ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία

Development of an EtherCAT-based motion control system and a semantic segmentation yaw control algorithm for quadruped robots

Ανάπτυξη ενός συστήματος ελέγχου κίνησης βασισμένο σε EtherCAT και ενός αλγόριθμου ελέγχου κατεύθυνσης μέσω τμηματοποίησης εικόνας για τετράποδα ρομπότ

Νεοκλής Βαϊνδηρλής

Επιβλέπων Καθηγητής: Ε. Γ. Παπαδόπουλος

Αθήνα, Ιούνιος 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία

Development of an EtherCAT-based motion control system and a semantic segmentation yaw control algorithm for quadruped robots

Ανάπτυξη ενός συστήματος ελέγχου κίνησης βασισμένο σε EtherCAT και ενός αλγόριθμου ελέγχου κατεύθυνσης μέσω τμηματοποίησης εικόνας για τετράποδα ρομπότ

Νεοκλής Βαϊνδηρλής

Επιβλέπων Καθηγητής: Ε. Γ. Παπαδόπουλος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Ιουνίου 2023

..……	……...…….	……
Ε. Παπαδόπουλος	Δ. Σούντρης	Π. Σωτηριάδης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2023

…….
Νεοκλής Βαϊνδηρλής
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η εργασία αυτή αποτελείται από δύο μέρη. Στο πρώτο μέρος της εργασίας έγινε αναβάθμιση των ηλεκτρονικών και ηλεκτρικών υποσυστημάτων του τετράποδου ρομπότ Laelaps. Στο δεύτερο και κύριο μέρος της εργασίας έγιναν τα πρώτα βήματα για την ανάπτυξη ενός αλγόριθμου ελέγχου της κατεύθυνσης ενός τετράποδου γεωργικού ρομπότ με την χρήση νευρωνικών δικτύων βαθιάς μάθησης. Το ρομπότ αναπτύσσεται υπό την επίβλεψη του καθηγητή Ευάγγελου Παπαδόπουλου, στο εργαστήριο Αυτομάτου Ελέγχου της Σχολής Μηχανολόγων Μηχανικών Ε.Μ.Π.

Στο πρώτο μέρος της εργασίας περιγράφεται η διαδικασία σχεδιασμού και υλοποίησης των νέων ηλεκτρονικών και ηλεκτρικών υποσυστημάτων του τετράποδου ρομπότ. Η νέα αρχιτεκτονική έγινε με γνώμονα την περαιτέρω βελτίωση των χρόνων απόκρισης του συστήματος Ethercat, καθώς και την βελτίωση της ηλεκτρικής σταθερότητας και της γενικότερης αξιοπιστίας του συστήματος ελέγχου των ηλεκτρικών κινητήρων.

Στο δεύτερο μέρος της εργασίας περιγράφεται η διαδικασία σχεδιασμού και υλοποίησης ενός αυτόνομου συστήματος πλοήγησης για γεωργικά ρομπότ ικανά να κινούνται μέσα σε ένα αμπέλι με την βοήθεια νευρωνικών δικτύων βαθιάς μάθησης. Εν ολίγοις, μέσω μίας κάμερας καταγράφεται η εικόνα του αμπελιού μπροστά από το ρομπότ και σε πραγματικό χρόνο μετατρέπεται μέσω του μοντέλου βαθιάς μάθησης σε μία τμηματοποιημένη εικόνα που χρησιμοποιείται για τον έλεγχο της κατεύθυνσής του.

Αρχικά υλοποιήθηκε το μοντέλο μηχανικής μάθησης στο διαδεδομένο περιβάλλον μηχανικής μάθησης PyTorch. Έπειτα εφαρμόστηκαν διάφορες τεχνικές με σκοπό την βελτίωση της ακρίβειας του μοντέλου. Για την επικύρωσή της ορθής λειτουργίας του συστήματος, μέσω πρωτότυπων τεχνικών, δημιουργήθηκε ένα σύνολο δεδομένων εικόνων, από μία ειδικά διαμορφωμένη αίθουσα με τεχνητό αμπέλι, πάνω στο οποίο εκπαιδεύτηκε το μοντέλο.

Έπειτα, σχεδιάστηκε και επικυρώθηκε ένας αλγόριθμος εξαγωγής του σφάλματος απόκλισης της πορείας του ρομπότ από την τμηματοποιημένη εικόνα. Ο αλγόριθμός αυτός πραγματοποιεί ακόμα έλεγχο της κατεύθυνσης του, βάση του σφάλματος αυτού. Η ανάπτυξή του έγινε σε μία προϋπάρχουσα ρομποτική πλατφόρμα και με την βοήθεια του τεχνητού αμπελιού. Τέλος υλοποιήθηκε και μία απλή διαδικασία αναγνώρισης και αλλαγής σειράς αμπελιών.

Λέξεις Κλειδιά

Τετράποδα Ρομπότ, Μηχανική Μάθηση, Τμηματοποίηση Εικόνας, Έλεγχος Κατεύθυνσης, Ηλεκτρονικό Σύστημα, Σχεδιασμός Συστήματος, Ενσωματωμένα Συστήματα, Μικροελεγκτής, F28388D, Σχεδιασμός PCB

Abstract

This thesis consists of two parts. In the first part, upgrades were made to the electronic and electrical subsystems of the quadrupedal robot Laelaps. The second part involved the development of an algorithm for controlling the direction of an agricultural robot using deep learning neural networks. The robot is being developed under the supervision of Professor Evangelos Papadopoulos, in the Automatic Control Systems Laboratory of the School of Mechanical Engineering National Technical University of Athens.

Regarding the first part of the thesis, the process of designing and implementing part of the new electronic and electrical subsystems of the quadruped robot is analyzed. The main design goal of the new architecture was improving response times of the EtherCAT system, as well as improving the electrical stability and the reliability of the motion control system.

The aim of the second part of the thesis is to describe the design and implementation process of an autonomous navigation system using deep learning for agricultural robots in vineyards. In short, an image of the vine in front of the robot is captured and transformed in real time, through a deep learning model, into a segmented image used to control its direction.

The machine learning model was implemented in the popular machine learning environment PyTorch. Then various techniques were applied in order to improve the accuracy of the model. To validate the correct operation of the system, through novel techniques, a dataset of images was created, from a specially configured room with a synthetic vineyard, on which the model was trained.

Then, an algorithm to extract the robot path deviation error from the segmented image was designed and validated. This algorithm also controls the robot's direction based on this error. Its development was mainly done on a pre-existing robotic platform and with the help of the synthetic vineyard. Finally, a simple procedure for identifying and changing the row in a vineyard was implemented.

Keywords

Legged Robots, Machine Learning, Image Segmentation, Yaw Control, Electronic System, System Design, Integrated Systems, Microcontroller, F28388D, PCB Design

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά όλους όσους με στήριξαν στην εκπόνηση της διπλωματικής μου εργασίας. Πρώτα από όλους, θα ήθελα να ευχαριστήσω ιδιαιτέρως τον επιβλέποντα καθηγητή κ. Ευάγγελο Παπαδόπουλο για την εμπιστοσύνη που μου έδειξε, δίνοντάς μου την ευκαιρία να συμμετάσχω σε διάφορες εργασίες και να πραγματοποιήσω τη διπλωματική εργασία μου στο εργαστήριο Αυτομάτου Ελέγχου, από τα πρώτα κιόλας έτη της ακαδημαϊκής μου πορείας.

