και πολλά χρόνια, αλλά άρχισε να αναπτύσσεται πραγματικά τον 20ο αιώνα. Ο βασικός στόχος της χρήσης των ρομπότ είναι να διευκολύνουν την ζωή των δημιουργών τους. Τον τελευταίο καιρό, η έρευνα στην ρομποτική έχει σημειώσει σημαντική πρόοδο, βελτιώνοντας ουσιαστικά την ανθρώπινη ζωή και οδηγώντας σε πρωτοποριακές νέες ιδέες.

Η Ρομποτική δημιουργεί μηχανολογικά συστήματα ικανά να αντικαταστήσουν τον κατασκευαστή τους και να μιμούνται τις κινήσεις του ανθρώπου. Τα ρομπότ βρίσκουν εφαρμογή σε διάφορες περιστάσεις και για διαφορετικούς στόχους, αλλά εν τη παρούσα εποχή πολλά από αυτά χρησιμοποιούνται σε περιβάλλοντα απαιτητικά και με υψηλό βαθμό κινδύνου (όπως η επιθεώρηση υλικών που εκπέμπουν ακτινοβολία, η ανίχνευση και η εξουδετέρωση εκρηκτικών μηχανισμών), σε κατασκευαστικές διεργασίες ή σε μέρη όπου η ανθρώπινη επιβίωση είναι αδύνατη (όπως στο διάστημα ή στον βυθό της θάλασσας). Στα συστήματα αυτά δεν υπάρχει περιορισμός όσον αφορά την μορφή τους, αλλά κάποια σχεδιάζονται να έχουν εμφάνιση ανθρώπου. Η εμφάνιση τους αυτή θεωρείται ότι συμβάλλει στην αποδοχή τους, ειδικά σε δραστηριότητες που συνήθως εκτελούνται από ανθρώπους.

Για να είναι αποτελεσματικό, ένα ρομπότ πρέπει να διαθέτει την ικανότητα ευελιξίας και προσαρμογής στον χώρο χρήσης του. Αυτό επιτυγχάνεται με τη χρήση ενός συστήματος ελέγχου και αισθητήρων. Οι αισθητήρες καταγράφουν πληροφορίες από το χώρο λειτουργίας τους, όπως για παράδειγμα την μέτρηση θερμοκρασίας ή την επεξεργασία εικόνων, ενώ το ελεγκτικό σύστημα αναλύει αυτά τα δεδομένα και προσαρμόζει τη συμπεριφορά του αναλόγως.

Το ελεγκτικό σύστημα ενός ρομπότ κατηγοριοποιείται με βάση τον τρόπο λειτουργίας του σε δύο τύπους. Ο πρώτος τύπος είναι ο έλεγχος ανοικτού βρόχου, εκεί όπου οι εντολές δίνονται ανάλογα με τα δεδομένα που συλλέγονται από τους αισθητήρες, αλλά δεν είναι υπαρκτή η επιστροφή πληροφοριών που να επιτρέπει την προσαρμογή της συμπεριφοράς του ρομπότ. Ο δεύτερος τύπος είναι ο έλεγχος κλειστού βρόχου, οπού οι αισθητήρες αποστέλλουν δεδομένα για την κατάσταση αυτού και οι συλλεχθέντες πληροφορίες είναι καθοριστικές στην ρύθμιση των εντολών και της συμπεριφοράς του.

(Υποκεφάλαιο 1.2) ΥΠΟΒΡΥΧΙΑ ΡΟΜΠΟΤΙΚΑ ΟΧΗΜΑΤΑ (ROVs)

Στην εποχή μας, παρατηρούμε μια αυξανόμενη παρουσία τηλεχειριζόμενων υποβρύχιων οχημάτων, τα οποία χρησιμοποιούνται συχνά. Αυτά τα υποβρύχια ρομποτικά οχήματα αποτελούνται από ένα υποβρύχιο μηχανισμό, την επιφανειακή ελεγκτική μονάδα και το καλώδιο δημιουργεί την σύνδεση. Η καθοδήγηση τους επιτυγχάνεται από τον χειριστή μέσω της ελεγκτικής μονάδας, η οποία πρέπει να είναι παρούσα συνεχώς για την διευκόλυνση της εκτέλεσης των αποστολών τους. Τα αυτόνομα υποβρύχια ρομποτικά οχήματα (AUVs) [2] είναι σχεδιασμένα ώστε η λειτουργία τους να είναι αυτόνομη δίχως την παρέμβαση ανθρώπων. Χρησιμοποιούνται για διάφορες αποστολές, όπως διερεύνηση υποθαλάσσιων περιοχών, έρευνα των ωκεανών, επιτήρηση του περιβάλλοντος, εύρεση και ανάκτηση αντικειμένων και άλλες χρήσεις. Συνήθως εξοπλίζονται με αισθητήρες διαφορετικών ειδών, όπως ακουστικούς (sonar), οπτικούς (κάμερες) και αισθητήρες που μετρούν φυσικές παραμέτρους, που είναι σημαντικοί για την επιτυχή εκτέλεση των έργων τους.

Επιπρόσθετα, τα AUVs έχουν ελεγκτικό σύστημα πλοήγησης, με το οποίο γίνεται επιτρεπτή η αυτόματη πλοήγηση και η παρακολούθηση τους. Η τεχνολογία όπου χρησιμοποιούν αυτά ,αναβαθμίζεται συνεχώς, με αποτέλεσμα την συνεχή δοκιμή νέων αισθητήρων σε αυτά, που μεγαλώνει το εύρος αυτονομίας τους, αλλά και την μέγιστη ταχύτητα και τη βέλτιστη ακρίβεια των πλοηγικών λειτουργιών τους. Ακόμη, τα AUVs έχουν την δυνατότητα δημιουργίας ομάδων μεταξύ τους ώστε να υπάρχει συνεργασία για την εκτέλεση συντονισμένων αποστολών και την ανταλλαγή πληροφοριών.

Συνολικώς, τα Αυτόνομα Υποβρύχια Οχήματα αποτελούν ένα σημαντικό κομμάτι της τεχνολογίας των ρομποτικών οχημάτων, προσφέροντας μια λύση αξιοπιστίας για τη συλλογή και αποκόμιση πληροφοριών από τον υποβρύχιο χώρο, δίχως την ανάγκη διαρκούς επιτήρησης και καθοδήγησης από έναν χειριστή.

(Υποκεφάλαιο 1.3) ΧΡΗΣΗ ΚΑΙ ΕΦΑΡΜΟΓΕΣ ΤΩΝ ROVs

Μικρότερο του 20% του υποθαλάσσιου περιβάλλοντος έχει ανακαλυφθεί μέχρι σήμερα. Αυτό οφείλεται στις δύσκολες συνθήκες που καθιστούν δύσκολη την επιβίωση του ανθρώπου σε αυτά. Μία από τις κύριες προκλήσεις που έρχεται αντιμέτωπος η ανθρώπινη εξερεύνηση του βυθού είναι οι αντίξοες συνθήκες που υπάρχουν κάτω από την επιφάνεια της θάλασσας, όπως η επιβλαβής επίδραση του οξυγόνου και η ασθένεια των δυτών. Εξαιτίας αυτού, η κατάδυση ενός δύτη στα βαθιά είναι εξαιρετικά επικίνδυνη και δαπανηρή. Παρόλα αυτά, υπάρχουν πολλές υποβρύχιες αποστολές, όπως η αλλαγή ή η επισκευή υποθαλάσσιων καλωδιώσεων και η χαρτογράφηση της θαλάσσιας επιφάνειας.

Όλο και περισσότερες ναυτιλιακές εταιρείες χρησιμοποιούν drones και τηλεχειριζόμενα οχήματα (ROV) για να πραγματοποιούν ελέγχους και επιθεωρήσεις, χωρίς να απαιτείται η φυσική παρουσία μελών του πληρώματος ή ειδικών. [3] Μία από τις ναυτιλιακές εταιρείες που έχουν επενδύσει σε drones είναι η KLine (Kawasaki Kisen Kaisha). Η ιαπωνική εταιρεία ανακοίνωσε πρόσφατα ότι προχωρά σε δοκιμαστικές επιθεωρήσεις και επισκευές της γάστρας των πλοίων της με τη χρήση drones και ενός συστήματος ανάλυσης εικόνας, που αναπτύχθηκαν σε συνεργασία με την ιαπωνική εταιρεία Technos Mihara Corporation.

Μια ακόμη ιαπωνική εταιρεία, η MOL, ακολουθεί το παράδειγμα της KLine. Πρόσφατα ανακοίνωσε ότι χρησιμοποίησε δοκιμαστικά ένα υποβρύχιο όχημα (ROV) για να επιθεωρήσει τμήματα της γάστρας ενός πλοίου της, το οποίο χρησιμοποιείται για την πόντιση υποβρύχιων καλωδίων. Μέχρι τώρα, τέτοιες έρευνες γίνονταν από δύτες και μόνο όταν το επέτρεπαν οι καιρικές συνθήκες, κάτι που συχνά αύξανε σημαντικά τον χρόνο και το κόστος των επιθεωρήσεων. Πλέον, η χρήση των ROVs αναμένεται να αλλάξει δραστικά τις συνθήκες επιθεωρήσεων και επισκευών στα πλοία.

(Υποκεφάλαιο 1.4) ΤΙ ΕΠΑΚΟΛΟΥΘΕΙ ΣΤΟ ΜΕΛΛΟΝ

Ανεξάρτητα από την τεχνολογία που αναπτύσσεται, η δυνατότητα να αντιμετωπίζονται περισσότερες υποβρύχιες εργασίες ανοίγει νέες αγορές [4] . Η αυξημένη ασφάλεια που προκύπτει από τη μείωση της ανάγκης για επανδρωμένες καταδύσεις οδηγεί επίσης σε μεγαλύτερη χρήση των UUVs. Η απαίτηση για εκτέλεση περισσότερων εργασιών με πιο μικρές ρομποτικές λύσεις γίνεται εφικτή με τη μείωση του χώρου που απαιτείται στο κατάστρωμα και το μικρότερο μέγεθος των σκαφών, καθώς και με γρηγορότερη και λιγότερο κοστοβόρα κινητοποίηση. Αυτές οι λύσεις επιτρέπουν την εργασία σε πιο δυνατά ρεύματα και βαθύτερα νερά.

Οι τεχνολογίες αυτονομίας και ενσωμάτωσης είναι οι πρωτεύων οδηγοί αυτής της εξέλιξης. Τα συστήματα αυτονομίας των ROV αναγκάζουν να επανεξεταστεί ο τρόπος που οργανώνονται οι επιχειρήσεις που λαμβάνουν χώρα εκτός των ακτών, δίνοντας τη δυνατότητα να παραμένουν τα πληρώματα στη στεριά αντί για την ανοιχτή θάλασσα. Αυτές οι τεχνολογίες δίνουν τη δυνατότητα στη χρήση υποθαλάσσιων συστημάτων και ρομποτικών οχημάτων από μια μεγαλύτερη ομάδα χρηστών, όπως πρώτους ανταποκριτές, ειδικές δυνάμεις και παρόχους υπηρεσιών πλοιοκτησίας. Ωστόσο, δεν είναι η αυτονομία από μόνη της που είναι σημαντική, αλλά και το πώς γίνεται η ενσωμάτωσή της στο συνολικό σύστημα, είτε μιλάμε για ROV, UUV ή το ανθρώπινο στοιχείο.

Μικραίνουν οι διαφορές ανάμεσα στα "ROV" και "AUV" και μειώνεται η τεχνική απόσταση ανάμεσα επανδρωμένων και μη συστημάτων. Επιτρέπεται από την αυτονομία να κατανοούμε και να συνεργαζόμαστε με πιο έξυπνα οχήματα σε διάφορες εφαρμογές, είτε με επίβλεψη είτε χωρίς. Η αλλαγή αυτή κάνει πιο σύνηθες την χρήση των UUV, καθώς δεν επιβάλλεται πλέον να είσαι ειδικός χειριστής ROV για να τα χρησιμοποιήσεις.

Η αγορά πιέζει για πιο αποδοτικά και αποτελεσματικά οχήματα, με πρωτεύων στόχο την ενίσχυση της αξίας για τους χρήστες. Το παραδοσιακό μοντέλο λειτουργίας βασιζόταν σε ομάδες που περιλάμβαναν χειριστές οχημάτων, πιλότους, τεχνικούς και εμπειρογνώμονες . Για να ελεγχθεί το κόστος, γίνεται μελέτη μείωσης του προσωπικού που χρειάζεται για να λειτουργήσουν αυτά. Έτσι, οι πιλότοι ROV και οι τεχνικοί εκπαιδεύονται να γίνουν εμπειρογνώμονες , ή οι εμπειρογνώμονες εκπαιδεύονται να γίνουν πιλότοι ROV. Εναλλακτικά, κατασκευάζονται οχήματα μεγαλύτερης ποιότητας που δεν χρειάζονται επιτόπου τεχνική υποστήριξη και μπορούν να λειτουργούν με ακριβείς οδηγίες από τους κατασκευαστές.

