Πανεπιστήμιο Δυτικής Αττικής

Σχολή Μηχανικών

Τμήμα Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής

Τομέας Ηλεκτρονικού Αυτοματισμού και Τηλεματικής

Εξομοίωση Αυτόνομου Επίγειου Οχήματος Βασισμένο στη Μηχανική Όραση

Διπλωματική Εργασία

Του

Μιχαήλ Ποταμίτη

Επιβλέπων: Δρ. Γρηγόριος Νικολάου

1 Σεπτεμβρίου 2020

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο / Η κάτωθι υπογεγραμμένος / η **ΠΟΤΑΜΙΤΗΣ ΜΙΧΑΗΛ**,

Του **ΔΗΜΗΤΡΙΟΥ** , με αριθμό μητρώου **43946** φοιτητής / τρια του Τμήματος **Μηχανικών Βιομηχανκής Σχεδίασης και Παραγωγής,** του **Πανεπιστημίου Δυτικής Αττικής** πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

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

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

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

Ο Δηλών **ΠΟΤΑΜΙΤΗΣ ΜΙΧΑΗΛ** Ημερομηνία **9/10/2020**

Περίληψη

Η περιοχή των επίγειων αυτοκινούμενων οχημάτων αποτελεί ένα σημαντικό πεδίο για μελέτη, τόσο στον ακαδημαϊκό, όσο και στον βιομηχανικό τομέα. Η εξέλιξη των πληροφοριακών υπολογιστικών συστημάτων και προηγμένων αλγορίθμων έχουν σημειώσει τρομερή άνοδο, με στόχο την ασφαλή οδήγηση, ως προς τους οδηγούς, αλλά και ως προς το περιβάλλον το οποίο αλληλοεπιδρά το όχημα. Στην παρούσα εργασία αναπτύσσεται ένας αλγόριθμος πορείας του οχήματος, βασισμένος στη μηχανική όραση, χρησιμοποιώντας διάφορα ψηφιακά φίλτρα. Κύριος στόχος του συστήματος αποτελεί ο εντοπισμός των λωρίδων και έπειτα η παρακολούθηση της πορείας των λωρίδων του δρόμου. Το σύστημα αποτελείται από το Turtlebot3 Burger και από μία κάμερα τοποθετημένη στο πάνω μέρος του οχήματος κατά τον διαμήκη άξονα του. Μέσω της κάμερας συλλέγονται όλες οι απαραίτητες πληροφορίες για τη περαιτέρω διόρθωση της γωνίας του οχήματος από τον νόμο ελέγχου, μέσω της εντολής στρέψης, στους κινητήρες. Γίνεται χρήση ενός αναλογικού ελεγκτή καθώς και ενός τριγωνομετρικού. Η επεξεργασία των πληροφοριών γίνεται με τη βιβλιοθήκη OpenCV χρησιμοποιώντας τη γλώσσα προγραμματισμού Python, σε λειτουργικό σύστημα Linux. Η εξομοίωση λειτουργίας του συστήματος πραγματοποιείται στο μεταλειτουργικό πρόγραμμα Robotic Operating System(R.O.S).

Abstract

The area of autonomous ground vehicles is an important field of study, both academically and industrially. The evolution of information computational systems and advanced algorithms have increased tremendously, with the aim of safety, for drivers, but also for the environment in which the vehicle interacts. In the present work, a vision-based autonomous road following algorithm is developed, using various digital filters. The main goal of the system is the road lanes detection and then, the vehicle be able to track these lanes. The system consists of the Turtlebot3 Burger and a camera mounted on the top of the vehicle along its longitudinal axis. The camera collects all the necessary information to further process and correct the vehicle's angle, by the control law, through the steering command, to the engines. A proportional and a trigonometric controllers are used. The information is processed with the Open-CV library using the Python programming language in a Linux operating system. The operation of the system is performed in a simulation environment, called R.O.S(Robotic Operating System).

SUBJECT AREA: ROBOTICS
KEYWORDS: COMPUTER VISION, ROS, IMAGE PROCESSING, ROBOTICS, VISION-BASED NAVIGATION

Ευχαριστίες

Ένα μέρος της παρούσας διπλωματικής εργασίας ξεκίνησε κατά την περίοδο της πρακτικής μου άσκησης στο Πανεπιστήμιο του Κράνφιλντ, με την συμβολή του επιβλέποντος καθηγητή μου, Δρ. Γρηγόριου Νικολάου, στον οποίο οφείλω ένα μεγάλο ευχαριστώ για την άριστη καθοδήγησή του και την συνεχή υποστήριξη που μου παρείχε, καθώς και τον υπεύθυνο μου στην Αγγλία τον Δρ. Al. Savvaris.

Επίσης, για την επιτυχή ολοκλήρωση της εργασίας συνέβαλαν σημαντικά οι συμφοιτητές μου, Πάρης Χατζιθάνος και Άγγελος Αντικατζίδης , τους οποίους ευχαριστώ πάρα πολύ, για την επίλυση διαφόρων ζητημάτων κατά την πορεία της εργασίας.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου για τη σημαντική στήριξη που μου παρείχε κατά τη διάρκεια των σπουδών μου.

Table of Contents

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

Κεφάλαιο 1 – Εισαγωγή

1.1 Σκοπός της Εργασίας

Τις προηγούμενες δεκαετίες, οι έρευνες για τα μη επανδρωμένα οχήματα, επίγεια, εναέρια, υποβρύχια, χρησιμοποιούνταν κυρίως από τις στρατιωτικές επιχειρήσεις και όχι για εμπορική χρήση. Πλέον, ολοένα και περισσότερο, αυξάνεται η τάση για εμπορευματοποίηση των αυτόνομων οχημάτων από μεγάλες εταιρίες και κατά συνέπεια για έρευνα στον ακαδημαϊκό χώρο. Η ανάπτυξη της τεχνολογίας και η ταυτόχρονη μείωση του κόστους διάφορων υπολογιστικών συστημάτων, η ανάπτυξη αισθητηρίων ακριβείας και η ποικιλία πλατφορμών ανάπτυξης ρομποτικών εφαρμογών ανοιχτού κώδικα συνέβαλαν δραστικά στην ανάπτυξη ενός ιδιαίτερα φιλικού περιβάλλοντος για την προσαρμογή τους στην καθημερινή ζωή.

Στη συγκεκριμένη εργασία, μελετάται ένα αυτόνομο σύστημα πλοήγησης, όπου το όχημα κινείται μεταξύ των λωρίδων του δρόμου (lane following) , με μοναδικό αισθητήριο, την λήψη εικόνας μέσω της κάμερας. Η εξομοίωση του οχήματος εξελίσσεται σε μία πίστα άγνωστης διαδρομής. Η αξιοποίηση των δυνατοτήτων της μηχανικής όρασης συνδυαστικά με τον νόμο ελέγχου, αποσκοπεί στην επιτυχή ολοκλήρωση της διαδρομής του οχήματος από το σημείο έναρξης έως σημείο το σημείο του τερματισμού. Διαμέσου του ελέγχου των κινητήρων, πραγματοποιείται συνεχής διόρθωση της γωνίας που υπολογίζεται. Αποτέλεσμα αυτού είναι η σωστή τοποθέτησή του στον δρόμο. Το όχημα που χρησιμοποιείται είναι το Turtlebot3. Ο χάρτης που χρησιμοποιείται προέκυψε ως αποτέλεσμα τροποποίησης του δισδιάστατου καρτεσιανού χάρτη Autorace, που προσφέρει το ROS.
Συνολικά παρακάτω φαίνεται ένα πλήρες διάγραμμα για το τι έχει γίνει και πρόκειται να αναλυθεί μέσα στην εργασία:

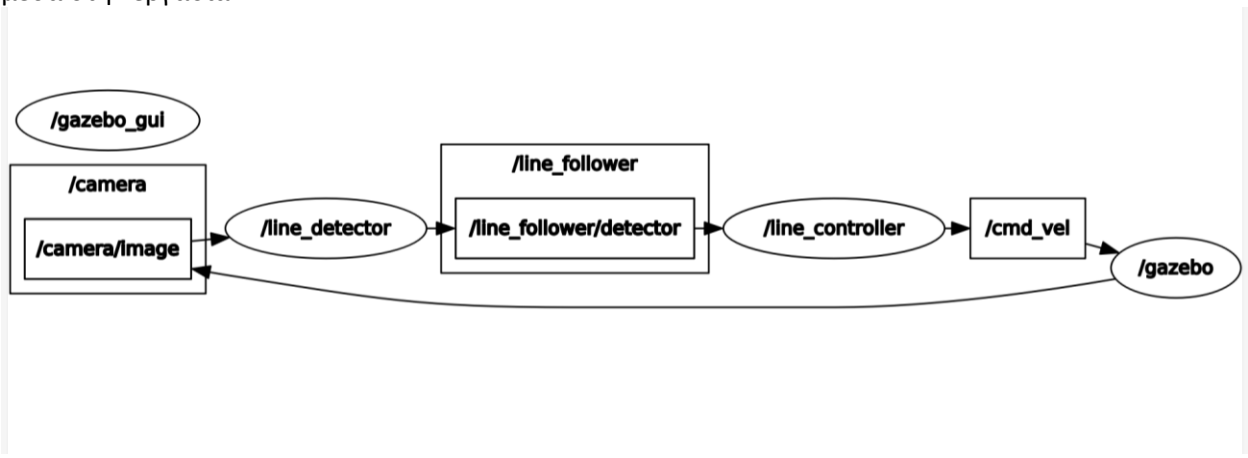


Figure 1 - ROS Nodes και Topics

Ανωτέρω απεικονίζεται γραφικά όλη η διαδικασία που εκτελείται καθώς και τι πρόκειται να αναλυθεί στη παρούσα εργασία. Η εικόνα της κάμερας(/camera/image) εισάγεται στο πρόγραμμα(line_detector) που εκτελεί όλη την επεξεργασία για τον εντοπισμό των γραμμών και τον υπολογισμό της γωνίας στρέψης. Στη συνέχεια, ειδικού τύπου μηνύματα από το προηγούμενο πρόγραμμα αποθηκεύονται σε ένα συγκεκριμένο τύπο αρχείου(line_follower) όπως θα αναλυθεί στα επόμενα κεφάλαια, τα οποία

στέλνονται για επεξεργασία στον ελεγκτή(line_controller). Ο ελεγκτής με τη σειρά του στέλνει στο όχημα πόσο πρέπει να στραφούν οι τροχοί του οχήματος μέσω του cmd_vel. Η διαδικασία επαναλαμβάνεται. Στα επόμενα κεφάλαια περιγράφονται λεπτομερώς τα blocks της εικόνας. Πιο συγκεκριμένα:

Το Κεφάλαιο 2, αποσκοπεί στην περιγραφή της πλατφόρμας ROS, Robotic Operational System, αναφέροντας τα στοιχεία που διατίθενται και τον λόγο για τον οποίο έχει καθιερωθεί ως τόσο διαδεδομένη πλατφόρμα.

Το Κεφάλαιο 3, εξηγεί την πορεία της προ επεξεργασίας της εικόνας καθώς και τους λόγους που εφαρμόζεται κάθε φίλτρο, εκμαιεύοντας χρήσιμες πληροφορίες για την επεξεργασία.

Το Κεφάλαιο 4, αναλύει τον αλγόριθμο που έχει χρησιμοποιηθεί για την εξεύρεση των γραμμών από την προ υπάρχουσα επεξεργασμένη εικόνα από το κεφάλαιο 3, με σκοπό τον υπολογισμό της γωνίας στρέψης του οχήματος στις περιπτώσεις που υπάρχουν στροφές και πρέπει να στραφεί, αλλά και όταν το όχημα δεν βρίσκεται στο κέντρο του δρόμου.

Το Κεφάλαιο 5, περιγράφει τους νόμους ελέγχου για την στρέψη του οχήματος που έχουν δοκιμαστεί, όπως, ο αναλογικός ελεγκτής P και ο τριγωνομετρικός ελεγκτής.

Το Κεφάλαιο 6, περιέχει τα αποτελέσματα των πειράματος από την επεξεργασία εικόνας της κάμερας και τα σφάλματα των δύο ελεγκτών.

Το Κεφάλαιο 7, περιλαμβάνει γενικά συμπεράσματα για την εργασία και μελλοντικούς τρόπους βελτίωσης.

Το Κεφάλαιο 8, περιλαμβάνει τις μελλοντικές προτάσεις προς έρευνα.

Κεφάλαιο 2 – ROS Robot Operating System

2.1 Εισαγωγή

Στο πεδίο των ρομποτικών συστημάτων, υπάρχει η ανάγκη για ανάπτυξη πλατφορμών ώστε να φιλοξενούνται τα προγράμματα λειτουργίας και το υλικό μέρος για εξομοιώσεις και υπολογισμούς.

Βασικά εργαλεία που χρησιμοποιούνται συνήθως είναι η πλοήγηση, η λήψη τιμών των αισθητήρων για το μετέπειτα έλεγχο του οχήματος, slam, ο χειρισμός των ρομπότ κ.α.

Για αυτόν τον λόγο η ακαδημαϊκή κοινότητα και η βιομηχανία χρειάζεται μια ενιαία πλατφόρμα ώστε να αναπτύσσει τα προγράμματα σε πλατφόρμα ανοιχτού κώδικα. Η πλατφόρμα που εξυπηρετεί τα προαναφερόμενα είναι το ROS.

Το ROS περιλαμβάνει έτοιμες συναρτήσεις για την επίτευξη διάφορων λειτουργιών όπως είναι η κίνηση του οχήματος, γραφικά περιβάλλοντα τριών διαστάσεων και άλλα. Επίσης διαθέτει πολύ χρήσιμο διαδικτυακό υλικό για νέους προγραμματιστές που θέλουν να ασχοληθούν με την ανάπτυξη ρομποτικών εφαρμογών. Επειδή, το ROS αποτελεί πλατφόρμα ανοιχτού κώδικα δίνει τη δυνατότητα βελτίωσης των υπαρχουσών βιβλιοθηκών, καθώς και την ανάπτυξη νέων από εξωτερικούς προγραμματιστές. Τέλος δίνει τη δυνατότητα στους χρήστες να μπορούν να γράψουν σε C++ ή Python γλώσσες προγραμματισμού και οι 2 γλώσσες να επικοινωνούν μεταξύ τους.

Το ROS για να φτιάξει οποιοδήποτε project χρησιμοποιεί το catkin. Αυτό δημιουργεί ένα χώρο εργασίας. Αρχικά δημιουργείται ένας φάκελος catkin_ws. Έπειτα με την εντολή catkin_make δημιουργούνται τρεις υποφάκελοι. Ο ένας είναι ο src, ο δεύτερος ο build και ο τρίτος ο devel. Μέσα στον φάκελο src δημιουργείται ένας φάκελος ο οποίος περιέχει το package της παρούσας εργασίας. Αυτά τα πακέτα λέγονται και catkin packages. Για δημιουργηθεί ένα πακέτο από τον τερματικό των Ubuntu πληκτρολογείται η εντολή:

$ cd ~ /catkin_ws/src

Με αυτή την εντολή το σύστημα έχει πρόσβαση στο φάκελο catkin_ws/src. Έπειτα για να δημιουργηθεί το πακέτο πληκτρολογείται:

$ catkin_create_pkg line_follower std_msgs rospy roscpp

Το παραπάνω δημιουργεί ένα φάκελο τον line_follower ο οποίος περιέχει ένα package.xml αρχείο. Τα ορίσματα std_msgs, rospy, roscpp είναι η λίστα των εξαρτήσεων(dependencies) πάνω στις οποίες εξαρτάται το πακέτο line_follower.

Κάθε φορά αν υπάρχει αλλαγή στο πακέτο, για να ανανεωθεί όλο το πακέτο αφού εκτελεστεί η εντολή

$ cd ~ /catkin_ws

Πληκτρολογείται η εντολή:

$ source devel/setup.bash.

2.2 Στοιχεία του ROS

- Node

  Το node πρακτικά είναι μια υπολογιστική διεργασία. Λειτουργεί ως ένα εκτελέσιμο πρόγραμμα που επιτελεί ένα σκοπό. Μέσα σε ένα ρομποτικό σύστημα συνυπάρχουν πολλά nodes. Ένα node μπορεί να ελέγχει τους κινητήρες του οχήματος, άλλο node να αποφεύγει εμπόδια και άλλο node να αναγνωρίζει αντικείμενα. Όλα τα nodes επικοινωνούν μεταξύ τους μέσω των messages(μηνυμάτων). Πίσω από όλα τα nodes υπάρχει ένα Master Node το οποίο επιβλέπει και επικοινωνεί όλα τα nodes μεταξύ τους, καθώς έχει και τις διευθύνσεις όλων των nodes που εκτελούνται. Υπάρχουν τρεις τύποι επικοινωνιών μεταξύ των nodes. Τα topics, services και Actions. Για να εκτελεστεί ένα node χρησιμοποιείται η εντολή rosrun η οποία συντάσσεται ως εξής: rosrun όνομα_πακέτου όνομα_node.py. Όπως είναι φανερό τα nodes είναι εκτελέσιμα προγράμματα της μορφής όνομα_προγράμματος.py στη Python γλώσσα. Υπάρχουν επίσης κάποιες χρήσιμες εντολές για τα nodes.

  a) Για να φανερωθούν τα ενεργά Nodes:

     $ rosnode list

  b) Για την λήψη πληροφοριών συγκεκριμένου node:

     $ rosnode info /<όνομα_node>

  c) Για την παύση λειτουργίας ενός node:

     $ rosnode kill /< όνομα_node >

  Στη παρούσα εργασία το πακέτο που δημιουργήθηκε είναι το line_follower το οποίο περιέχει τα nodes, δηλαδή τα προγράμματα που χρησιμοποιούνται για την εκτέλεση όλης της διαδικασίας. Αυτά είναι το line_detector.py και το line_controller.py τα οποία περιέχονται μέσα στο src φάκελο του πακέτου. Αυτά συμβολίζονται σχηματικά με ελλείψεις. Επίσης γίνεται και χρήση του topic /cmd vel. Αυτό το topic περιλαμβάνεται στο πακέτο του Turtlebot. Μέσω αυτού στέλνεται η επιθυμητή γραμμική και γωνιακή ταχύτητα στο Turtlebot.

- Messages

  Τα messages αποτελούν τη δομή δεδομένων για τα Topics, Service, Action για τη μεταφορά των δεδομένων μεταξύ των nodes.

  Για τα topics τα messages έχουν μορφή αρχείου όνομα_αρχείου.msg

  ΤύποςΜεταβλητής1 ΌνομαΜεταβλητής1
  ΤύποςΜεταβλητής2 ΌνομαΜεταβλητής2
  ΤύποςΜεταβλητής3 ΌνομαΜεταβλητής3

Για τις services τα messages έχουν μορφή αρχείου όνομα_αρχείου.srv. Αυτά σε σύγκριση με τα Messages, έχουν δύο μέρη. Το ένα μέρος είναι το αίτημα(Request) και το άλλο η απάντηση(Response). Τα δύο αυτά μέρη διαχωρίζονται με το σύμβολο ‘’---‘’

#Request
ΤύποςΜεταβλητής1 ΌνομαΜεταβλητής1
ΤύποςΜεταβλητής2 ΌνομαΜεταβλητής2

#Response
ΤύποςΜεταβλητής3 ΌνομαΜεταβλητής3
ΤύποςΜεταβλητής4 ΌνομαΜεταβλητής4

Για τα actions έχουν μορφή αρχείου όνομα_αρχείου.active. Αυτά τα μηνύματα έχουν τρία μέρη. Το goal, result, feedback(στόχος, αποτέλεσμα, ανάδραση). Χωρίζονται μεταξύ τους με το σύμβολο ‘’---‘’.

#Goal
ΤύποςΜεταβλητής1 ΌνομαΜεταβλητής1

#Result
ΤύποςΜεταβλητής3 ΌνομαΜεταβλητής3

#Feedback
ΤύποςΜεταβλητής5 ΌνομαΜεταβλητής5

Μερικές εντολές του ROS για τα μηνύματα είναι οι παρακάτω:

a) Για την εμφάνιση της λίστας των μηνυμάτων των topics και actions:
   $ rosmsg list

b) Για την εμφάνιση της λίστας των μηνυμάτων για τις services:
   $ rossrv list

c) Για να εμφανίσει όλα τα μηνύματα που περιέχονται στα πακέτα:
   $ rosmsg packages

d) Για να εμφανίσει τη δομή των μηνυμάτων:
   $ rosmsg show όνομα_πακέτου/όνομα_μηνύματος

Στη συγκεκριμένη εργασία το μήνυμα που χρησιμοποιείται είναι το line_follower.msg τύπος δεδομένου, επειδή γίνεται χρήση topic.

- Topics

Τα topics είναι οι δίαυλοι επικοινωνίας μεταξύ των nodes ώστε τα nodes να ανταλλάσσουν πληροφορίες. Οι είσοδοι και οι έξοδοι των nodes δεν επικοινωνούν άμεσα μεταξύ τους. Αντ’

αυτού, ένα node για να αντλήσει δεδομένα εγγράφεται σε ένα topic(subscriber) και ένα node μπορεί να δημοσιεύει(publisher) δεδομένα σε ένα topic.

Σε ένα topic μπορούν να υπάρξουν πολλοί publishers και subscribers. Στη παρούσα εργασία τα topics που χρησιμοποιούνται είναι αυτά με που είναι μέσα στα τετράγωνα. Η εικόνα της κάμερας, ο line follower και το cmd_vel. Καθένα από αυτά θα αναλυθούν στη συνέχεια της εργασίας.

Χρήσιμες εντολές για τα topics παρατίθενται παρακάτω:

a) Εμφάνιση της λίστας των topics
   $ rostopic list

b) Για τη λήψη πληροφοριών ενός topic
   $ rostopic info /όνομα_topic

c) Για να εμφανιστεί ο τύπος μηνύματος του topic
   $ rostopic type / όνομα_topic

d) Για να εμφανίσει τα δεδομένα ενός topic στον τερματικό
   $ rostopic echo / όνομα_topic

e) Για να δημοσιεύσει τα δεδομένα ενός topic από τον τερματικό
   $ rostopic pub όνομα_topic τύπος_μηνύματος ‘δεδομένα’

- Service

Τα services είναι ένας ακόμα τρόπος μεταφοράς πληροφοριών στο ROS. Είναι προγράμματα που καλούνται από τα ROS nodes και κάνουν μια διεργασία. Ο service server (ο οποίος παρέχει την υπηρεσία/πρόγραμμα) καθορίζει μια επιστροφή κλήσης (callback)που είναι η διεργασία η οποία πρέπει να εκτελεστεί για την αντιμετώπιση του service request και εκτελεί το service. Ο service client (που καλεί την υπηρεσία) αποκτά πρόσβαση σε αυτήν την υπηρεσία μέσω ενός τοπικού διακομιστή μεσολάβησης (local proxy).
Τα services συνήθως είναι μικρά προγράμματα τα οποία τρέχουν ενίοτε και παίρνουν πολύ λίγο χρόνο να εκτελεστούν. Αυτό γίνεται γιατί όταν γίνει κλήση για service το πρόγραμμα εκτελεί το service και μετά συνεχίζει ότι έκανε. [1]

Μερικές από τις εντολές που υπάρχουν διαθέσιμες είναι:

a) Εμφάνιση της λίστας των services
   $ rosservice list

b) Για τη λήψη πληροφοριών ενός service
   $ rosservice info /όνομα_service

c) Για να εμφανιστεί ο τύπος μηνύματος του service
$ rosservice type / όνομα_service

d) Για τη λήψη του ονόματος του κόμβου που παρέχει μία συγκεκριμένη υπηρεσία
$ rosservice node / όνομα_service

e) Για τη κλίση ενός service από τον τερματικό
$ rosservice call / όνομα_service service-args

- Action

Τα ROS Actions είναι πολύ παρόμοια με τα services. Ένα ψεγάδι των services είναι ότι πρέπει πρώτα να τελειώσει το service και έπειτα να συνεχίσει το πρόγραμμα. Αν για κάποιο λόγο το service κάνει πολύ ώρα μέχρι να εκτελεστεί δεν υπάρχει η επιλογή να σταματήσει. Τα actions δεν έχουν αυτό το μειονέκτημα. Στην ουσία γεμίζουν αυτό το κενό. Εκτελούν από πίσω προγράμματα και μπορούν διακοπούν αν ο χρήστης το θελήσει. Επίσης ο χρήστης μπορεί να ζητήσει και ανατροφοδότηση των αποτελεσμάτων. Άρα ολοκληρώνοντας, υπάρχει ο Action Client και το Action Server. Ο Action Client στέλνει αιτήματα επίτευξης λειτουργίας ή μπορεί να στείλει και μηνύματα διακοπής λειτουργίας. Ο Action Server στέλνει στο Action Client το status, το result και το feedback.
.