Ένα μεγάλο ευχαριστώ οφείλω στους υποψήφιους διδάκτορες Κώστα Κουτσούκη, Θανάση Μαστρογεωργίου και ιδιαιτέρως στον Κωνσταντίνο Μαχαιρά για την άριστη συνεργασία που είχαμε, καθώς και για την εμπιστοσύνη που μου έδειξαν. Από τις πρώτες μου κιόλας εργασίες στο εργαστήριο ο Κ. Μαχαιράς συγκεκριμένα μου έμαθε πώς να λειτουργώ σε μεγάλα και απαιτητικά projects και ήταν πάντα πρόθυμος να με βοηθήσει σε οποιαδήποτε δυσκολία αντιμετώπιζα. Ακόμα θα ήθελα να συγχαρώ τον Γιώργο Μπολανάκη και ιδαιτέρως τον Αριστοτέλη Παπαθεοδώρου, για την δουλειά που κάνανε πάνω στο ηλεκτρονικό συστημα του laelaps, και κυρίως για την αναλυτική επεξήγηση που είχαν και οι 2 στις διπλωματικές τους εργασίες, ώστε να με βοηθήσουν να κατανοήσω την περίπλοκη δομή του και να την εξελίξω με όσες λιγότερες δυσκολίες γινόταν.

Επίσης θα ήθελα να ευχαριστήσω τον καθηγητή κ. Λεωνίδα Αλεξόπουλο που με έφερε σε επαφή όχι μόνο με το δικό του εργαστήριο, αλλά και του κ. Παπαδόπουλου και γενικότερα για τις συμβουλές του και την καθοδήγηση που μου προσέφερε όλα αυτά τα χρόνια. Τέλος, το μεγαλύτερο ευχαριστώ το οφείλω στην οικογένειά μου, για την καθημερινή, υλική και ηθική, στήριξη καθ’ όλη τη διάρκεια των σπουδών μου.

Αφιερώνεται στην γιαγιά μου Λεμονιά

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1 Εισαγωγή

1.1 Σκοπός Εργασίας

Σκοπός του πρώτου μέρους της εργασίας ήταν αρχικά η βελτίωση των χρόνων απόκρισης του συστήματος EtherCAT με ανανεωμένο υλικό, διατηρώντας όμως τις ίδιες τεχνικές σχεδιασμού κυκλωμάτων και ανάπτυξης του λογισμικού. Αυτό επιτρέπει μία ομοιογένεια μεταξύ των εκδόσεων του τετράποδου ρομπότ Laelaps, εξασφαλίζοντας ότι ο εκάστοτε χρήστης που αναπτύσσει οποιοδήποτε μέρος του συστήματός του, δεν θα χαθεί μέσα στην μεγάλη πολυπλοκότητά του. Οι καινούριες πλακέτες που θα χρησιμοποιηθούν απαιτούν κάποιες ιδιαίτερες τεχνικές κόλλησης της οποίες θα εξερευνήσουμε, καθώς δεν έχουν χρησιμοποιηθεί αντίστοιχου τύπου στο παρελθόν. Επίσης σκοπός είναι να αντιμετωπισθεί και το χαμηλό όριο ρεύματος των ηλεκτρονικών οδήγησης των ηλεκτρικών κινητήρων, λόγω υπερθέρμανσης, με αναβάθμισή τους σε πιο ισχυρούς και παράλληλα με προσθήκη λειτουργιών καταγραφής του ρεύματος κατανάλωσης των κινητήρων.

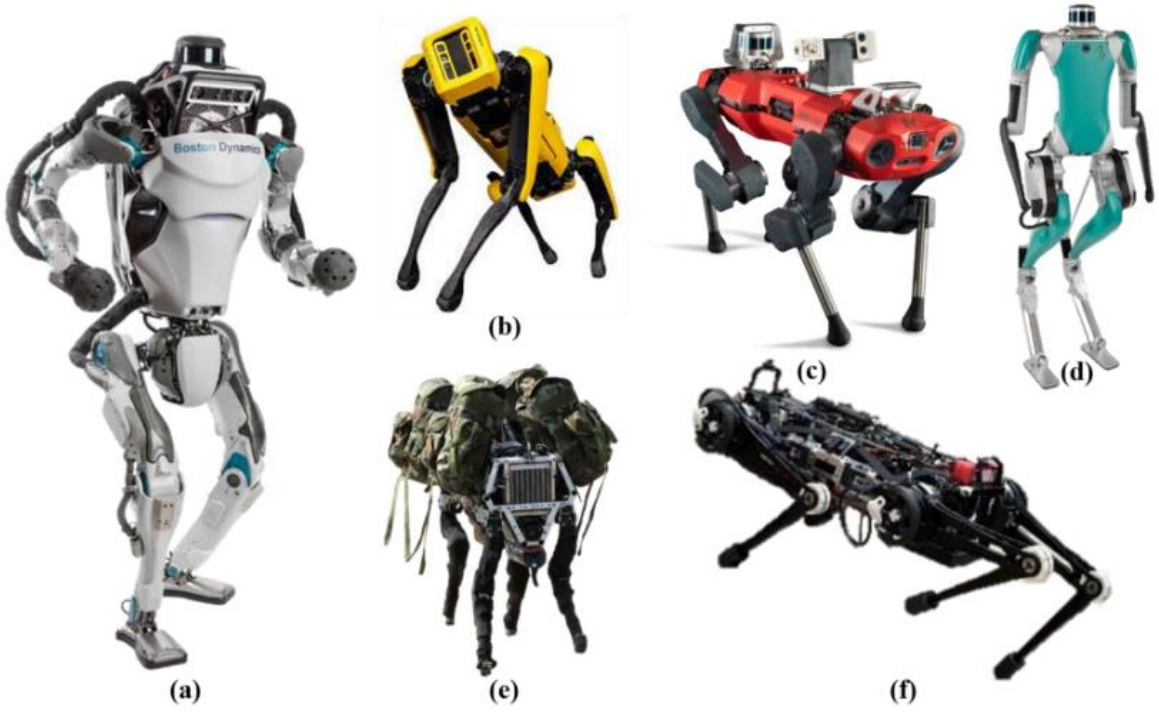
Στο δεύτερο μέρος της εργασίας σκοπός ήταν η κίνηση ενός ρομπότ μέσα στο τεχνητό αμπέλι του εργαστηρίου, αποκλειστικά με την χρήση τμηματοποιημένης εικόνας. Τα γεωργικά ρομπότ παραδοσιακά βασίζονταν σε συνδυασμό πολλαπλών ακριβών αισθητήρων για την καθοδήγησή τους, όπως Lidar και GPS υψηλής ακριβείας. Στόχος εδώ είναι το ρομπότ να μην χρησιμοποιεί ακριβούς αισθητήρες, εκμεταλλευόμενο την όραση υπολογιστή μέσω της μηχανικής μάθησης.