Μεγαλύτερη αποτελεσματικότητα μπορεί να επιτευχθεί με την ενίσχυση της αυτονομίας στις αποστολές των ROV. Όσο μεγαλύτερη είναι η αυτονομία των ROV, τόσο επιτρέπεται η χρήση τους σε ευρύτερο φάσμα χρηστών, περιλαμβανομένου χειριστών χωρίς να έχουν λάβει ειδική εκπαίδευση χρήσης τους, να χρησιμοποιούν τα ROVs για υποθαλάσσιες διεργασίες.

ΚΕΦΑΛΑΙΟ 2 ΠΕΡΙΓΡΑΦΗ ΕΡΓΑΛΕΙΩΝ

(Υποκεφάλαιο 2.1) BLUEROV2

(Ενότητα 2.1.α)ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Το BlueRov 2 [5] είναι ένα ROV (remotely operated vehicle), δηλαδή ένα υποβρύχιο όχημα αυτόματου ελέγχου το οποίο έχει δημιουργηθεί για διεργασίες υποθαλάσσιας ρομποτικής όπως η εξερεύνησή της θάλασσας ή η χαρτογράφησης αυτής. Το ρομπότ αυτό αναπτύχθηκε από την εταιρία Blue Robotics [6], που έχει ειδίκευση στην κατασκευή οικονομικών υποβρυχίων.

Το ρομπότ αυτό έχει δυνατή κατασκευή αφού του επιτρέπεται να βυθιστεί έως και 300 μέτρα κάτω από την επιφάνεια της θάλασσας. Το βάρος του απλού BlueRov2, διότι υπάρχει και το heavy το οποίο ζυγίζει περισσότερο από το κανονικό, είναι 18 κιλά και το μήκος του απλού είναι κάπου κοντά στο μισό μέτρο και οι κινητήρες που έχει τροφοδοτούνται από ηλεκτρικές μπαταρίες που του προσφέρουν πρωτόγνωρη αυτονομία.

Όπως τα περισσότερα ROV οχήματα , έτσι και αυτό έχει αισθητήρες που του δίνουν την δυνατότητα να διαβάζει το περιβάλλον γύρω του, να ανιχνεύει την θερμοκρασία του νερού και την πίεση που υπάρχει στην συγκεκριμένη θέση, αλλά και κάμερες υψηλής ευκρίνειας για την λήψη εικόνων και βίντεο καθ΄ όλη την διάρκεια της αποστολής. Αυτό το αυτόματο υποβρύχιο ρομπότ όταν υποστηρίζεται από το λογισμικό που ονομάζεται ROS, το οποίο βοηθάει στην λειτουργία γενικά πολλών ρομπότ, αποκτάει την δυνατότητα επίτευξης αποστολών αυτόνομα.

Επιπρόσθετα, όπως έχει ειπωθεί στο πρώτο κεφάλαιο, πλέον αναλαμβάνουν αποστολές που αφορούν τον τομέα της ναυτιλίας , σαν τη έρευνα του υποθαλάσσιου κόσμου, τον έλεγχο πλοίων αλλά και την επισκευή τους. Το χαμηλό κόστος αγοράς του το κάνει προσιτό για ερευνητικά κέντρα αλλά και για ερασιτέχνες στον τομέα της ρομποτικής. Εν κατακλείδι , το BlueRov2 είναι πολύ δημοφιλές υποβρύχιο στην ρομποτική κοινότητα γιατί προσφέρει ευελιξία και αυτονομία.

ΟΝΟΜΑ: BlueRov2

ΚΑΤΑΣΚΕΥΑΣΤΗΣ: Blue Robotics

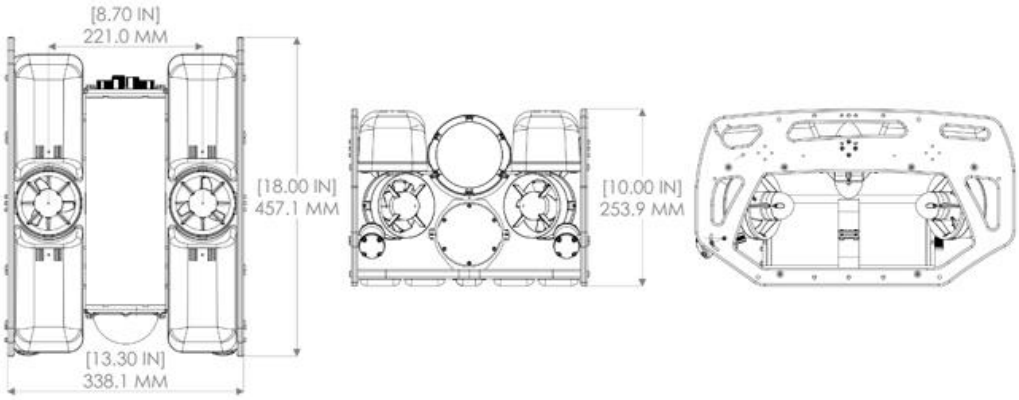
ΔΙΑΣΤΑΣΕΙΣ: 457 mm x 338 mm x 254 mm

ΚΙΝΗΤΗΡΕΣ: 6 Blue Robotics T200

ΒΑΡΟΣ (απλού μοντέλου): 12kg

ΜΕΓΙΣΤΟ ΒΑΘΟΣ ΚΑΤΑΔΥΣΗΣ: 300m

(Ενότητα 2.1.β)ΕΙΚΟΝΕΣ



(Εικόνα 1η: Σχεδιασμός και διαστάσεις BlueRov2)

(Εικόνα 2η: Φυσική απεικόνιση του υποβρυχίου BlueRov2)

(Εικόνα 3η και 4η: Διαφορετικές οπτικές γωνίες του υποβρυχίου)

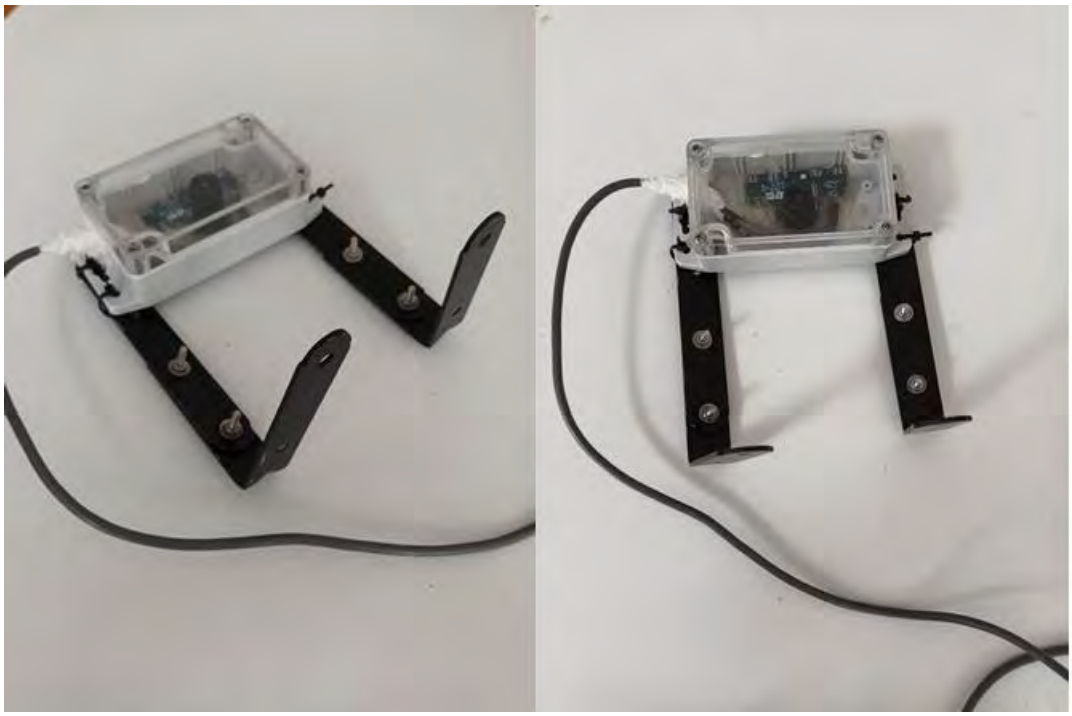
(Υποκεφάλαιο 2.2) ΚΑΜΕΡΑ(SONY PS3)

(Ενότητα 2.2.α)ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΚΑΜΕΡΑΣ

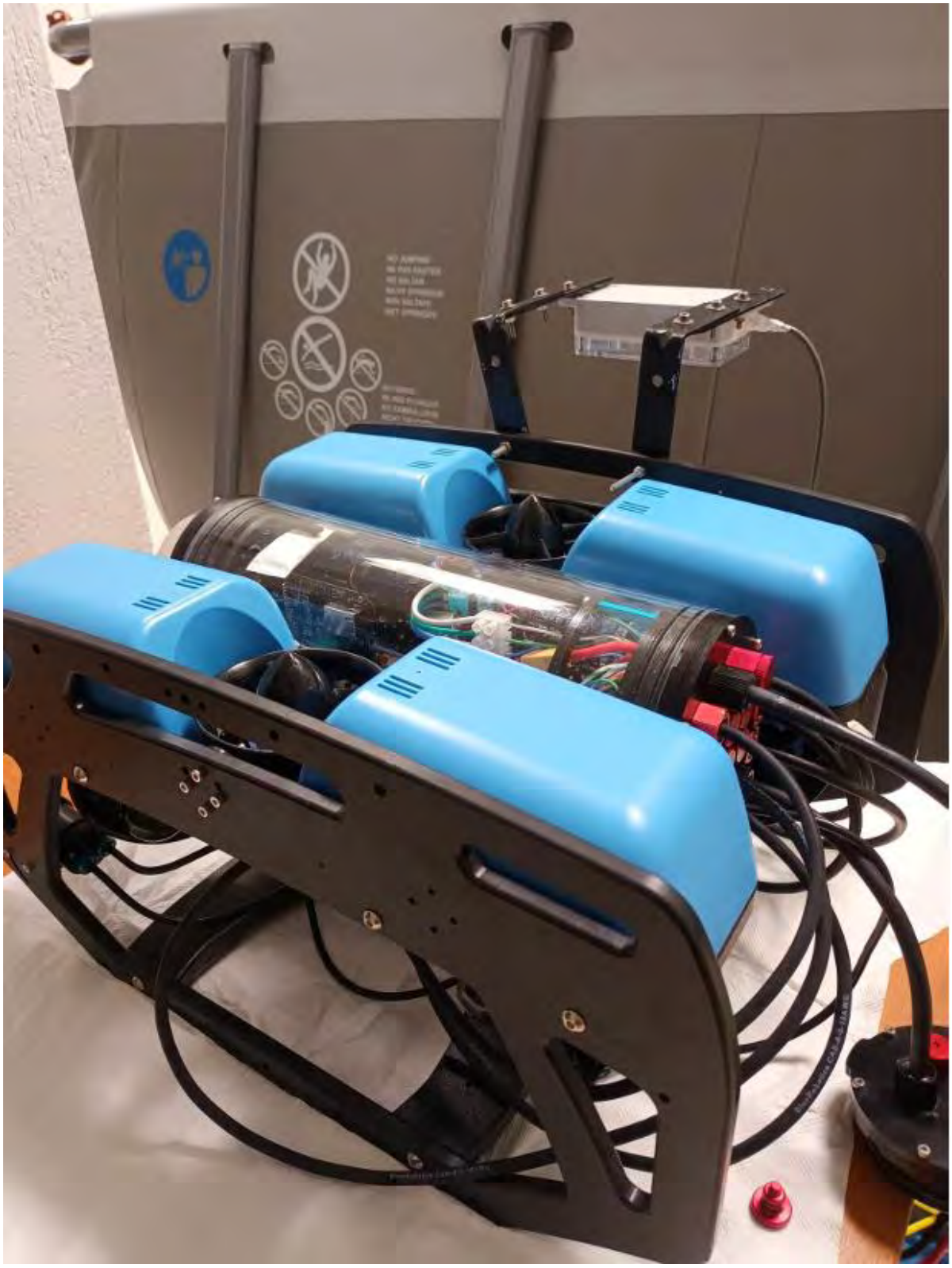
Η κάμερα που βοήθησε στην εύρεση των markers της πισίνας, δηλαδή την δημιουργία ενός χάρτη αυτής, είναι η SONY-PS3. Η κάμερα τοποθετήθηκε δεξιά του υποβρυχίου , αφού πρώτα προστέθηκε σε αυτή μία μεταλλική βάση για να βοηθάει στην ισορρόπησή της. Όμως η κάμερα δεν είναι αδιάβροχη και για να προστατευτεί από το νερό επικαλύφθηκε με ένα αδιάβροχο κάλυμμα το οποίο πάρθηκε έτοιμο διότι είχε γίνει παλαιότερα υλοποίηση αλγορίθμου ανίχνευσης οπτικών δεικτών από το εργαστήριο ρομποτικής του Πανεπιστημίου Θεσσαλίας. Για να υπάρχει ισορροπία στην πλεύση του ρομπότ , έπρεπε να αφαιρεθεί ένα από τα βαρίδια που διαθέτει αυτό.