- *Goal(Στόχος)*

Για την υλοποίηση των λειτουργιών χρησιμοποιώντας τα Actions, το goal περιέχει ουσιαστικά το τι πρέπει να υπολογιστεί στο Action. Αν πρέπει να περιστραφεί για παράδειγμα το όχημα κάποια στιγμή και να σταματήσει τη προηγούμενη λειτουργία του στέλνει μήνυμα στο cmd_vel το οποίο περιέχει τα μηνύματα και τις τιμές της γραμμικής και γωνιακής ταχύτητας.

- Feedback(Ανάδραση)

Ο ActionServer παρέχει feedback πληροφοριών προόδου της κατάστασης επίτευξης του goal στον ActionClient. Δηλαδή, δίνει την πραγματική τιμή κατάστασης εξέλιξης του goal. Στο παράδειγμα με το cmd_vel, δίνει τη πραγματική στάση του Turtlebot.

- Result(Αποτέλεσμα)

Ο ActionServer μπορεί να παρέχει επίσης και το αποτέλεσμα του στόχου που ζητήθηκε. Σε αντίθεση με το feedback, αυτό στέλνεται μία φορά. Είναι ιδιαίτερα σημαντικό, αν ο στόχος είναι να παρέχει πληροφορίες για κάτι. Στον ActionClient στέλνεται το τελικό μήνυμα της κατάστασης. [2]

Η διαδικασία φαίνεται παραστατικά στη παρακάτω εικόνα.

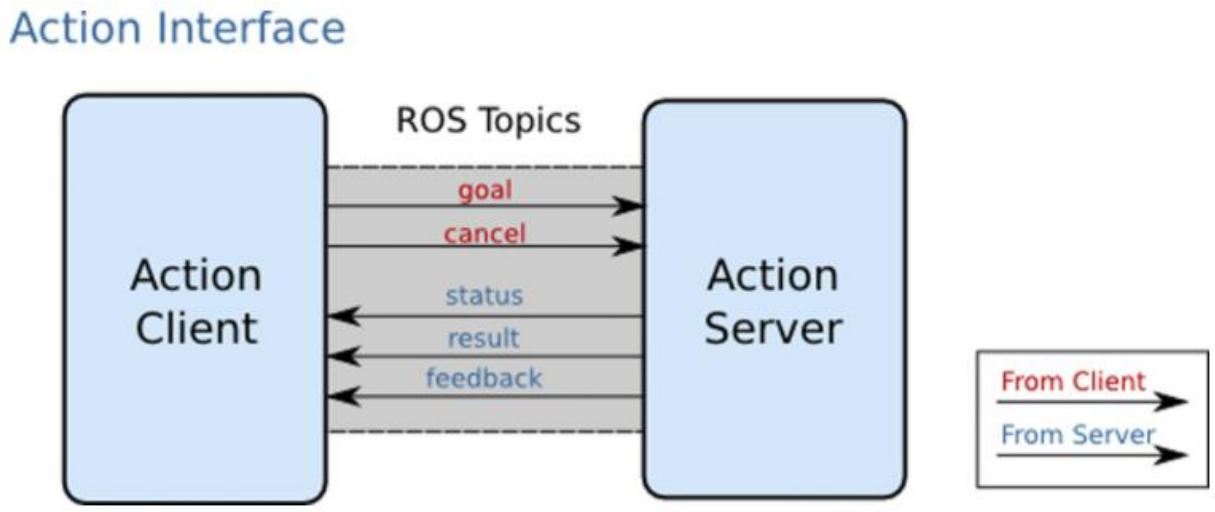


Figure 2 - Action Περίγραμμα

- Master

Το ROS master είναι ένα συγκεκριμένο rosnode το οποίο «ξεκινάει» το περιβάλλον του ROS και χειρίζεται την επικοινωνία μεταξύ των υπόλοιπων ros nodes. Συγκεκριμένα το rosmaster καταγράφει την επικοινωνία μεταξύ publishers και subscribers σε συγκεκριμένα topics και διαφορετικές διεργασίες όπως τα services ή τα actions. Ο σκοπός του ros master είναι να επιτρέπει σε μεμονωμένα rosnodes να εντοπίζει το ένα με το άλλο και αφού εντοπιστούν, επικοινωνούν μεταξύ τους peer-to-peer (P2P). Το ROS Master παρέχει ένα API που βασίζεται σε XMLRPC, το οποίο χρησιμοποιεί βιβλιοθήκες του ROS, όπως το roscpp και το rospy, καλούνται για αποθήκευση και ανάκτηση πληροφοριών. [3]

Με την εκκίνηση του ros master ξεκινάει και το Parameter server.

- Parameter server

Ο parameter server είναι μια κοινή βάση δεδομένων που μπορεί να προσπελαστεί από όλο το ROS. Αυτή η βάση περιέχει πληροφορίες διαμόρφωσης, διαφορετικές παραμέτρους και τιμές οι οποίες είναι προσβάσιμες από κάθε σημείο του ROS (rosnodes, terminal ή όταν ξεκινάει το ROS). Οι παράμετροι που μπορούν να συμπεριληφθούν είναι κυρίως στατικοί δηλαδή δεν αλλάζουν συνήθως κατά την διάρκεια της εκτέλεσης, γιατί ο parameter server έχει σχεδιαστεί για γρήγορη εκτέλεση πράγμα που δεν είναι εφικτό αν οι παράμετροι αλλάζουν συχνά. Ένα παράδειγμα παραμέτρων είναι τα εξής: Το όνομα, το σύστημα συντεταγμένων, οι ενδογενείς και εξωγενείς παράμετροι της κάμερας. [4]

2.3 Gazebo

Συνήθως, για την ανάπτυξη ενός πειράματος το οποίο συνδυάζει τη χρήση hardware-software και μίας κατασκευής τοποθέτησης αυτών των εξαρτημάτων, χρησιμοποιούνται περιβάλλοντα εξομοίωσης τα οποία δεν εξαρτώνται σε αληθινά robot. Κύριος σκοπός της εξομοίωσης αποτελεί η εξοικονόμηση χρηματικών πόρων, καθώς και η ελαχιστοποίηση των λαθών που μπορεί να προκύψουν λόγω κατασκευαστικής αστοχίας, για παράδειγμα , κάποια λάθος συνδεσμολογία μεταξύ των κινητήρων και του επεξεργαστή, ο συντονισμός της περιστροφής των τροχών του οχήματος και άλλα. Τέτοιου είδους τεχνικά σφάλματα κοστίζουν πολύτιμο χρόνο. Καλοσχεδιασμένα περιβάλλοντα εξομοίωσης, αν δεν υστερούν σε πληροφορία και αντικατοπτρίζουν ένα μεγάλο ποσοστό του πραγματικού περιβάλλοντος, αυξάνουν τα ποσοστά βεβαιότητας της μελλοντικής κατασκευής ως προς την επιτυχία της. Στη παρούσα εργασία χρησιμοποιείται ως περιβάλλον εξομοίωσης το GAZEBO.

Το GAZEBO είναι ένας τρισδιάστατος εξομοιωτής ρομποτικών εφαρμογών. Το ROS μπορεί να παρέχει ειδικά πακέτα για το GAZEBO. Αυτά τα πακέτα παρέχουν απαραίτητες πληροφορίες για την προσομοίωση ενός robot. Περιέχει διάφορα μοντέλα πιστών για οχήματα, τρισδιάστατα σχήματα χώρων, καθώς προσφέρει και την ανάπτυξη μοντέλων αν θέλει ο χρήστης. Το πιο σημαντικό γεγονός αποτελεί ότι υποστηρίζει το ROS και το καθιστά ικανό για την εξομοίωση πολύπλοκων ρομποτικών εφαρμογών σε εσωτερικούς, είτε σε εξωτερικούς χώρους. [5] Το περιβάλλον εργασίας, η πίστα και το Robot Turtlebot3 που χρησιμοποιείται και κατασκευάστηκε στη παρούσα εργασία φαίνεται στη παρακάτω εικόνα:

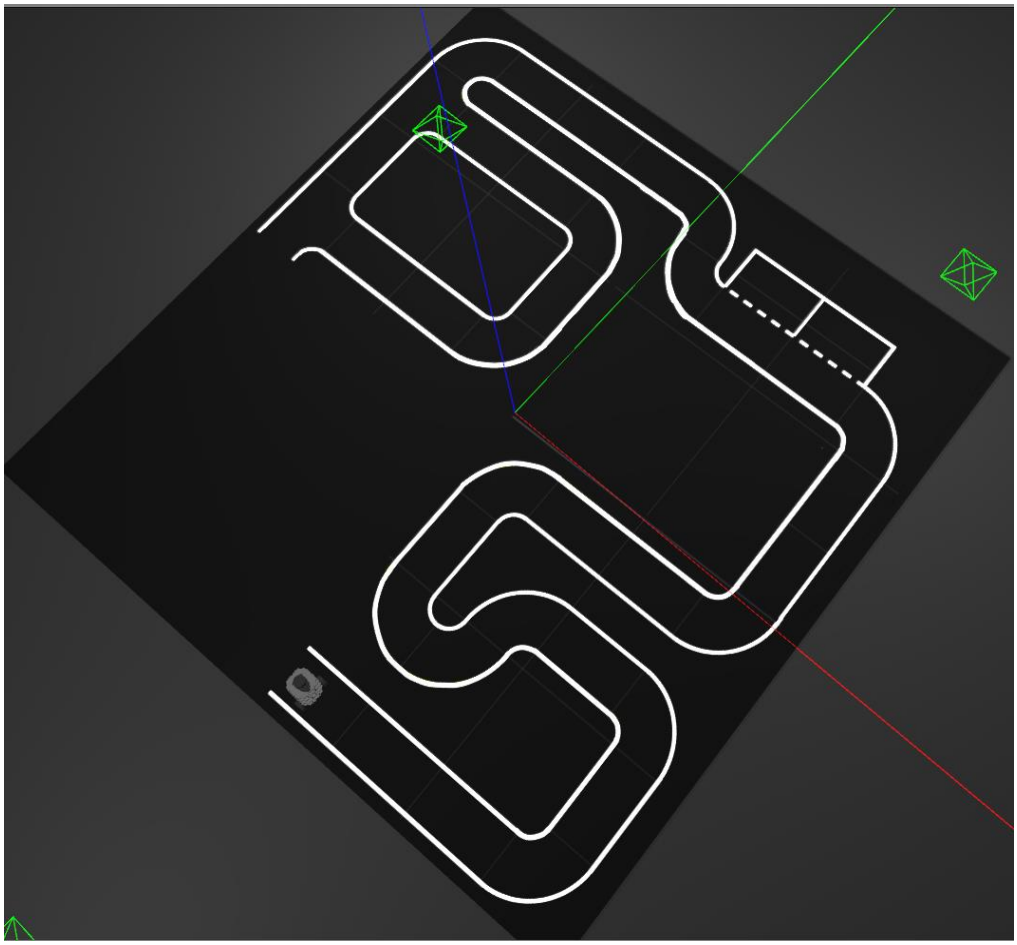


Figure 3 - Gazebo

2.4 Turtlebot3

Το Turtlebot είναι μία ρομποτική πλατφόρμα του ROS. Στην παρούσα εργασία χρησιμοποιείται το TurtleBot3 Burger το οποίο δημιουργήθηκε το 2017 με ακόμη πιο βελτιωμένα χαρακτηριστικά από τα προηγούμενα μοντέλα (Turtlebot1, Turtlebot2) σύμφωνα με τις απαιτήσεις των πελατών. Το Turtlebot3 διαθέτει από την ROBOTIS τον σερβοκινητήρα DYNAMIXEL, έναν νέο, βελτιωμένο, σύγχρονο κινητήρα τροχών. Πρόκειται για ένα ρομπότ απόλυτα συμβατό με το ROS το οποίο μπορεί να χρησιμοποιηθεί ευρέως για ερευνητικούς σκοπούς και στην εκπαίδευση για εκμάθηση. Έχει χαμηλό κόστος, είναι μικρό και εύχρηστο χωρίς να υστερεί σε ποιότητα.
Έχει κατασκευαστεί έτσι ώστε να δέχεται διάφορα περιφερειακά όπως π.χ έναν μικροεπεξεργαστή, κάμερα , και διάφορα αισθητήρια. Οι βασικές λειτουργίες που μπορεί να εκτελέσει είναι το SLAM(Simultaneous localization and mapping), δηλαδή το ρομπότ βασισμένο στα αισθητήρια του και σε υπολογιστικούς αλγορίθμους να χαρτογραφεί ένα άγνωστο περιβάλλον με αποτέλεσμα μετά την σάρωση του άγνωστου χώρου να γνωρίζει την περιβάλλουσα θέση του. Επίσης εκτελεί πλοήγηση, δηλαδή να διαλέγει ασφαλείς διαδρομές μέσα στο χώρο χωρίς συγκρούσεις και τέλος να εκτελεί τις

λειτουργίες που επιβάλλει ο προγραμματιστής του, όπως π.χ για την χαρτογράφηση ενός δωματίου. Ο χειρισμός του μπορεί να πραγματοποιηθεί τηλεκατευθυνόμενα από έναν υπολογιστή, ειδικό χειριστήριο, είτε ακόμη από μια Android εφαρμογή. Επιπρόσθετα μπορεί να εφαρμοστεί πάνω στο Turtlebot3 ένα άκρο το οποίο προσφέρεται από το ROS και είναι απολύτως συμβατό, το OpenManipulator. Χρησιμοποιείται για διάφορες χρήσεις, το οποίο το καθιστά μετά ένα υπηρεσιακό Robot μαζί με τις τεχνολογίες του SLAM και της πλοήγησης στο χώρο. Με τον συνδυασμό της λήψης εικόνας από μία κάμερα και τεχνικές της μηχανικής όρασης και των προηγούμενων μπορεί για παράδειγμα να εντοπίσει ένα αντικείμενο και να το μετακινήσει από ένα σημείο σε ένα άλλο. Το βασικότερο χαρακτηριστικό το οποίο το καθιστά τέλειο για χρήση, αποτελεί το γεγονός ότι υπάρχει ήδη έτοιμο ως βιβλιοθήκη μέσα στο ROS με τα ανάλογα πακέτα του Turtlebot3 σε ότι αφορά τον έλεγχο των κινητήρων, την κινηματική και την δυναμική του μοντέλου. Αυτό το καθιστά κατάλληλο εργαλείο για εξομοιώσεις, καθώς ένα Turtlebot3 στην εξομοίωση διαθέτει ακριβώς τα ίδια χαρακτηριστικά και στην πραγματικότητα. Αυτό έχει ως αποτέλεσμα όταν γραφτούν τα προγράμματα στο ROS και δοκιμαστούν οι λειτουργίες του στο Gazebo να είναι σε θέση να τις εκτελέσει και στη πραγματικότητα. Τέλος ως προς το μοντέλο του κατατάσσεται στα τύπου unicycle, διαφορικής οδήγησης ρομπότ. [6]

Παρακάτω παρατίθεται το TurtleBot3 burger ένας πίνακας με τα χαρακτηριστική του Turtlebot3 burger. [7]

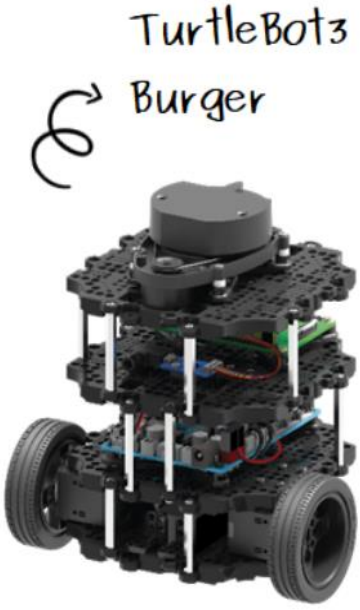


Figure 4 - Turtlebot3 Burger

Στοιχεία	**Turlebot3 Burger**
Μέγιστη γραμμική ταχύτητα	0.22 m/s
Μέγιστη γωνιακή ταχύτητα	2.84 rad/s (162.72 deg/s)
Μέγιστο Φορτίο	15kg
Μέγεθος (ΜήκοςxΠλάτοςxΎψος)	138mm x 178mm x 192mm
Βάρος (SBC + Μπαταρία + Αισθητήρες)	1kg
Κατώφλι Ανάβασης	10 mm ή μικρότερο
Αναμενόμενος Χρόνος Λειτουργίας	2ώρες 30λεπτά
Αναμενόμενος Χρόνος Φόρτισης	2h 30m
SBC	Raspberry Pi 3 Model B and B+
MCU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
Χειριστήριο	-
Ενεργοποιητής	XL430-W250
Laser Αισθητήρας Απόστασης	360 Laser Distance Sensor LDS-01
Camera	-
IMU	Γυροσκόπιο 3 Axis Αξελερόμετρο 3 Axis Μαγνητόμετρο 3 Axis
Συνδέσεις Ισχύος	3.3V / 800mA 5V / 4A 12V / 1A
Pins επέκτασης	GPIO 18 pins Arduino 32 pin
Περιφερειακά	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4
DYNAMIXEL θύρες	RS485 x 3, TTL x 3
Ήχος	Διάφορα προγραμματίσιμα ακολουθιακά Beep
Προγραμματίσιμα LEDs	User LED x 4

Κατάσταση LEDs	Κατάσταση Πλακέτας LED x 1 Arduino LED x 1 Ισχύς LED x 1
Κουμπιά και διακόπτες	Buttons x 2, Επανεκκίνησης κουμπί x 1, Dip διακόπτης x 2
Μπαταρία	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
Σύνδεση PC	USB
Firmware Αναβάθμιση	μέσω USB / μέσω JTAG
Προσαρμογέας Ισχύος (SMPS)	Είσοδος : 100-240V, AC 50/60Hz, 1.5A @max Έξοδος: 12V DC, 5A

Table 1 - Τεχνικά Χαρακτηριστικά Turtlebot3 Burger

Στη παρακάτω εικόνα φαίνεται το Turtlebot3 Burger στην εξομοίωση:

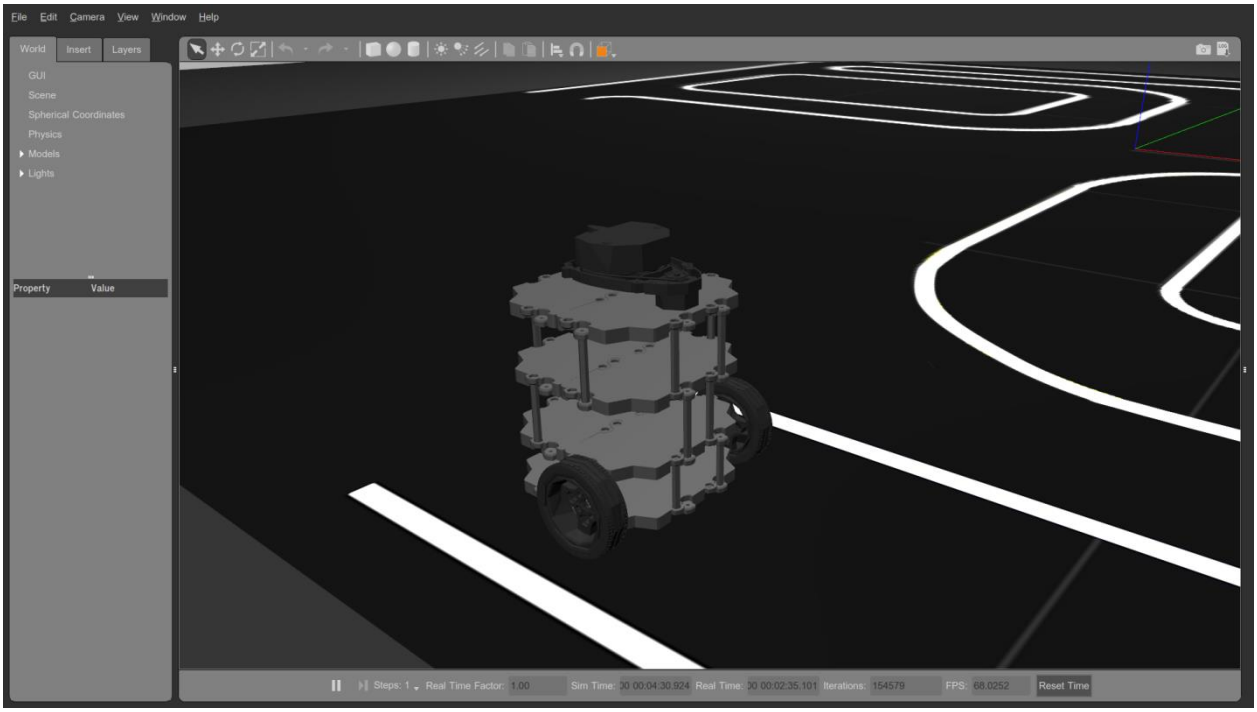


Figure 5 - Turtlebot3 Burger στο Gazebo

Κεφάλαιο 3 - Προεπεξεργασία εικόνας

Για την επεξεργασία εικόνας γίνεται χρήση της OPENCV (Open Source Computer Vision Library) βιβλιοθήκης. Αποτελεί ανοιχτού κώδικα βιβλιοθήκη η οποία περιέχει συναρτήσεις για μηχανική όραση και για μηχανική μάθηση. Η OpenCV δημιουργήθηκε για τη παροχή υπηρεσιών κάτω από μία ενιαία υποδομή για εφαρμογές μηχανικής όρασης καθώς και να χρησιμοποιηθεί δραστικά σε εμπορικά προϊόντα. Η OpenCV έχει πολύ μεγάλο αριθμό αλγορίθμων από μηχανική όραση και μάθηση. Αυτοί οι αλγόριθμοι μπορούν να εφαρμοστούν για παράδειγμα, στον εντοπισμό και αναγνώριση προσώπων, αντικειμένων, στην κατηγοριοποίηση διάφορων ανθρώπινων πράξεων σε βίντεο, εύρεση ίδιων παρόμοιων εικόνων σε βάση δεδομένων και άλλα. Αυτή τη βιβλιοθήκη τη χρησιμοποιούν παραπάνω από 47 χιλιάδες χρήστες και ένας εκτιμώμενος αριθμός λήψεων της βιβλιοθήκης είναι 18 εκατομμύρια. Χρησιμοποιείται εξίσου από τον ακαδημαϊκό και τον βιομηχανικό κλάδο. Υποστηρίζει την C++, Python, Java και Matlab γλώσσες προγραμματισμού και λειτουργεί σε όλα τα λογισμικά. Windows, Linux, Android και Mac OS. [8]

Στη παρούσα εργασία γίνεται χρήση μερικών από το σύνολο των συναρτήσεων που διαθέτει η OpenCV. Όλο το κεφάλαιο 3 μαζί με το κεφάλαιο 4 υπόκεινται στο node lineDetector. Πρώτο στάδιο αποτελεί η απόκτηση της εικόνας μέσω της κάμερας, ώστε το Turtlebot3 να έχει την ικανότητα να διαβάζει το περιβάλλον της πίστας ως είσοδο για το σύστημα. Επειδή η εξομοίωση γίνεται στο Gazebo αυτό που είναι απαραίτητο, αποτελεί η απομόνωση των άσπρων λωρίδων της εικόνας για τη περαιτέρω επεξεργασία. Παρακάτω φαίνεται η πίστα μέσα από τη κάμερα:

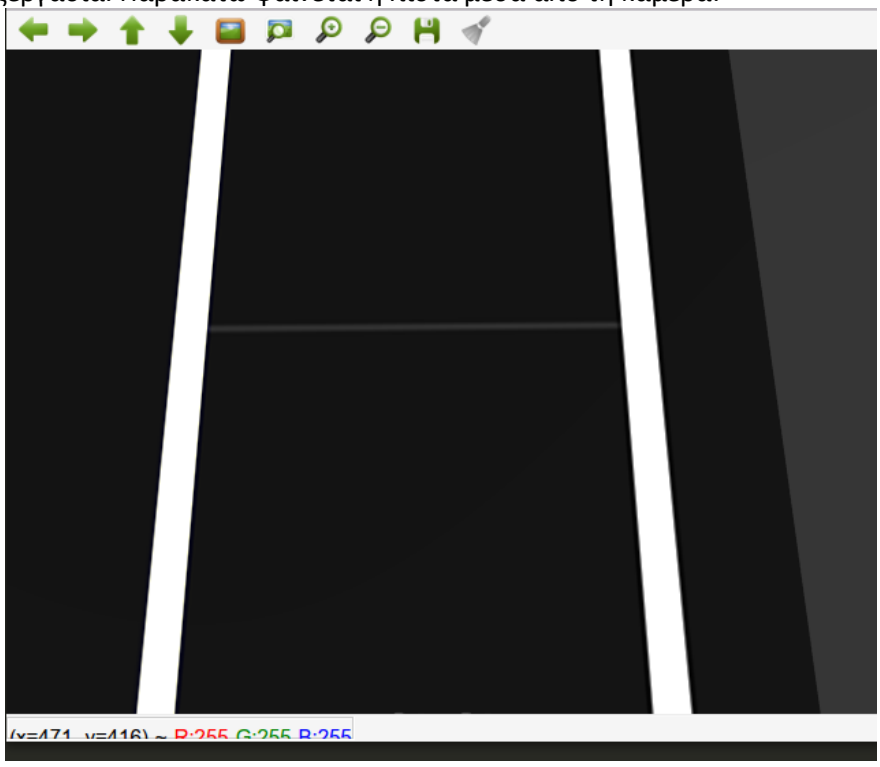


Figure 6 - Εικόνα Κάμερας

Ας περιγραφούν πρώτα όμως, κάποια βασικά στοιχεία για μια εικόνα. Κάθε εικόνα αποτελείται από pixels σε έναν δυσδιάστατο πίνακα. Δηλαδή μία εικόνα $100x100$ έχει 100 γραμμές και 100 στήλες που είναι ίσο με 10^4 pixels. Αυτή είναι και η μοναδική πηγή πληροφορίας που μπορεί να έχει μία εικόνα. Μία εικόνα αποτελείται από τα χρώματα των pixels. Αυτά τα χρώματα φτιάχνονται από έναν συνδυασμό μπλε, κόκκινου, πράσινου. Ο συνδυασμός αυτών των τιμών των χρωμάτων αποτελούν ένα pixel. Οι τιμές του κάθε χρώματος για παράδειγμα του Κόκκινου είναι μεταξύ $[0 - 255]$. Το ίδιο ισχύει και για τα υπόλοιπα χρώματα, Πράσινου και Μπλε. Άρα για παράδειγμα ένα μαύρο pixel έχει την μορφή $(0,0,0)$ που σημαίνει απουσία Κόκκινου,Πράσινου,Μπλε. Η αρχή αξόνων σε μια εικόνα στο OpenCV ορίζεται η αριστερή πάνω γωνία. Το μέγεθος της εικόνας ως προς τον Χ άξονα(στήλες) ορίζεται ώς $ΌνομαΕικόνας.shape[1]$ και το μέγεθος της εικόνας ως προς τον Υ άξονα ως $ΌνομαΕικόνας.shape[0]$.