Απώτερος στόχος της μελέτης είναι η εφαρμογή του αλγορίθμου πλοήγησης στο τετράποδο ρομπότ Laelaps, στην κίνηση του οποίου αφιερώνεται και το πρώτο μέρος της εργασίας. Ως πρώτο όμως βήμα, σε αυτή την εργασία αξιολογείται η ιδέα και η πρώτη υλοποίηση του αλγορίθμου ελέγχου κατεύθυνσης σε μία απλούστερη ρομποτική πλατφόρμα η οποία επιτρέπει κίνηση σε κάθε κατεύθυνση χρησιμοποιώντας τροχούς τύπου “mecanum” και είναι ό,τι πιο κοντά υπάρχει στις δυνατότητες κίνησης ενός τετράποδου ρομπότ. Το τελικό βήμα της αξιοποίησης του αλγορίθμου στο τετράποδο Laelaps αλλά και σε αλλά τετράποδα ρομπότ της ομάδας αφήνεται ως μελλοντική εργασία.

Όσον αφορά την τμηματοποίηση εικόνας, σκοπός είναι η εύρεση ενός αποδοτικού μοντέλου μηχανικής μάθησης και η εκπαίδευσή του με βέλτιστο και συστηματικό τρόπο, ώστε να είναι εύκολη η περαιτέρω ανάπτυξη αυτού του εγχειρήματος από τους επόμενους μετά το πέρας αυτής της εργασίας. Πέρα από την σωστή παραμετροποίηση του, η σημαντικότερη προϋπόθεση για την σωστή λειτουργία του είναι η ποιότητα και η ποσότητα των δεδομένων εκπαίδευσης. Σε αυτή την εργασία λοιπόν, θα ασχοληθούμε και με την καταγραφή δεδομένων εικόνας στον χώρο του αμπελιού και την δημιουργία των τελικών δεδομένων εκπαίδευσης με τον πιο αποδοτικό χρονικά και ποιοτικά τρόπο, καθώς αυτή είναι μία πολύ χρονοβόρα διαδικασία που απαιτεί ιδιαίτερη λεπτομέρεια. Τέλος θα ασχοληθούμε με την εξερεύνηση διάφορων τρόπων που μπορεί να χρησιμοποιηθεί η τμηματοποιημένη εικόνα για τον έλεγχο της κατεύθυνσης του ρομπότ. Αυτό απαιτεί ανάλυση των υπαρχουσών τεχνικών που χρησιμοποιούνται καθώς και εξερεύνηση νέων μεθόδων, μέσω πειραμάτων στο τεχνητό αμπέλι. Σκοπός είναι να βρεθεί η τεχνική με την καλύτερη επίδοση και αξιοπιστία.

1.2 Βιβλιογραφική Ανασκόπηση

Τα τελευταία χρόνια έχει αυξηθεί σε μεγάλο βαθμό το ερευνητικό ενδιαφέρον του ευρύτερου τομέα της ρομποτικής στα τετράποδα ρομπότ. Με τη βοήθεια των τεσσάρων ποδιών τους, μπορούν να αντιμετωπίσουν ανώμαλα εδάφη, να ανεβαίνουν σκάλες και να ξεπερνούν άλλα εμπόδια με ασφάλεια και ευκολία, εκεί όπου άλλες αρχιτεκτονικές θα αποτύγχαναν ή θα καθυστερούσαν ιδιαίτερα. Επιπλέον, τα πόδια τους επιτρέπουν ευελιξία και προσαρμοστικότητα στο περιβάλλον. Αυτό τους επιτρέπει να πλοηγούνται σε στενούς χώρους, να επιδίδονται σε εξερευνητικές αποστολές και να προσαρμόζονται σε διάφορες εργασίες, όπως αναζήτηση και διάσωση, γεωργία, κατασκευές και περιβαλλοντική έρευνα [9] . Με αυτά τα πλεονεκτήματα, τα τετράποδα ρομπότ αποτελούν ισχυρούς και ευέλικτους συνεργάτες στον χώρο της ρομποτικής. Γενικότερα, όλο ένα και περισσότερα ακαδημαϊκά ιδρύματα και εταιρείες τεχνολογίας δημιουργούν τετράποδα ρομπότ προσπαθώντας να αντιμετωπίσουν τα βασικότερα προβλήματα. Τα σχέδια λοιπόν βελτιώνονται σταδιακά με στόχο την αποτελεσματική ενσωμάτωση στις εφαρμογές που αναφέρθηκαν. Τα ρομπότ Atlas [19] , Spot [20] και BigDog [21] της Boston Dynamics, μαζί με άλλους, όπως το ANYmal της Anybotics και τον πρόσφατο διάδοχό του ANYmal C [22] , το MIT Cheetah [23] και το πιο πρόσφατο δίποδο DIGIT της Agility [24] , είναι μερικές απόπειρες τελευταίας τεχνολογίας που καταδεικνύουν τις τρέχουσες ερευνητικές τάσεις (Σχήμα 1-1).



Σχήμα 1-1. (a) Boston Dynamics Atlas. (b) Boston Dynamics Spot. (c) ANYbotics ANYmal-C. (d) Agility Robotics DIGIT. (e) Boston Dynamics BigDog. (f) MIT Cheetah 3.

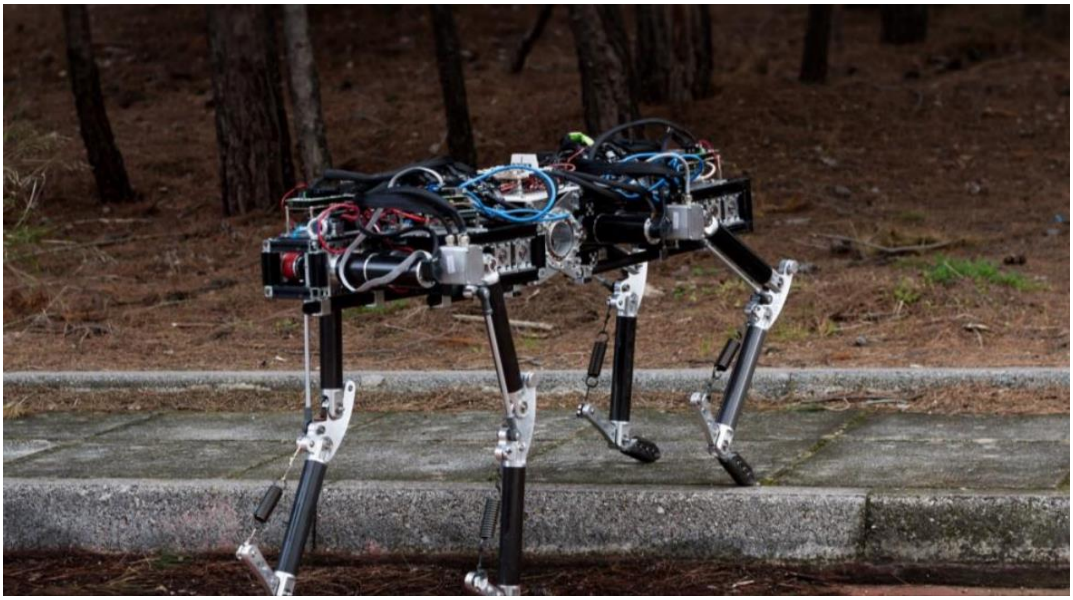
Ειδικότερα στον χώρο της γεωργίας όπου έχουμε ανώμαλο έδαφος και μεταβαλλόμενες συνθήκες τα τετράποδα ρομπότ έχουν αρχίσει και εξερευνώνται όλο και περισσότερο τελευταία [10] . Ο καθορισμός της γραμμής πλοήγησης είναι κρίσιμος για την αυτόνομη πλοήγηση αγροτικών ρομπότ στην γεωργία. Η τμηματοποίηση εικόνας σε πραγματικό χρόνο είναι μια εξαιρετική προσέγγιση για την επίλυση αυτού του προβλήματος, καθώς προσαρμόζεται στη δυναμική φύση του αγροτικού περιβάλλοντος και δεν βασίζεται σε