Η κάμερα συνδέεται άμεσα με τον ηλεκτρονικό υπολογιστή που γίνεται και η σύνδεση του υποβρυχίου, μέσω ενός καλωδίου που διαθέτει αυτή , μήκους περίπου δέκα μέτρων. Αυτή η σύνδεση δεν περιορίζει σε καμία περίπτωση την κίνηση του ρομπότ αφού η πισίνα έχει μήκος 6 μέτρα.

(Ενότητα 2.2.β)ΟΠΤΙΚΗ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΚΑΜΕΡΑΣ



(Εικόνα 5η: Φυσική απεικόνιση της κάμερας SONY-PS3 και του αδιάβροχου καλύμματός της)



(Εικόνα 6η: Απεικόνιση της κάμερας SONY-PS3 συνδεδεμένη στο υποβρύχιο BlueRov2)

(Εικόνα 7η και 8η: Η παραπάνω απεικόνιση από διαφορετικές οπτικές γωνίες)

(Υποκεφάλαιο 2.3) ΠΙΣΙΝΑ

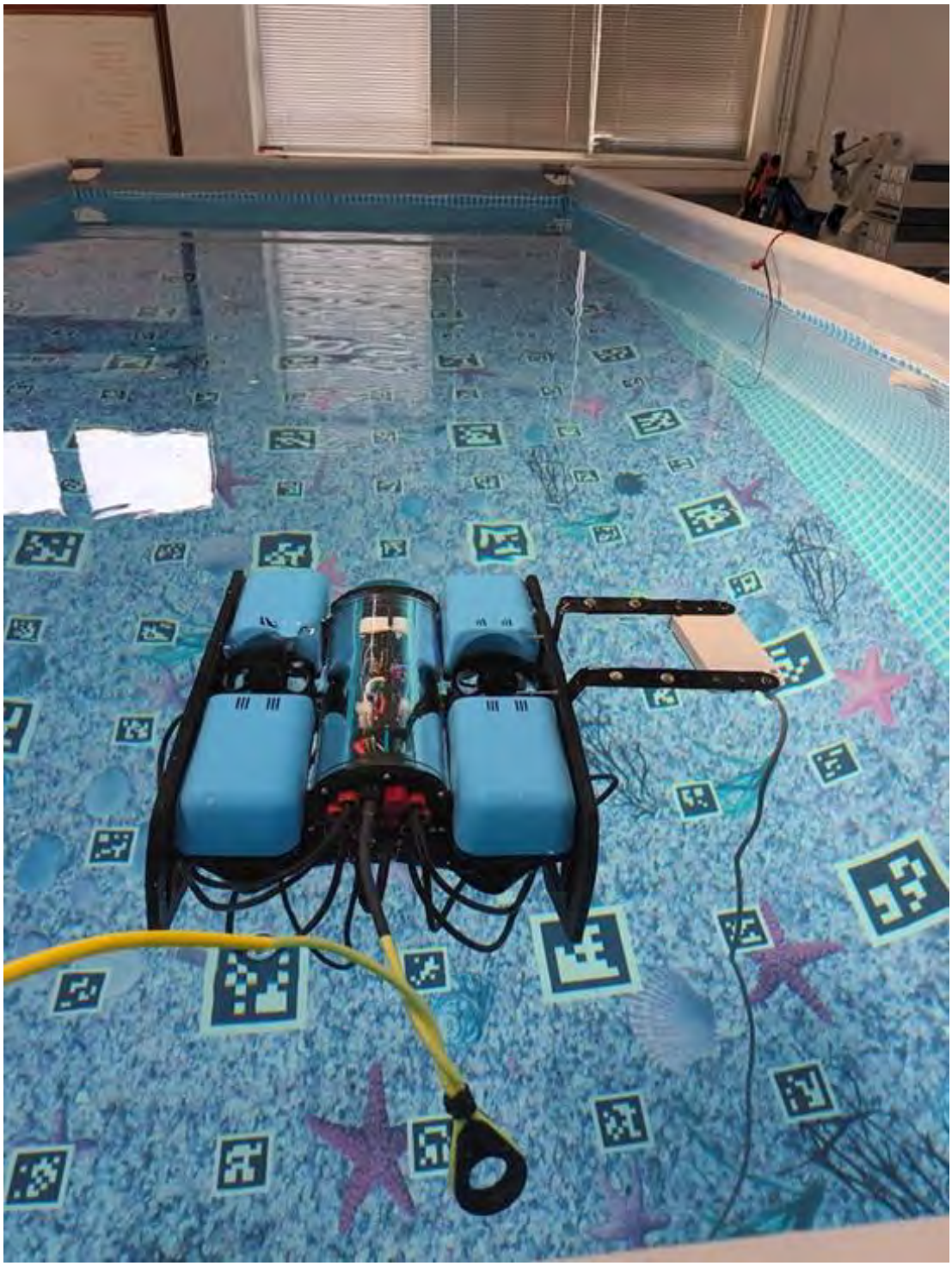
(Ενότητα 2.3.α) ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΠΙΣΙΝΑΣ

Για να πραγματοποιηθεί κάποιο πείραμα που αφορά υποβρύχια ρομποτικά οχήματα, το εργαστήριο Ρομποτικής του Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Θεσσαλίας έχει πισίνα η οποία απεικονίζεται στις παρακάτω εικόνες. Αυτή έχει μήκος και πλάτος 6 και 3 μέτρα αντίστοιχα, καθώς και βάθος 1,5 μέτρων, κάτι που δεν επιτρέπει την εύκολη τοποθέτηση της κάμερας και για αυτό τον λόγω όπως είδαμε παραπάνω η κάμερα μπήκε δεξιά στο υποβρύχιο και όχι στο κέντρο αυτού, για να έχει καλύτερο οπτικό πεδίο κάτι που θα κάνει την χαρτογράφηση του βυθού της πισίνας και των 171 δεικτών (Aruco Markers) με περίμετρο 10εκ. x 10εκ πιο εύκολη και πιο γρήγορη.

(Ενότητα 2.3.β) ΟΠΤΙΚΗ ΑΠΕΙΚΩΝΙΣΗ ΤΗΣ ΠΙΣΙΝΑΣ

(Εικόνα 9η:Πισίνα εργαστηρίου Ρομποτικής διαστάσεων 6m x 3m x 1,5m)

(Εικόνα 10η:Πισίνα Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Θεσσαλίας)



(Εικόνα 11η:Βύθιση του υποβρυχίου στην πισίνα)

(Υποκεφάλαιο 2.4) ROS (ROBOT OPERATING SYSTEM)

(Ενότητα 2.4.α) ΕΙΣΑΓΩΓΗ ΣΤΟ ROS

Το ROS (ROBOT OPERATING SYSTEM) [7] είναι μία πλατφόρμα που αποτελείται από λογισμικό ανοικτού κώδικα, το οποίο δημιουργήθηκε για να βοηθήσει στην ανάπτυξη εφαρμογών που αφορούν ρομποτικά οχήματα. Ένα ερευνητικό εργαστήριο ξεκίνησε την ανάπτυξη του , το Willow Garage [8], προτού δοθεί για την περαιτέρω εξέλιξή του και συντήρησή στο Open Source Robotics Foundation (OSRF) [9].

Το λογισμικό αυτό, διαθέτει πληθώρα εργαλείων, λειτουργιών αλλά και βιβλιοθηκών που δίνουν την δυνατότητα στην δημιουργία, την δοκιμή αλλά και την λειτουργία ρομποτικών συστημάτων. Το ROS στοχεύει στην εύκολη ανταλλαγή πληροφοριών μεταξύ των πολλαπλών υποσυστημάτων που διαθέτει το ρομπότ αλλά και στην επανεκτέλεση των διαφόρων αλγορίθμων και του κώδικα σωστής λειτουργίας που χρησιμοποιεί αυτό.

Αν και το ROS, το οποίο μεταφράζεται ως robot operating system, δηλαδή ρομποτικό λειτουργικό σύστημα, δεν αποτελεί πραγματικό operating system ,αλλά είναι ένα λογισμικό πλαίσιο που λειτουργεί πάνω σε ήδη υπάρχοντα λειτουργικά συστήματα όπως είναι τα Windows ή τα Linux.

Παρακάτω θα αναφερθούν με λίγα λόγια οι βασικές αρχές λειτουργίας του ROS:

- Master: Χωρίς αυτό, δεν θα ήταν δυνατή η εύρεση των κόμβων μεταξύ τους και αυτό επακολουθεί πρόβλημα στην ανταλλαγή μηνυμάτων ή στο κάλεσμα υπηρεσιών. Το ROS Master δίνει την δυνατότητα καταχώρισης ονόματος και εύρεση στο Υπολογιστικό Γράφημα.

- Κόμβοι (Nodes): Το ROS βασίζεται σε μία αρχιτεκτονική, δυαδική, που θεμελιώνεται με τους κόμβους. Κάθε ένας από αυτούς ,σαν ανεξάρτητο υποσύστημα, είναι εκτελεστική αρχή για μοναδικές λειτουργίες. Η επικοινωνία μεταξύ τους θεσπίζεται μέσω των μηνυμάτων.

- Μηνύματα (Messages): Το λειτουργικό ROS για την επίτευξη της σωστής μεταφοράς δεδομένων, διαθέτει σύστημα που ορίζει τα κατάλληλα μηνύματα για την διεργασία αυτή. Αυτά είναι πολύτιμα για την ανταλλαγή πληροφοριών μεταξύ των κόμβων.

- Θέματα (Topics): Ο κάθε κόμβος έχει την δυνατότητα να δημοσιεύει πληροφορίες στα θέματα, και ο κάθε ένας από αυτούς μπορεί εγγραφεί στο θέμα το οποίο έχει τις επιθυμητές πληροφορίες. Οπότε, τα θέματα αποτελούν μηχανισμό που δουλεύουν με την σχέση δημοσίευσης/συνδρομής και δημιουργούν την δυνατότητα ανταλλαγή μηνυμάτων μεταξύ πολλών κόμβων.

- Υπηρεσίες (Services): Η πρόσβαση ενός κόμβου σε λειτουργίες που ανήκουν σε άλλους κόμβους γίνεται δυνατή από τις υπηρεσίες. Αυτές προσφέρουν αντιθέτως με την ξεπερασμένη ανταλλαγή μηνυμάτων όπως γίνεται στα θέματα, εκσυγχρονισμένη σχέση κλήσης/απόκρισης.

- Πακέτα (Packages): Η οργάνωση του κώδικα και των βιβλιοθηκών του ROS γίνεται σε πακέτα. Το κάθε πακέτο κρατά αρχεία πόρων αλλά και τον κώδικα για λειτουργικό σύνολο συγκεκριμένο.

- Bags: Τα bags αποτελούν μία μορφή αποθήκευσης και αναπαραγωγής των δεδομένων των μηνυμάτων του συστήματος. Αυτά είναι κομβικός μηχανισμός για την αποθήκευση δεδομένων, που μπορεί να είναι δύσκολη η συλλογή τους αλλά είναι σημαντική αν το επιθυμητό αποτέλεσμα είναι η δοκιμή ή η ανάπτυξη κάποιου αλγορίθμου.

Το ROS διαθέτει ποικιλία εργαλείων για την επίτευξη στόχων όπως η οπτικοποίηση δεδομένων, η διόρθωση σφαλμάτων, η ενημέρωση του χρήστη για την κατάσταση του λειτουργικού συστήματος και διαφόρων άλλων. Επιπροσθέτως, υπάρχει ενεργή κοινότητα χρηστών που προσφέρουν πολύτιμη βοήθεια αλλά και πακέτα που έχουν δημιουργήσει οι ίδιοι, ώστε να υπάρξει μεγάλη υποστήριξη μεταξύ των χρηστών που συνεπάγεται στην σφοδρή ανάπτυξη πολλών τομέων της ρομποτικής.