3.1 Φίλτρο Απομόνωσης Λευκών Λωρίδων

Χρησιμοποιώντας βασικές λειτουργίες κατωφλίου μέσα από συναρτήσεις της OpenCV βιβλιοθήκης επιτυγχάνεται ο εντοπισμός του άσπρου χρώματος της εικόνας που είναι οι λωρίδες του δρόμου. Αυτό γίνεται ορίζοντας ένα εύρος τιμών των pixels προς επεξεργασία. Πιο συγκεκριμένα δέχεται ως ορίσματα την εικόνα προς επεξεργασία ένα κάτω όριο χρωμάτων RGB καθώς και πάνω όριο. Αυτό το εύρος ορίζει πια pixels της εικόνας θα παραμείνουν και ποια pixels θα μηδενιστούν στην εικόνα. [9]

Στην Python ορίζεται ως:

$$\boldsymbol{cv2.inRange(image, lower, upper)}$$

Όπου:

- Image: Τα frames της εικόνας που έρχονται από τη κάμερα
- Lower: Κάτω όριο RGB επιλέγοντας τις τιμές, $[\ 0, 160,\ \ 0]$
- Upper: Άνω όριο RGB με τιμές, [255, 255, 255]

Άρα, η παραγόμενη τελική εικόνα είναι η:

$$\boldsymbol{WhiteFilter = cv2.inRange(image, lower, upper)}$$

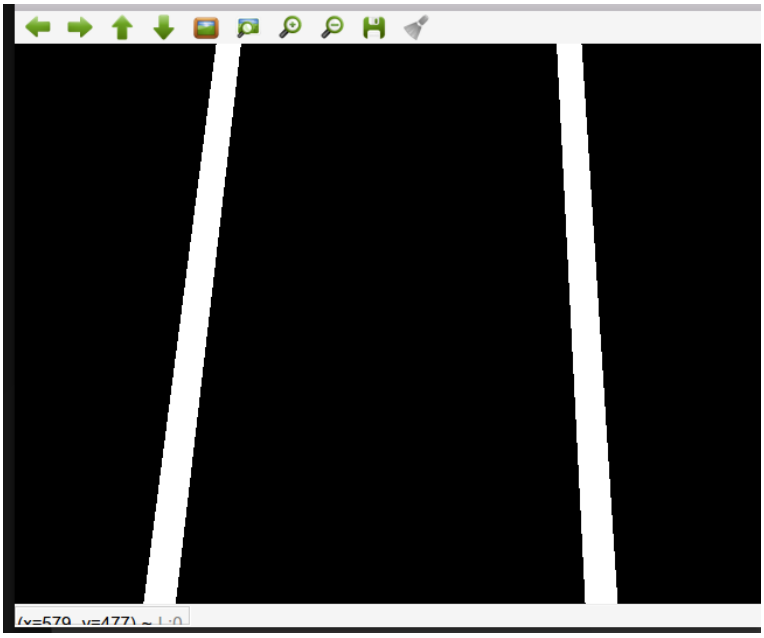


Figure 7 - Άσπρο Φίλτρο

3.2 Γκαουσιανό Φίλτρο

Το συγκεκριμένο φίλτρο χρησιμοποιείται με σκοπό την ομαλοποίηση εικόνων, δηλαδή την απλοποίηση τους όσον αφορά το βαθμό λεπτομέρειας και την αφαίρεση pixel τα οποία αποτελούν «θόρυβο». Είναι γνωστό ότι η κανονική – Γκαουσιανή κατανομή στον δισδιάστατο χώρο αναπαρίσταται από την εξής συνάρτηση:

$$\boldsymbol{G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\pi\sigma^2}}}$$

Από τιμές οι οποίες προέρχονται από την παραπάνω κατανομή δημιουργείται ένας δισδιάστατος πίνακας αριθμών ο οποίος χρησιμοποιείται για εφαρμογή συνελικτικών πράξεων επάνω στον πίνακα pixel της εικόνας. Ενώ το σ είναι μια παράμετρος η οποία καθορίζεται από τον χρήστη και καθορίζει την ένταση εξομάλυνσης του φίλτρου , οι διαστάσεις του πίνακα $N * N$ συνήθως καθορίζονται με βάση το σ. Ως συνήθης κανόνας, αν και όχι καθολικός, το N διαλέγεται ως τρείς φορές το ακέραιο μέρος της τυπικής απόκλισης δηλαδή του $\sqrt{\sigma}$. Ακολουθεί παράδειγμα Γκαουσιανού πίνακα $3 * 3$.

$$\begin{pmatrix} \frac{1}{2\pi\sigma^2}e^{-\frac{1^2+1^2}{2\pi\sigma^2}} & \frac{1}{2\pi\sigma^2}e^{-\frac{0^2+1^2}{2\pi\sigma^2}} & \frac{1}{2\pi\sigma^2}e^{-\frac{1^2+1^2}{2\pi\sigma^2}} \\ \frac{1}{2\pi\sigma^2}e^{-\frac{1^2+0^2}{2\pi\sigma^2}} & \frac{1}{2\pi\sigma^2}e^{-\frac{0^2+0^2}{2\pi\sigma^2}} & \frac{1}{2\pi\sigma^2}e^{-\frac{1^2+0^2}{2\pi\sigma^2}} \\ \frac{1}{2\pi\sigma^2}e^{-\frac{1^2+1^2}{2\pi\sigma^2}} & \frac{1}{2\pi\sigma^2}e^{-\frac{0^2+1^2}{2\pi\sigma^2}} & \frac{1}{2\pi\sigma^2}e^{-\frac{1^2+1^2}{2\pi\sigma^2}} \end{pmatrix}$$

Κατά την εφαρμογή του φίλτρου κάθε pixel το οποίο επεξεργάζεται μετατρέπεται σε ένα σταθμισμένο μέσο των γειτονικών του όπου τα βάρη είναι οι τιμές του Γκαουσιανού πίνακα. Το ίδιο το pixel έχει την μεγαλύτερη τιμή βάρους ενώ τα υπόλοιπα μικρότερη, αναλόγως της απόστασης τους από αυτό.

Ο Γκαουσιανός πίνακας είναι ισοτροπικός με αποτέλεσμα η συνελικτική διαδικασία να μπορεί να σπάσει σε δύο επιμέρους στους άξονες x και y μια ιδιότητα ιδιαίτερα χρήσιμη όσον αφορά την υπολογιστική πολυπλοκότητα. [10]

Η εφαρμογή του εν λόγω φίλτρου γίνεται με την OpenCV για τη γλώσσα Python χρησιμοποιώντας την συνάρτηση:

$$\boldsymbol{cv2.GaussianBlur(Img, (ksize, ksize), sigma)}$$

Όπου:

- Img: Η εικόνα προς επεξεργασία. Στην συγκεκριμένη περίπτωση είναι η εικόνα *WhiteFilter* του προηγούμενου υποκεφαλαίου
- Ksize: Οι διαστάσεις του πίνακα οι οποίες είναι (5,5)
- Sigma: Το νούμερο της διακύμανσης κατά άξονα. Επιλέγεται $sigma = 0$

Τελικά λαμβάνεται η εικόνα:

$$\boldsymbol{Blur = cv2.GaussianBlur(Img, (ksize, ksize), sigma)}$$

Figure 8 - Γκαουσιανό Φίλτρο

3.3 Φίλτρο Ακμών

Το φίλτρο ακμών έχει σκοπό τον μετασχηματισμό της εικόνας σε μια νέα, η οποία θα απαρτίζεται από τις ακμές ενός ή περισσοτέρων αντικειμένων τα οποία απεικονίζονται σε αυτή. Η συγκεκριμένη λειτουργία είναι ιδιαίτερα σημαντική εφόσον τα pixels που τελικά απομένουν, χρησιμοποιούνται για τον εντοπισμό των ευθείων γραμμών μέσω του πιθανοτικού μετασχηματισμού γραμμών(Hough Lines P). [11]

Ο αλγόριθμος ο οποίος χρησιμοποιείται, εφαρμόζεται σε ασπρόμαυρες εικόνες και εκτελείται εντός των ακολούθων πέντε σταδίων:

- Απαλοιφή θορύβου εικόνας (Noise Reduction): Εφαρμογή Γκαουσιανού φίλτρου όπως αυτό αναπτύχθηκε κατά την παράγραφο 3.2
- Εύρεση κλίσης ακμών (Intensity Gradient of the Image): Εν συνεχεία η εξομαλυμένη εικόνα υπόκειται σε μετασχηματισμό με φίλτρο Sobel. Από την εφαρμογή του στον σε κατακόρυφη και οριζόντια κατεύθυνση ανακλώνται η πρώτη παράγωγος στην οριζόντια κατεύθυνση (G_x) και η

πρώτη παράγωγος στην κάθετη κατεύθυνση (G_y). Από αυτές τις δύο εικόνες βρίσκεται έπειτα η gradient και η κατεύθυνση για κάθε pixel:

$$Edge\ Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Όπου τα φίλτρα Sobel είναι:

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{pmatrix}$$

- Non-Maximum Suppression. Αυτό το βήμα σκοπό έχει την λέπτυνση των ακμών. Στον πίνακα τον οποίο σχηματίζεται με βάση το προηγούμενο βήμα (Gradient Intensity matrix). Κάθε Pixel σε αυτόν το πίνακα ελέγχεται για την κλίση του και την ένταση του. Εάν σε κάποιο από τα pixel τα οποία επεξεργάζονται την δεδομένη στιγμή υπάρχει στην ίδια κατεύθυνση γειτονικό το οποίο παρουσιάζει μεγαλύτερη ένταση τότε είναι το μόνο το οποίο και κρατείται.

- Double Threshold: Σε αυτό το βήμα ορίζονται συγκεκριμένα όρια, άνω και κάτω, τα οποία βοηθούν στην κατηγοριοποίηση των Pixel. Άνω του ορίου pixel θεωρείται ότι συμβάλλουν στην εύρεση ακμών σε αντίθεση με αυτά κάτω του ορίου. Τα υπολειπόμενα, τα οποία βρίσκονται σε εύρος ανάμεσα των ορίων επεξεργάζονται περαιτέρω στο επόμενο βήμα.
- Εύρεση τελικώς των ακμών: Βάσει των αποτελεσμάτων του προηγούμενου βήματος σε κάθε Pixel το οποίο επεξεργάζεται εάν και μόνο αν υπάρχει ένα γειτονικό το οποίο είναι κατηγοριοποιημένο ως «δυνατό», δηλαδή συμβάλλει στην εύρεση ακμών, τότε και αυτό μετατρέπεται αντίστοιχα σε «δυνατό».

Η συνάρτηση που λαμβάνεται στη Python είναι:

$$\boldsymbol{cv2.Canny(Image, LowThreshold, HighThreshold)}$$

Όπου:

- Image: Η εικόνα που εισάγεται. Στη συγκεκριμένη περίπτωση είναι η Blur.
- Low_threshold: Κάτω όριο κατωφλίου με τιμή 50
- High_threshold: Άνω όριο κατωφλίου με τιμή 150

Παρακάτω λαμβάνεται η τελική εικόνα του φίλτρου ακμών:

$$\boldsymbol{Edges = cv2.Canny(Image, LowThreshold, HighThreshold}$$

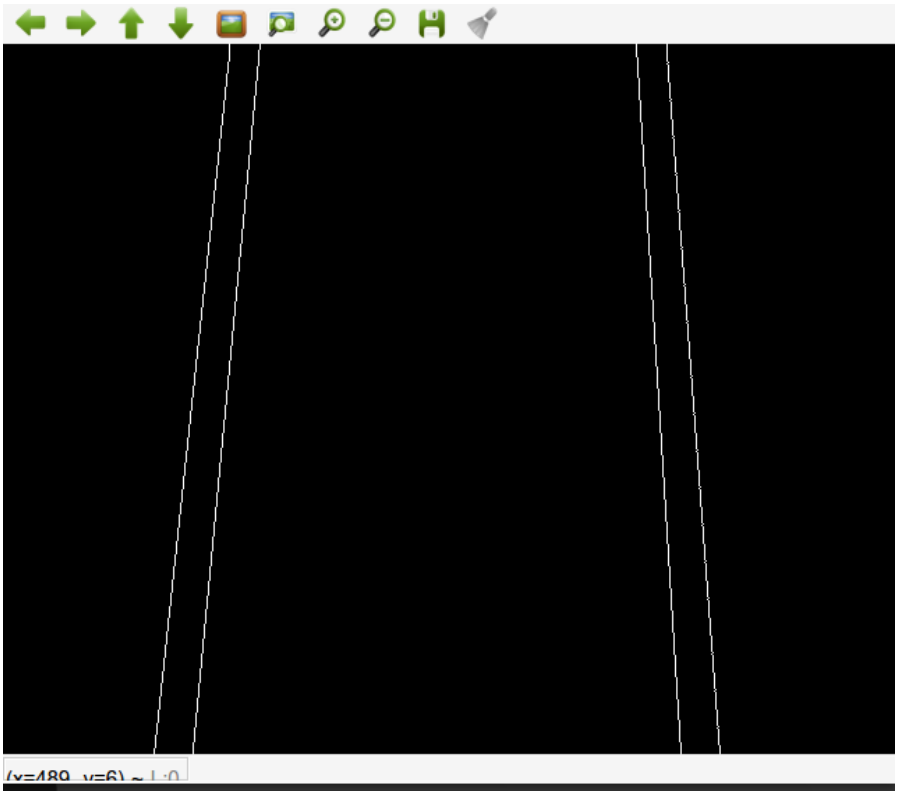


Figure 9 - Φίλτρο Ακμών

.

3.4 Φίλτρο Αλλαγής Προοπτικής

Σε αυτό το σημείο της εργασίας χρησιμοποιούνται δύο γεωμετρικοί μετασχηματισμοί εικόνας. Ο *GetPerspectiveTransform* και ο *WrapPerspective*. Αυτοί οι μετασχηματισμοί ανήκουν στους δύο διαστάσεων γεωμετρικούς μετασχηματισμούς. Δεν αλλάζουν το περιεχόμενο της εικόνας αλλάζουν όμως, τη θέση του pixel της εικόνας και το στέλνουν σε μία νέα εικόνα αποστολής.

- GetPerspectiveTransform

Υπολογίζει έναν μετασχηματισμό προοπτικής από 4 ζευγάρια των αντίστοιχων σημείων. Στη Python δέχεται δύο ορίσματα. Το source(src) και το destination(dst).

Στην Python ορίζεται ως:

$$\boldsymbol{Cv2.getPerspectiveTransform(src, dst)}$$

Src: Συντεταγμένες των κορυφών του τετράγωνου της περιοχής ενδιαφέροντος.

Dst: Συντεταγμένες των αντίστοιχων κορυφών τετραγώνου που θα σταλθούν τα σημεία του src στην εικόνα προορισμού. Στη περίπτωση των γραμμών που εστιάζει η εργασία, πρέπει να απομονωθεί ένα μέρος των γραμμών από το τετράγωνο που αναφέρθηκε. Πιο συγκεκριμένα:

$$offset1 = edges.shape[1]/2\,,$$

$$Ύψος\ Πάνω\ Αριστερής\ και\ Δεξιάς\ Γωνίας\ Τετραγώνου = HeightTop = 275,$$

$$Ύψος\ Κάτω\ Αριστερής\ και\ Δεξιάς\ Γωνίας\ Τετραγώνου = HeightBot = edges.shape[0]$$

$$Μήκος\ Πάνω\ Σημείων\ =\ 228$$

$$Μήκος\ Κάτω\ Σημείων\ =\ 228$$

Τελικά το τετράγωνο ενδιαφέροντος που λαμβάνεται έχει σημεία τα:

$$\alpha = (offset1 - Μήκος\ Πάνω\ Σημείων\ , HeightTop)$$

$$\beta = (offset1 - Μήκος\ Κάτω\ Σημείων, edges.shape[0])$$

$$\gamma = (offset1 + Μήκος\ Κάτω\ Σημείων, edges.shape[0])$$

$$\delta = (offset1 + Μήκος\ Πάνω\ Σημείων, HeightTop)$$

Στη παρακάτω εικόνα φαίνεται το τετράγωνο ενδιαφέροντος:

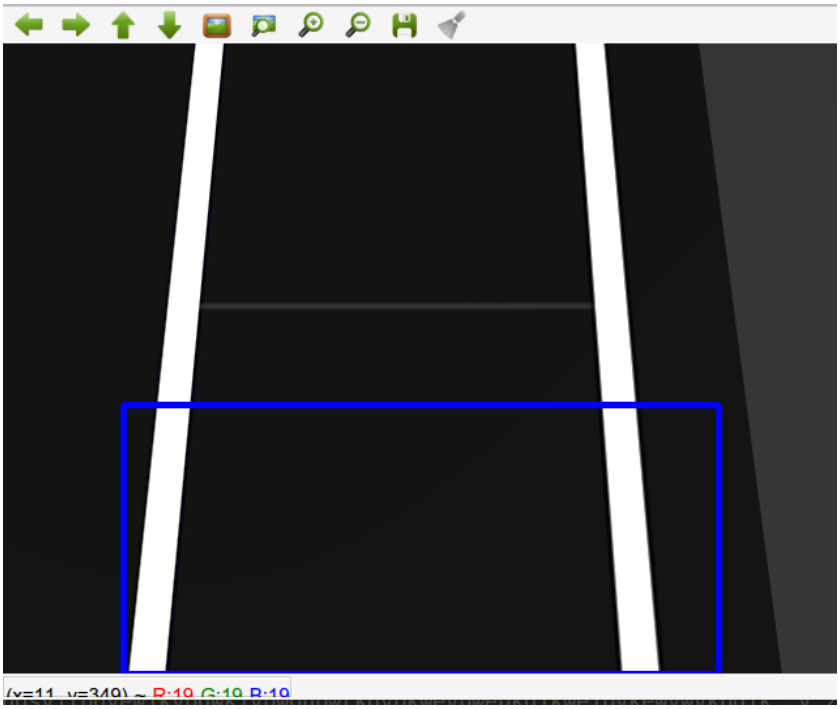


Figure 10 - Τετράγωνο Ενδιαφέροντος

$$\boldsymbol{Src} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta})$$

$$offset = 210\,,$$

$$(Ύψος\ Πάνω\ Αριστερής\ και\ Δεξιάς\ Γωνίας\ Τετραγώνου)' = (HeightTop)' = 0,$$

$$(Ύψος\ Κάτω\ Αριστερής\ και\ Δεξιάς\ Γωνίας\ Τετραγώνου)' = HeightBot = edges.shape[0]$$

$$Μήκος\ Πάνω\ Σημείων\ =\ edges.shape[1]/2$$

$$Μήκος\ Κάτω\ Σημείων\ =\ edges.shape[1]/2$$

$$\alpha' = (Μήκος\ Πάνω\ Σημείων - offset\,, HeightTop)$$

$$\beta' = (Μήκος\ Κάτω\ Σημείων - offset, HeightBot)$$

$$\gamma' = (offset + Μήκος\ Κάτω\ Σημείων, HeightBot)$$

$$\delta' = (offset + Μήκος\ Πάνω\ Σημείων, HeightTop)$$

$$\boldsymbol{Dst} = (\boldsymbol{\alpha}', \boldsymbol{\beta}', \boldsymbol{\gamma}', \boldsymbol{\delta}')$$

$$\boldsymbol{M = cv2.getPerspectiveTransform(src, dst)}$$

Στην ουσία η συνάρτηση υπολογίζει έναν $3x3$ πίνακα του μετασχηματισμού προοπτικής ώστε να ισχύει:

$$\begin{bmatrix} t_i x_i' \\ t_i y_i' \\ t_i \end{bmatrix} = map_{matrix} * \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\text{Όπου: } dst(i) = (x_i', y_i'), src(i) = (x_i, y_i), i = 0,1,2,3$$

Αφού, τώρα υπολογίστηκε το M μητρώο η διαδικασία συνεχίζεται με τον *WrapPerspective* μετασχηματισμό.

- *WrapPerspective* [12]

Εφαρμόζει έναν μετασχηματισμό προοπτικής στην εικόνα.

Στη Python ορίζεται ως*:*

$$\boldsymbol{Cv2.WarpPerspective(src, M, (dsize)}$$

Src: Εικόνα εισόδου. Αυτή η εικόνα στην συγκεκριμένη περίπτωση είναι η η έξοδος του φίλτρου των ακμών. Δηλαδή η εικόνα $Edges$.

M: Το υπολογισμένο μητρώο $\boldsymbol{M = cv2.getPerspectiveTransform(src, dst)}$**,** της προηγούμενης παραγράφου.

$dsize:$ Μέγεθος της εικόνας εξόδου. Αποτελείται από τα (maxWidth, maxHeight). Το maxWidth και το maxHeight πρέπει να είναι ίδια με της εικόνας Src. Δηλαδή:

$$max_{width} = edges.shape[1]$$

$$max_{height} = edges.shape[0]$$

Ως έξοδος αυτού του μετασχηματισμού λαμβάνεται η εικόνα $Dst(x, y)$. Η μετατροπή γίνεται σύμφωνα με τον παρακάτω υπολογισμό:

$$Dst(x, y) = Src\left(\frac{M_{11} * x + M_{12} * y + M_{13}}{M_{31} * x + M_{32} * y + M_{33}}, \frac{M_{21} * x + M_{22} * y + M_{23}}{M_{31} * x + M_{32} * y + M_{33}}\right)$$

Παρακάτω φαίνεται η εικόνα dst δρόμου υπό ευθεία και υπό στροφή μετά τον μετασχηματισμό.