πολύπλοκο υλικό. Δημοφιλείς βιβλιοθήκες βαθιάς μάθησης όπως η PyTorch περιλαμβάνουν αλγόριθμους τμηματοποίησης υψηλής ακρίβειας [1] που μπορούν να εκπαιδευτούν και να χρησιμοποιηθούν για αγροτική πλοήγηση. Ως εκ τούτου, πολυάριθμες δημοσιεύσεις έχουν κυκλοφορήσει πρόσφατα χρησιμοποιώντας με επιτυχία τη σημασιολογική τμηματοποίηση ειδικά σε γεωργικά ρομπότ. Οι Xia Li και άλλοι χρησιμοποίησαν μια απλή κάμερα RGB και το Faster-U-net CNN (Convolutional Neural Network), εστιάζοντας στον ταχύτερο χρόνο επεξεργασίας και πέτυχαν ακρίβεια pixel > 0,89 σε διάφορες καλλιέργειες. Οι Yan Song και άλλοι [4] χρησιμοποίησαν μια κάμερα Kinect 2.0 RGB-D και ένα προσαρμοσμένο FCN (Fully Convolutional Network) βασισμένο στο VGG16 και πέτυχαν ακρίβεια pixel > 0,92, συγκρίνοντας την ακρίβεια μεταξύ βάθους και δεδομένων RGB. Οι Diego Aghi και άλλοι ακολούθησαν επίσης μια παρόμοια προσέγγιση χρησιμοποιώντας μια κάμερα Intel Realsense d435i RGB-D και ένα προσαρμοσμένο CNN βασισμένο στο MobileNet-v3, με ακρίβεια pixel 0,92. Φυσικά, έλεγχος με τμηματοποίηση εικόνας έχει γίνει και σε τετράποδα ρομπότ από τους Ganyu Deng και άλλους. Οι συγκεκριμένοι έλεγξαν ένα τετράποδο ρομπότ με αλγόριθμο path-planning βασισμένο σε τμηματοποίηση εικόνας όπως και οι υπόλοιποι αλλά χρησιμοποίησαν παράλληλα και έναν αλγόριθμο διόρθωσης σε περίπτωσης σφάλματος του δικτύου λόγω περιορισμένης εκπαίδευσης.

2 Ανάπτυξη συστήματος ελέγχου κίνησης βασισμένο σε EtherCAT

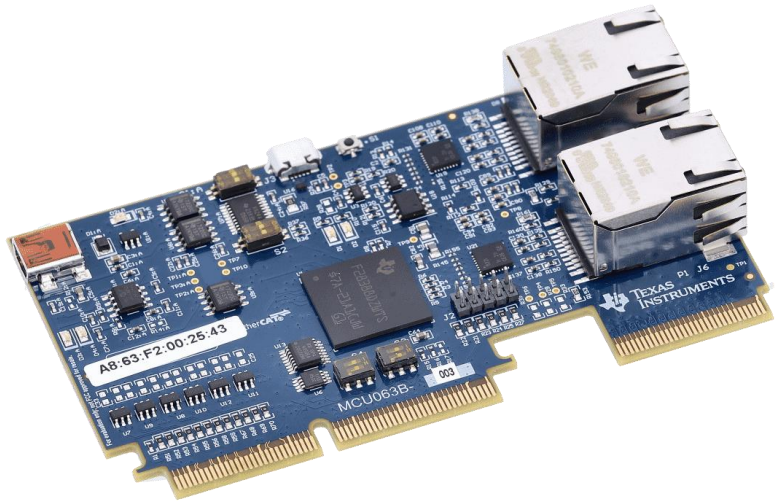
2.1 Εισαγωγή

Σκοπός του πρώτου μέρους της παρούσας διπλωματικής εργασίας ήταν ο σχεδιασμός και η υλοποίηση του νέου ηλεκτρονικού συστήματος του τετράποδου ρομπότ πολλαπλών αρθρώσεων «Λαίλαψ» (Σχήμα 2-1). Το συγκεκριμένο ρομπότ έχει σχεδιάστει εξ' ολοκλήρου στο εργαστήριο αυτόματου ελέγχου του κ. Παπαδόπουλου της σχολής Μηχανολόγων Μηχανικών Ε.Μ.Π. [14] από διπλωματούχους καθώς και μεταπτυχιακούς φοιτητές [17] [18] [16] [15] [6] Έγινε προσπάθεια να μην επεκταθούμε σε περιττές πληροφορίες και να εστιάσουμε στις μεθοδολογίες που χρησιμοποιήθηκαν, ώστε η εργασία που εκπονήθηκε να αποτελεί βοήθημα για τη συστηματική μελέτη, τον σχεδιασμό και την υλοποίηση των ηλεκτρονικών συστημάτων του τετράποδου ρομπότ.



Σχήμα 2-1. Το τετράποδο ρομπότ Laelaps.

Η νέα έκδοση του Laelaps διαθέτει πιο ισχυρούς μικροελεγκτές (MCUs) και ελεγκτές κινητήρων (motor controllers) εξασφαλίζοντας πιο γρήγορη και αξιόπιστη λειτουργία του συστήματος. Ο καινούριος F28388D MCU της Texas Instruments έχει ενσωματωμένη μονάδα επικοινωνίας Ethercat μειώνοντας σημαντικά τον λανθάνοντα χρόνο επικοινωνίας (latency). Όλες οι λειτουργίες του συγκεκριμένου MCU εξυπηρετούνται μέσω ενός edge connector που συνδέεται στην κεντρική πλακέτα σαν μία τυπική desktop κάρτα γραφικών όπως φαίνεται και στο Σχήμα 2-2. Απαιτούνται λοιπόν νέες στρατηγικές σχεδίασης που θα αναλυθούν στο παρακάτω κεφάλαιο. Επιπρόσθετα χρησιμοποιείται ο καινούριος AZBDC20A8 motor controller που διαθέτει βελτιωμένες προδιαγραφές ανώτατου ρεύματος, ελαχιστοποιώντας τις πιθανότητες υπερθέρμανσης και συνεπακόλουθης μείωσης της απόδοσης του ελέγχου των κινητήρων.



Σχήμα 2-2. Ο καινούριος F28388D MCU που χρησιμοποιείται στην νέα έκδοση του Laelaps.