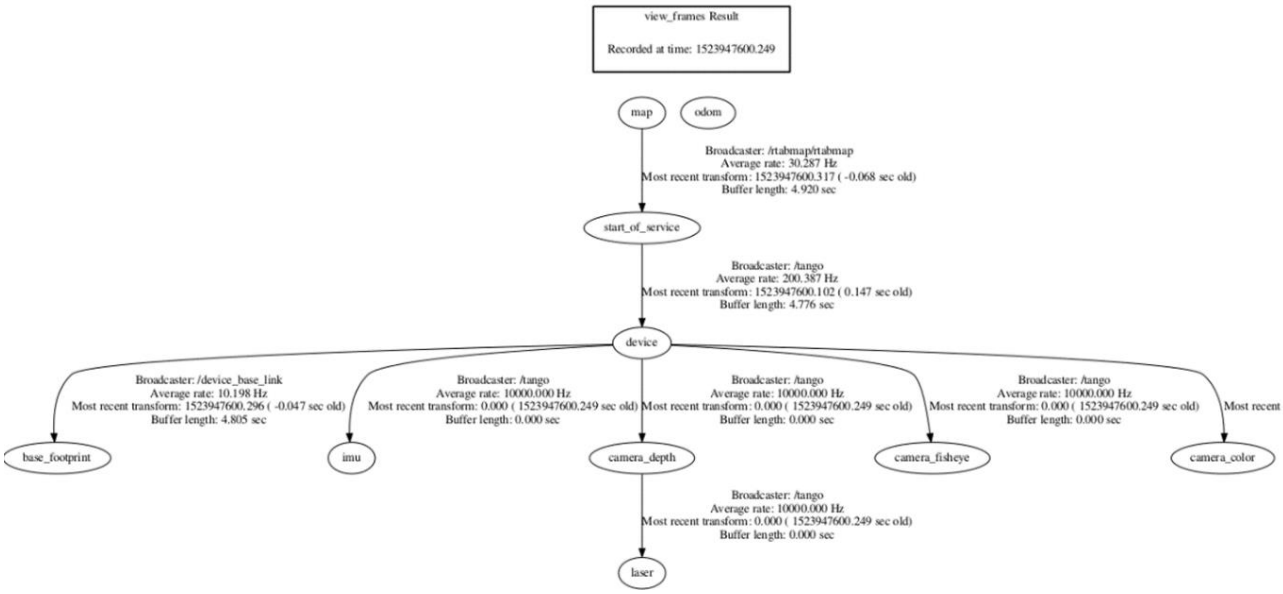
Τέλος , αυτό είναι κρίσιμο εργαλείο στον ακαδημαϊκό κόσμο, αφού επιλέγεται για ακαδημαϊκή έρευνα και την ανάπτυξη ρομποτικών συστημάτων για βιομηχανίες. Το ROS έχοντας στην διάθεσή του μεγάλη συλλογή εργαλείων είναι ένα πρωτεύον εργαλείο για την επίλυση προβλημάτων στον κόσμο της ρομποτικής.

(Ενότητα 2.4.β) ΕΙΣΑΓΩΓΗ ΣΤΑ TRANSFORMATIONS

Η αλληλεπίδραση των ρομποτικών οχημάτων με τον πραγματικό κόσμο απαιτεί απαραίτητη προϋπόθεση να γίνει περιγραφή θέσεως των αισθητήρων του ρομποτικού οχήματος μέσω συντεταγμένων, καθώς και των θέσεων των αντικειμένων που θα αλληλοεπιδράσει το ρομποτικό όχημα. Τα Transformations (TF) [10] είναι ένα σύστημα coordinates frame και ορίζει τις συντεταγμένες μέσω του λειτουργικού συστήματος ROS.

Για να κάνει το σύστημα TF λήψη των πληροφοριών, πρέπει να γνωρίζει τον τρόπο σύνδεσης των θέσεων των εξαρτημάτων του ρομπότ. Συνήθως, το κάθε ρομποτικό σύστημα έχει αρκετά πλαίσια συντεταγμένων, τρισδιάστατα, τα οποία μεταβάλλονται με την πάροδο του χρόνου. Το σύστημα αυτό επιβλέπει όλα τα πλαίσια καθ' όλη την διάρκεια του χρόνου και δίνει την δυνατότητα στον χρήστη να δει την θέση του ρομπότ στον χάρτη, τις σχέσεις των πλαισίων ή την θέση τους. Κάποια παραδείγματα πλαισίων είναι αυτό της βάσης, της λαβής, της κεφαλής αλλά και το παγκόσμιο πλαίσιο.

Παρακάτω θα γίνει οπτική απεικόνιση ενός γενικού σχεδίου μετασχηματισμών σε μορφή transformations tree (δέντρου):



(Εικόνα 12η:Απεικόνιση ενός δέντρου μετασχηματισμών)

Παρατηρώντας την παραπάνω απεικόνιση του δέντρου, είναι προφανές ότι στην βάση αυτού είναι το πλαίσιο του map (χάρτη) το οποίο έχει συνδεθεί με το start of service (εκκίνηση υπηρεσίας), η οποία καλείται να πραγματοποιήσει την ολοκλήρωση του ρομποτικού οχήματος. Στην συνέχεια, απεικονίζονται όλα τα πλαίσια που χρησιμοποιούνται από το ρομπότ (imu, camera color κτλ.) αλλά και τα αποτυπώματα αυτού (base footprint). Τέλος, στον camera depth αισθητήρα βλέπουμε το πλαίσιο laser. Εκτός από τα πλαίσια που απεικονίζονται στο δέντρο, μπορούμε να δούμε και τους κόμβους που στέλνουν πληροφορίες μετασχηματισμού, πόση είναι η συχνότητα αποστολής μηνυμάτων αλλά και την χρονική διάρκεια του κάθε πλαισίου.

(Ενότητα 2.4.γ) ΕΙΣΑΓΩΓΗ ΣΤΟ RQT

Το RQT (ROS Qt) [11] είναι πακέτο του ROS λογισμικού, το οποίο έχει την δυνατότητα να δώσει στον χρήστη ένα GUI γραφικό περιβάλλον, που βοηθάει στην εκτέλεση, την δημιουργία αλλά και τον έλεγχο διαφόρων ρομποτικών εφαρμογών. Το πακέτο αυτό έχει βάση το Qt framework, το οποίο είναι εργαλείο που ειδικεύεται στην ανάπτυξη λογισμικού με γραφική διεπαφή από τον χρήστη.

ΤΟ Πακέτο λογισμικού αυτό προσφέρει διάφορα επιπρόσθετα plugins που δημιουργούν τις προϋποθέσεις για τον έλεγχο, την ανάλυση αλλά και την οπτικοποίηση

δεδομένων που αναφέρονται στο ρομποτικό σύστημα. Παρακάτω αναφέρονται κάποιες από τις λειτουργίες του:

- Rqt graph: Η λειτουργία αυτή απεικονίζει γραφικά τα nodes (κόμβοι) και τα topics (θέματα) του ROS, δίνοντας στον χρήστη την δομή του ρομπότ ως οπτική απεικόνιση.

- Rqt console: Προκαλεί την εμφάνιση των log messages (μηνυμάτων καταγραφής) που παράγουν οι κόμβοι, δίνοντας στον χρήστη την δυνατότητα να παρακολουθεί τα γεγονότα και να αντιμετωπίσει απρόσμενα προβλήματα.

- Rqt plot: Η λειτουργία αυτή απεικονίζει και αναλύει τα δεδομένα που στέλνουν οι αισθητήρες και άλλοι μετρητές με την μορφή γραφήματος.

- Rqt bag: Το rqt_bag κάνει επιτρεπτή την ανάλυση των recordings (γραφημάτων) του ROS, ευρέως γνωστά και ως bags (τσάντες) που έχουν πληροφορίες οι οποίες έχουν συλλεχθεί από το ρομπότ, την επεξεργασία αλλά και την αναπαραγωγή τους.

Το RQT δίνει δηλαδή στον χρήστη ένα περιβάλλον φιλικό προς αυτόν, που μπορεί να χρησιμοποιηθεί για την διόρθωση ή την εύρεση σφαλμάτων σε ρομποτικές εφαρμογές αλλά και την ανάπτυξη αυτών, και δημιουργεί έναν οπτικό τρόπο επικοινωνίας με το λειτουργικό ROS.

(Ενότητα 2.4.δ) ΕΙΣΑΓΩΓΗ ΣΤΟ RVIZ

Το RVIZ [12] αποτελεί κύριο εργαλείο που έχει εξ ορισμού το ROS για να απεικονίζει και να ελέγχει το ρομπότ και τα δεδομένα που στέλνει αυτό. Αυτό το περιβάλλον είναι ικανό στην οπτικοποίηση των αντικειμένων , των δεδομένων που αποστέλλονται, τους αισθητήρες αλλά και την σχέση αλληλεπίδρασης μεταξύ τους και τον έλεγχο αυτής.

Οι κύριες λειτουργίες του RVIZ είναι οι εξής:

- Οπτικοποίηση αισθητήρων: Υπάρχει η δυνατότητα εμφάνισης δεδομένων από οποιοδήποτε αισθητήρα, όπως την κάμερα, λέιζερ, πυξίδες κτλ. Ο χρήστης έχει την δυνατότητα διαμόρφωσης των δεδομένων, που λαμβάνονται ως περιβαλλοντικές αναπαραστάσεις, ανάλογα με τις παροντικές του ανάγκες.

- Οπτικοποίηση χάρτη: Το πακέτο αυτό του ROS μπορεί να δημιουργήσει μια περιβαλλοντική αναπαράσταση η οποία να έχει εμπόδια και διάφορα άλλα

στοιχεία, η οποία κατασκευάζεται από διάφορες τεχνικές όπως π.χ. την τεχνική Simultaneous Localization and Mapping (SLAM).

- Οπτικοποίηση ρομποτικών μοντέλων: Γίνεται η εισαγωγή των αντικειμένων ή των ρομποτικών μοντέλων που αλληλοεπιδρούν μεταξύ τους, με σκοπό τον έλεγχο των θέσεων τους και την κίνηση των μοντέλων.

- Έλεγχος του ρομπότ: Με την βοήθεια του εργαλείου RVIZ, γίνεται δυνατή η έκδοση εντολών κίνησης για το ρομπότ και η παρακολούθηση αυτής. Επιπροσθέτως, υπάρχει η επιλογή αλλαγής της ταχύτητας του ρομποτικού συστήματος.

Τελικά, το RVIZ αποτελεί ένα από τα χρησιμότερα εργαλεία που έχει το ROS στην συλλογή του, το οποίο χρησιμοποιείται σε μεγάλο βαθμό από τη ρομποτική κοινότητα για την ανάπτυξη, τον έλεγχο αλλά και την επίδειξη ρομποτικών εφαρμογών.

(Υποκεφάλαιο 2.5) ROS ΚΑΙ BLUEROV2

(Ενότητα 2.5.α) ΣΥΝΔΕΣΗ ΜΕΤΑΞΥ ROS ΚΑΙ BLUEROV2

Για να επιτευχθεί η σύνδεση μεταξύ του ROS του BLUEROV2 χρειάστηκε το πακέτο "bluerov_ros_playground" [13]. Το πακέτο αυτό αποτελείται από ανοικτό κώδικα και έχει δημιουργηθεί για το συγκεκριμένο υποβρύχιο. Αυτό προσφέρει στο υποβρύχιο δυνατότητες ελέγχου, ανάπτυξης αλλά και την προσομοίωσή του στο περιβάλλον του ROS. Ο δημιουργός του πακέτου, ο Patrick Jose Pereira [14], το έφτιαξε και συνέβαλε στην ενσωμάτωση του υποβρυχίου με το ROS.

Με το πακέτο αυτό γίνεται ο έλεγχος της κίνησης του ρομπότ, όπως και η προβολή της μπροστινής κάμερας που έχει το υποβρύχιο από μόνο του αλλά και η λήψη πληροφοριών από τους πολλούς αισθητήρες που έχει το BLUEROV2. Επίσης, είναι δυνατή η υποστήριξη της χρήσεως του υποβρυχίου χωρίς την πραγματική του παρουσία, δηλαδή την υποστήριξη μίας προσομοίωσης.

(Ενότητα 2.5.β) ΕΠΕΞΗΓΗΣΗ ΤΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΤΟΥ BLUEROV_ROS_PLAYGROUND

Το bluerov_ros_playground αποτελεί χρήσιμο πακέτο όχι μόνο για το υποβρύχιο που αναφέρεται στην συγκεκριμένη πτυχιακή αλλά και για πολλά άλλα. Όμως, χρειάζεται προσοχή στην ορθή αξιοποίηση του πακέτου για να μην υπάρξει πιθανότητα βλάβης ή ατύχημα με αποτέλεσμα την καταστροφή του υποβρυχίου. Και αυτός είναι ο κύριος λόγος που προτείνεται στην αρχή η χρήση του πακέτου να γίνει πειραματικά σε προσομοιωτή για να αποφευχθούν τυχόν βλάβες ή δυσλειτουργίες στο πακέτο αλλά και την συνήθεια χρήσης του από τον χειριστή του ρομπότ.

Για την ορθή λειτουργία του πακέτου χρειάζεται:

- Οποιαδήποτε έκδοση του λειτουργικού περιβάλλοντος ROS
- Να επιτευχθεί η ορθή εγκατάσταση, από αξιόπιστη πηγή, του πακέτου
- Η σωστή χρήση του πακέτου MAVROS [15] για την σωστή ανταλλαγή πληροφοριών μεταξύ του ρομπότ και του ROS

- Ορθή εγκατάσταση του πακέτου freefloating_gazebo [16] ώστε να επιτευχθεί η σωστή σύνδεση της προσομοίωσης μεταξύ του υποβρυχίου και του ROS.