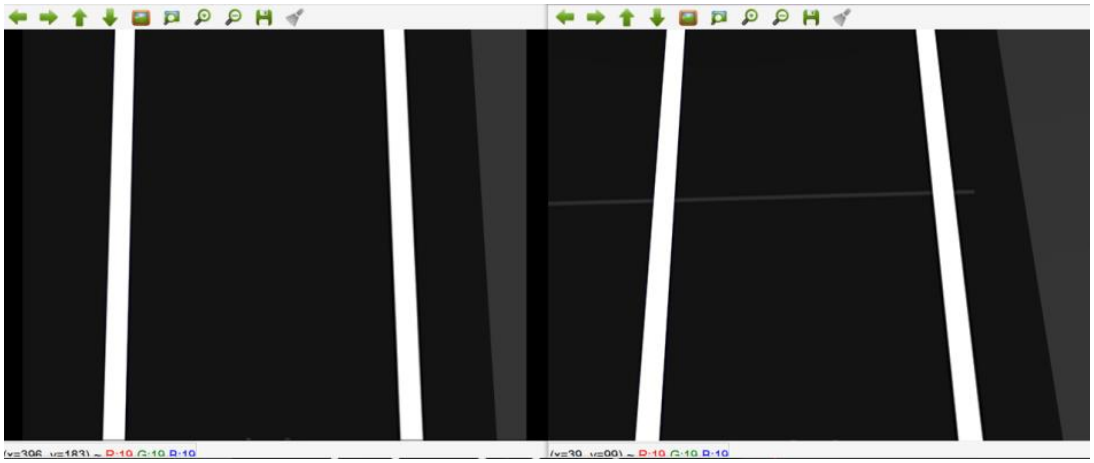


Figure 11 - Απόκριση Γεωμετρικού Μετασχηματισμού σε Ευθεία

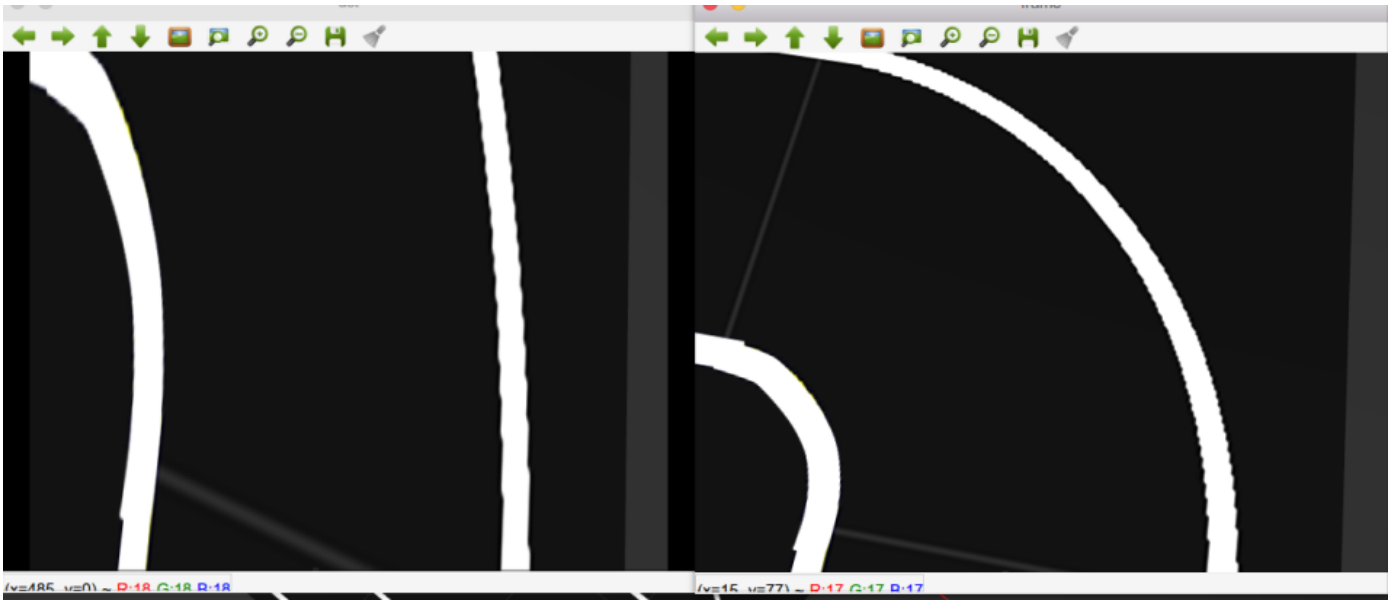


Figure 12 - - Απόκριση Γεωμετρικού Μετασχηματισμού σε Ευθεία

Στη παρακάτω εικόνα αναπαρίστανται συνολικά τα προηγούμενα βήματα που αναφέρθηκαν:

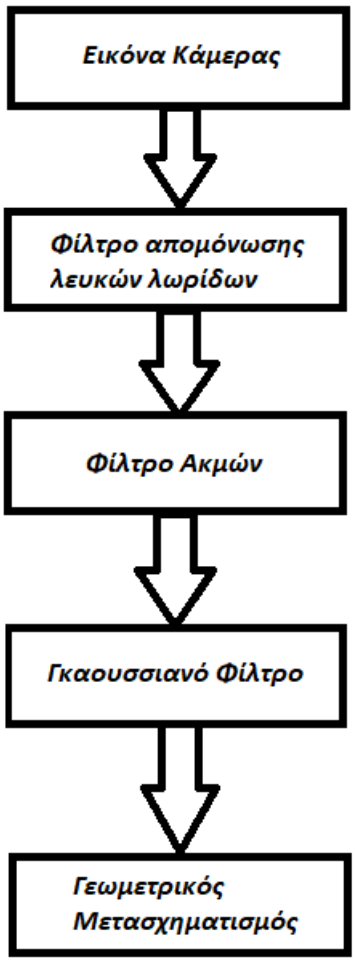


Figure 13 - Διάγραμμα Διαδικασίας Προεπεξεργασίας Εικόνας

Κεφάλαιο 4 - Αλγόριθμος Εύρεσης Λωρίδων και Υπολογισμός Γωνίας Στρέψης

Οποιαδήποτε επεξεργασία για να συμβεί στις γραμμές που έχουν απομονωθεί από τα προηγούμενα βήματα, κρίνεται απαραίτητο να βρεθούν τα σημεία που αντιπροσωπεύουν αυτές οι γραμμές. Ορίζοντας 2 σημεία, μπορεί να αναπαρασταθεί μία και μόνο γραμμή. Κεντρική ιδέα είναι να βρεθούν δύο βασικές γραμμές, που η μία να αντιπροσωπεύει το πλήθος των αριστερών γραμμών και η άλλη το πλήθος των δεξιών γραμμών. Από αυτές τις δύο γραμμές, παίρνοντας τον μέσο όρο των σημείων των αριστερών και δεξιών, δημιουργείται μία γραμμή κατεύθυνσης. Όταν το όχημα καλείται να στρίψει λόγω μίας δεξιάς ή αριστερής στροφής η γραμμή κατεύθυνσης δημιουργεί γωνία φ με τον X άξονα. Αυτό συμβαίνει επίσης και όταν, το όχημα σε έναν ευθύ δρόμο είναι στραμμένο προς τα αριστερά η δεξιά. Έτσι λοιπόν, για την εύρεση της γωνίας φ δημιουργείται επίσης και μία κεντρική κάθετη σταθερή γραμμή στο κέντρο του παραθύρου. Όταν η γραμμή κατεύθυνσης συμπίπτει πάνω στην κεντρική γραμμή σημαίνει ότι το όχημα βρίσκεται στη σωστή θέση στο δρόμο, αφού η απόσταση των σημείων της γραμμής κατεύθυνσης και της κεντρικής γραμμής, ελαχιστοποιείται . Αυτό συμβαίνει διότι, η κεντρική γραμμή διέρχεται από τον X άξονα τοπικών συντεταγμένων του οχήματος και το κέντρο μάζας του, άρα δηλώνει και την πορεία του αυτοκινήτου τη κάθε στιγμή. Οποιαδήποτε γωνία σχηματίζεται μεταξύ της κεντρικής γραμμής και της γραμμής κατεύθυνσης πρέπει να διορθωθεί και οι κινητήρες να δημιουργήσουν μία τέτοια ροπή στρέψης ώστε να περιστραφεί το όχημα. Μηδενική γωνία στρέψης σημαίνει και μηδενικό σφάλμα, άρα ότι το όχημα βρίσκεται στη σωστή πορεία. Όμως, αυτό είναι μερικώς αληθές, ανάλογα το που ζητείται να βρίσκεται το όχημα.

Όταν η θέση του οχήματος δεν βρίσκεται στο κέντρο του δρόμου αλλά μετατοπισμένο μόνο πιο δεξιά ή πιο αριστερά η γωνία, φ που σχηματίζεται είναι πάλι $\varphi = 0°$. Άρα, σε αυτή τη περίπτωση μηδενικού σφάλματος γωνίας πρέπει να ληφθεί υπόψιν και η απόσταση μεταξύ της κεντρικής γραμμής και της γραμμής κατεύθυνσης του X άξονα. Επειδή ο ελεγκτής που θα περιγραφεί στο Κεφάλαιο 5, λαμβάνει ως είσοδο τη γωνία στρέψης με $\varphi = 0°$ ο ελεγκτής θεωρεί μηδενικό σφάλμα, άρα οι κινητήρες στρέφονται με σταθερές και ίσες ροπές. Για τιμές πολύ κοντά στο 0 δηλαδή, $\varphi \approx 0°$ οι οποίες αντιπροσωπεύουν και τη πραγματικότητα, ο ελεγκτής δεν είναι τόσο άμεσος, επειδή θεωρείται σχεδόν μηδενικό σφάλμα στους υπολογισμούς λόγω της χρήσης τριγωνομετρικών εξισώσεων. Για την επίλυση αυτού του προβλήματος δημιουργείται μία πολύ μικρή τεχνητή γωνία στρέψης, μεγαλύτερη από τη γωνία που υπολογίζεται αλλά και τόσο μικρή ώστε να διασφαλίζεται η ομαλή απόκριση του οχήματος από απότομες κινήσεις περιστροφής και να οδηγείται το σύστημα στην αστάθεια ή σε ανεπιθύμητες ταλαντώσεις γύρω από την $\varphi \approx 0°$. Αυτή η γωνία προφανώς δεν είναι πραγματική, αλλά λύνει το συγκεκριμένο πρόβλημα, με σχετικά απλό τρόπο, ώστε ο ελεγκτής να μπορεί να διορθώσει μία μικρή τιμή γωνίας για να στραφεί το όχημα.

Κάθε γραμμή σε μια εικόνα αποτελείται από ένα ζεύγος σημείων ώστε να είναι μοναδική. Η μορφή της είναι:

$$y = ax + c$$

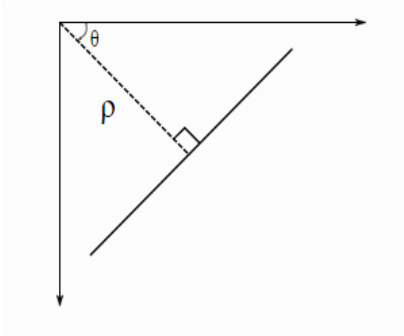
Όπου (x, y) κάθε σημείο της ευθείας.

Η παραμετρική μορφή της παραπάνω εξίσωσης είναι:

$$\rho = xcos\theta + ysin\theta$$

Όπου ρ είναι η κάθετη απόσταση που σχηματίζεται από την αρχή των αξόνων έως την γραμμή και θ η γωνία που σχηματίζεται από την κάθετη ρ γραμμή και τον οριζόντιο άξονα. Γίνεται χρήση της παραμετρικής μορφής διότι η κλίση της πρώτης μορφής, όταν οι ευθείες είναι κάθετες, απειρίζεται. Κάθε ζεύγος σημείων (x_i, y_i) μετατρέπεται σε ημιτονοειδείς καμπύλες στο χώρο $Hough\ \theta - \rho$ ως:

$$\rho = x_i cos\theta + y_i sin\theta$$



Η απόσταση ρ έχει θετικό πρόσημο και $\theta < 180°$ όταν η γραμμή βρίσκεται κάτω από την αρχή των αξόνων $(0,0)$ στη πάνω αριστερή γωνία. Αν βρεθεί η γραμμή πάνω από την αρχή των αξόνων η θ γωνία λαμβάνεται μικρότερη των $180°$ με $\rho < 0$. Κάθε κάθετη γραμμή έχει $\theta = 0°$ και κάθε οριζόντια γραμμή έχει $\theta = 90°$. [13]

Ο Hough μετασχηματισμός λειτουργεί ως εξής:

Κάθε γραμμή, όπως προαναφέρθηκε παραπάνω από τη παραμετρική μορφή της ευθείας μπορεί να γραφεί ως ζεύγος, (ρ, θ). Πρώτα αρχικοποιείται ένας πίνακας συσσώρευσης δύο διαστάσεων με τη τιμή 0 για τις τιμές του, (ρ, θ), όπου ρ είναι οι γραμμές του πίνακα και θ οι στήλες του. Η ακρίβεια του των υπολογισμών εξαρτάται από το μέγεθος του πίνακα. Αν η ακρίβεια ζητείται να είναι 1 μοίρα τότε το θ γεμίζει με 181 τιμές, από $0,1,2 \ldots, 180$. Για την ρ μεταβλητή η μέγιστη απόσταση που υπάρχει είναι η διαγώνιος απόσταση της εικόνας, παίρνοντας 1 pixel ακρίβεια.

Έστω παράθυρο εικόνας $100x100$ με οριζόντια γραμμή στο μέσο του παραθύρου. Τα σημεία (x_0, y_0) είναι γνωστά από το φίλτρο των ακμών.

Στην εξίσωση $\rho = x_i cos\theta + y_i sin\theta$ γίνεται αντικατάσταση το πρώτο σημείο (x_0, y_0). Μετά την αντικατάσταση για κάθε $\theta = 0,1,2, \ldots 180$ λαμβάνεται το αντίστοιχο ρ. Σε κάθε κελί του συσσωρευτή αυξάνεται η τιμή κατά 1 για κάθε (ρ, θ) που βρέθηκε. Έτσι στο κελί $(50,90)$ έχει αποθηκευτεί η τιμή 1.

Η διαδικασία συνεχίζεται και επαναλαμβάνεται για το δεύτερο σημείο. Για κάθε (θ, ρ) η τιμή του συσσωρευτεί αυξάνεται κατά 1. Αυτή τη φορά το κελί $(50,90)$ αυξήθηκε πάλι κατά 1. Συνολικά η τιμή

του κελιού τώρα μαζί με την προηγούμενη επανάληψη έγινε 2. Η διαδικασία συνεχίζεται για όλα τα σημεία. Έτσι στο τέλος της διαδικασίας το σημείο $(50,90)$ θα έχει το μέγιστο αριθμό. Αυτό σημαίνει ότι υπάρχει γραμμή με μήκος 50 και γωνία 90°.

Ολοκληρώνοντας, όπως φάνηκε από το παράδειγμα πολλά σημεία μίας γραμμής αντιστοιχούν σε πολλαπλά ίδια σημεία στο Hough επίπεδο (ρ, θ). Έτσι όταν υπάρχουν σημεία τα οποία ανήκουν σε μία γραμμή ανιχνεύονται. Όσα πιο πολλά σημεία υπάρχουν στην ίδια γραμμή τόσο μεγαλύτερο είναι και το μέγιστο άθροισμα των αντίστοιχων κελιών του συσσωρευτή.

Η συνάρτηση η οποία εντοπίζει τις γραμμές είναι η HoughlinesP. Ως έξοδο δίνει τα σημεία ή αλλιώς τις συντεταγμένες των pixel στην οθόνη του υπολογιστή. Από όλα αυτά τα σημεία που δίνονται ως έξοδος της συνάρτησης αυτής, πρέπει να φτιαχτούν δύο ευθείες που να αντιπροσωπεύουν το σύνολο των αριστερών και των δεξιών γραμμών.

Έστω ο παρακάτω πίνακας ως αποτέλεσμα της συνάρτησης που προαναφέρθηκε:

$$[\,\boldsymbol{X_1}\,,\ \boldsymbol{Y_1}\,,\ \boldsymbol{X_2}\,,\boldsymbol{Y_2}\,]$$

Αυτό το ζεύγος σημείων αναπαριστά γραφικά οποιαδήποτε ευθεία που βρίσκεται κατά την εκτέλεση της συνάρτησης.

Στην εικόνα φαίνονται οι γραμμές που βρέθηκαν από τα παραπάνω σημεία. Αυτές δεν συμπίπτουν με τις άσπρες του δρόμου γιατί ο υπολογιστής διαβάζει τις γραμμές της αριστερής πλευράς που έχουν μετατοπιστεί λόγω του γεωμετρικού μετασχηματισμού.

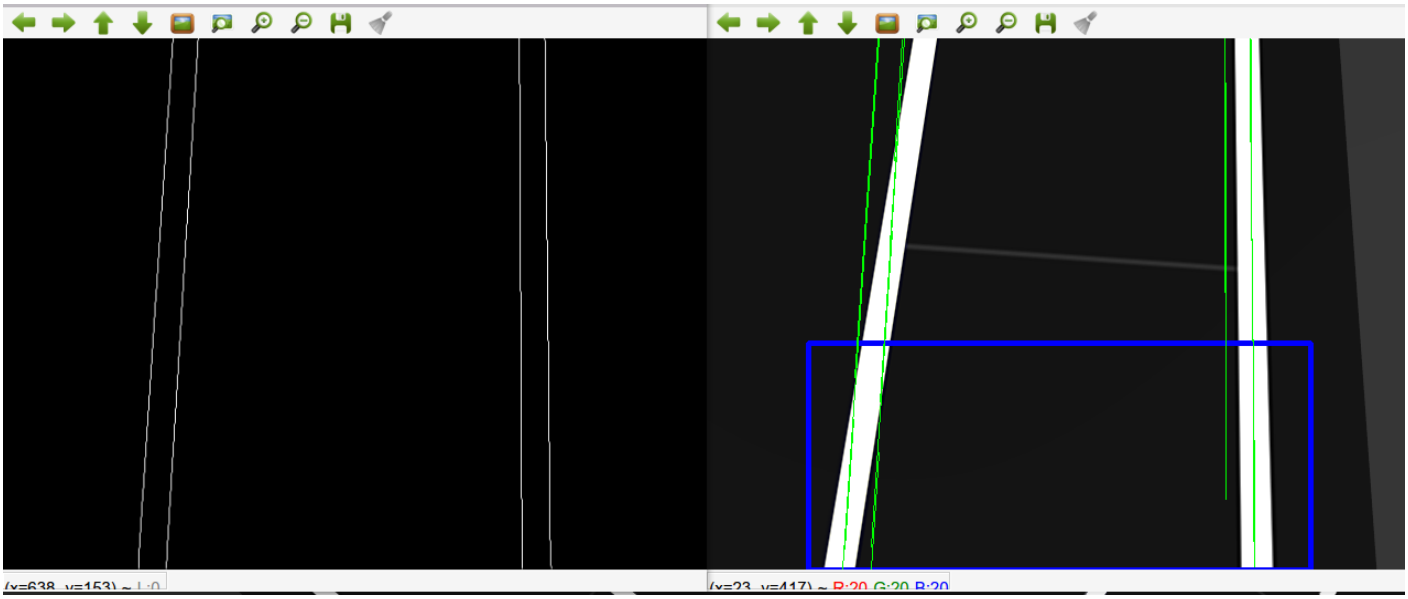


Figure 14 - Εντοπισμός Γραμμών

Στην παρακάτω εικόνα φαίνονται οι γραμμές που εντοπίζονται με το κόκκινο χρώμα χωρίς την εφαρμογή του γεωμετρικού μετασχηματισμού.

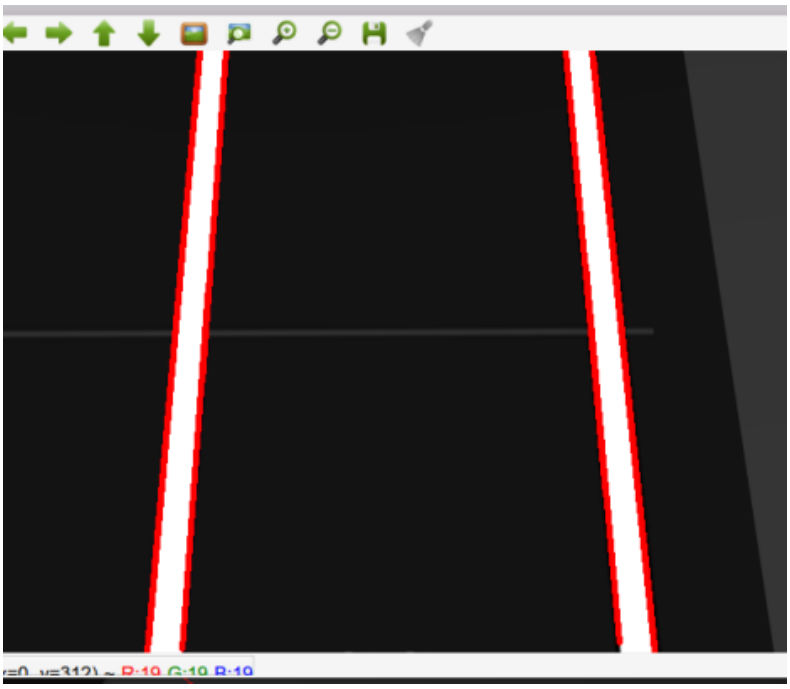


Figure 15 - Εντοπισμός Γραμμών

Η κλίση της κάθε ευθείας αυτού του ζεύγους είναι:

$$\boldsymbol{Κλίση} = \frac{\boldsymbol{Y_2} - \boldsymbol{Y_1}}{\boldsymbol{X_2} - \boldsymbol{X_1}}$$

Ο επόμενος στόχος είναι να βρεθεί η προέκταση της ευθείας σε ποιο σημείο τέμνει τον Χ άξονα.

Έστω ότι μία $ευθεία(\varepsilon)$ αναπαρίσταται από μία συνάρτηση $f(x)$. Τότε η κλίση μίας γραμμικής συνάρτησης $f(x)$ είναι:

$$\boldsymbol{s} = \boldsymbol{Κλίση_{f(x)}} = \frac{\boldsymbol{f(y_2) - f(y_1)}}{\boldsymbol{x_2} - \boldsymbol{x_1}} \rightarrow \boldsymbol{s} * (\boldsymbol{x_2} - \boldsymbol{x_1}) = \boldsymbol{f(y_2) - f(y_1)} \rightarrow$$

$$\boldsymbol{s} * \boldsymbol{x_2} - \boldsymbol{s} * \boldsymbol{x_1} = \boldsymbol{f(y_2) - f(y_1)} \rightarrow \boldsymbol{x_2} = \frac{\boldsymbol{f(y_2) - f(y_1)}}{\boldsymbol{s}} + \boldsymbol{x_1}$$

Επειδή ζητείται το σημείο τομής της ευθείας με τον X άξονα, αυτό είναι το σημείο $(X, Shape[0])$.

Άρα, $\boldsymbol{X = x_2 \xrightarrow{f(y_1)=\mathbf{Shape[0]}} X = \frac{f(y_2) - \mathbf{Shape[0]}}{s} + x_1}$

Το μήκος της κάθε ευθείας είναι ορίζεται ως:

$$\boldsymbol{length = \sqrt{(Y_2 - Y_1)^2 + (X_2 - X_1)^2}}$$

Το επόμενο βήμα που ακολουθείται στον αλγόριθμο για την κατηγοριοποίηση των αριστερών και των δεξιών ευθειών, που σημαίνει αυτόματα την ανάγκη για δημιουργία μίας κύριας αριστερής και μίας κύριας δεξιάς ευθείας από το σύνολο των σημείων που είναι γνωστά, είναι το εξής:

Αρχικά δημιουργούνται δύο πίνακες, ο $Left\_lines$ και ο $Right\_lines$.

- 1η Περίπτωση

  Αν το σημείο X που έχει βρεθεί είναι μικρότερο της απόστασης $\frac{\text{Shape[1]}}{2}$, $X < \frac{\text{Shape[1]}}{2}$,τότε δημιουργείται ένας πίνακας $Left\_lines(X, s) \in \mathbb{R}1x2$ με αυτά τα X και τις αντίστοιχες κλίσεις και ένας πίνακας $Left\_weights(length) \in \mathbb{R}1x1$ με τα αντίστοιχα μήκη.

- 2η Περίπτωση

  Αν το σημείο X που έχει βρεθεί είναι μεγαλύτερο της απόστασης $\frac{\text{Shape[1]}}{2}$, $X > \frac{\text{Shape[1]}}{2}$, τότε δημιουργείται ένας πίνακας $Right\_lines(X, s) \in \mathbb{R}1x2$ με αυτά τα X και τις αντίστοιχες κλίσεις και ένας πίνακας $Right\_weights(length) \in \mathbb{R}1x1$ με τα αντίστοιχα μήκη.