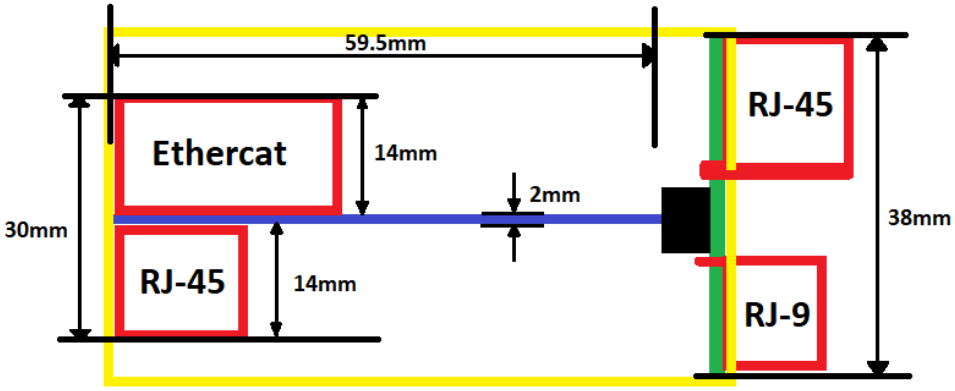
2.2 Η πλακέτα επέκτασης της F28388D control CARD

2.2.1 Βελτιστοποίηση της συναρμολόγησης των πλακετών ως προς τον χώρο

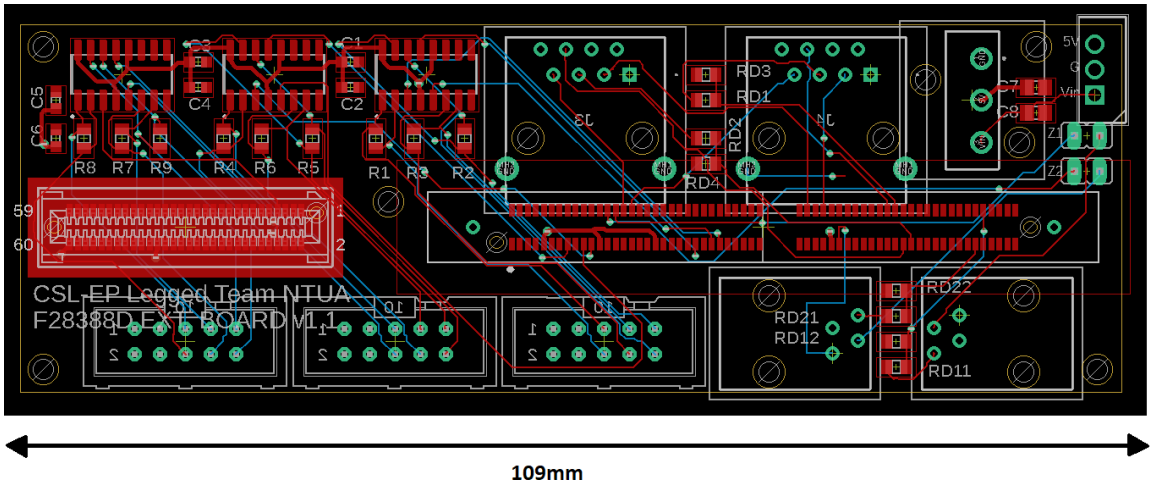
Η πλακέτα επέκτασης έχει σχεδιαστεί για την διασύνδεση κάθε MCU με τους motor controllers και τους κωδικοποιητές (encoders) που απαιτούνται για τον έλεγχο των κινητήρων του ισχίου και του γόνατος σε κάθε ένα από τα τέσσερα άκρα. Η επικοινωνία με κάθε AZBDC20A8 motor controller πραγματοποιείται με καλώδιο τύπου RJ-45, όπου η ζητούμενη ροπή του κινητήρα μεταφέρεται μέσω ενός σήματος PWM. Κάθε άρθρωση διαθέτει έναν HEDS-5540 rotary encoder (αυξητικό κωδικοποιητή) ο οποίος συνδέεται μέσω ενός 10-pin IDC connector, καθώς και έναν απόλυτο κωδικοποιητή RMF44VE10BA10 magnetic absolute encoder [30] ο οποίος συνδέεται με καλώδιο τύπου RJ-9.

Όπως αναφέρθηκε, ο καινούριος MCU [27] συνδέεται με τελείως διαφορετική διάταξη, όπως μία κάρτα γραφικών, σε αντίθεση με τον προηγούμενο [6] (LAUNCHXL-F28379D), ο οποίος είχε κλασσικές σειρές από headers. Έχοντας αυτόν τον περιορισμό υπ' όψη, ο κύριος στόχος του σχεδιασμού της νέας πλακέτας επέκτασης είναι να ελαχιστοποιηθεί ο συνολικός όγκος της συναρμολόγησης. Αυτό επιτεύχθηκε συνδέοντας **κάθετα** την εμπρόσθια όψη της πλακέτας επέκτασης με τον MCU. Τοποθετώντας όλους τους κονέκτορες στην οπίσθια όψη, εξοικονομούμε πολύ χώρο, καθώς δεν απαιτείται περισσότερο πλάτος, όπως φαίνεται και στο Σχήμα 2-3.

Η πλακέτα επέκτασης έχει συνολικό μήκος 109mm, όπως φαίνεται και στο Σχήμα 2-4, όντας ελάχιστα μεγαλύτερη από την κάρτα MCU (106.5mm). Όσον αφορά το συνολικό ύψος της κατασκευής, αθροίζοντας το πλάτος των κονεκτόρων RJ-45, εκατέρωθεν της κάρτας MCU καθώς και το πάχος του PCB, το συνολικό ύψος του είναι 30mm, όπως φαίνεται και αριστερά στο Σχήμα 2-3. Παρόλα αυτά η πλακέτα επέκτασης έχει πλάτος 38mm (8mm παραπάνω) καθώς αυτό είναι το ελάχιστο πλάτος ώστε να μην συγκρούονται οι κονέκτορες RJ-45, Edge, RJ-9, όπως φαίνεται και δεξιά στο Σχήμα 2-3. Η συνολική διάσταση του συναρμολογημένου συστήματος είναι 109x38x83.



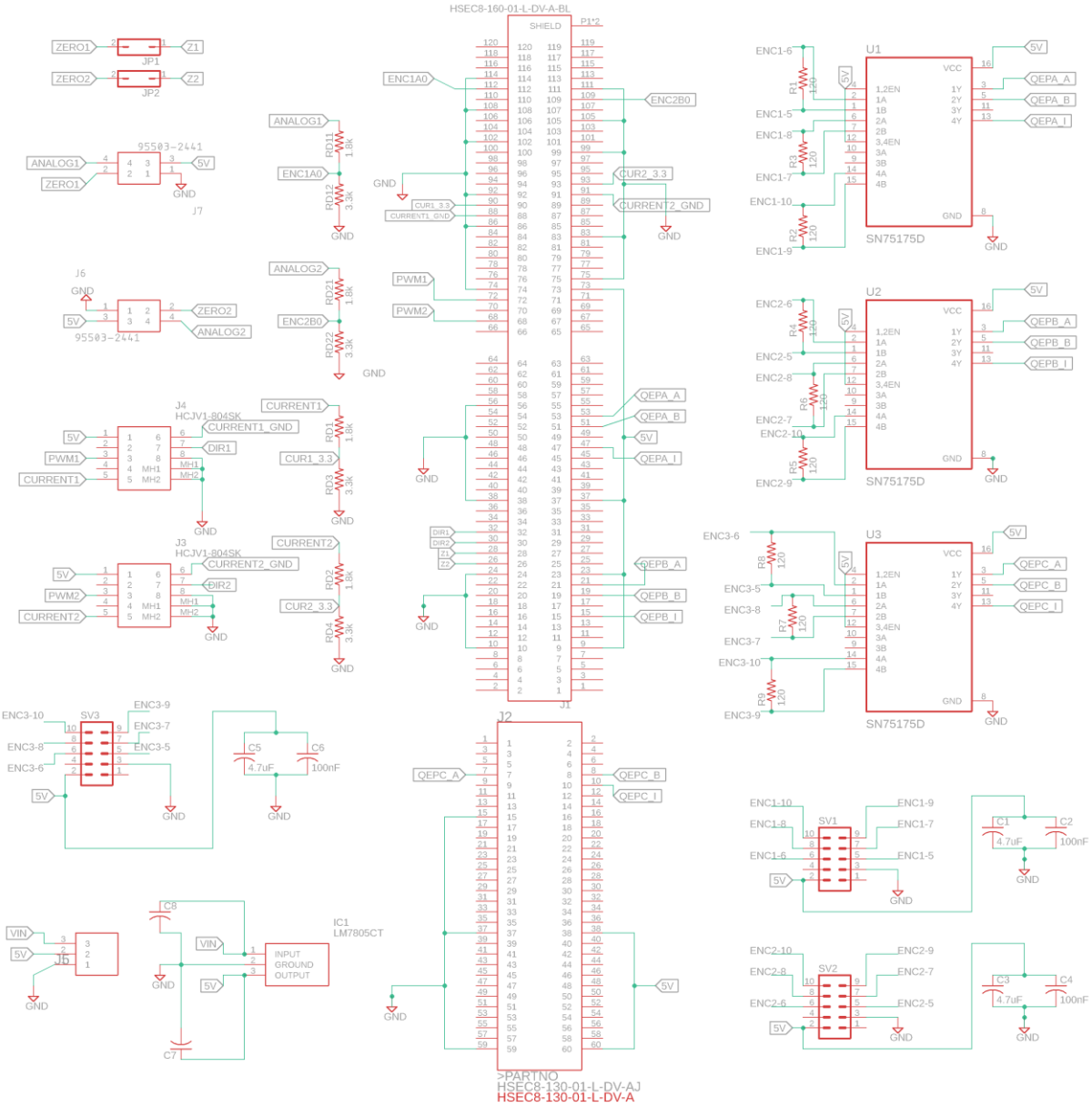
Σχήμα 2-3. Πλάγια όψη του συναρμολογήματος.