(Ενότητα 2.5.γ) ΕΠΕΞΗΓΗΣΗ ΤΗΣ ΧΡΗΣΗΣ ΤΟΥ BLUEROV_ROS_PLAYGROUND

Για την σωστή η λειτουργία του πακέτου, πρέπει να γίνει η σύνδεση του υποβρυχίου ρομπότ bluerov2 με τον ηλεκτρονικό υπολογιστή και να μην υπάρχει κάποιο τεχνικό πρόβλημα. Επιπροσθέτως, με την εντολή που θα αναγράφεται παρακάτω, ο χειριστής έχει την δυνατότητα να χρησιμοποιεί πλήρως το ρομπότ με την βοήθεια τηλεχειριστηρίου και να δει πληροφορίες για την κατάσταση του (κατάσταση μπαταρίας, έλεγχος διαρροής, κτλ.).

```
$ roslaunch bluerov_ros_playground bluerov_node.launch
```

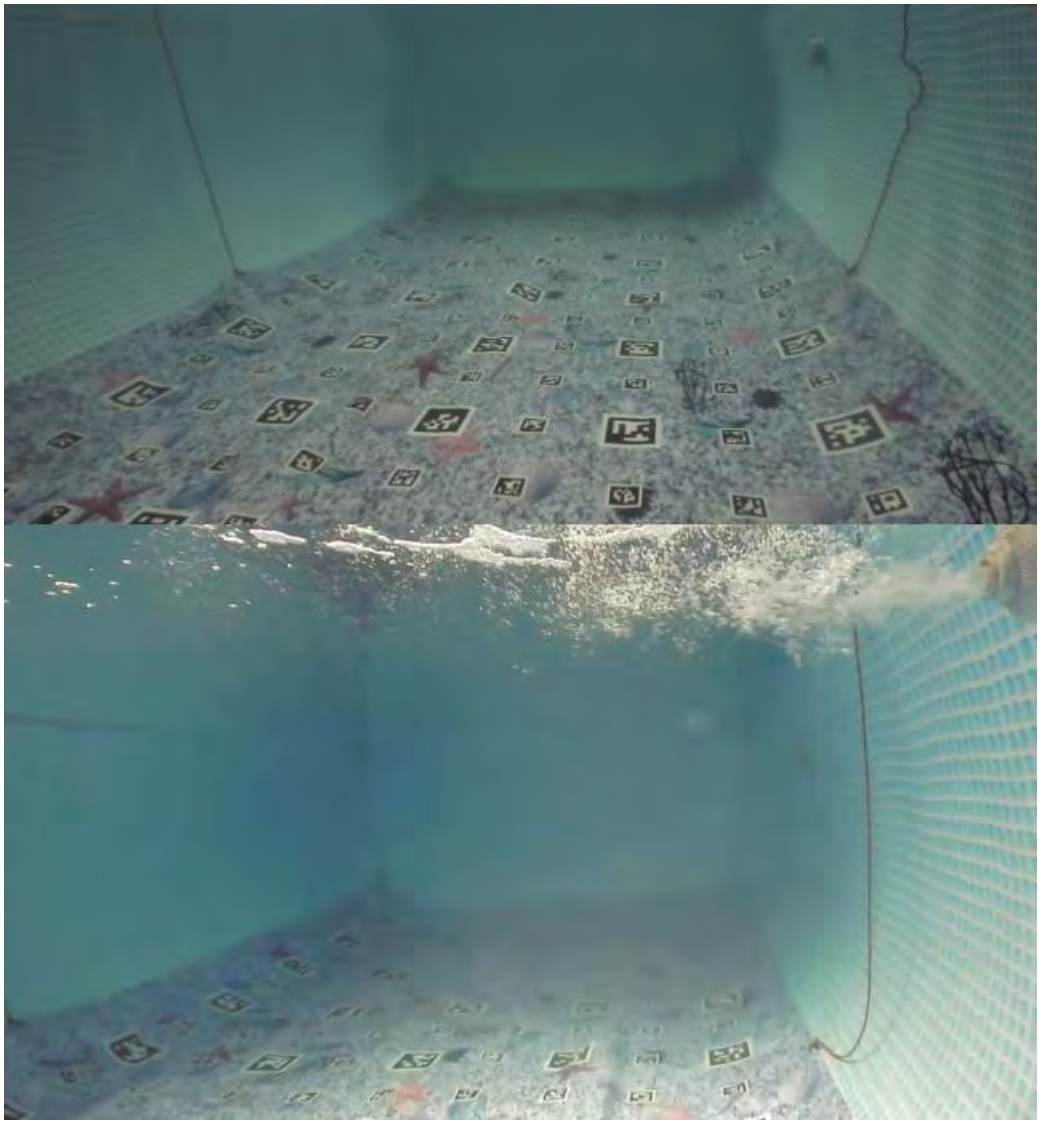
Παρακάτω, θα δούμε το αποτέλεσμα της εκτέλεσης της εντολής <<rostopic list>> , δηλαδή μία λίστα με όλες τις πληροφορίες:

```
/BlueRov2/arm - std_msgs/Bool
/BlueRov2/battery - sensor_msgs/BatteryState
/BlueRov2/camera/image_raw - sensor_msgs/Image
/BlueRov2/imu/data - sensor_msgs/Imu
/BlueRov2/mode/set - std_msgs/String
/BlueRov2/odometry - nav_msgs/Odometry
/BlueRov2/rc_channel1/set_pwm - std_msgs/UInt16
/BlueRov2/rc_channel2/set_pwm - std_msgs/UInt16
/BlueRov2/rc_channel3/set_pwm - std_msgs/UInt16
/BlueRov2/rc_channel4/set_pwm - std_msgs/UInt16
/BlueRov2/rc_channel5/set_pwm - std_msgs/UInt16
/BlueRov2/rc_channel6/set_pwm - std_msgs/UInt16
/BlueRov2/rc_channel7/set_pwm - std_msgs/UInt16
/BlueRov2/rc_channel8/set_pwm - std_msgs/UInt16
/BlueRov2/servo1/set_pwm - std_msgs/UInt16
/BlueRov2/servo2/set_pwm - std_msgs/UInt16
/BlueRov2/servo3/set_pwm - std_msgs/UInt16
/BlueRov2/servo4/set_pwm - std_msgs/UInt16
/BlueRov2/servo5/set_pwm - std_msgs/UInt16
/BlueRov2/servo6/set_pwm - std_msgs/UInt16
/BlueRov2/servo7/set_pwm - std_msgs/UInt16
/BlueRov2/servo8/set_pwm - std_msgs/UInt16
/BlueRov2/setpoint_velocity/cmd_vel - geometry_msgs/TwistStamped
/BlueRov2/state - std_msgs/String
/rosout - rosgraph_msgs/Log
/rosout_agg - rosgraph_msgs/Log
```

(Εικόνα 13η:Απεικόνιση του αποτελέσματος της εντολής rostopic list)

Και με την εντολή που αναγράφεται παρακάτω θα προβληθεί η μπροστινή κάμερα του υποβρυχίου:

```
$ rosrun image_view image_view image:=/BlueRov2/camera/image_raw
```



(Εικόνα 14η: Απεικόνιση της μπροστινής κάμερας του υποβρυχίου μέσα στην πισίνα)

ΚΕΦΑΛΑΙΟ 3 LOCALIZATION,MAPPING AND MAVROS

(Υποκεφάλαιο 3.1)VISUAL SLAM

Η Visual SLAM [17] είναι μία μέθοδος που αναφέρει την πραγματοποίηση χαρτογράφησης και χωροθέτησης με την χρήση είτε μίας είτε πολλών καμερών, η οποία

προτιμάται διότι συγκριτικά με άλλους αισθητήρες έχει πιο λίγο κόστος και δίνει μία πιο πλούσια παρουσίαση του περιβάλλοντος.

Η γνώση που αφορά τις αυτόνομες ρομποτικές εφαρμογές, του Visual SLAM, είναι πολύτιμη στην πτήση των drones ,στην αυτόνομη οδήγηση I.X αλλά και στα υποβρύχια ρομπότ και διάφορες άλλες εφαρμογές. Από αυτό συνδυάζεται η οπτική αντίληψη και ο υπολογισμός της θέσης του ρομπότ για να ανιχνεύονται οι κινήσεις αυτού και να υπολογίζεται η θέση του στον χάρτη που δημιουργείται . Η νοοτροπία που περνάει το Visual SLAM είναι ότι το ρομποτικό σύστημα κάνει χρήση μίας ή περισσοτέρων καμερών για να γίνει δυνατή η ανίχνευση των χαρακτηριστικών του περιβάλλοντος στο οποίο βρίσκεται, όπως γραμμές, γωνίες και άλλα σημεία ενδιαφέροντος. Με αυτά τα χαρακτηριστικά γίνεται η εκτίμηση της κίνησης αλλά και της θέσης του ρομπότ, καθώς συμβάλει και στο κτίσιμο ενός χάρτη του περιβάλλοντος που βρίσκεται το ρομποτικό σύστημα.

Η ακολουθία ξεκινά με την εξαγωγή χαρακτήρων από τις εικόνες, την εύρεση διαφορετικών χαρακτηριστικών των στιγμιότυπων των εικόνων μέσω της σύγκρισης αυτών, την εκτίμηση της θέσεως και της κίνησης του ρομποτικού συστήματος και τελικά την φόρτωση των ληφθέντων πληροφοριών σε έναν χάρτη. Η επανάληψη της διαδικασίας είναι συνεχής όσο το ρομπότ κινείται και είναι αυτόματη.

Υπάρχει ποικιλία ερευνών και μελετών για την μέθοδο αυτή, όπου σε αυτές αναγράφονται τεχνικές και γίνεται σύγκριση απόδοσης και ακρίβειας σε καθεμία από αυτές. Σε κάποιες από τις έρευνες, έγινε χρήση πολλαπλών καμερών για πολλαπλούς πράκτορες , για την δημιουργία του χάρτη με μεγαλύτερη ταχύτητα και την επιτάχυνση της λειτουργίας σε δυναμικά αλλά και σε στατικά περιβάλλοντα, αν και μέθοδος αυτή λειτουργεί σε πραγματικό χρόνο (real time) και απαιτεί μεγάλη υπολογιστική ισχύ. Άλλη μελέτη αναφέρει ότι με την χρήση της κάμερας και με την υποβοήθηση διαφόρων αισθητήρων, το σύστημα είναι ικανό να διαχειριστεί καταστάσεις που ένα απλό συμβατικό μοντέλο SLAM δεν είναι ικανό να το κάνει, με μειονέκτημα στον συγχρονισμό της κάμερας και των αισθητήρων.

(Υποκεφάλαιο 3.2) FIDUCIAL SLAM

(Ενότητα 3.2.α) ΕΠΕΞΗΓΗΣΗ ΤΩΝ FIDUCIAL MARKERS

Την σημερινή εποχή ποικίλες εφαρμογές κάνουν χρήση δεικτών κωδικών (Aruco Markers, κωδικοί QR κ.τ.λ.),με την δημοτικότητά τους να έχει εκτοξευθεί και την ανάγκη σύνδεσης φυσικών αντικειμένων με τα αντίστοιχα οπτικά ορόσημα. Με την διαδικασία μάθησης για την εφαρμογή ενός τέτοιου αλγορίθμου αποκτάται μεγαλύτερη κατανόηση των αλγορίθμων που βοηθούν στην ανίχνευσή τους, των παραμέτρων που συμβάλουν στην σχεδίασή τους αλλά και την προσφορά τους σε μία ρομποτική εφαρμογή.

Η κάθε εκτίμηση μίας θέσης του ρομποτικού συστήματος έχει βάση στα δεδομένα που επιστρέφουν οι οπτικοί αισθητήρες το οποίο αποτελεί βασικό χαρακτηριστικό σε πολλαπλές ρομποτικές εφαρμογές, παραδείγματος χάρει τον εντοπισμό, το SLAM, την πλοήγηση του ρομπότ και διάφορες άλλες. Η συγκεκριμένη διαδικασία βασίζεται στην εύρεση αντιστοιχιών μεταξύ χαρακτηριστικών σημείων στο περιβάλλον και την προβολή τους σε μία εικόνα δύο διαστάσεων(μία κάμερα).Τα συνθετικά Markers είναι χρήσιμα για την εξαγωγή κατάλληλων σημείων χαρακτηριστικών [21].

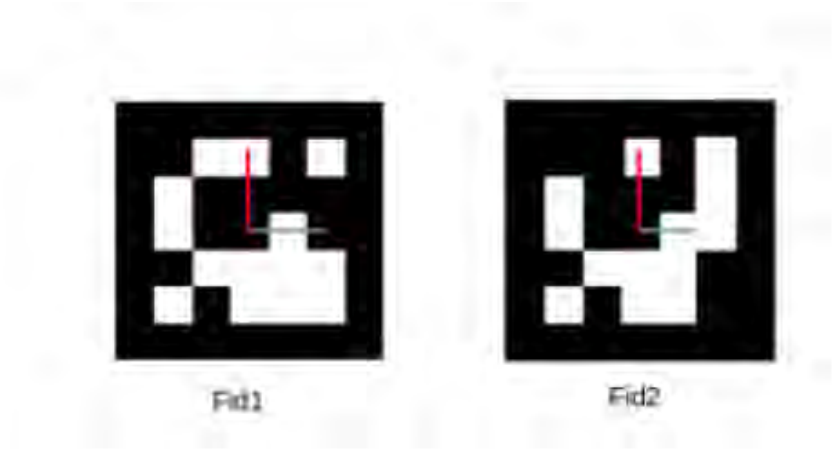
Υπάρχουν προκαθορισμένες αναλογίες και προκαθορισμένο μέγεθος του κάθε Fiducial Marker, που επιτρέπει την εκτίμηση της απόστασης από την κάμερα έως το marker που βλέπει. Μία από τις δημοφιλέστερες προσεγγίσεις είναι οι markers δυαδικών δεικτών τετραγώνου σχήματος που απεικονίζονται παρακάτω.