Η κύρια Κλίση και σημείο τομής της αριστερή γραμμή προκύπτει ως πίνακας στοιχείων A:

$$\boldsymbol{A = \frac{Left\_lines(X, s) * Left\_weights(length)}{\sum((Left\_weights(length))}} \in \mathbb{R}^{1x2}$$

Η νέα Κλίση και το νέο σημείο τομής που υπολογίζονται από τον πίνακα A είναι η $Slope_1$ και X_1 αντίστοιχα.

$$\boldsymbol{A = (S_1, X_1)}$$

Η κύρια Κλίση και σημείο τομής της δεξιάς γραμμής προκύπτει ως ο πίνακας στοιχείων B:

$$\boldsymbol{B} = \frac{\boldsymbol{Right\_lines(X,s) * Right\_weights(length)}}{\sum((\boldsymbol{Right\_weights(length)})} \in \mathbb{R}^{\mathbf{1x2}}$$

Η νέα Κλίση και το νέο σημείο τομής που υπολογίζονται από τον πίνακα B είναι η $Slope2$ $και$ $X2$ αντίστοιχα.

$$\boldsymbol{B} = (\boldsymbol{S_2}, \boldsymbol{X_2})$$

Η γενική μορφή εξίσωσης, μίας γραμμικής συνάρτησης $f(x)$, η οποία διέρχεται από δύο σημεία είναι:

$$f(\boldsymbol{x}) - \boldsymbol{y1} = \boldsymbol{\lambda} * (\boldsymbol{x} - \boldsymbol{x1})$$

Όπου $\boldsymbol{\lambda} = \frac{\boldsymbol{y2-y1}}{\boldsymbol{x2-x1}}$

Η αριστερή και δεξιά κύριες ευθείες, αποτελούνται από δύο σημεία αντίστοιχα. Με βάση πλέον, τους παραπάνω υπολογισμούς μπορούν να γραφτούν τα παρακάτω ζεύγη σημείων.

- Τα σημεία $Point0, Point1$ ανήκουν στην αριστερή ευθεία.
- Τα σημεία $Point2, Point3$ ανήκουν στην δεξιά ευθεία.

Το πρώτο σημείο $Point0$, ορίζεται ως:

$$\boldsymbol{Point0} = (\boldsymbol{X_1}, \boldsymbol{Shape}[\mathbf{0}])$$

Για την εύρεση του σημείου $Point1$, όπως αποδείχτηκε από τις προηγούμενες σχέσεις, ισχύει ότι:

$$\boldsymbol{X} = \frac{\boldsymbol{f(y_2)} - \mathbf{Shape}[\mathbf{0}]}{\boldsymbol{s}} + \boldsymbol{x_1}$$

Με κατάλληλη αντικατάσταση, με τη γνώση των παρόντων δεδομένων η σχέση γίνεται:

$$\boldsymbol{X_1} = \frac{\mathbf{Shape}[\mathbf{0}] - \boldsymbol{f(y_2)}}{\boldsymbol{S_1}} + \boldsymbol{x_2}$$

Η αλλαγή των προσήμων του αριθμητή $(Shape[0] - f(y\_2))$ δικαιολογείται από το διαφορετικό σημείο αρχής μέτρησης$(0,0)$ των αξόνων. Η αρχή των αξόνων για την οθόνη του υπολογιστή ξεκινάει από την πάνω αριστερή γωνία. Άρα, το $Shape[0]$ πάντα είναι η μέγιστη απόσταση του παραθύρου που ορίζεται από την αρχή των αξόνων ως προς το ύψος ή αλλιώς ως προς τον Y άξονα.

Επίσης η τιμή, $f(y_2)$, μπορεί να οριστεί και ως:

$$\boldsymbol{f(y_2) = HeightR * Shape[0]}$$

Όπου, $\mathbf{HeightR = a, a \in \mathbb{R}}$ $\boldsymbol{και}$ $\boldsymbol{0 < a \leq 1}$

Με αντικατάσταση της παραπάνω σχέσης στην, παραπάνω σχέση, γίνεται

$$\boldsymbol{X_1 = \frac{Shape[0] - HeightR * Shape[0]}{S_1} + x_2}$$

Λύνοντας τελικά τη σχέση ως προς x_2, καταλήγει να παίρνει την εξής μορφή:

$$\boldsymbol{x_2 = \frac{Shape[0] * (HeightR - 1)}{S_1} + X_1}$$

Άρα, το σημείο, $Point1$, είναι:

$$\boldsymbol{Point1 = (x_2, HeightR * Shape[0])}$$

Το τρίτο σημείο, $Point2$, ορίζεται ως:

$$\boldsymbol{Point2 = (X_2, Shape[0])}$$

Για την εύρεση του σημείου $Point3$, ακολουθείται η ίδια λογική του $Point1$, όπως αποδείχτηκε παραπάνω.
Η τιμή του, x_3, ισούται με:

$$\boldsymbol{x_3 = \frac{Shape[0] * (HeightR - 1)}{S_2} + X_2}$$

Άρα, το σημείο $Point3$, ορίζεται ως:

$$\boldsymbol{Point3 = (x_3, HeightR * Shape[0])}$$

Επόμενος στόχος αποτελεί να βρεθεί το σύνολο όλων των σημείων που διέρχονται από την κάθε ευθεία, έτσι ώστε να παρθεί ο μέσος όρος του κάθε αριστερού και του αντίστοιχου δεξιού σημείου. Αποτέλεσμα αυτού είναι η εύρεση της κεντρικής δυναμικής ευθείας, η οποία δηλώνει την επιθυμητή κατεύθυνση του οχήματος.

Κάθε ευθεία μπορεί να οριστεί ως:

$$\boldsymbol{y = a * x + b}$$

Στον αλγόριθμο διέρχονται πρακτικά άπειρα σημεία από την ευθεία. Αν στον αλγόριθμο χρησιμοποιούνται όλο αυτό το πλήθος σημείων, το σύστημα γίνεται πολύ ευαίσθητο στις μεταβολές, επειδή δέχεται πολύ πληροφορία. Ένας σχετικά μικρός αριθμός επιθυμητών σημείων κρίνεται αποδοτικός. Το βήμα δειγματοληψίας δεν βγαίνει πάντα ακέραιος αριθμός. Αυτό συμβαίνει επειδή οι συντεταγμένες των σημείων, που μεταφράζονται σε pixels, είναι πάντα ακέραιοι αριθμοί το πλήθος των επιθυμητών σημείων ως προς την απόσταση των σημείων μπορεί να βγει πραγματικός αριθμός. Είναι γνωστό ότι σε μία επαναληπτική δομή δεν επιτρέπεται δεκαδικό βήμα. Στην συνέχεια βρίσκεται το βήμα δειγματοληψίας με βάση τον επιθυμητό αριθμό στοιχείων.

$\boldsymbol{Επιθυμητός\ Αριθμός\ Σημείων = K, με\ K \in \mathbb{Z}\ και\ 2 \leq K \leq 4}$

Επίσης ο αριθμός των σημείων που εμφανίζονται στην αριστερή δεν είναι ο ίδιος με τη δεξιά. Όταν δεν είναι ο ίδιος ο μέσος όρος που θα βγει, θα περιέχει σημεία τα οποία δεν έχουν αντίστοιχα της απέναντι γραμμής, με αποτέλεσμα να εκτίθεται ο αλγόριθμος σε ανεπιθύμητα σφάλματα.

- Εύρεση βήματος δειγματοληψίας αριστερής γραμμής

Για τον καθορισμό του βήματος δειγματοληψίας, έτσι ώστε να εμφανίζονται στην οθόνη το πλήθος των επιθυμητών σημείων αρχικά υπολογίζεται η απόσταση που απέχουν οι συντεταγμένες στον X άξονα, των σημείων:

$$\boldsymbol{Point0 = (X_1, Shape[0])}$$

$$\boldsymbol{Point1 = (x_2, HeightR * Shape[0])}$$

$$\boldsymbol{r1 = x_2 - X_1}$$

Αφού ορίστηκε το K και το $r1$, το βήμα δειγματοληψίας της αριστερής γραμμής βρίσκεται ως:

$$\boldsymbol{step1 = \frac{r1}{K}}$$

$$\boldsymbol{step1 = \begin{cases} Αν\ r1\ είναι\ ακέραιο\ πολλαπλάσιο\ μεγαλύτερο\ ή\ ίσο\ του\ K, step1 \in \mathbb{Z} \\ Σε\ κάθε\ άλλη\ περίπτωση\ step1 \in \mathbb{R} \end{cases}}$$

Για αυτόν τον λόγο πρώτα εκτελείται η διαίρεση κανονικά, στρογγυλοποιείται ο αριθμός και έπειτα λαμβάνεται το ακέραιο μέρος του αριθμού.

Αν ο αριθμός $r1 < K$ τότε το πηλίκο της σχέσης καταλήγει να είναι 0, που σημαίνει ότι δεν θα παρθεί κανένα σημείο. Έτσι το $step1$ ορίζεται ως:

$$\boldsymbol{step1 = 1}$$

δηλαδή ο μικρότερος αριθμός βήματος που μπορεί να υπάρξει παίρνοντας όλα τα σημεία από το σύνολο, πλήθους $\boldsymbol{r1 = x_2 - X_1}$.

Το τελικό πλήθος στοιχείων της αριστερής γραμμής, αντί για K, εν τέλη είναι:

- Εύρεση βήματος δειγματοληψίας δεξιάς γραμμής

Η λογική που ακολουθείται είναι η ίδια με την αριστερή γραμμή. Τα σημεία της δεξιάς γραμμής είναι:

$$\boldsymbol{Point2} = (\boldsymbol{X_2}, \boldsymbol{Shape}[\mathbf{0}])$$

$$\boldsymbol{Point3} = (\boldsymbol{x_3}, \boldsymbol{HeightR} * \boldsymbol{Shape}[\mathbf{0}])$$

Η απόσταση μεταξύ του x_3 $και$ X_2 είναι:

$$\boldsymbol{r2} = \boldsymbol{X_2} - \boldsymbol{x_3}$$

Το βήμα δειγματοληψίας της δεξιάς γραμμής είναι:

$$\boldsymbol{step2} = \frac{\boldsymbol{r2}}{\boldsymbol{K}}$$

Αν $r2 < K$, τότε:

$$\boldsymbol{step2} = \mathbf{1}$$

και με τελικό αριθμό στοιχείων, αντί K,

$$\boldsymbol{Πλήθος\ στοιχείων} = \boldsymbol{\mu}$$

Έτσι πλέον από τις αποστάσεις $r1$ $και$ $r2$ δεν παίρνονται όλα τα ενδιάμεσα σημεία αλλά ένας κοντινός αριθμός στον K, ν , για την αριστερή γραμμή με βήμα $step1$ και μ για την δεξιά γραμμή με βήμα $step2$.

- Ο πίνακας των X συντεταγμένων που δημιουργείται για την αριστερή γραμμή, $EveryX_1$, έχει την ακόλουθη μορφή:

$$\boldsymbol{EveryX_1} = (\boldsymbol{x_2}, \ldots), \boldsymbol{EveryX_1} \in \mathbf{Z}^{\mathbf{1xv}}$$

Από την γενική εξίσωση ευθείας της παραπάνω σχέσης, πρέπει να υπολογιστεί το b, ώστε στη συνέχεια να οι θέσεις των συντεταγμένων στον Y άξονα που αντιστοιχούν στον πίνακα $EveryX_1$.

$$\boldsymbol{b} = \boldsymbol{y} - \boldsymbol{a} * \boldsymbol{x}$$

Για την αριστερή γραμμή κάνοντας αντικατάσταση τις συντεταγμένες του $Point0 = (X_1, Shape[0])$, η σχέση γίνεται:

$$\boldsymbol{b1} = \boldsymbol{Shape}[\mathbf{0}] - \boldsymbol{S_1} * \boldsymbol{X_1}$$

Συνεχίζοντας, οι αντίστοιχες Y συντεταγμένες του, $EveryX_1$, είναι ο πίνακας $EveryY_1$.

Αφού έχει βρεθεί το $b1$ ο πίνακας $EveryY_1$ υπολογίζεται μέσα από τη προηγούμενη σχέση λύνοντας ως προς, y. Με αντικατάσταση του, y, με τον πίνακα $EveryY_1$, νέος πίνακας ορίζεται ως:

$$\boldsymbol{EveryY_1 = (\,EveryX_1 * S_1 + b1\,), EveryY_1 \in \mathbf{Z^{1xv}}}$$

- Ακολουθώντας την ίδια μεθοδολογία για τη δεξιά γραμμή, ο πίνακας των Χ συντεταγμένων που δημιουργείται για την δεξιά γραμμή, $EveryX_2$, έχει την ακόλουθη μορφή:

$$\boldsymbol{EveryX_2 = (X_2, \dots), EveryX_2 \in \mathbf{Z^{1x\mu}}}$$

Για την εύρεση του $b2$ γίνεται αντικατάσταση του σημείου, $Point2 = (\,X_2, Shape[0])$, στη σχέση παραπάνω, όπως δείχτηκε, καταλήγοντας με:

$$\boldsymbol{b2 = Shape[0] - S_2 * X_2}$$

Ο πίνακας $EveryY_2$ ο οποίος περιέχει τις συντεταγμένες για τα σημεία στον Υ άξονα προκύπτει ως:

$$\boldsymbol{EveryY_2 = (\,EveryX_2 * S_2 + b2\,), EveryY_2 \in \mathbf{Z^{1x\mu}}}$$

Το πλήθος των στοιχείων των πινάκων $EveryX_1$ και $EveryX_2$ με τους αντίστοιχους πίνακες $EveryY_1$ και $EveryY_2$ δεν είναι πάντα το ίδιο, δηλαδή $\mu \neq \nu$. Με τον τρόπο που ακολουθήθηκε το μ, ν καταλήγουν να είναι πολύ κοντινοί αριθμοί με αποτέλεσμα, αν αφαιρεθούν τόσα στοιχεία από τον ένα πίνακα, ώστε να καταλήξει στον πλήθος στοιχείων του άλλου να χάνεται ελάχιστη πληροφορία αλλά από την άλλη το πρόγραμμα να μην οδηγείται σε σφάλματα. Αυτό επιτυγχάνεται με τον παρακάτω τρόπο. Υπάρχουν δύο περιπτώσεις:

- Αν $\mu > \nu$:

$$\boldsymbol{diff = \mu - \nu}$$

Δηλαδή αν ο πίνακας $EveryX_2$, πλήθους μ στοιχείων έχει παραπάνω , $diff$ στοιχεία από το πλήθος των στοιχείου του $EveryX_1$, πλήθους, ν, τότε το πλήθος του πίνακα αυτού, γίνεται $\mu - diff$, με:

$$\boldsymbol{\nu' = \mu - diff}$$

Άρα ο πίνακας $EveryX_2$ ορίζεται όπως ήταν πριν, αλλά με πλήθος $diff$ λιγότερων στοιχείων.

$$\boldsymbol{EveryX_2 = (X_2, \dots), EveryX_2 \in \mathbf{Z^{1x\nu'}}}$$

- Αν $\mu < \nu$:

$$\boldsymbol{diff = \nu - \mu}$$

Ο πίνακας $EveryX_1$, πλήθους ν στοιχείων έχει παραπάνω , $diff$ στοιχεία από το πλήθος των στοιχείου του $EveryX_2$, πλήθους, μ, τότε το πλήθος του πίνακα αυτού, γίνεται $\nu - diff$, με:

$$\boldsymbol{\mu' = \nu - diff}$$

Άρα ο πίνακας $EveryX_1$ ορίζεται όπως ήταν πριν, αλλά με πλήθος $diff$ λιγότερων στοιχείων.

$$\boldsymbol{EveryX_1} = (\boldsymbol{X_1}, \dots), \boldsymbol{EveryX_1} \in \mathbf{Z}^{\mathbf{1x\mu'}}$$

Τώρα πλέον διασφαλίστηκε ότι οι πίνακες έχουν ίσο αριθμό στοιχείων, είτε ίσο με ν', είτε ίσο με, μ'. Παρακάτω φαίνονται οι γραμμές των σημείων με το φούξια χρώμα:

$$(\boldsymbol{EveryX_1}, \boldsymbol{EveryY_1})$$

$$(\boldsymbol{EveryX_2}, \boldsymbol{EveryY_2})$$

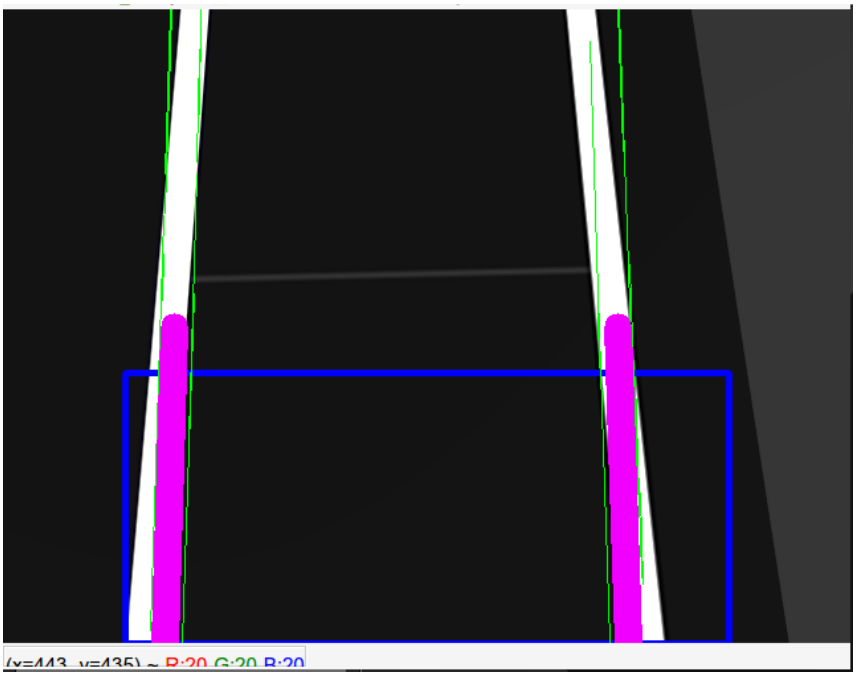


Figure 16 - Τελικές Γραμμές από τον Εντοπισμό Ευθειών

Επόμενο βήμα είναι να δημιουργηθούν δύο νέοι πίνακες, ο $NewXpoints$ και $NewYpoints$ οι οποίοι θα υπολογίζονται ως εξής:

$$\boldsymbol{NewXpoints} = (\boldsymbol{EveryX_1} + \boldsymbol{EveryX_2}) * \mathbf{0.5}$$

$$\boldsymbol{NewYpoints} = (\boldsymbol{EveryY_1} + \boldsymbol{EveryY_2}) * \mathbf{0.5}$$

Τα στοιχεία των παραπάνω πινάκων είναι πραγματικοί αριθμοί, άρα από κάθε στοιχείο του πίνακα κρατιέται μόνο το ακέραιο μέρος του. Έτσι, ισχύει ότι:

$$\boldsymbol{NewXpoints} \in \mathbf{Z}^{\mathbf{1x\mu'}} \text{ ή}$$

$$\boldsymbol{NewXpoints} \in \mathbf{Z}^{\mathbf{1x\nu'}}$$

Και

$$\boldsymbol{NewYpoints} \in \mathbf{Z}^{\mathbf{1x}\mu'} \text{ ή}$$

$$\boldsymbol{NewYpoints} \in \mathbf{Z}^{\mathbf{1x}\nu'}$$

Τέλος, ορίζεται ως, P, το τελικό πλήθος των πινάκων, $NewXpoints, NewYpoints$ με,

$$\boldsymbol{P = \mu'} \text{ ή } \boldsymbol{P = \nu'}$$

Στην εικόνα από κάτω φαίνεται η νέα ευθεία που προέκυψε από τον μέσο όρος των προηγούμενων σημείων. Είναι η μπλε γραμμή.

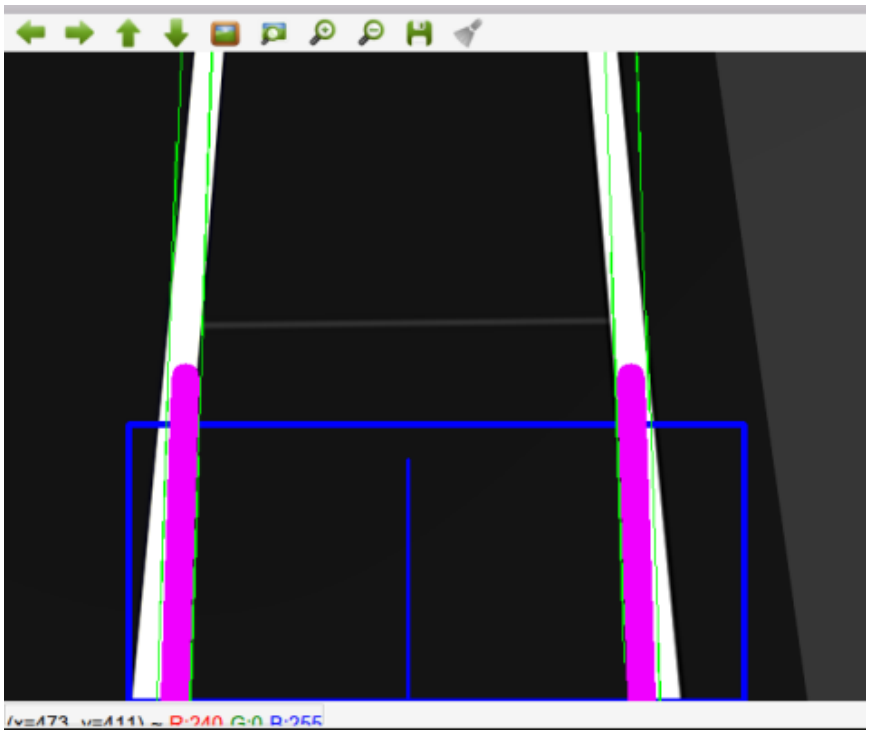


Figure 17 - Μέσος Όρος των Δύο Φούξια Γραμμών

Αφού υπολογίστηκαν οι παραπάνω πίνακες, πρέπει να δημιουργηθεί μία κάθετη γραμμή η οποία θα είναι σταθερή πάντα στο κέντρο του παραθύρου. Η γωνία που σχηματίζει η κάθετη γραμμή με την $ευθεία\ (\varepsilon)$ που αποτελείται από τα σημεία των πινάκων $NewXpoints, NewXpoints,$ είναι το σφάλμα το οποίο ο ελεγκτής καλείται να διορθώσει, ώστε να υπάρξει η επιθυμητή πορεία στο όχημα.

Η κάθετη γραμμή έχει εξίσωση:

$$\boldsymbol{x = Shape[1]/2}$$

Παρακάτω φαίνεται η κάθετη γραμμή στο κέντρο του παραθύρου. Είναι μεγαλύτερη από τις μπλε.

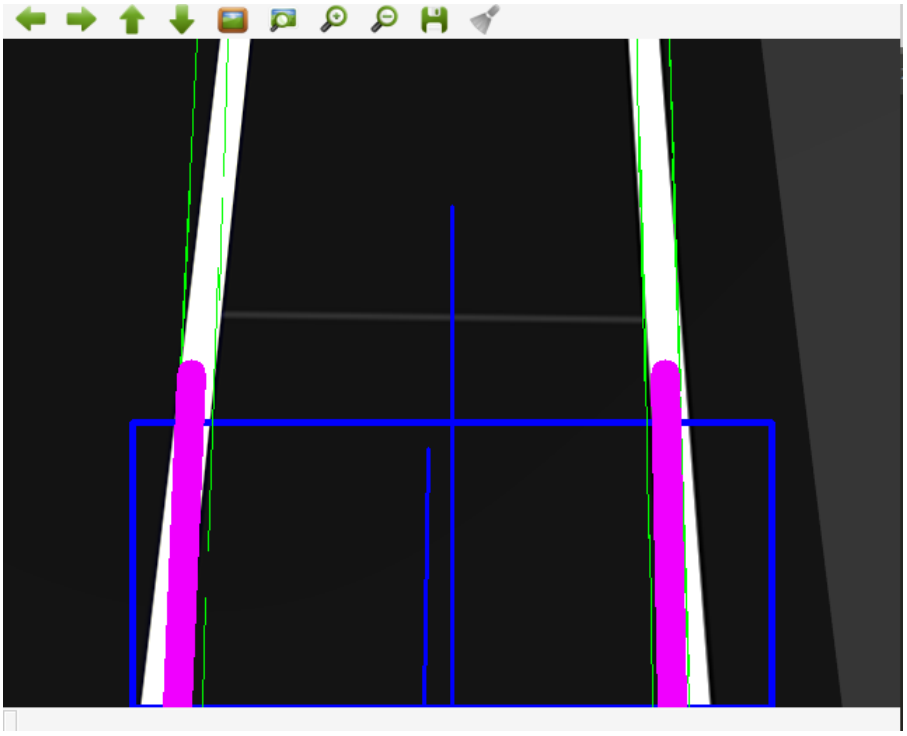


Figure 18 – Κεντρική Μπλε Κάθετη Γραμμή

Για να υπολογιστεί η γωνία μεταξύ της κεντρικής σταθερής κάθετης ευθείας και της ευθείας (ε) πρέπει να βρεθεί αρχικά το σημείο τομής τους.