Σχήμα 2-4. Κάτοψη της πλακέτας επέκτασης.

2.2.2 Σχεδιασμός πλακέτας επέκτασης

Για την ελαχιστοποίηση του ηλεκτρομαγνητικού θορύβου μεταξύ των εξαρτημάτων στην πλακέτα, αλλά και την απλοποίηση του σχεδίου χρησιμοποιήθηκαν **ground (GND) και VCC planes** στην εμπρόσθια και στην οπίσθια όψη του PCB αντίστοιχα. Το διάγραμμα της πλακέτας φαίνεται στο Σχήμα 2-4. Όλες οι συνδέσεις μεταξύ της κάρτας MCU και των υπόλοιπων εξαρτημάτων βρίσκονται στον Πίνακα 2-1. Το πλήθος των εξαρτημάτων που χρειάζονται για την κατασκευή της πλακέτας βρίσκονται στον Πίνακα 8-1.



Σχήμα 2-5. Διάγραμμα κυκλώματος πλακέτας επέκτασης F28388D.

Πίνακας 2-1. Συνδέσεις της κάρτας MCU.

Function	HSEC pin	MCU pin
ANALOG ENCODER 1	9	ADC-A0
ANALOG ENCODER 2	12	ADC-B0
CURRENT MONITOR 1	31	ADC-C2
CURRENT MONITOR 1 GND	33	ADC-C3
CURRENT MONITOR 2	28	ADC-D0
CURRENT MONITOR 2 GND	30	ADC-D1
MOTOR CONTROLLER 1 PWM	49	PWM1A
MOTOR CONTROLLER 2 PWM	51	PWM1B
MOTOR CONTROLLER 1 DIR	89	GPIO40
MOTOR CONTROLLER 2 DIR	91	GPIO41

INC.ENCODER 1 A	68	GPIO20/QEPA-A
INC.ENCODER 1 B	70	GPIO21/QEPA-B
INC.ENCODER 1 INDEX	74	GPIO23/QEPA-I
INC.ENCODER 2 A	100	GPIO54/QEPB-A
INC.ENCODER 2 B	102	GPIO55/QEPB-B
INC.ENCODER 2 INDEX	106	GPIO57/QEPB-I
INC.ENCODER 3 A	127	GPIO62/QEPC-A
INC.ENCODER 3 B	128	GPIO63/QEPC-B
INC.ENCODER 3 INDEX	130	GPIO65/QEPC-I

2.2.3 Τροφοδοσία της πλακέτας επέκτασης

Όσον αφορά την τροφοδοσία της πλακέτας, παρόλο που η τάση λειτουργίας της είναι 3.3V, η τάση της τροφοδοσίας εισόδου της πλακέτας πρέπει να βρίσκεται στα 5, καθώς διαθέτει δικό της ενσωματωμένο ρυθμιστή τάσης ο οποίος αναλαμβάνει την ρύθμισή στα 3.3V. Συνεπώς δεν απαιτείται κάποιος εξωτερικός ρυθμιστής τάσης 3.3V όπως στην προηγούμενη πλακέτα, ούτε κάποιο 3.3V enable/disable jumper [6] . Έτσι η κάρτα MCU, καθώς και τα υπόλοιπα εξαρτήματα στην πλακέτα επέκτασης τα οποία έχουν επίσης τάση λειτουργίας 5V, τροφοδοτούνται από ένα γραμμικό ρυθμιστή τάσης (LDO) 5V LM7805, ο οποίος έχει μέγιστη τάση εισόδου 35V επιτρέποντας απευθείας τροφοδοσία της πλακέτα από την κεντρική πηγή τροφοδοσίας η οποία έχει τάση λειτουργίας 10V.

Παρόλο που αυτός ο ρυθμιστής, λόγω της φύσης του δεν είναι όσο αποδοτικός, όσο ένας διακοπτόμενος ρυθμιστής τάσης (switching regulator), έχει πολύ μικρότερα επίπεδα θορύβου. Το Laelaps βρίσκεται ακόμα σε στάδια ανάπτυξης, επομενως θελουμε να ελαχιστοποιήσουμε τυχον προβλημάτα που μπορεί να την καθυστερησουν. Η λειτουργία διάφορων ευαίσθητων εξαρτημάτων, όπως αναλογικά και ασύρματα συστήματα που είναι ιδιαίτερα επιρρεπή, απαιτεί πηγή τάσης με τον δυνατό λιγότερο θόρυβο. Στην προηγούμενη του έκδοση υπήρχε μόνο η δυνατότητα τροφοδοσίας από switching regulators, οι οποίοι αν και έχουν αναπτυχθεί ιδιαίτερα τα τελευταία χρόνια, από την φύση τους έχουν ακόμα κάποιο θόρυβο. Θέλοντας να εξασφαλίσουμε τα χαμηλότερα δυνατά επίπεδα θορύβου, αυτό επιτυγχάνεται με γραμμικό ρυθμιστή τάσης.