(Εικόνα 15η: Απεικόνιση μίας βιβλιοθήκης οπτικών οροσήμων)

Η τοποθέτηση των Fiducial Markers γίνεται σε μία σταθερή επιφάνεια. Μετά, κατά την εκκίνηση μίας διαδικασίας εντοπισμού και κίνησης το μη επανδρωμένο θα πρέπει να βλέπει συνέχεια κάποιον δείκτη για να γίνει η ανάλογη κίνηση δηλαδή ο marker θα πρέπει να βρίσκεται καθ' όλη την διάρκεια της χαρτογράφησης στο οπτικό πεδίο της κάμερας. Αυτό μπορεί να γίνει πολύ απλά, με την χρήση μίας ομάδας Markers. Με αυτόν τον τρόπο, όσο μεγαλύτερο είναι το πλήθος των δεικτών, τόσο πιο εύκολο είναι να πραγματοποιηθεί η χαρτογράφησή τους. Αντιστοίχως, όσο πιο μεγάλο το πλήθος τόσο μεγαλύτερος είναι ο χρόνος που θα πραγματοποιηθεί μία χαρτογράφηση.

(Ενότητα 3.2.β) ΠΑΚΕΤΟ FIDUCIALS ΣΤΟ ROS

Στο συγκεκριμένο πείραμα έγινε χρήση του ROS πακέτου FIDUCIALS [18]. Το πακέτο αυτό προσφέρει την δυνατότητα σε ένα ρομποτικό σύστημα να προσδιορίζει την θέση του και τον προσανατολισμό του βλέποντας τα markers που έχουν τοποθετηθεί στο περιβάλλον του. Η διαδικασία είναι απλή, πρώτα γίνεται ο καθορισμός της θέσης του δείκτη, μετά γίνεται η δημιουργία του χάρτη (μορφής αρχείου 6 DOF θέσεων) βλέποντας τα ζεύγη των δεικτών και δίνοντας τη τοποθεσία μεταξύ τους.



(Εικόνα 16η: Απεικόνιση marker θέσεως και προσανατολισμού)

(Ενότητα 3.2.γ) ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ ΤΟ ΠΑΚΕΤΟ FIDUCIALS

Η Ubiquity Robotics έχει εφεύρει ένα σύστημα εντοπισμού που κάνει χρήση έναν αριθμό δεικτών γνωστού μεγέθους για να προσδιορίσει την θέση του ρομπότ. Ο κόμβος που κάνει την ανίχνευση των markers ονομάζεται aruco detect [19]. Για κάθε δείκτη που φαίνεται στην εικόνα, παράγεται σύνολο κορυφών συντεταγμένων εικόνας. Επειδή οι παράμετροι της κάμερας και τα μεγέθη των δεικτών είναι γνωστά, είναι δυνατή η εκτίμηση της θέσης των δεικτών σε σχέση με αυτή της κάμερας.

Κάθε κορυφή εικόνας έχει συντεταγμένες (x, y) και γίνεται αντιστοιχία σε μία ακτίνα της κάμερας. Ο κώδικας εκτίμησης της θέσης δίνει την λύση σε ένα σύνολο γραμμικών εξισώσεων που προσδιορίζουν τις συντεταγμένες του κόσμου (x, y, z) για κάθε κορυφή. Αυτό μας δίνει τον μετασχηματισμό από το σύστημα συντεταγμένων του πακέτου στο σύστημα συντεταγμένων της κάμερας T_fid_cam. Αυτό δείχνει την θέση του δείκτη στο σύστημα συντεταγμένων της κάμερας. Παρακάτω απεικονίζονται δύο fiducials, fid1 και fid2. Αν το

πρώτο έχει γνωστή θέση στον κόσμο, την T_map_fid1, και είναι γνωστός ο μετασχηματισμός του δείκτη σε κάμερα για παραπάνω από έναν δείκτη, είναι δυνατό να υπολογιστεί η στάση του Fid2.

$$T\_map\_fid2 = T\_map\_fid1 * T\_cam\_fid2 * T\_fid1\_cam \quad [20]$$

Με τον παραπάνω τρόπο, η δημιουργία του χάρτη επιτυγχάνεται καθώς γίνεται παρατήρηση ζευγών των markers (δεικτών). Μεγάλο άθροισμα παρατηρήσεων συνδυάζεται με στάθμιση, για να γίνει η παραγωγή μίας εκτίμησης της θέσεως όλων των δεικτών, αλλά και του ρομπότ.

(Ενότητα 3.2.δ) ΠΩΣ ΓΙΝΕΤΑΙ Η ΧΡΗΣΗ ΤΟΥ ΠΑΚΕΤΟΥ FIDUCIALS

Για να γίνει δυνατή η χρήση του πακέτου FIDUCIALS στο ROS θα πρέπει να υπάρχουν οι παρακάτω προϋποθέσεις:

- Περιβάλλον ROS το οποίο θα είναι λειτουργικό
- Ορθή εγκατάσταση των απαραίτητων πακέτων
- Χρήση κάμερας ικανής να δει τους δείκτες
- Γνωστός στατικός μετασχηματισμός (θέση και προσανατολισμός της κάμερας σε σχέση με το ρομποτικό σύστημα)

Θα γίνει χρήση των παρακάτω εντολών:

Για την δημιουργία των markers:

```
rosrun aruco_detect create_markers.py 100 112 fiducials.pdf
```

Όταν πραγματοποιηθεί η δημιουργία τους , μπορούν να εκτυπωθούν και να τοποθετηθούν στο περιβάλλον χωρίς να υπάρχει κάποιο συγκεκριμένο μοτίβο, αλλά θα πρέπει να υπάρξει μία πυκνότητα στην οποία η κάμερα θα μπορεί να βλέπει παραπάνω από δύο markers για να γίνει η κατασκευή του χάρτη.

Κατ' επέκταση, θα πρέπει να γίνεται σωστή λειτουργία δύο nodes, του aruco detect που είναι υπεύθυνο για τον εντοπισμό των markers, και του fiducial slam το οποίο κάνει την εκτίμηση της θέσης και την δημιουργία του χάρτη. Ο χάρτης αποθηκεύεται αυτόματα σε μορφή αρχείου (Text Document).

Τα αρχεία των δύο κόμβων είναι έτοιμα και εκτελούνται ως εξής:

```
roslaunch aruco_detect aruco_detect.launch
roslaunch fiducial_slam fiducial_slam.launch
```

Υπάρχει και έτοιμο αρχείο που κάνει την οπτικοποίηση του χάρτη στο RVIZ. Με την παρακάτω εντολή θα εμφανιστεί ο χάρτης κατά την διάρκεια της δημιουργίας του καθώς και η προβολή της κάμερα.

```
roslaunch fiducial_slam fiducial_rviz.launch
```

(Ενότητα 3.2.ε) ΚΩΔΙΚΑΣ ΤΟΥ ΠΑΚΕΤΟΥ FIDUCIALS

Για να πραγματοποιηθεί η χαρτογράφηση της πισίνας στην εργασία, έγιναν μερικές αλλαγές στις προκαθορισμένες παραμέτρους που έχει το πακέτο FIDUCIALS, διότι τα χαρακτηριστικά σύνδεσης του ρομπότ, της κάμερας αλλά και ο τύπος και το μέγεθος των δεικτών έχουν καθοριστικό ρόλο στην σωστή λειτουργία του αλγορίθμου.

Παρακάτω γίνεται η απεικόνιση του κόμβου aruco_detect.launch που πραγματοποιεί την αναγνώριση των δεικτών από την κάμερα:

```
<!-- Run the aruco_detect node -->
<launch>
  <!-- namespace for camera input -->
  <arg name="camera" default="usb_cam"/>
  <arg name="image" default="image_raw"/>
  <arg name="transport" default="compressed"/>
  <arg name="fiducial_len" default="0.1"/>
  <arg name="dictionary" default="3"/>
  <arg name="do_pose_estimation" default="true"/>
  <!-- If vis_msgs set to true, pose estimation will be published with ROS standard vision_msgs -->
  <arg name="vis_msgs" default="false"/>
  <arg name="ignore_fiducials" default="" />
  <arg name="fiducial_len_override" default="" />
  <arg name="verbose" default="false"/>

  <node pkg="aruco_detect" name="aruco_detect"
    type="aruco_detect" output="screen" respawn="false">
    <param name="image_transport" value="$(arg transport)"/>
    <param name="publish_images" value="true" />
    <param name="fiducial_len" value="$(arg fiducial_len)"/>
    <param name="dictionary" value="$(arg dictionary)"/>
    <param name="do_pose_estimation" value="$(arg do_pose_estimation)"/>
    <param name="vis_msgs" value="$(arg vis_msgs)"/>
    <param name="ignore_fiducials" value="$(arg ignore_fiducials)"/>
    <param name="fiducial_len_override" value="$(arg fiducial_len_override)"/>
    <param name="verbose" value="$(arg verbose)"/>
    <remap from="camera/compressed"
        to="$(arg camera)/$(arg image)/$(arg transport)"/>
    <remap from="camera_info" to="$(arg camera)/camera_info"/>
  </node>
</launch>
```

(Εικόνα 17η: Απεικόνιση αρχείου aruco_detect.launch)

Όπως φαίνεται παραπάνω, στο αρχείο εκκίνησης του κόμβου πρέπει να οριστούν οι παρακάτω παράμετροι:

- Οι πληροφορίες της κάμερας (**<arg name=”camera” default=”/usb_cam”/>**)
- Η εικόνα (**<arg name=”image” default=”/image_raw”/>**)
- Το μέγεθος των δεικτών (**<arg name=”fiducial_len” default=”0.1”/>**)
- Η σωστή βιβλιοθήκη των δεικτών, που στην συγκεκριμένη περίπτωση στον βυθό της πισίνας βρίσκονται δείκτες που ανήκουν στην βιβλιοθήκη 3 (**<arg name=” dictionary ” default=”3”/>**)

Στη συνέχεια, θα γίνει απεικόνιση του αρχείου εκκίνησης fiducial_slam.launch, το οποίο κάνει την δημιουργία του χάρτη και την εκτίμηση της θέσης του ρομπότ με την βοήθεια της θέσης των δεικτών:

Open ▾ fiducial_slam.launch ~/catkin_ws/src/fiducials/fiducial_slam/launch Save

```
<!--
   Run the fiducial_slam node
-->
<launch>
  <arg name="camera" default="/head_camera"/>
  <arg name="map_frame" default="map"/>
  <arg name="odom_frame" default="odom"/>
  <arg name="base_frame" default="base_link"/>
  <arg name="publish_tf" default="true"/>
  <arg name="tf_publish_interval" default="1.0"/>
  <arg name="future_date_transforms" default="0.1"/>
  <arg name="publish_6dof_pose" default="false"/>
  <arg name="pose_publish_rate" default="20"/>
  <arg name="systematic_error" default="0.01"/>
  <arg name="covariance_diagonal" default=""/>

  <node type="fiducial_slam" pkg="fiducial_slam" output="screen"
    name="fiducial_slam">
    <param name="map_file" value="$(env HOME)/.ros/slam/map.txt" />
    <param name="map_frame" value="$(arg map_frame)" />
    <param name="read_only_map" value="true" />
    <param name="odom_frame" value="$(arg odom_frame)" />
    <param name="base_frame" value="$(arg base_frame)" />
    <param name="publish_tf" value="$(arg publish_tf)" />
    <param name="tf_publish_interval" value="$(arg tf_publish_interval)" />
    <param name="future_date_transforms" value="$(arg future_date_transforms)" />
    <param name="publish_6dof_pose" value="$(arg publish_6dof_pose)" />
    <param name="pose_publish_rate" value="$(arg pose_publish_rate)" />
    <param name="sum_error_in_quadrature" value="true"/>
    <rosparam param="covariance_diagonal" subst_value="True">$(arg covariance_diagonal)</rosparam>
    <remap from="/camera_info" to="$(arg camera)/camera_info"/>

  </node>
</launch>
```

HTML ▾ Tab Width: 8 ▾ Ln 21, Col 4 ▾ INS

(Εικόνα 18η: Απεικόνιση αρχείου fiducial_slam.launch)

Και οι αλλαγές που χρειάστηκαν:

- Η κάμερα (**<arg name=”camera” default=”head_camera”/>**)
- Η αποθήκευση του χάρτη (**<arg name=”map_frame” default=”map”/>**)
- Η οδομετρία (**<arg name=”odom_frame” default=”odom”/>**)
- Η βάση του ρομπότ (**<arg name=”base_frame” default=”base_link”/>**)

Και όταν φτιαχτεί ο χάρτης είναι θεμιτό να γράφεται η εντολή για ανάγνωση του χάρτη και όχι ανανέωσής του.