Η X συντεταγμένη τομής της (ε) είναι πάντα σταθερή με,

$$\boldsymbol{NewX_2 = \frac{Shape[1]}{2}}$$

Επειδή το $NewX_2$ καταλήγει να είναι δεκαδικός αριθμός, λαμβάνεται το ακέραιο μέρος του αριθμού.

Χρησιμοποιώντας την σχέση,

$$\boldsymbol{f(x) - y1 = \lambda * (x - x1)}\text{, } \boldsymbol{\mu\varepsilon\ \kappa\lambda\acute{\iota}\sigma\eta\ \lambda = \frac{y2-y1}{x2-x1}}$$

και λύνοντας ως προς $f(x)$ προκύπτει:

$$\boldsymbol{f(x) = \lambda * (x - x1) + y1}$$

Και με αντικατάσταση των νέων δεδομένων με

$$\boldsymbol{f(x) = NewY_2,}$$

$$\boldsymbol{\lambda = \frac{NewYpoints(P-1) - NewYpoints(0)}{NewXpoints(P-1) - NewXpoints(0)}},$$

$$\boldsymbol{x = \frac{Shape[1]}{2}} \ \boldsymbol{και} \ \boldsymbol{x1 = NewXpoints(0)},$$

$$\boldsymbol{y1 = NewYpoints(0)}$$

προκύπτει:

$$\boldsymbol{NewY_2 = \frac{NewYpoints(P-1) - NewYpoints(0)}{NewXpoints(P-1) - NewXpoints(0)} * \left(\frac{Shape[1]}{2} - NewXpoints(0)\right) + NewYpoints(0)}$$

Άρα το σημείο τομής με την κάθετη ευθεία είναι:

$$\boldsymbol{ΣημΤομής = (NewX_2, NewY_2)}$$

Σημειώνεται ότι στους πίνακες το $(P-1)$, υποδηλώνει το τελευταίο σημείο του πίνακα και το (0), το πρώτο στοιχείο του πίνακα.

Το άλλο σημείο που χρειάζεται στη πορεία είναι το σημείο τομής της (ε) με τον X άξονα. Αυτό εύκολα βρίσκεται από τον $NewXpoints$, για την X συντεταγμένη και είναι το $NewX_1 = NewXpoints(0)$, καθώς για την Y συντεταγμένη το $NewY_1 = Shape[0]$. Άρα το ζητούμενο σημείο είναι:

$$\boldsymbol{ΣημΤομηςX = (NewX_1, NewY_1)}$$

Για τον υπολογισμό της γωνίας φ, μεταξύ της (ε) $και$ της κεντρικής κάθετης, λαμβάνεται η εφαπτόμενη της.

$$\boldsymbol{\tan(\varphi) = \frac{d_2}{d_1}}$$

Για να υπολογιστεί όμως η γωνία φ,

$$\boldsymbol{\varphi = tan^{-1}(\frac{d_2}{d_1})}$$

Όπου d_2 είναι η απόσταση στον άξονα X μεταξύ του τελευταίου και πρώτου σημείου του πίνακα $NewXpoints$,

$$\boldsymbol{d_2 = |NewX_2 - NewX_1|}$$

Και d_1, η απόσταση της προσκείμενης πλευράς, με:

$$\boldsymbol{d_1 = |NewY_2 - NewY_1|}$$

Παρακάτω φαίνεται η τομή των δύο ευθειών όταν το όχημα δει μπροστά του στροφή, καθώς οι αντίστοιχες πλευρές και γωνία που έχουν υπολογιστεί.

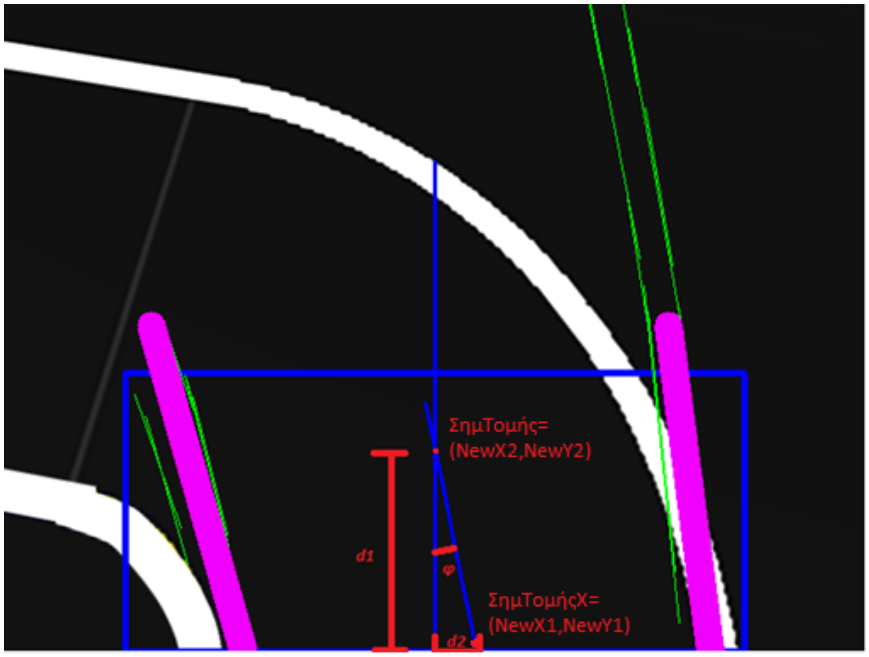


Figure 19 - Τομή Ευθείας και Κάθετης Γραμμής

Ο υπολογισμός της γωνίας καθ' αυτός δεν μπορεί να αποβεί αποτελεσματικός. Για αυτό η γωνία στρέψης που βρέθηκε πρέπει να συνδυαστεί με περιπτώσεις, ανάλογα το όχημα αν βρίσκεται στην αριστερή ή δεξιά πλευρά του δρόμου και να κινηθεί προς το κέντρο, είτε όταν υπάρχουν αριστερές ή δεξιές στροφές.

Για να γίνει αντιληπτό ότι το όχημα πρέπει να στραφεί πρέπει να ισχύουν κάποιες συνθήκες:

Το node του ελεγκτή για να λειτουργήσει πρέπει να αξιοποιηθούν οι πληροφορίες των αποτελεσμάτων των παρακάτω περιπτώσεων. Στο κεφάλαιο 2 αναφέρθηκε η χρήση του topic line_follower.msg, στο οποίο εν τέλη εισάγονται τα μηνύματα από την έξοδο του node line_detector.py

Άρα στο τέλος κάθε περίπτωσης θα στέλνει τα εξής μηνύματα στο αρχείο line_follower.msg:

- Μήνυμα λογικής μεταβλητής(Boolean) αν έχει βρεθεί γραμμή.
- Μήνυμα χρόνου εκτέλεσης της εξομοίωσης στο Gazebo από τη στιγμή που αρχίζει να τρέχει το node(
- Μήνυμα κατεύθυνσης προς τα που στρέφεται το όχημα, αριστερά ή δεξιά.
- Μήνυμα διόρθωσης γωνίας φ που υπολογίστηκε παραπάνω.

1. *Περίπτωση του οχήματος να έχει αριστερή στροφή μπροστά του.*

$$Αν\ ((NewX_1 > NewX_2) KAI (\ NewY_1 > NewY_2\ )) KAI\ \varphi > 2rad$$

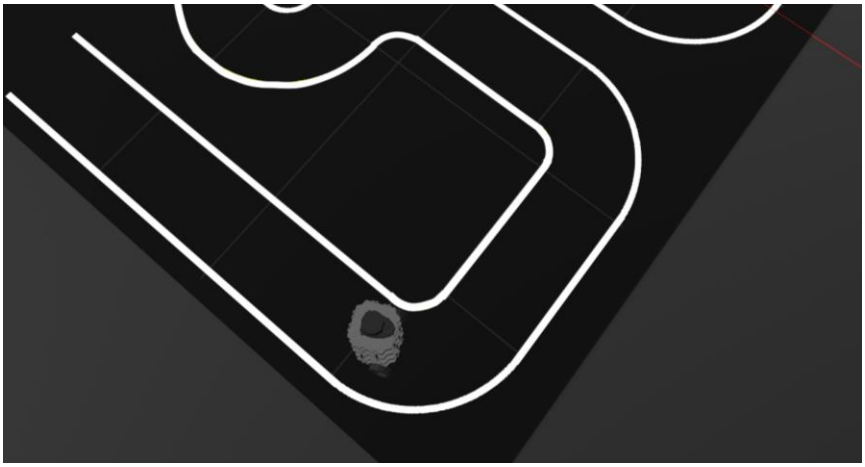


Figure 20 – Θέση Οχήματος για την Πρώτη Περίπτωση Αριστερής Στροφής

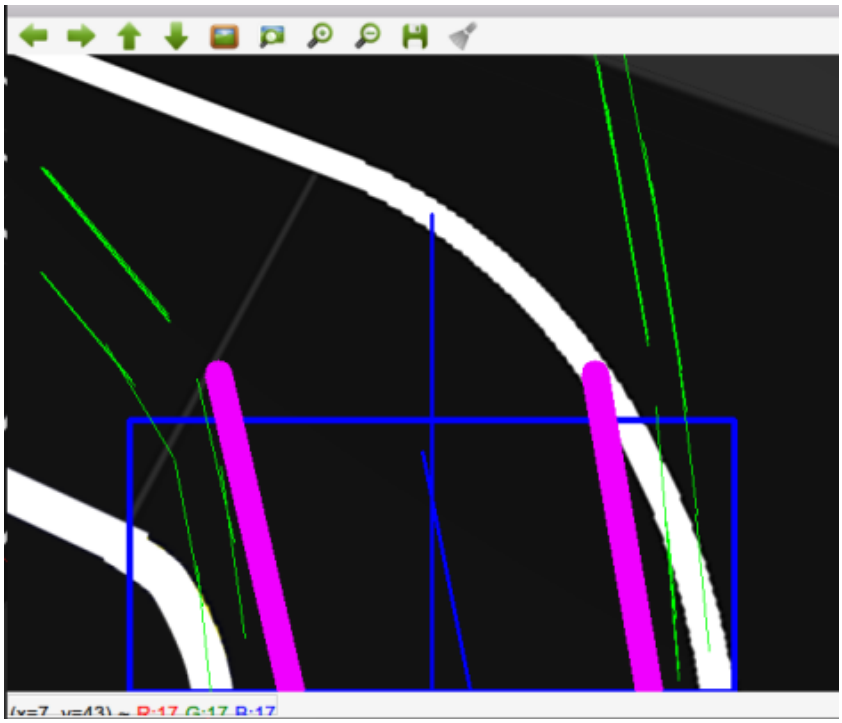


Figure 21 - Εικόνα Κάμερας Αριστερής στροφής Πρώτης Περίπτωσης

$$Αν\ ((NewX_1 < NewX_2) KAI (\ NewY_1 < NewY_2\ ))\ KAI\ \varphi > 2rad$$

Figure 22 - – Θέση Οχήματος για την Δεύτερη Περίπτωση Αριστερής Στροφής

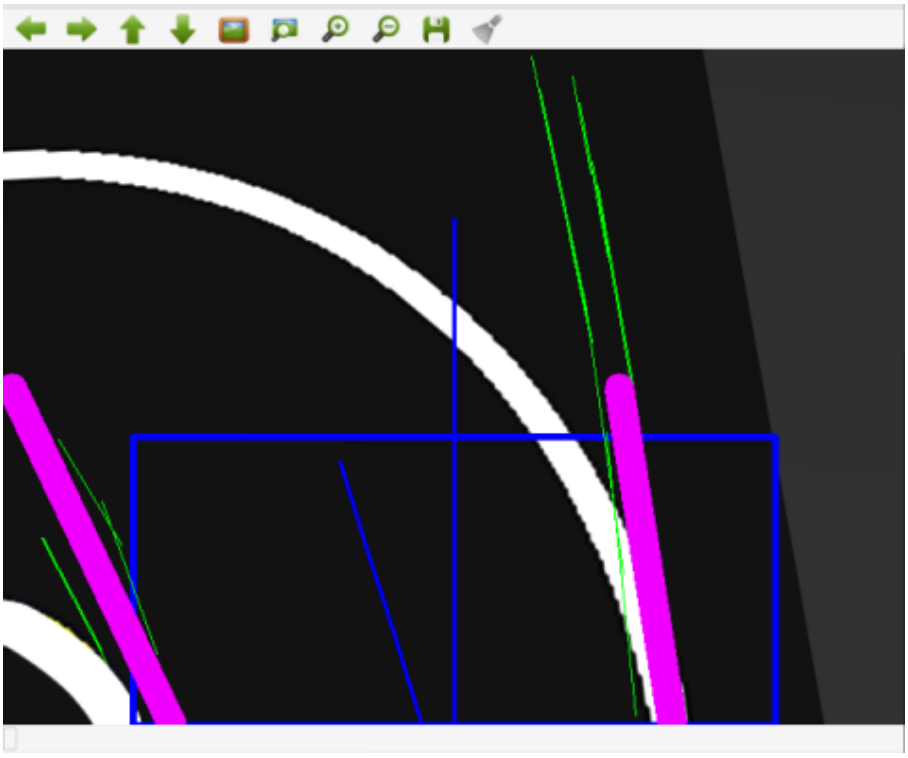


Figure 23 - Εικόνα Κάμερας Αριστερής στροφής Δεύτερης Περίπτωσης Αριστερής Στροφής

$\boldsymbol{Τότε}$

$$\boldsymbol{Εντοπισμός Γραμμής = Αληθής}$$

$$\boldsymbol{Χρόνος = Μετρούμενος\ Χρόνος}$$

$$\boldsymbol{Κατεύθυνση = "Αριστερά"}$$

$$\boldsymbol{Σφάλμα Γωνίας = \varphi}$$

2. *Περίπτωση του οχήματος να έχει δεξιά στροφή μπροστά του.*

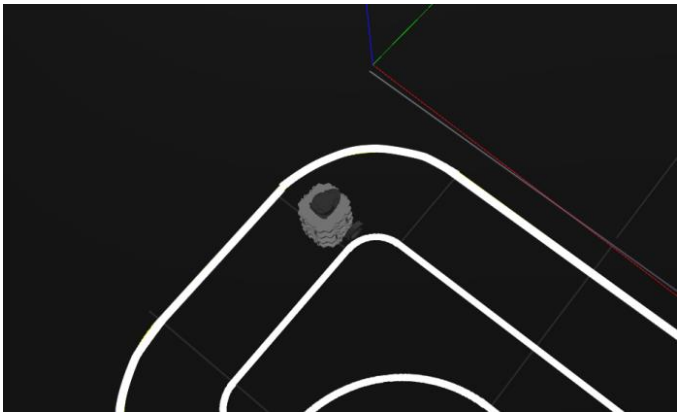


Figure 24 - Θέση Οχήματος για την Πρώτη Περίπτωση Δεξιάς Στροφής

$$\boldsymbol{Αν\ \left((NewX_1 > NewX_2) KAI (\ NewY_1 < NewY_2\ )\right) KAI\ \varphi > 2rad}$$

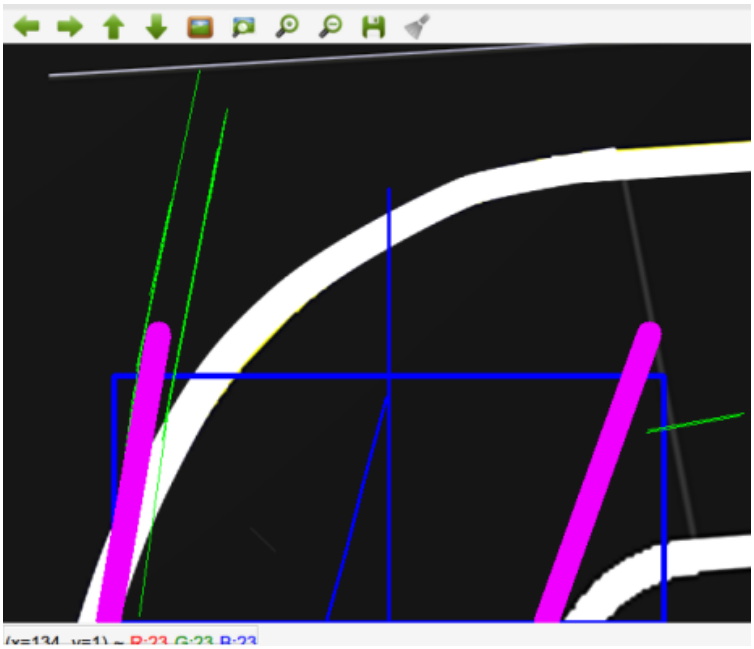


Figure 25 - Εικόνα Κάμερας Πρώτης Περίπτωσης Δεξιάς Στροφής

$$\boldsymbol{Αν\ \left((NewX_1 < NewX_2) ΚΑΙ (\ NewY_1 > NewY_2\ )\right)\ ΚΑΙ\ \varphi > 2rad}$$

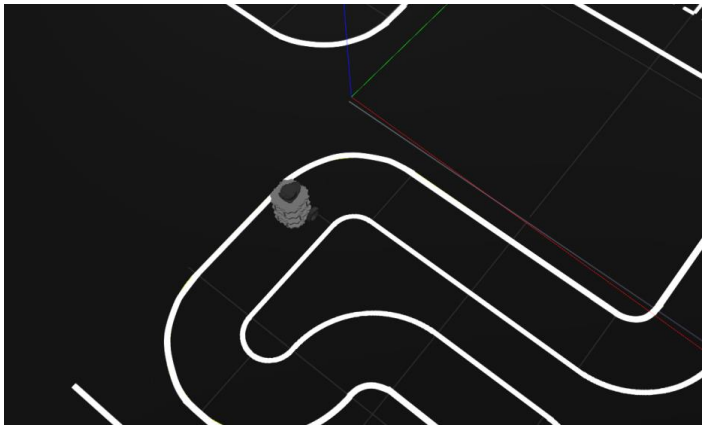


Figure 26 - - Θέση Οχήματος για την Δεύτερη Περίπτωση Δεξιάς Στροφής

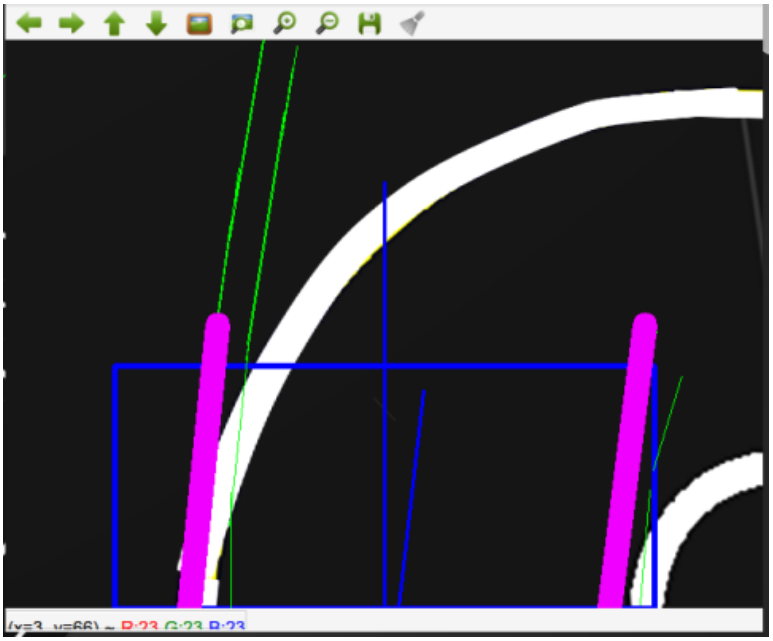


Figure 27 - - Εικόνα Κάμερας Δεύτερης Περίπτωσης Δεξιάς Στροφής

$$\boldsymbol{Τότε}$$

$$\boldsymbol{ΕντοπισμόςΓραμμής = Αληθής}$$

$$\boldsymbol{Χρόνος = Μετρούμενος\ Χρόνος}$$

$$\boldsymbol{Κατεύθυνση = "Δεξιά"}$$

$$\boldsymbol{ΣφάλμαΓωνίας = \varphi}$$

3. *Περίπτωση του οχήματος να βρίσκεται στη αριστερή πλευρά του δρόμου.*

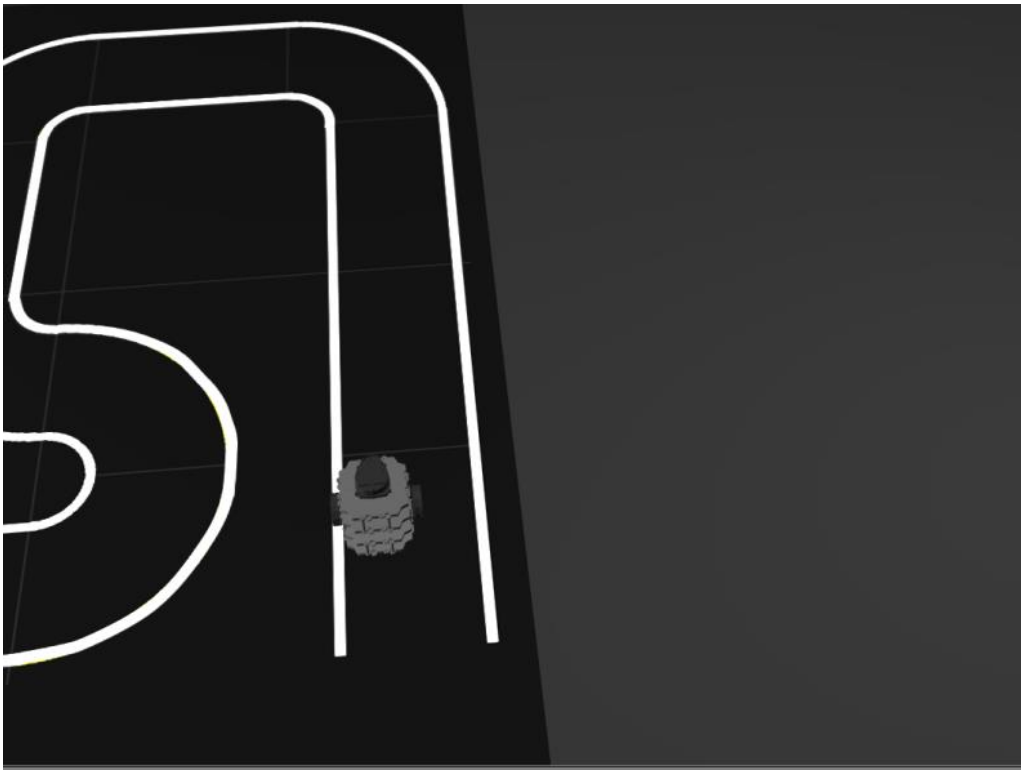


Figure 28 - Το Όχημα Βρίσκεται στην Αριστερή Πλευρά του Δρόμου

Όταν συμβαίνει αυτό το φαινόμενο η γωνία φ είναι αρκετά μικρή ή μπορεί να είναι και $\varphi = 0$.

Για αυτόν το λόγο χρησιμοποιείται στον X άξονα, η απόσταση της κεντρικής ευθείας από την συντεταγμένη X της (ε). Αυτή η απόσταση έχει υπολογιστεί και προηγουμένως και ισούται με

$$\boldsymbol{d_2 = |\, NewX_2 - \, NewX_1|}$$

Όμως, με κάποιο τρόπο πρέπει η απόσταση να αναχθεί σε μια λογική τιμή γωνίας στρέψης. Ως αποτέλεσμα αυτού βρέθηκε ότι

$$\boldsymbol{\varphi = \frac{d_2}{parameter}}.$$

Πειραματικά βρέθηκε ότι $700 \leq parameter \leq 950$. Όσο μεγαλώνει ο παρονομαστής του κλάσματος η γωνία φ μικραίνει άρα, το σύστημα γίνεται λιγότερο ευαίσθητο στις μεταβολές.

Όσο ο παρονομαστής μικραίνει, μεγαλώνει η γωνία στρέψης με αποτέλεσμα στο όχημα να υποβάλλεται μία απότομη και μεγάλη γωνία στρέψης, πράγμα το οποίο δεν αντικατοπτρίζει την πραγματικότητα.