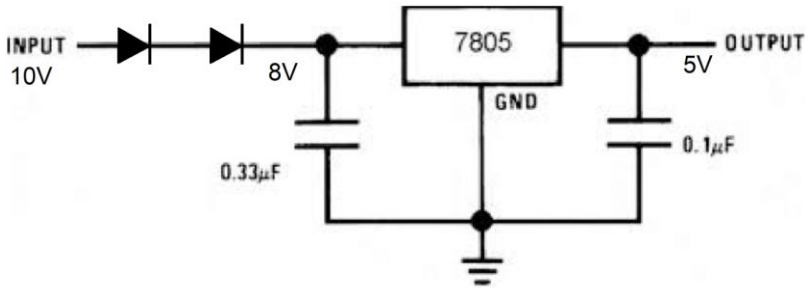
Εντούτοις, η χρήση του LDO ρυθμιστή τάσης στην πλακέτα απαιτεί ιδιαίτερη προσοχή, καθώς δεν πρέπει να υπερβούμε τα μέγιστα επίπεδα θερμότητας που υποστηρίζει το συγκεκριμένο IC. Ο συγκεκριμένος ρυθμιστής τάσης έχει συσκευασία TO-3 και σύμφωνα με το φύλλο δεδομένων, το μέγιστο επίπεδο θερμότητας, με την τοποθέτηση μίας μικρής ψήκτρας ισούται με 4W. Για τον υπολογισμό της συνολικής απαγωγής θερμότητας θα χρησιμοποιήσουμε τον τύπο:

$$P = (V_I - V_o)I_{out} \tag{2-1}$$

Η μέγιστη κατανάλωση ρεύματος I_{out} ισούται με ~960mA και υπολογίστηκε αθροίζοντας την κατανάλωση κάθε εξαρτήματος στην πλακέτα επέκτασης, ενώ λειτουργούν στην μέγιστη τους ισχύ. Οι μετρήσεις έγιναν με ψηφιακό πολύμετρο και είναι οι εξής:

- 1x MCU 500mA
- 3x Digital rotary encoder with differential converter 100mA
- 2x Motor controller Optocoupler circuit 50mA
- 2x Analog rotary encoder 35mA

Συνεπώς για την μείωση της απαγόμενης θερμότητας οφείλουμε να έχουμε τάση εισόδου όσο πιο κοντά γίνεται στην τάση εξόδου στον ρυθμιστή τάσης. Αυτό μπορεί να επιτευχθεί τοποθετώντας σε σειρά με τον ρυθμιστή δύο 1N5408 διόδους ισχύος, μειώνοντας την τελική τάση εισόδου στον ρυθμιστή τάσης κατά περίπου 2V. Η τεχνική αυτή όχι μόνο μειώνει την διαφορά τάσης V_I-V_O από 10-5=5V σε 8-5=3V (Σχήμα 2-6), αλλά και προστατεύει το κύκλωμα από ανάποδη πολικότητα. Ουσιαστικά, η συνολική απαγωγή θερμότητας μοιράζεται μεταξύ του ρυθμιστή τάσης και των δύο διόδων, οι οποίες έχουν πολύ μεγαλύτερη ικανότητα απαγωγής θερμότητας. Έτσι τελικά η συνολική απαγωγή θερμότητας του ρυθμιστή τάσης ισούται με (8-5)x0.9=2.88W, που είναι άνετα εντός του ορίου των 4W.



Σχήμα 2-6. Κύκλωμα LDO ρυθμιστή τάσης 7805.

2.2.4 Διαχείριση διαφορικών σημάτων των αυξητικών κωδικοποιητών

Ο αυξητικός κωδικοποιητής HEDS-5540 χρησιμοποιεί διαφορικά σήματα (QEPA, QEPA-, QEPB, QEPB-, QEPI, QEPI-) για την μετάδοση της πληροφορίας για λόγους αξιοπιστίας. Για την μετατροπή των ζευγών των διαφορικών σημάτων σε απλά σήματα που θα διαβάσει η κάρτα MCU χρησιμοποιούμε το ενσωματωμένο SN75175, έναν τετραπλό differential line receiver. Στην είσοδο τροφοδοσίας VCC κάθε IC τοποθετηθήκαν δύο 4.7μF και 100nF 16V πυκνωτές αποσύνδεσης για την σταθεροποίηση της ισχύος. Κάθε ζεύγος διαφορικού σήματος βραχυκυκλώθηκε με αντίσταση 120Ω όπως συνιστάται από το φύλλο δεδομένων του κατασκευαστή. Η σύνδεση των αυξητικών κωδικοποιητών με την πλακέτα επέκτασης γίνεται με έναν 10-pin IDC κονέκτορα.

2.2.5 Παρακολούθηση ρεύματος κινητήρων

Σε σχέση με το Laelaps II προστέθηκε μία ακόμα λειτουργία, η καταγραφή του ρεύματος που καταναλώνουν οι κινητήρες, μέσω του σήματος **CURRENT_MONITOR** που παρέχεται από τους AZBDC20A8 motor controllers. Το σήμα αυτό, μαζί με την γείωση VEE η οποία δεν είναι κοινή με αυτή της κάρτας MCU (λόγω του optocoupler που περιγράφεται στο επόμενο κεφάλαιο) λαμβάνονται από το καλώδιο RJ-45 που στέλνει το σήμα PWM και καταγράφονται από την μονάδα ADC της MCU σε λειτουργία διαφορικής ανάγνωσης, ώστε να μετρηθεί η διαφορά τάσης μεταξύ των δύο αυτών σημάτων. Το σήμα αυτό έχει τάση 0-5V, οπότε ένας διαιρέτης τάσης αποτελούμενος από 3.3kΩ, 1.8kΩ αντιστάσεις αντιστοιχίζει την τάση αυτή στα 0-3.3V, όπως και η έξοδος των απόλυτων κωδικών. Τα σήματα αυτά λαμβάνονται από το ίδιο RJ-45 καλώδιο που μεταδίδει το σήμα PWM.

2.2.6 Τεχνικές κόλλησης των εξαρτημάτων στην πλακέτα επέκτασης

Η καινούρια κάρτα MCU χρησιμοποιεί τον κονέκτορα HSEC που φαίνεται στο Σχήμα 2-7. Όπως φαίνεται και από την φωτογραφία, ο κονέκτορας αυτός έχει πολύ λεπτά πινάκια τα

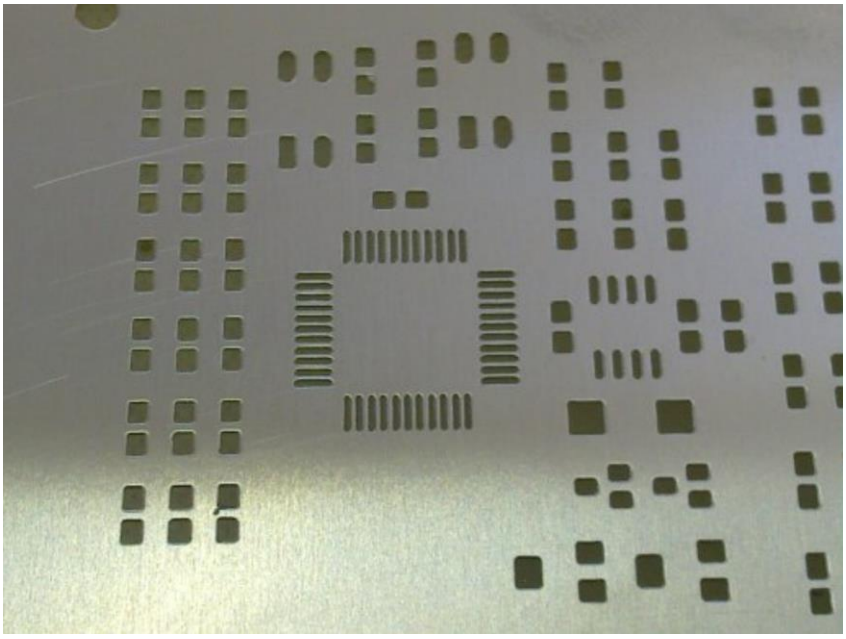
οποία βρίσκονται πολύ κοντά μεταξύ τους. Ακόμα ακουμπούν με το πλαστικό περίβλημα το οποίο προφανώς έχει κάποια όρια θερμοκρασίας. Όλα τα παραπάνω καθιστούν την κόλληση αυτού του κονέκτορα ιδιαίτερα δύσκολη και ίσως αδύνατη με παραδοσιακές τεχνικές, όπως με την χρήση κολλητηριού.