<param name=” read_only_map” value=”true”/>

Τέλος, το αρχείο εκκίνησης του Fiducial_rviz.launch ,το οποίο κάνει την οπτικοποίηση του χάρτη κατά τη διάρκεια κατασκευής του.

```
<?xml version="1.0"?>

<launch>
  <node pkg="rviz" type="rviz" name="$(anon rviz)" args="-d $(find fiducial_slam)/fiducials.rviz"/>
  <arg name="model" default="$(find bluerov_ros_playground)/model/BlueRov2.urdf.xacro"/>
  <arg name="rvizconfig" default="$(find bluerov_ros_playground)/model/model.rviz" />

  <node name="rviz" pkg="rviz" type="rviz" required="true"  args="-d $(arg rvizconfig)"/>

  <param name="robot_description" command="$(find xacro)/xacro.py --inorder $(arg model)"/>

  <node pkg="robot_state_publisher" type="robot_state_publisher"  name="robot_state_publisher">
    <param name="publish_frequency" type="double" value="10.0" />
  </node>

</launch>
```

(Εικόνα 19η: Απεικόνιση αρχείου fiducial_rviz.launch)

Στο παραπάνω αρχείο έχει γίνει η πρόσθεση της επιλογής της οπτικοποίησης του ρομπότ από το RVIZ για την ευκολότερη κατανόηση του προσανατολισμού του υποβρυχίου. Για τον σκοπό αυτό χρησιμοποιήθηκε ένα έτοιμο αρχείο 3D του υποβρυχίου που μας δίνεται από το πακέτο bluerov_ros_playground.

Αυτό γίνεται με τις εξής εντολές:

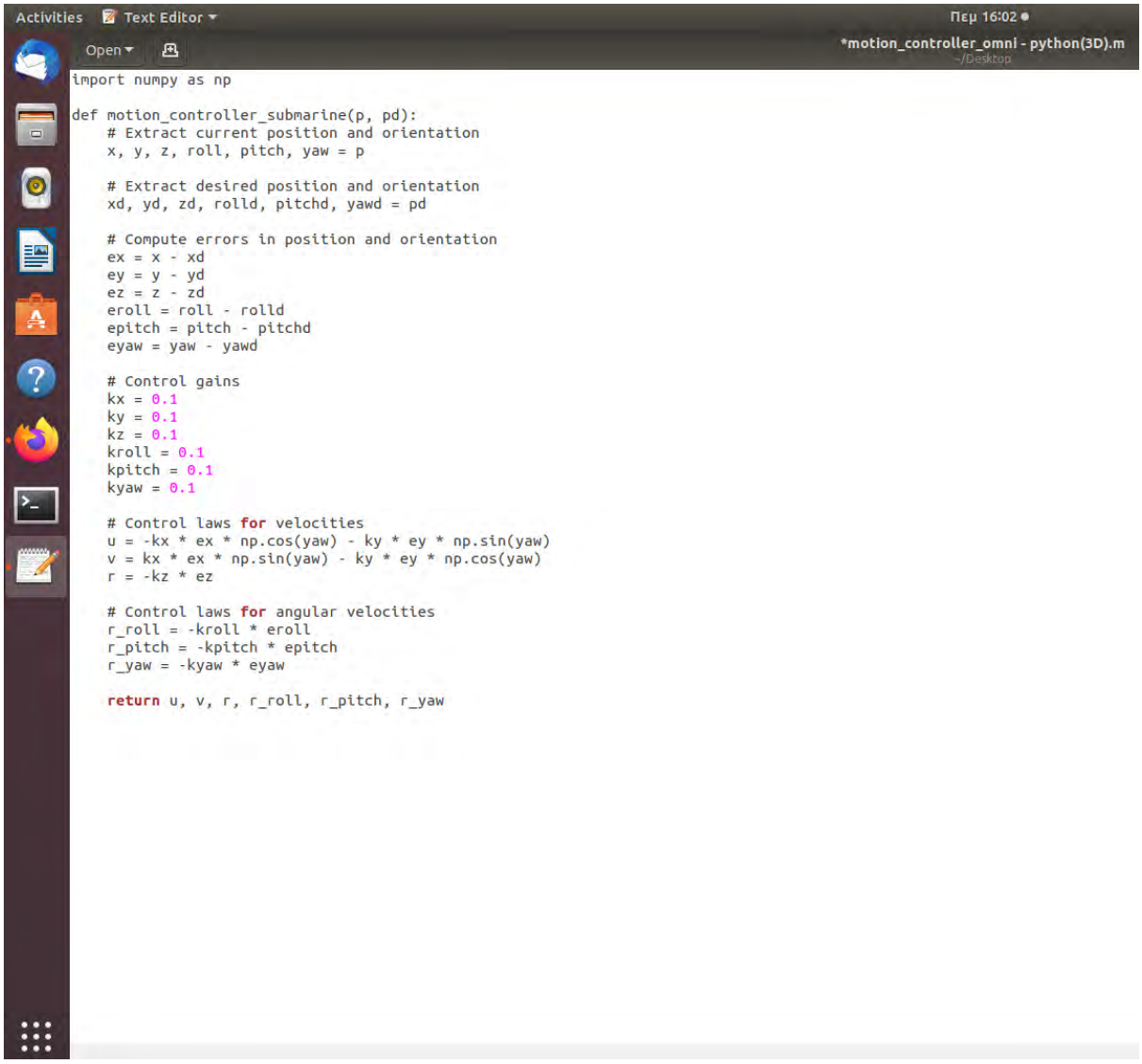
<arg name=”model” default=$(find bluerov2_description)/model/bluerov2_default.xacro”/>

<arg name=”rvizconfig” default=$(find bluerov_ros_playground)/model/model.rviz”/>

ΚΕΦΑΛΑΙΟ 4 ΑΛΓΟΡΙΘΜΟΣ ΕΛΕΓΧΟΥ ΚΙΝΗΣΗΣ

Ο ελεγκτής κίνησης αυτός έχει γραφθεί για να γίνει η χρήση του στην γλώσσα προγραμματισμού python. Αρχικά, ο αλγόριθμος αυτός υπολογίζει τις απαραίτητες ταχύτητες που πρέπει να έχει το ρομποτικό όχημα για να μειώσει το σφάλμα στη θέση και τον προσανατολισμό του σε σχέση με μία επιθυμητή θέση και έναν επιθυμητό προσανατολισμό.

Πρόκειται για μία απλή ,αλλά αποτελεσματική μέθοδο ελέγχου, που βασίζεται στις διαφορές (σφάλματα) της τρέχουσας και της επιθυμητής κατάστασης κάνοντας χρήση αναλογικούς νόμους ελέγχου (proportional control laws).



Ο παραπάνω αλγόριθμος υλοποιεί έναν ελεγκτή κίνησης για το υποβρύχιο. Αρχικά, εισάγεται η βιβλιοθήκη numpy ως np, η οποία χρησιμοποιείται για αριθμητικούς υπολογισμούς, ιδιαίτερα με πίνακες. Μετά ορίζεται η συνάρτηση motion_controller_submarine, η οποία δέχεται δύο παραμέτρους p (η τρέχουσα θέση και προσανατολισμός του υποβρυχίου) και pd (η επιθυμητή θέση και προσανατολισμός).

Στην συνέχεια, εξάγονται οι τιμές της τρέχουσας θέσης (x, y, z) και προσανατολισμού (roll, pitch, yaw) από το διάνυσμα p και οι τιμές της επιθυμητής θέσης (xd, yd, zd) και προσανατολισμού (rolld, pitchd, yawd) από το διάνυσμα pd. Υπολογίζονται τα σφάλματα (διαφορές) μεταξύ της τρέχουσας και της επιθυμητής θέσης (ex, ey, ez) και του προσανατολισμού (eroll, epitch, eyaw).

Μετέπειτα, ορίζονται οι συντελεστές ενίσχυσης (control gains) για τον έλεγχο της κίνησης και του προσανατολισμού του υποβρυχίου. Οι τιμές αυτές καθορίζουν την "ένταση" της αντίδρασης του συστήματος στα σφάλματα που υπολογίστηκαν προηγουμένως. Υπολογίζονται οι ταχύτητες (u, v, w) του υποβρυχίου στις τρεις κατευθύνσεις. Οι ταχύτητες αυτές εξαρτώνται από τα σφάλματα θέσης και τον προσανατολισμό (μέσω των συναρτήσεων ημιτόνου και συνημίτονου) και υπολογίζονται οι γωνιακές ταχύτητες (r_roll, r_pitch, r_yaw) για τον έλεγχο του προσανατολισμού του υποβρυχίου, βασιζόμενες στα σφάλματα προσανατολισμού.

Τέλος, η συνάρτηση επιστρέφει τις υπολογισμένες ταχύτητες (u, v, w) και γωνιακές ταχύτητες (r_roll, r_pitch, r_yaw), οι οποίες χρησιμοποιούνται για να κατευθύνουν το υποβρύχιο προς την επιθυμητή θέση και προσανατολισμό.

ΚΕΦΑΛΑΙΟ 5 ΣΥΜΠΕΡΑΣΜΑΤΑ

Σκοπός της πτυχιακής εργασίας είναι η πειραματική υλοποίηση σχήματος ελέγχου κίνησης με οπτική ανατροφοδότηση για υποβρύχιο ρομποτικό όχημα. Δηλαδή, με την χρήση ενός αλγορίθμου ελέγχου κίνησης και με την βοήθεια οπτικών μέσων (στην προκειμένη περίπτωση της κάμερας), που δημιουργεί τον χάρτη των δεικτών που βρίσκονται στον πυθμένα της πισίνας του εργαστηρίου ρομποτικής του τμήματος Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Θεσσαλίας , να είναι δυνατός ο έλεγχος της κίνηση του υποβρυχίου στέλνοντάς το σε συγκεκριμένο σημείο του χάρτη αυτού.

Επιπροσθέτως , υπάρχουν κάποια αξιοσημείωτα συμπεράσματα που θα διευκολύνουν μελλοντικά είτε την χρήση κάποιου πακέτο που χρησιμοποιήθηκε στην εργασία αυτή , είτε στην συνέχεια του πειράματος σε ποιο εξιδεικευμένο στάδιο. Στην αρχή της χαρτογράφησης το υποβρύχιο πρέπει να κινείται αργά, αλλά όσο αυξάνεται ο χάρτης οροσήμων τόσο η χαρτογράφηση γινόταν ταχύτερα και έδινε στο ROV μεγαλύτερη ελευθερία κίνησης. Και αυτό συμβαίνει διότι ο αλγόριθμος Fiducial Slam είναι πολύ πιο ακριβής ως προς τη χαρτογράφηση και τη χωροθέτηση του υποβρυχίου, κι αυτό ευθύνεται ως προς την αυξημένη βεβαιότητα που λαμβάνει ο αλγόριθμος από την ύπαρξη των παραπάνω οπτικών οροσήμων.

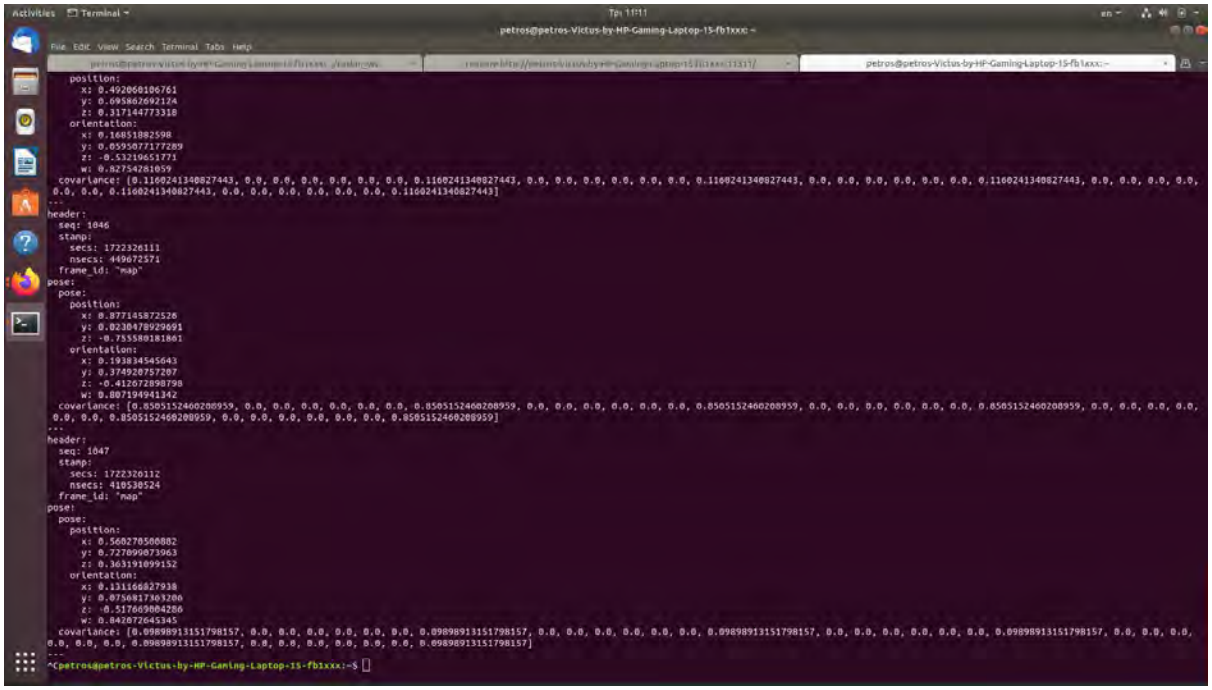
Επίσης , η κάμερα που έχει τοποθετηθεί στο υποβρύχιο δεν είναι τόσο σταθερή, λόγω της κίνησης αυτού, με αποτέλεσμα κάποιες φορές η χαρτογράφηση να είναι ανακριβής, αλλά με μικρή πιθανότητα λάθους προσανατολισμού των δεικτών. Για να γίνει δυνατή η επίλυση του προβλήματος αυτού, χρειάστηκε το υποβρύχιο όχημα να περάσει πολλές φορές από τα ορόσημα αυτά , ώστε να ανιχνεύσει τα γειτονικά τους και έτσι να διορθωθεί το λάθος του προσανατολισμού.