Η συνθήκη η οποία πρέπει να ισχύει ώστε το όχημα να εκτελέσει τον υπολογισμό αυτόν είναι:

$$\boldsymbol{Αν}\ ((\boldsymbol{NewY_2} > \boldsymbol{480})\ Ή\ (\boldsymbol{NewY_2} < \boldsymbol{0}))\ \boldsymbol{ΚΑΙ}\ (\boldsymbol{\varphi} < \boldsymbol{2rad})\ \boldsymbol{ΚΑΙ}\ (\boldsymbol{\varphi} \neq \boldsymbol{0})\ \boldsymbol{ΚΑΙ}\ \ (\boldsymbol{NewX_1} > \boldsymbol{NewX_2})$$

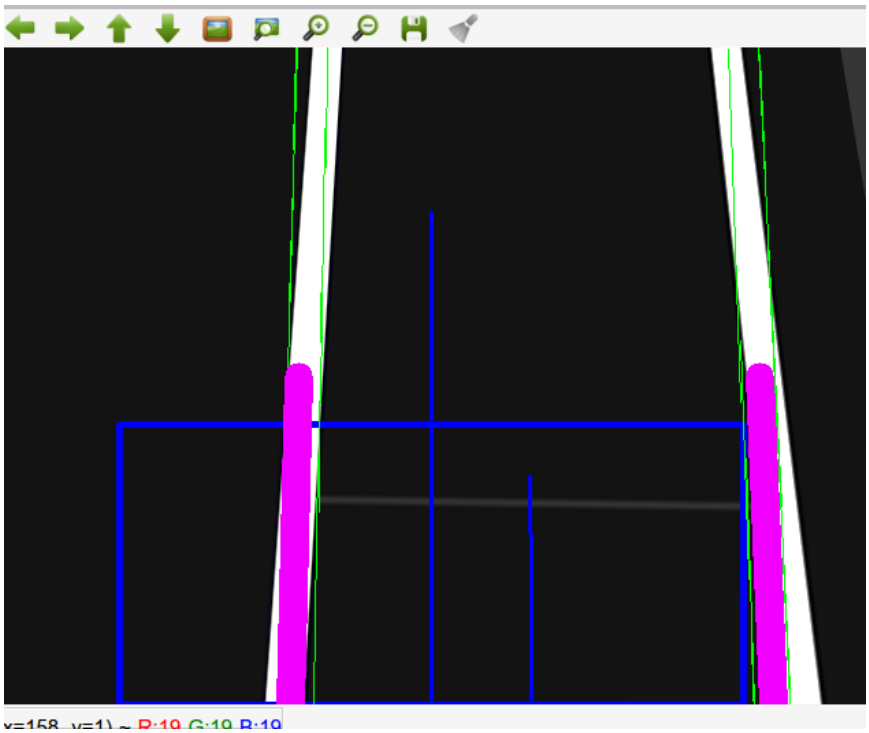


Figure 29 - Εικόνα Κάμερας Από τη Αριστερή Πλευρά του Δρόμου

$$\boldsymbol{Τότε}$$

$$\boldsymbol{ΕντοπισμόςΓραμμής} = \boldsymbol{Αληθής}$$

$$\boldsymbol{Χρόνος} = \boldsymbol{ΜετρούμενοςΧρόνος}$$

$$\boldsymbol{Κατεύθυνση} = \text{"}\boldsymbol{Δεξιά}\text{"}$$

$$\boldsymbol{ΣφάλμαΓωνίας} = \boldsymbol{\varphi}$$

4. *Περίπτωση του οχήματος να βρίσκεται στη δεξιά πλευρά του δρόμου.*

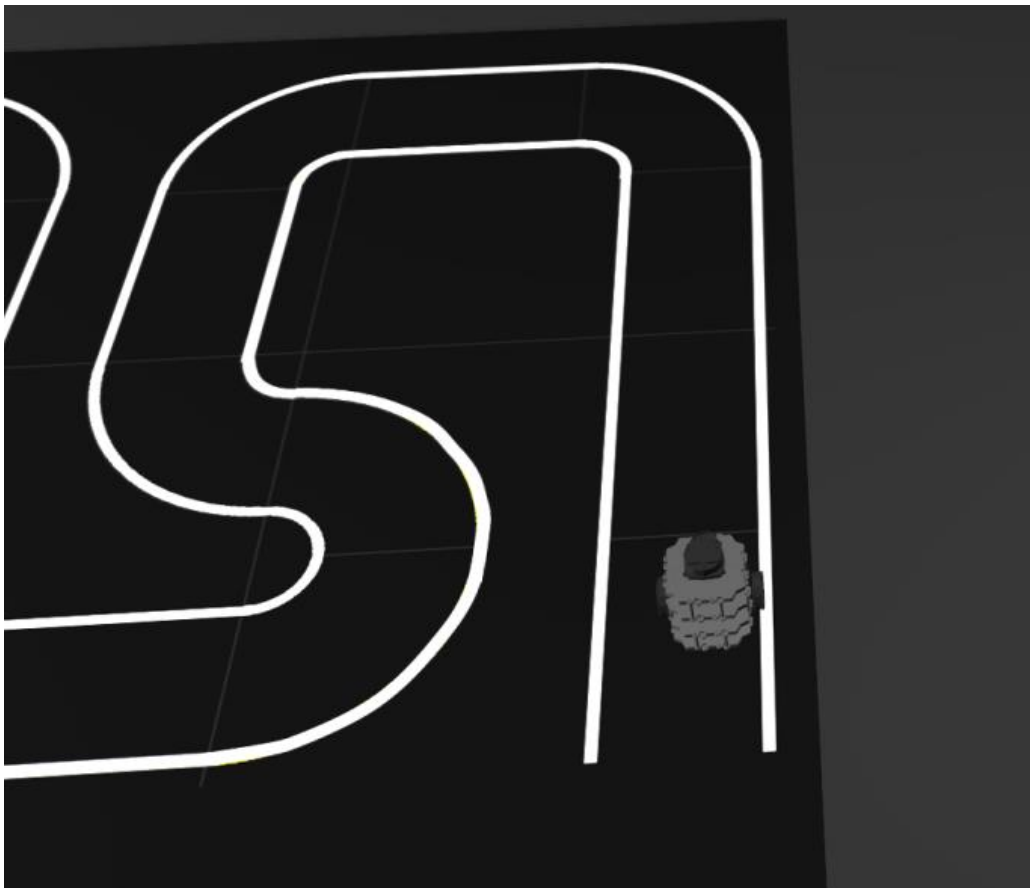


Figure 30 - Το Όχημα Βρίσκεται στην Δεξιά Πλευρά του Δρόμου

Όπως και στην προηγούμενη περίπτωση ισχύει η ίδια μεθοδολογία. Η διαφορά εντοπίζεται στη συνθήκη. Αρχικά υπολογίζονται τα d_2, φ όπως και πριν:

$$\boldsymbol{d_2 = |\, NewX_2 - \, NewX_1|}$$

$$\boldsymbol{\varphi = \frac{d_2}{parameter}}$$

Έπειτα η συνθήκη η οποία πρέπει να ισχύει είναι η παρακάτω:

$$\boldsymbol{Αν}\ ((\boldsymbol{NewY_2} > \boldsymbol{480})\ Ή\ (\boldsymbol{NewY_2} < \boldsymbol{0}))\ \boldsymbol{ΚΑΙ}\ (\boldsymbol{\varphi} < \boldsymbol{2rad})\ \boldsymbol{ΚΑΙ}\ (\boldsymbol{\varphi} \neq \boldsymbol{0})\ \boldsymbol{ΚΑΙ}\ (\boldsymbol{NewX_1} < \boldsymbol{NewX_2})$$

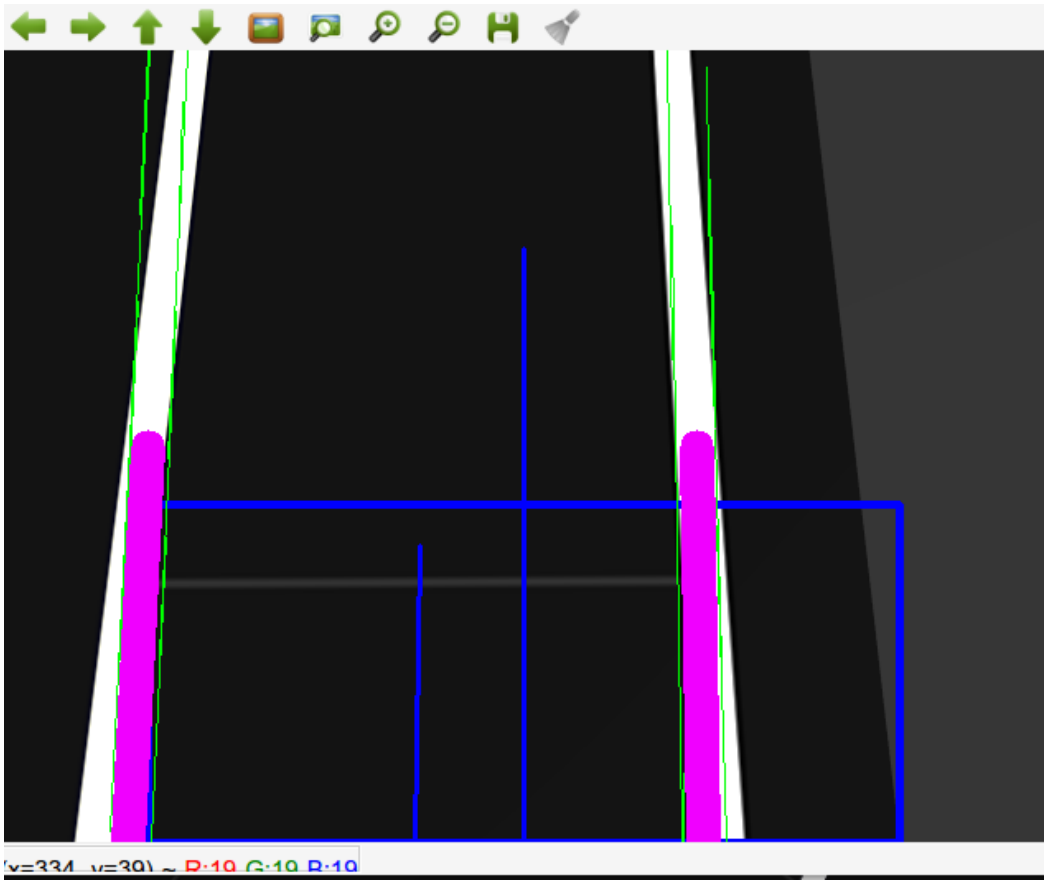


Figure 31 - - Εικόνα Κάμερας Από τη Δεξιά Πλευρά του Δρόμου

$$\boldsymbol{Τότε}$$

$$\boldsymbol{ΕντοπισμόςΓραμμής} = \boldsymbol{Αληθής}$$

$$\boldsymbol{Χρόνος} = \boldsymbol{ΜετρούμενοςΧρόνος}$$

$$\boldsymbol{Κατεύθυνση} = \text{"}\boldsymbol{Αριστερά}\text{"}$$

$$\boldsymbol{ΣφάλμαΓωνίας} = \boldsymbol{\varphi}$$

Κεφάλαιο 5 - Σχεδίαση Νόμων Ελέγχου

Η λειτουργία που επιτελεί ένας ελεγκτής αφορά την εξάλειψη του σφάλματος. Ένα σφάλμα προκύπτει από την Επιθυμητή Τιμή αφαιρώντας την Πραγματική. Ένας ελεγκτής είναι μία επαναληπτική διαδικασία, όπου σε κάθε επανάληψη σκοπός του είναι να μειώνει το σφάλμα, όσο πιο πολύ γίνεται και να φτάσει αν είναι δυνατόν στο 0. Ο πιο κλασσικός έλεγχος που μπορεί να εφαρμοστεί είναι ο γνωστός ελεγκτής PID(Proportional, Integral, Derivative). Δεν είναι απαραίτητο να εφαρμοστούν και οι τρεις όροι. Στην προκειμένη περίπτωση δοκιμάζεται ένας αναλογικός έλεγχος, P, καθώς και ένας τριγωνομετρικός ελεγκτής συγκεκριμένα για την παρούσα εργασία. Η γωνία που έχει υπολογιστεί από το προηγούμενο κεφάλαιο, αποτελεί κατευθείαν και το σφάλμα που πρέπει να διορθωθεί.

Επίσης, επειδή η κάμερα είναι τοποθετημένη στο κέντρο μάζας του οχήματος και το σύστημα συντεταγμένων της στον X άξονα συμπίπτει με τον X άξονα κατεύθυνσης του οχήματος δεν κρίνεται απαραίτητος κάποιος μετασχηματισμός στις γωνίες στρέψης.

Ένα γενικό παράδειγμα ελέγχου φαίνεται παρακάτω:

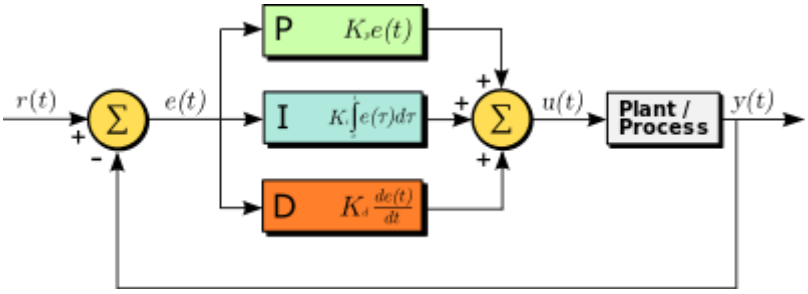


Figure 32 - PID Ελεγκτής

Η συνάρτηση ενός PID ελέγχου είναι η παρακάτω:

$$\boldsymbol{u(t) = K_p e(t) + K_i \int e(t) + K_d \frac{de(t)}{dt}}$$

Για λειτουργήσει αυτή η διαδικασία, στην πιο απλή περίπτωση, χρειάζεται ένα αισθητήριο, το οποίο να μετράει την πραγματική τιμή του συστήματος και να έχει δηλωθεί μία επιθυμητή τιμή. Στην αρχή μετρείται το σφάλμα, δηλαδή πόσο απέχει η πραγματική τιμή από την επιθυμητή. Έπειτα το $\sigma\phi\acute{\alpha}\lambda\mu\alpha = e(t)$, πρέπει να μειώνεται σε κάθε επανάληψη στον υπολογιστή καθώς αποτελεί μια διαφορική εξίσωση ενός δυναμικού συστήματος, σύμφωνα με την παραπάνω εξίσωση. Η παραπάνω εξίσωση έχει 3 αγνώστους παραμέτρους. K_p, K_i, K_d. Η εύρεση αυτών των τριών τιμών, κρίνει και το πόσο καλά θα δουλέψει και ο ελεγκτής, ώστε να υπάρξει μια επιθυμητή απόκριση. Ο K_p όρος προσφέρει στο σύστημα ένα συνολικό έλεγχο δράσης στο σύστημα μέσω τον παράγοντα του κέρδους, ο K_i όρος μειώνει το σφάλμα μόνιμης κατάστασης και ο K_dόρος βελτιώνει τη μεταβατική απόκριση του συστήματος. Γενικά ένας πίνακας με το τι προσφέρει κάθε όρος φαίνεται παρακάτω:

Απόκριση Κλειστού Συστήματος	Χρόνος Ανύψωσης	Υπερύψωση	Χρόνος Αποκατάστασης	Σφάλμα Μόνιμης Κατάστασης	Ευστάθεια
Αυξάνοντας το K_p	Μειώνεται	Αυξάνεται	Μικρή Αύξηση	Μειώνεται	Μειώνεται
Αυξάνοντας το K_i	Μικρή Μείωση	Αύξηση	Αύξηση	Μεγάλη Μείωση	Μείωση
Αυξάνοντας το K_d	Μικρή Μείωση	Μειώνεται	Μειώνεται	Μεγάλη Αλλαγή	Βελτίωση

Table 2 - Αποκρίσεις Kp, Ki, Kd

Το αποτέλεσμα αυτής της εξίσωσης μπαίνει στο σύστημα ως είσοδος με αποτέλεσμα να αλλάζει η έξοδος του συστήματος. Η έξοδος του συστήματος, πρέπει να έχει την επιθυμητή απόκριση που ζητείται, δηλαδή να φτάσει την επιθυμητή τιμή. [14]

Στην προκειμένη περίπτωση το αισθητήριο είναι η κάμερα, από την οποία υπολογίζεται το σφάλμα της γωνίας του οχήματος. Για την εξάλειψη του σφάλματος έχουν δοκιμαστεί δύο τύποι ελεγκτών. Ο ένας είναι ο P(Proportional) αναλογικός ελεγκτής και ο δεύτερος ο τριγωνομετρικός ελεγκτής. Απώτερος σκοπός και δύο αυτών ελεγκτών, είναι να σταθεροποιήσουν ασταθές σύστημα. Γενικά ευσταθές σύστημα θεωρείται αυτό που η έξοδος του τείνει σε μια τιμή και δεν απειρίζεται. Σε κάθε επανάληψη λειτουργίας του προγράμματος δίνουν μία νέα τιμή στη γραμμική και τη γωνιακή ταχύτητα του Turtlebot3 έως ότου να μηδενίζεται κάθε φορά σφάλμα. Το σφάλμα είναι:

$$e(t) = επιθυμητή\ γωνία - μετρούμενη\ γωνία$$

Η επιθυμητή γωνία είναι: $επιθυμητή\ γωνία = 0°$

Η μετρούμενη γωνία, είναι: $Μετρούμενη\ γωνία = Γωνία\ Στρέψης = \varphi$

- *Αναλογικός Ελεγκτής P.*

Η γενική μορφή του ελεγκτή είναι:

$$u(t) = K_p e(t)$$

Ο ελεγκτής P είναι η πιο απλή μορφή ελεγκτή που μπορεί να χρησιμοποιηθεί. Ο κύριος σκοπός του είναι να μειώσει τη τιμή του σταθερού μόνιμου σφάλματος. Όσο αυξάνει η τιμή του K_p το σφάλμα της

μόνιμης κατάστασης και ο χρόνου ανόδου, μειώνονται. Για μεγάλες τιμές K_p στο σύστημα προκαλείται υπερύψωση.

$$U = u = Γραμμική\ Ταχύτητα = σταθερή$$

$$\omega = Γωνιακή\ Ταχύτητα = K_p e(t)$$

- *Τριγωνομετρικός Ελεγκτής*

Οι ελεγκτής P που χρησιμοποιήθηκε, ή αν εφαρμοζόταν ο PID ή κάποιοι από τους όρους του PID δηλαδή, PI, PD είναι πολύ αποτελεσματικοί γενικής χρήσης ελεγκτές. Το πρόβλημα που υπάρχει όμως, είναι εύρεση των σταθερών K_p, K_i, K_d, κάτι το οποίο είναι δύσκολο. Υπάρχουν μέθοδοι εύρεσης των τιμών, αλλά πρέπει να υπάρχει διαθέσιμο ένα αρκετά καλό μοντέλο του συστήματος. Αν το σύστημα είναι μη γραμμικό, δυσκολεύει ακόμη πιο πολύ η κατάσταση. Αυτό το πρόβλημα το λύνει ο τριγωνομετρικός ελεγκτής πιο εύκολα ο οποίος λειτουργεί αρκετά αποτελεσματικά. [15]

Η προσέγγιση που έγινε στα προηγούμενα κεφάλαια ήταν εν τέλη η γωνία περιστροφής του οχήματος, ώστε να διορθώσει το σφάλμα.

Η εξισώσεις που διέπουν τον τριγωνομετρικό ελεγκτή είναι:

$$U = u * cos(e(t))$$

$$\omega = \frac{u}{l} * sin(e(t))$$

Όπως διαπιστώνεται η γραμμική ταχύτητα U είναι μέγιστη όταν $e(t) = 0$ και μειώνεται σύμφωνα με τη γραφική του συνημίτονου όταν το $e(t)$ αυξάνει. Αυτό βοηθά πάρα πολύ στο χρόνο που κάνει το Turtlebot να τελειώσει τη πίστα, γιατί στις περιπτώσεις που εμφανίζεται σχεδόν μηδενικό σφάλμα, η γραμμική ταχύτητα είναι η μέγιστη.

Η γωνιακή ταχύτητα εξαρτάται από τον όρο του ημιτόνου. Όταν το σφάλμα είναι 0 η γωνιακή ταχύτητα είναι 0. Δηλαδή το όχημα δε στρίβει, που σημαίνει ότι το όχημα βρίσκεται στη σωστή πορεία.

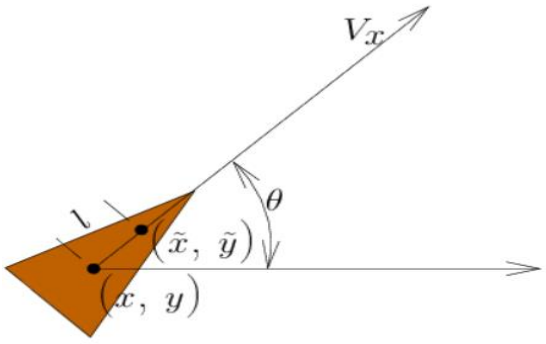


Figure 33 - Αναπαράσταση του l

Η μεταβλητή l δηλώνει μια πολύ κοντινή θέση μπροστά από το όχημα. Αν το σημείο που βρίσκεται το όχημα είναι Χ(x,y) και Χ'(x',y') η επόμενη θεωρητική θέση του οχήματος το $l = X' - X$. Στην ουσία η μεταβλητή l είναι και η μοναδική τιμή που χρειάζεται εύρεση. Προκύπτει μέσα από πειραματικές δοκιμές. Στον όρο $\frac{u}{l}$ όσο αυξάνει l, μειώνεται και η ταχύτητα στροφής, που αυτό σημαίνει ότι δε στέλνεται η κατάλληλη γωνία στρέψης και εν τέλη το όχημα βρίσκεται εκτός πορείας. Για πολύ μικρά l, η γωνιακή ταχύτητα αυξάνει πολύ με αποτέλεσμα να στρέφεται πολύ γρήγορα, με αποτέλεσμα να γίνονται απότομες κινήσεις στρέψης του οχήματος και να μην συνεχίζεται η διαδικασία ομαλά.

Όταν το όχημα περιστρέφεται αριστερά ή δεξιά η γωνία, με τον τρόπο που έχει υπολογιστεί, είναι πάντα θετική. Για αυτό το λόγο γίνεται η χρήση μηνυμάτων μέσα στο node του ελεγκτή line_controller.py.

- $Αν\ το\ μήνυμα\ κατεύθυνσης =$ "Αριστερά", $Τότε\ ΓωνίαΠεριστροφής = e(t) = \varphi$
- $Αν\ το\ μήνυμα\ κατεύθυνσης =$ "Δεξιά", $Τότε\ ΓωνίαΠεριστροφής = e(t) = -\varphi$

Cmd_vel Topic

Το cmd_vel Topic του ROS είναι τύπος μηνύματος Twist.msg. Έχει την ακόλουθη μορφή:

Linear Velocity:

float64 x
float64 y
float64 z

Angular Velocity:

float64 x
float64 y
float64 z

Τα τρία πρώτα δηλώνουν Γραμμική Ταχύτητα στους τρεις άξονες x,y,z. Τα άλλα τρία δηλώνουν την περιστροφική ταχύτητα ως προς του τρεις άξονες x,y,z.