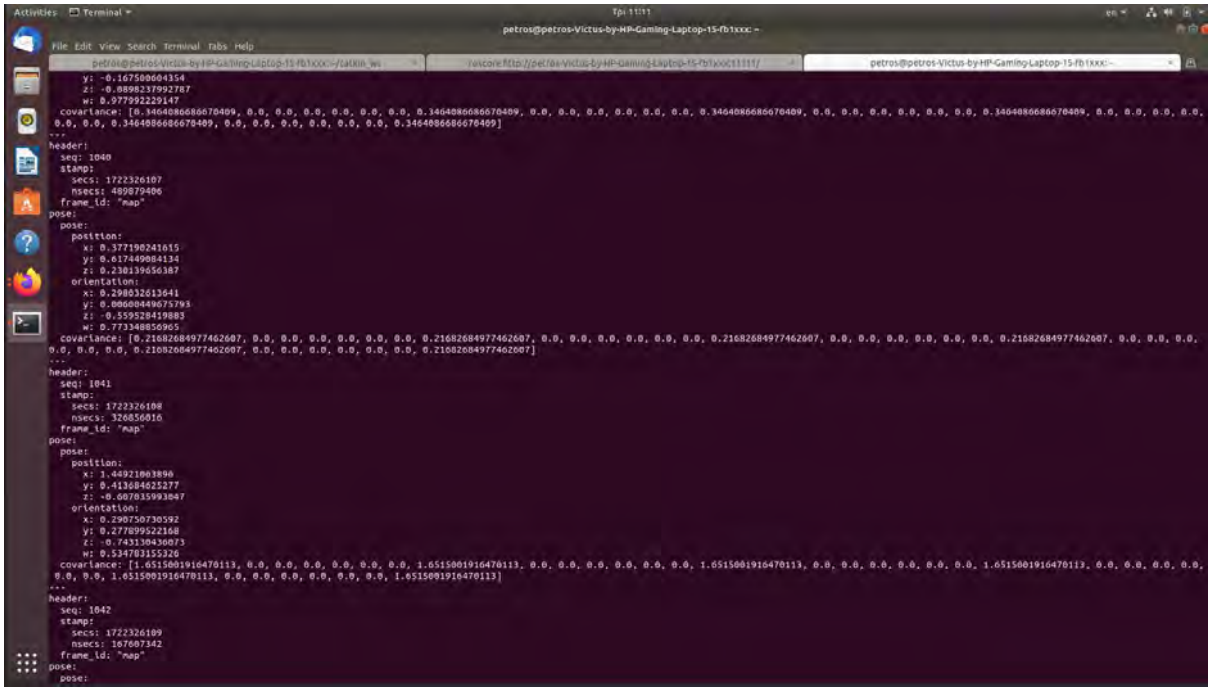
Τέλος στην σύνδεση του πακέτου Mavros με το ρομπότ υπάρχει έλλειψη συγκεκριμένων αισθητήρων ώστε να επιτραπεί από αυτό η χρήση μερικών modes κίνησης του υποβρυχίου. Με την εύρεση τρόπων προσπέρασης των παραπάνω δυσκολιών, που αναγράφονται παραπάνω , διεξήχθη πείραμα στο εργαστήριο Ρομποτικής και Αυτομάτου Ελέγχου, του Πανεπιστημίου Θεσσαλίας, διαβάζοντας των χάρτη που μας δημιουργεί ο αλγόριθμος Fiducial Slam με την βοήθεια της κάμερας και του πακέτου Mavros και με την χρήση του παρακάτω κώδικα είναι δυνατή η υλοποίηση σχήματος ελέγχου κίνησης με οπτική ανατροφοδότηση για το υποβρύχιο ρομποτικό όχημα.

Παρακάτω υπάρχει σε μορφή εικόνας ο κώδικας που χρησιμοποιήθηκε σε διεξαγωγή του πειράματος καθώς και εικόνες που δείχνουν την μεταβολή της θέσης του υποβρυχίου ρομπότ μέσα στον χαρτογραφημένο πυθμένα της πισίνας.

```python
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist, PoseStamped
import numpy as np
import tf

def motion_controller_submarine(p, pd):
    # Extract current position and orientation
    x, y, z, roll, pitch, yaw = p

    # Extract desired position and orientation
    xd, yd, zd, rolld, pitchd, yawd = pd

    # Compute errors in position and orientation
    ex = x - xd
    ey = y - yd
    ez = z - zd
    eroll = roll - rolld
    epitch = pitch - pitchd
    eyaw = yaw - yawd

    # Control gains
    kx = 0.1
    ky = 0.1
    kz = 0.1
    kroll = 0.1
    kpitch = 0.1
    kyaw = 0.1

    # Control laws for velocities
    u = -kx * ex * np.cos(yaw) - ky * ey * np.sin(yaw)
    v = kx * ex * np.sin(yaw) - ky * ey * np.cos(yaw)
    r = -kz * ez

    # Control laws for angular velocities
    r_roll = -kroll * eroll
    r_pitch = -kpitch * epitch
    r_yaw = -kyaw * eyaw

    return u, v, r, r_roll, r_pitch, r_yaw

class BlueROVController:
    def __init__(self):
        rospy.init_node('bluerov_control_node', anonymous=True)

        self.vel_pub = rospy.Publisher('/mavros/setpoint_velocity/cmd_vel_unstamped', Twist, queue_size=10)
        self.pose_sub = rospy.Subscriber('/fiducial_pose', PoseStamped, self.pose_callback)

        self.current_pose = None
         self.waypoints = [(0.5, 0.1, 0, 0, 0 ,0), (-0.6, 0.3, 0, 0, 0, 0), (-0.7, 0.1, 0, 0, 0 ,0), (1, -0.1, 0, 0, 0, 0)]
        self.current_waypoint_index = 0
        self.rate = rospy.Rate(10)

    def pose_callback(self, data):
        self.current_pose = data
```

```
        self.current_pose = None
         self.waypoints = [(0.5, 0.1, 0, 0, 0 ,0), (-0.6, 0.3, 0, 0, 0, 0), (-0.7, 0.1, 0, 0, 0 ,0), (1, -0.1, 0, 0, 0, 0)]
        self.current_waypoint_index = 0
        self.rate = rospy.Rate(10)

    def pose_callback(self, data):
        self.current_pose = data

    def move_to_waypoints(self):
        while not rospy.is_shutdown():
            if self.current_pose is None:
                continue

            if self.current_waypoint_index >= len(self.waypoints):
                break

            current_position = self.current_pose.pose.position
            current_orientation = self.current_pose.pose.orientation
            roll, pitch, yaw = self.get_rpy_from_quaternion(current_orientation)

            current_p = (current_position.x, current_position.y, current_position.z, roll, pitch, yaw)
            desired_p = self.waypoints[self.current_waypoint_index]

            u, v, r, r_roll, r_pitch, r_yaw = motion_controller_submarine(current_p, desired_p)

            twist = Twist()
            twist.linear.x = u
            twist.linear.y = v
            twist.linear.z = r
            twist.angular.x = r_roll
            twist.angular.y = r_pitch
            twist.angular.z = r_yaw

            self.vel_pub.publish(twist)

            distance_to_waypoint = np.linalg.norm(np.array(desired_p[:3]) - np.array(current_p[:3]))
            if distance_to_waypoint < 0.1:  # Tolerance to consider waypoint reached
                self.current_waypoint_index += 1

            self.rate.sleep()

        rospy.loginfo("All waypoints reached. Shutting down.")

    @staticmethod
    def get_rpy_from_quaternion(quat):
        quaternion = (quat.x, quat.y, quat.z, quat.r)
        euler = tf.transformations.euler_from_quaternion(quaternion)
        return euler  # Returns roll, pitch, yaw

if __name__ == '__main__':
    try:
        controller = BlueROVController()
        controller.move_to_waypoints()
    except rospy.ROSInterruptException:
        pass
```





ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Ρομποτική
https://el.wikipedia.org/wiki/%CE%A1%CE%BF%CE%BC%CF%80%CE%BF%CF%84%CE%B9%CE%BA%CE%AE
https://bigblue.academy/gr/ti-einai-i-rompotiki
https://builtin.com/robotics

[2] Υποβρύχια Ρομποτικά Οχήματα
https://en.wikipedia.org/wiki/Autonomous_underwater_vehicle
https://www.sciencedirect.com/topics/earth-and-planetary-sciences/autonomous-underwater-vehicle

[3] Χρήση και εφαρμογές των ROVs
https://www.naftikachronika.gr/2019/02/27/drones-kai-rov-stin-ypiresia-tis-naftilias/
https://reachrobotics.com/blog/what-is-an-underwater-rov/

[4] Έρευνες και άρθρα σχετικά με το μέλλον των ROV
https://el.marinetechnologynews.com/news/%CF%85%CF%80%CE%BF%CE%B8%CE%B1%CE%BB%CE%AC%CF%83%CF%83%CE%B9%CE%BF%CF%82-%CF%84%CE%BF%CE%BC%CE%AD%CE%B1%CF%82-%CE%BC%CE%AD%CE%BB%CE%BB%CE%BF%CE%BD-%CF%84%CF%89%CE%BD-%CE%B5%CF%80%CE%B1%CE%BD%CE%B4%CF%81%CF%89%CE%BC%CE%AD%CE%BD%CF%89%CE%BD-278607

[5] Πληροφορίες για το BlueRov2
https://bluerobotics.com/store/rov/bluerov2/
https://en.wikipedia.org/wiki/Robotics
https://users.sch.gr/jenyk/index.php/robotics

[6] Πληροφορίες για την Blue Robotics
Blue Robotics - Underwater ROVs, USVs, Thrusters and Sonars!

[7] ROS (Robot Operating System)
https://www.ros.org/
https://en.wikipedia.org/wiki/Robot_Operating_System
https://wiki.ros.org/ROS/Introduction

[8] Willow Garage
https://en.wikipedia.org/wiki/Willow_Garage

[9] Open Robotics
https://en.wikipedia.org/wiki/Open_Robotics

[10] ROS Transformations
https://wiki.ros.org/tf/Overview/Transformations
https://linklab-uva.github.io/autonomousracing/assets/files/ROS-tf.pdf

[11] RQT Framework
https://wiki.ros.org/rqt
https://roboticsbackend.com/rqt-graph-visualize-and-debug-your-ros-graph/

[12] RVIZ visualization tool for ROS
https://wiki.ros.org/rviz

[13] ROS package <bluerov_ros_playground>
https://github.com/patrickelectric/bluerov_ros_playground

[14] Creator of package <bluerov_ros_playground>
https://github.com/patrickelectric/bluerov_ros_playground

[15] MAVROS communication package for ROS
http://wiki.ros.org/mavros

[16] Simulation package <freefloating_gazebo>
https://github.com/freefloating-gazebo/freefloating_gazebo

[17] Visual SLAM (and approaches)
https://cvg.cit.tum.de/research/vslam
https://ipsjcva.springeropen.com/articles/10.1186/s41074-017-0027-2

[18] FIDUCIAL ROS Package
https://wiki.ros.org/fiducials

[19] ARUCO DETECT node
https://wiki.ros.org/aruco_detect

[20] Εντοπισμός Fiducial Markers από την κάμερα
https://wiki.ros.org/aruco_detect

[21] A. Zakiev, K. Shabalina, T. Tsoy, and E. Magid, "Pilot virtual experiments on aruco and artag systems comparison for fiducial marker rotation resistance", Proceedings of 14th International Conference on Electromechanics and Robotics Zavalishin's Readings, pp. 455-464, 2020.
https://www.researchgate.net/publication/335503059_Pilot_Virtual_Experiments_on_ArUco_and_ArTag_Systems_Comparison_for_Fiducial_Marker_Rotation_Resistance