Στη περίπτωση του Turtlebot οι μεταβλητές που χρησιμοποιούνται είναι από το Γραμμικό κομμάτι της ταχύτητας, ο x άξονας, και από την γωνιακή ταχύτητα η μεταβλητή z. Οι δύο παραπάνω υπολογισμένες ταχύτητες, η γραμμική και η περιστροφική, στέλνονται στο Γραμμικό_Χ και στο Γωνιακό_Z, αντίστοιχα .

$$X = Γραμμική\ Ταχύτητα$$

$$Z = Γωνιακή\ Ταχύτητα$$

Κεφάλαιο 6 - Αποτελέσματα

Το Turtlebot3 κατά την εξομοίωση ανταποκρίθηκε επιτυχώς. Μία συνολική εικόνα του συστήματος από την εύρεση των γραμμών και της γωνίας στρέψης φαίνεται παρακάτω:

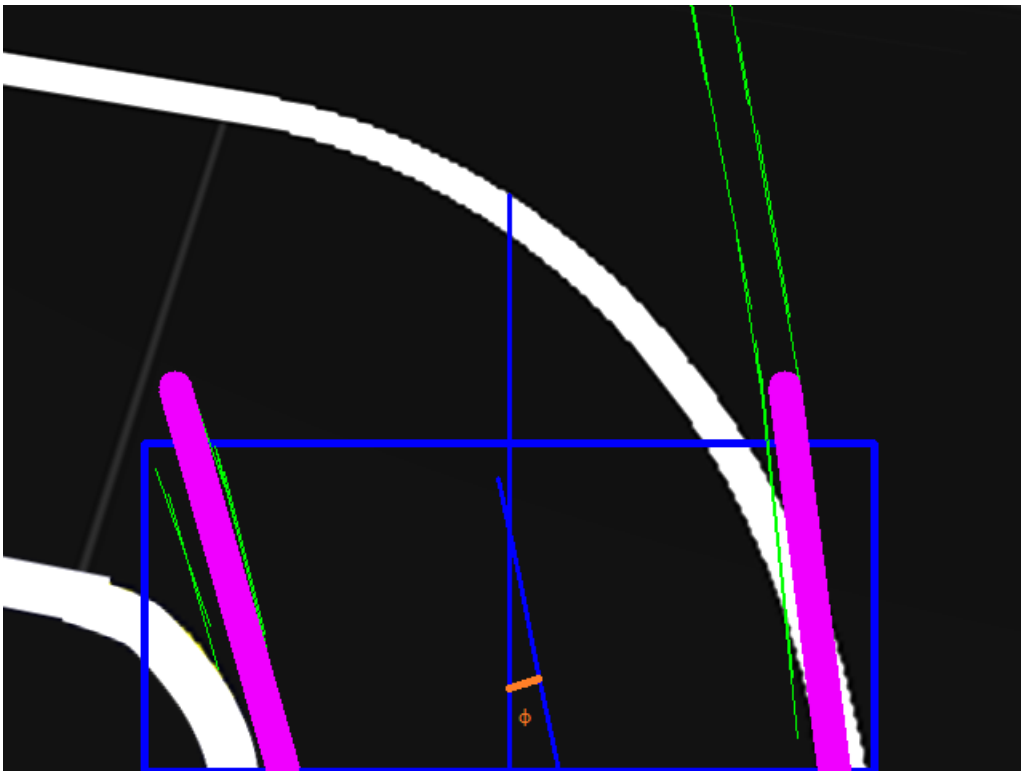


Figure 34 - Τελική Εικόνα Κάμερας

Όπως παρατηρείται οι πράσινες γραμμές δηλώνουν τις γραμμές οι οποίες βρέθηκαν. Δεν είναι πάνω στις άσπρες επειδή έχει προηγηθεί ο γεωμετρικός μετασχηματισμός. Οι φούξια αριστερή και δεξιά γραμμή αντιπροσωπεύουν σε μία γραμμή το σύνολο των πράσινων γραμμών που βρέθηκαν. Επίσης φαίνεται η γωνία στρέψης φ, η οποία σχηματίζεται από το σημείο τομής της μικρής μπλε γραμμής με την κεντρική μεγάλη κάθετη μπλε γραμμή. Όταν αυτές οι δύο γραμμές είναι σχεδόν παράλληλες μεταξύ τους η γωνία $\varphi = 0$ που σημαίνει ότι το όχημα βρίσκεται στη σωστή πορεία. Οι δοκιμασμένες μέγιστες γραμμικές ταχύτητες και στους δύο ελεγκτές είναι 0.27. Η μέγιστη γραμμική ταχύτητα του Turtlebot3 Burger είναι 0.22 στη πραγματικότητα. Αυτό σημαίνει ότι υπάρχουν περιθώρια λάθους σε πείραμα με πραγματική πίστα διότι οι ελεγκτές μπορούν να ανταποκριθούν και σε πιο απαιτητικές καταστάσεις λειτουργίας του Robot. Δοκιμάστηκαν επίσης και πολύ μεγάλες γραμμικές ταχύτητες της τάξεως του 0.4-0.6m/s, αλλά η εύρεση των παραμέτρων των ελεγκτών ήταν αρκετά δύσκολη. Επίσης ακόμη και αν έστριβε έγκαιρα το Robot έφευγε από την πορεία του και ολίσθαινε λόγω του βάρους του σε συνδυασμό με την τριβή της πίστας.

Οι τιμές τελικά που χρησιμοποιήθηκαν για την παρακάτω γραφική παράσταση (χρόνου, σφάλματος) για τον τριγωνομετρικό ελεγκτή είναι:

$$l = 0.038$$

$$u = 0.27$$

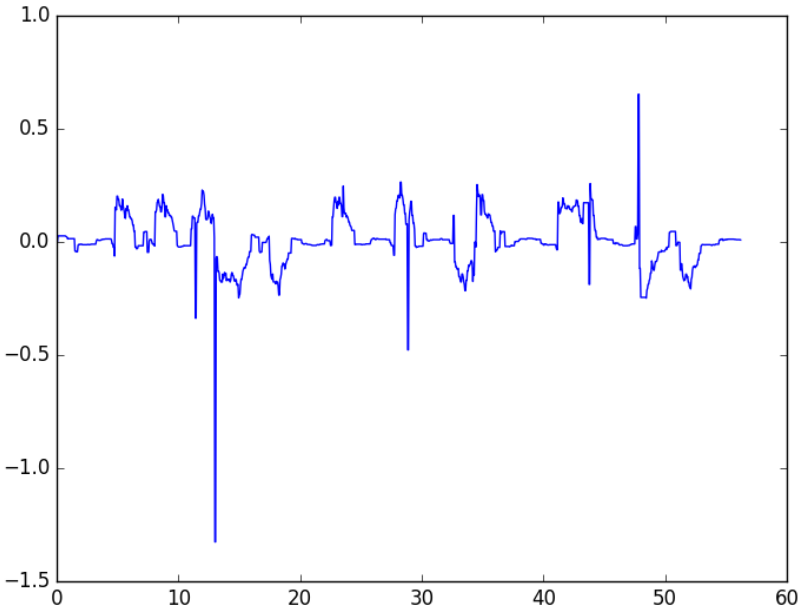


Figure 35 - Γραφική Σφάλματος σε Συνάρτηση με το Χρόνο Γεωμετρικού Ελεγκτή

Ο πίνακας τιμών που βρέθηκε αποτελείται από τη στήλη του σφάλματος και τη στήλη χρόνου από το εσωτερικό μετρητή χρόνου των nodes που προκύπτει από τη συνάρτηση rospy.get_time(). Παρατηρήθηκε ότι οι τιμές των χρόνων δεν είναι πραγματικές ως προς τον κόσμο. Όμως, η μία τιμή από την επόμενη έχουν διαφορά περίπου 0.05. Για αυτό το λόγο με βάση αυτή τη παρατήρηση, δημιουργήθηκε ένας νέος άξονας χρόνου που αυξάνεται κατά 0.05sec. Για παράδειγμα, η στήλη του σφάλματος του τριγωνομετρικού ελεγκτή κατέληξε από την εξομοίωση να έχει 1125 στοιχεία. Άρα, αν γίνει η πράξη $1125 * 0.05$, βρίσκεται ότι από τη στιγμή 0 μέχρι τη στιγμή 56.25 με βήμα 0.05 χωράνε 1125 τιμές, όσα είναι και τα στοιχεία της στήλης του σφάλματος.

Για την παρακάτω γραφική παράσταση (χρόνου,σφάλματος) οι παρακάτω τιμές για τον Αναλογικό ελεγκτή που δοκιμάστηκαν είναι:

$$k_P = 5$$

$$u = 0.27$$

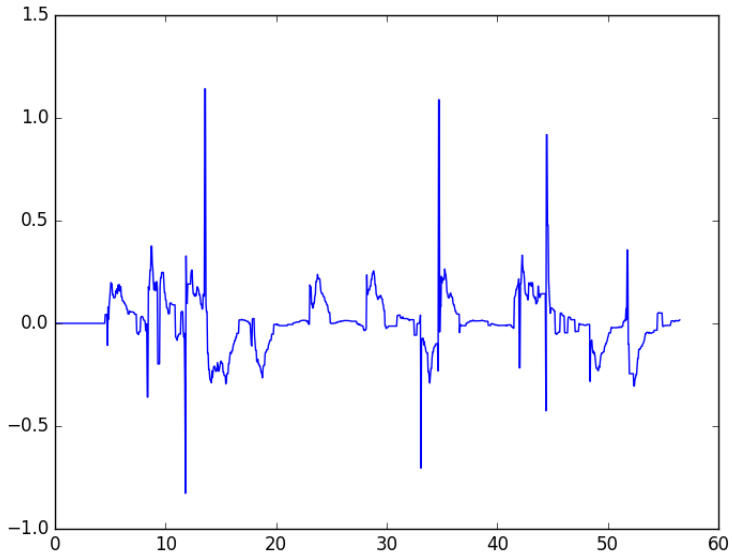


Figure 36 - Γραφική Σφάλματος σε Συνάρτηση με το Χρόνο Αναλογικού Ελεγκτή

Και στις δύο περιπτώσεις παρατηρούνται κάποιες απότομες αλλαγές στα σφάλματα(spikes). Μελλοντικά, ένα spike filter μπορεί να εφαρμοστεί στους ελεγκτές για την εξάλειψη των spikes. Ο αναλογικός ελεγκτής έχει περισσότερες απότομες αλλαγές και λίγο μεγαλύτερα σφάλματα σε κάποια αντίστοιχα σημεία συγκριτικά με τον τριγωνομετρικό. Επειδή οι απότομες αλλαγές του σφάλματος κρατάνε ελάχιστο χρόνο και καταλήγουν κατευθείαν σε μία κοντινή τιμή στο 0 θεωρούνται αμελητέες στο τελικό αποτέλεσμα, όπως και κατά τη κίνηση του οχήματος δεν παρατηρούνται. Οι θετικές τιμές σφαλμάτων δείχνουν ότι το Turtlebot3 έχει μπροστά του αριστερή στροφή ή πρέπει να στραφεί αριστερά για να διορθώσει τη θέση του στο δρόμο. Οι αρνητικές τιμές σφαλμάτων δείχνουν ότι το όχημα μπροστά έχει δεξιές στροφές ή πρέπει να στραφεί δεξιά. Επίσης παρατηρείται ότι μετά από κάθε αλλαγή είτε από τα θετικά ή αρνητικά σφάλματα, το σφάλμα καταλήγει να είναι πολύ κοντά στο 0, πράγμα που σημαίνει ότι ο έλεγχος λειτουργεί αποτελεσματικά και το όχημα δεν χάνει τη πορεία του. Το σφάλμα όπως έχει προαναφερθεί είναι η γωνία φ, που σημαίνει ότι οι απότομες αλλαγές μπορεί να οφείλονται και σε ασταθείς μετρήσεις από τον εντοπισμό των σημείων από τον μετασχηματισμό Hough στο πολύ αρχικό στάδιο. Τέλος το ότι το όχημα δεν χάνει τη πορεία του και πάντα ως σύστημα είναι ευσταθές, σημαίνει ότι τα φίλτρα και ο αλγόριθμος υλοποίησης λειτουργούν αποτελεσματικά. Αν και ο Αναλογικός ελεγκτής θεωρείται ο πιο απλός λειτουργεί εξίσου καλά με τον τριγωνομετρικό. Οι τιμές και των δύο ελεγκτών βρέθηκαν πειραματικά, που αυτό αυτόματα αφήνει περιθώρια για βελτίωση για την εύρεση ακόμη πιο αποδοτικών τιμών.

Αν και οι δύο ελεγκτές δούλεψαν πάρα πολύ αποτελεσματικά η δοκιμή ενός PID κρίνεται απαραίτητη, πράγμα το οποίο δεν έγινε στη παρούσα εργασία. Η εύρεση τριών παραμέτρων είναι αρκετά περίπλοκη διαδικασία όταν πραγματοποιείται πειραματικά. Ιδανικά ο πιο αποτελεσματικός τρόπος ελέγχου ενός συστήματος είναι όταν όλες οι εξισώσεις που διέπουν το σύστημα είναι γνωστές. Όσο πιο λεπτομερείς είναι οι εξισώσεις τόσο πιο μεγάλη πληροφορία λαμβάνεται για το σύστημα. Υπάρχουν αρκετές τεχνικές για την εύρεση των τιμών όταν το σύστημα είναι γνωστό, αλλά πολλές φορές οι εξισώσεις των συστημάτων δεν είναι γραμμικές και για αυτό σε κάποια μέρη του συστήματος χρησιμοποιούνται οι

γνωστές εξισώσεις για την σχεδίαση ενός άρτιου ελεγκτή και σε άλλες εύρεση γίνεται πειραματικά. Επίσης ένα πολύ χρήσιμο εργαλείο που υπάρχει είναι το System Identification του Matlab. Όταν είναι γνωστή η είσοδος και η έξοδος του συστήματος, αυτό προσπαθεί να προσεγγίσει τη συνάρτηση μεταφοράς που διέπει το σύστημα. Στη παρούσα εργασία δε κρίθηκε απαραίτητο γιατί η εύρεση των παραμέτρων των δύο ελεγκτών δεν ήταν αρκετά χρονοβόρα διαδικασία επειδή και στις δύο περιπτώσεις ήταν μία η άγνωστη παράμετρος. Τέλος η προσέγγιση των ευθειών έγινε με τον Hough LinesP. Το OpenCV προσφέρει και άλλη μία βιβλιοθήκη την Hough Circles η οποία μπορεί να δίνει καλύτερα αποτελέσματα στις στροφές, αφού είναι κυρτές και μπορούν να αναπαρασταθούν ως ημικύκλια, από τον Hough Lines. Ίσως ο συνδυασμός και των δύο να αποφέρει το βέλτιστο αποτέλεσμα. Ο Hough lines να εντοπίζει, γενικά τις κάθετες γραμμές και ο Hough Circle τις στροφές.

Κεφάλαιο 7 - Συμπεράσματα

Τελευταία, η ενασχόληση των ερευνητών, που συνδυάζει τις επιστήμες μηχανικών λογισμικού υπολογιστών και κλασσικής μηχανικής, έχουν επικεντρωθεί στη αυτόνομη αυτοκίνηση. Η ανάπτυξη τεχνικών μηχανικής μάθησης, σε συνδυασμό με τη μηχανική όραση και έξυπνων αλγορίθμων πλοήγησης έχουν οδηγήσει όλες τις εταιρίες αυτοκίνησης να επενδύουν κεφάλαια στη βελτίωση τέτοιου είδους συστημάτων. Στην παρούσα εργασία αναπτύχθηκε ένα σύστημα αυτόνομης πλοήγησης ρομποτικού επίγειου οχήματος τύπου Turtlebot3 Burger με χρήση μηχανικής όρασης. Σκοπός του συστήματος είναι η ομαλή διαδρομή του Turtlebot3 μεταξύ των λωρίδων του δρόμου, από την έναρξη της πίστας έως τον τερματισμό. Το σύστημα αποτελείται από το Turtlebot3 Burger και μία κάμερα στο πάνω μέρος του ως αισθητήριο για την κατανόηση του περιβάλλοντός του. Στο σύστημα εισάγεται η εικόνα από τη κάμερα. Έπειτα, γίνεται κατάλληλη επεξεργασία της εικόνας με τη χρήση ψηφιακών φίλτρων. Το τελικό αποτέλεσμα των φίλτρων αποτελεί απαραίτητο στάδιο για την σωστή λειτουργία του αλγόριθμου εντοπισμού των γραμμών και κατά συνέπεια τον υπολογισμό της γωνίας στρέψης. Αυτή η γωνία, δηλαδή της απόκλισης από την επιθυμητή γωνία η οποία είναι $0\,rad$, στέλνεται στον ελεγκτή του οχήματος, ώστε αυτός με τη σειρά του να μηδενίσει το σφάλμα. Το σύστημα ολοκληρώνει τη λειτουργία του και ακινητοποιεί το όχημα όταν δεν εντοπίζονται άλλες γραμμές προς επεξεργασία. Ως λειτουργικό πρόγραμμα χρησιμοποιείται το ROS, το οποίο παρέχει ως ρομποτική πλατφόρμα εξομοίωσης το Turtlebot. Το Turtlebot είναι κατασκευασμένο από την Open Source Robotics Foundation, όπως και το ROS. Για αυτό το λόγο είναι απόλυτα συμβατά μεταξύ τους. Αποτέλεσμα αυτού προκύπτει ότι, πολλές από τις λειτουργίες του Robot για παράδειγμα, ο έλεγχος των κινητήρων, αλγόριθμοι όπως το SLAM, και γενικότερα το μαθηματικό μοντέλο(εξισώσεις κίνησης και δυναμικής) που χρησιμοποιείται, παρέχονται ως έτοιμες συναρτήσεις στον χρήστη. Το πρόγραμμα εξομοίωσης είναι το GAZEBO μέσα στο οποίο αναπτύσσεται όλη η εξομοίωση και χρησιμοποιεί όλα τα προγράμματα και στοιχεία του ROS. Τέλος, ο χάρτης που χρησιμοποιείται στο GAZEBO αποτελεί τροποποίηση του δισδιάστατου καρτεσιανού χάρτη Autorace, που προσφέρει το ROS.

Για την επεξεργασία της εικόνας χρησιμοποιείται αρχικά ένα φίλτρο άσπρου χρώματος, το οποίο απομονώνει τις άσπρες λωρίδες του δρόμου, έπειτα ένα Γκαουσιανό φίλτρο που αφαιρεί τον θόρυβο και ομαλοποιεί την εικόνα, στη συνέχεια ένα φίλτρο ακμών το οποίο είναι απαραίτητο για τη συνάρτηση εντοπισμού γραμμών και ο γεωμετρικός μετασχηματισμός που αυξάνει την αποτελεσματικότητα κίνησης του Robot. Η έξοδος του κάθε φίλτρου είναι η είσοδος για το επόμενο όπου η τελική έξοδος εισάγεται στον μετασχηματισμό Hough Lines ο οποίος εντοπίζει οποιαδήποτε υποψήφια γραμμή και δίνει ως έξοδο τα σημεία των γραμμών. Αυτά υπόκεινται σε επεξεργασία ώστε να βρεθεί μία κεντρική γραμμή ως αποτέλεσμα του μέσου όρου των δύο βασικών γραμμών που έχουν αναπτυχθεί, τη δεξιά και την αριστερή γραμμή. Η τομή της νέας γραμμής με μία κάθετη σταθερή γραμμή που έχει δημιουργηθεί, στο κέντρο του παραθύρου, δίνει μία γωνία η οποία αποτελεί και το σφάλμα γωνίας που καλείται ο ελεγκτής να εξαλείψει. Στην εργασία δοκιμάζονται δύο ελεγκτές. Ο πρώτος είναι ο αναλογικός ελεγκτής P που αποτελεί έναν από τους πιο γνωστούς ελεγκτές και ο τριγωνομετρικός ο οποίος είναι κατάλληλος για περιπτώσεις ελέγχου Robot, που το πρόβλημα προσεγγίζεται με τον υπολογισμό της τρέχουσας γωνίας του οχήματος σε σύγκριση με την επιθυμητή. Και οι δύο ελεγκτές κατέληξαν να είναι πολύ αποδοτικοί και να διορθώνουν επιτυχώς το σφάλμα, με αποτέλεσμα την επιτυχή ολοκλήρωση αυτόνομης πορείας του Robot, ανάμεσα στις γραμμές, στην πίστα εξομοίωσης.

Κεφάλαιο 8 - Προτάσεις για Περαιτέρω Έρευνα

Διαφορετικές μέθοδοι αυτόματης πορείας του οχήματος μπορούν να αναπτυχθούν στο υπάρχον σύστημα της εργασίας. Το σύστημα μπορεί να εκπαιδευτεί από τις ανθρώπινες αντιδράσεις στον έλεγχο του οχήματος με τεχνικές της μηχανικής μάθησης. Επίσης η πίστα μπορεί να βελτιωθεί περαιτέρω προσθέτοντας στοιχεία όπως, φανάρια, πεζούς και διάφορα αντικείμενα, με σκοπό την αποτύπωση μίας ρεαλιστικής εικόνας εξομοίωσης του οχήματος σε συνθήκες λειτουργίας του πραγματικού κόσμου. Επιπλέον η παρούσα εργασία μπορεί να δοκιμαστεί και σε άλλες πίστες ώστε να αποτυπωθεί καλύτερα και η ολιστική λειτουργικότητα του αλγορίθμου. Το πείραμα έλαβε χώρα μόνο σε εξομοίωση. Μελλοντικές δοκιμές μπορούν να γίνουν σε μία πραγματική πίστα για επιβεβαίωση των αποτελεσμάτων λειτουργίας. Το κύριο χαρακτηριστικό της μηχανικής όρασης αποτελεί η θεώρησης της στατικής κατάσταση του περιβάλλοντος ως προς τα χρώματα της εικόνας που εισάγονται στη κάμερα. Οποιαδήποτε αλλαγή ως προς τον χρωματισμό του περιβάλλοντος, αλλάζει και το αποτέλεσμα των εξόδων των φίλτρων. Αν δηλαδή το περιβάλλον αποτελείται από σκοτεινότερο ή φωτεινότερο περιβάλλον αλλάζει αυτομάτως και η απόχρωση του λευκού χρώματος των λωρίδων. Έτσι, κρίνεται απαραίτητη η ανάπτυξη ενός φίλτρου συνεχούς αναπροσαρμογής των χρωμάτων για την ευρωστία του συστήματος.

ΠΙΝΑΚΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος Όρος	**Ελληνικός Όρος**
Messages	Μηνύματα
Nodes	Κόμβος
Topics	Θέματα
Services	Υπηρεσίες
Actions	Δράσεις
Response	Απάντηση
Goal	Στόχος
Result	Αποτέλεσμα
Feedback	Ανάδραση
Callback	Επανάκληση
Service Client	Πελάτης Υπηρεσίας
Service Request	Αίτημα Υπηρεσίας
Local Proxy	Τοπικός διακομιστής
Cmd_vel	Θέμα του ρομποτικού λειτουργικού συστήματος
Terminal	Τερματικός
Parameter Server	Διακομιστής Παραμέτρων
Hardware	Υλικό
Software	Λογισμικό
SLAM	Simultaneous localization and mapping
Unicycle	Μονόροδο
Firmware	Υλικολογισμικό
Blur	Θόλωση
Noise Reduction	Μείωση Θορύβου
Intensity Gradient	Κλίση Έντασης
Pixel	Εικονοστοιχείο
Pins	Καρφίτσες

Camera	Κάμερα
Sigma	Διακύμανση
Non-Maximum Suppression	Μη Μέγιστη Καταστολή
Double Threshold	Διπλό Κατώφλι
Length	Μήκος
Slope	Κλίση
Spikes	Ακραίες Απότομες Αλλαγές σε Τιμές Μεταβλητών
Spike filter	Φίλτρο Ακραίων Απότομων Αλλαγών σε Τιμές Μεταβλητών

ΑΚΡΩΝΥΜΙΑ

OpenCV	Open Source Computer Vision Library
RGB	Red Green Blue Colour Model
OpenManipulator	Open Source Manipulator
SLAM	Simultaneous localization and mapping
SMPS	Power supply and battery charging device
SBC	Single Board Computer
MCU	Microcontroller Unit
LED	Light Emitting Diode
PC	personal computer
KSIZE	Kernel size
e	Error
IMU	Inertial Measurement Unit

Βιβλιογραφία

[1] J. M. O'Kane, A Gentle Introduction to ROS, 2018.

[2] O. S. R. Foundation, «http://wiki.ros.org/actionlib,» [Ηλεκτρονικό].

[3] O. S. R. Foundation, «http://ros.informatik.uni-freiburg.de/roswiki/Master.html,» [Ηλεκτρονικό].

[4] O. S. R. Foundation, «http://wiki.ros.org/Parameter%20Server,» [Ηλεκτρονικό].

[5] O. S. R. Foundation, «http://gazebosim.org/tutorials?cat=guided_b&tut=guided_b1,» [Ηλεκτρονικό].

[6] O. S. R. R. Foundation, «https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#turtlebot,» [Ηλεκτρονικό].

[7] R. Open Source Robotics Foundation, «https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/#hardware-specifications,» [Ηλεκτρονικό].

[8] O. team, «https://opencv.org/about/,» [Ηλεκτρονικό].

[9] D. R. Szeliski, Computer Vision, Algorithms and Applications, 2011.

[10] A. K. R. Tinku Acharya, Image Processing: Principles and Applications, Wiley-Interscience; 1st Edition, 2005.

[11] J. F. Canny, «A Computational Approach To Edge Detection,» *IEEE Trans. Pattern Analysis and Machine Intelligence,* Τόμ. %1 από %2vol. PAMI-8, pp. 679-697, 1986.

[12] R. Szeliski, Computer Vision, Algorithms and Applications, 2011.

[13] R. O. a. P. E. H. ". o. t. H. T. t. D. L. a. C. i. P. C. A. V. 1. p. 1. (. 1. Duda, «Use of the Hough Transformation to Detect Lines and Curves in Pictures,» *Comm. ACM,* τόμ. 15, p. 11–15, January, 1972.

[14] G. C. Y. L. Kiam Heong Ang, «PID Control System Analysis, Design, and Technology,» 13 November 2007.

[15] T. Bower, «http://faculty.salina.k-state.edu/tim/robotics_sg/Control/controllers/trig_trick.html,» [Ηλεκτρονικό].