Σχήμα 2-7. Κονέκτορας HSEC.

Για να πετύχουμε λοιπόν υψηλή ακρίβεια στις κολλήσεις και αποφυγή βραχυκυκλωμάτων, πρέπει να εφαρμόσουμε τεχνικές κόλλησης εξαρτημάτων SMD. Η τεχνική που θα ακολουθήσουμε χρησιμοποιεί ένα stencil, πάστα κόλλησης χαμηλής θερμοκρασίας, καθώς και ένα πιστόλι κόλλησης θερμού αέρα.

Οι πλακέτες σχεδιάστηκαν στο EAGLE και τα σχέδια τους στάλθηκαν στην υπηρεσία JLCPCB η οποία προμήθευσε τα PCB καθώς και το stencil. Ουσιαστικά το stencil επιτρέπει να εφαρμόσουμε την ακριβή ποσότητα πάστας που χρειάζεται μόνο πάνω στα σημεία που απαιτείται, όπως φαίνεται και στο Σχήμα 2-88, αποφεύγοντας έτσι βραχυκυκλώματα ή αδύναμες κολλήσεις.

Έπειτα, χρησιμοποιούμε πάστα κόλλησης χαμηλής θερμοκρασίας [35] . Μετά την εφαρμογή της πάνω στο stencil, το αφαιρούμε προσεκτικά, τοποθετούμε όλα τα εξαρτήματα, εφαρμόζουμε θερμό αέρα στους 140^{o}C μέχρι η πάστα να μετασχηματιστεί πλήρως σε κόλληση. Ο λόγος απαίτησης πάστας χαμηλής θερμοκρασίας, είναι ότι το πλαστικό περίβλημα του κονέκτορα λιώνει όταν χρησιμοποιούνται κανονικές πάστες, διότι το πιστόλι θερμού αέρα πρέπει να ρυθμιστεί σε μεγαλύτερη θερμοκρασία.



Σχήμα 2-8. Παράδειγμα χρήσης του stencil.

2.3 Η πλακέτα επέκτασης της AZBDC20A8

2.3.1 Εισαγωγή

Η πλακέτα επέκτασης έχει τροποποιηθεί σε σχέση με την προηγούμενη έκδοση ώστε να δέχεται πλέον τον καινούριο, ισχυρότερο και ελαφρώς διαφορετικό στην συνδεσμολογία AZBDC20A8 motor controller καθώς και κάποιες επιπρόσθετες λειτουργίες καταγραφής. Πιο συγκεκριμένα, όπως υποδηλώνει και η ονομασία του, ο καινούριος motor controller επιτρέπει μέγιστη λειτουργία των μοτέρ στα 20A σε σχέση με τον προκάτοχό του AZBDC12A8 ο οποίος υποστήριζε μέχρι 12A. Ο τελευταίος αλλάχθηκε λόγω προβλημάτων υπερθέρμανσης, και συνεπακόλουθης αδυναμίας ελέγχου τον κινητήρων, καθώς λειτουργούσε για παρατεταμένο χρονικό διάστημα στις ανώτατες τιμές ρεύματος.

2.3.2 Απομόνωση ηλεκτρικών σημάτων με χρήση optocoupler

Το σύστημα απομόνωσης σημάτων διατηρήθηκε από την προηγούμενη έκδοση του Laelaps. Πιο συγκεκριμένα, για την επιπλέον απομόνωση των μικροελεγκτών από ηλεκτρομαγνητικό θόρυβο προερχόμενο από τον motor controller, χρησιμοποιήθηκε ένας 2-κάναλος 6N137 οπτοζεύκτης (optocoupler) [32] . Τα σήματα ελέγχου DIR και PWM απομονώνονται μέσω αυτού του IC και έπειτα ενισχύονται μέσω ενός διπολικού 2N7002 N-Channel MOSFET [31] και τροφοδοτούνται στις εισόδους ελέγχου DIR, PWM του AZBDC20A8. Οι οπτοζεύκτες και η λογική μονάδα του AZBDC20A8 τροφοδοτούνται από έναν εξωτερικό 5V MAX5035 ρυθμιστή τάσης, για επιπλέον απομόνωση του θορύβου.

2.3.3 Κύκλωμα και σχηματικό της πλακέτας επέκτασης

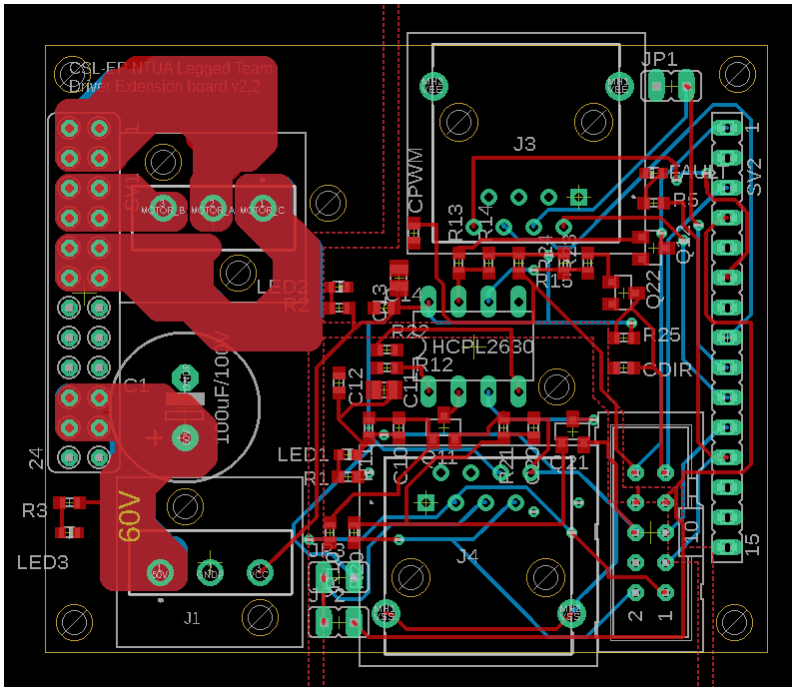
Το κύκλωμα και η διάταξη της πλακέτας επέκτασης είναι παρόμοια με εκείνα της προηγούμενης έκδοσης του Laelaps. Γενικότερα, οι λειτουργικές υπομονάδες αυτής της πλακέτας έχουν διαχωριστεί με Ground και VCC Pours. Η τεχνική αυτή εξασφαλίζει μειωμένο θόρυβο από τις γραμμές υψηλής τάσης για την τροφοδοσία των κινητήρων και

καθιστά τον σχεδιασμό πιο εύκολο επιτρέποντας τα εξαρτήματα να είναι πιο πυκνά τοποθετημένα.

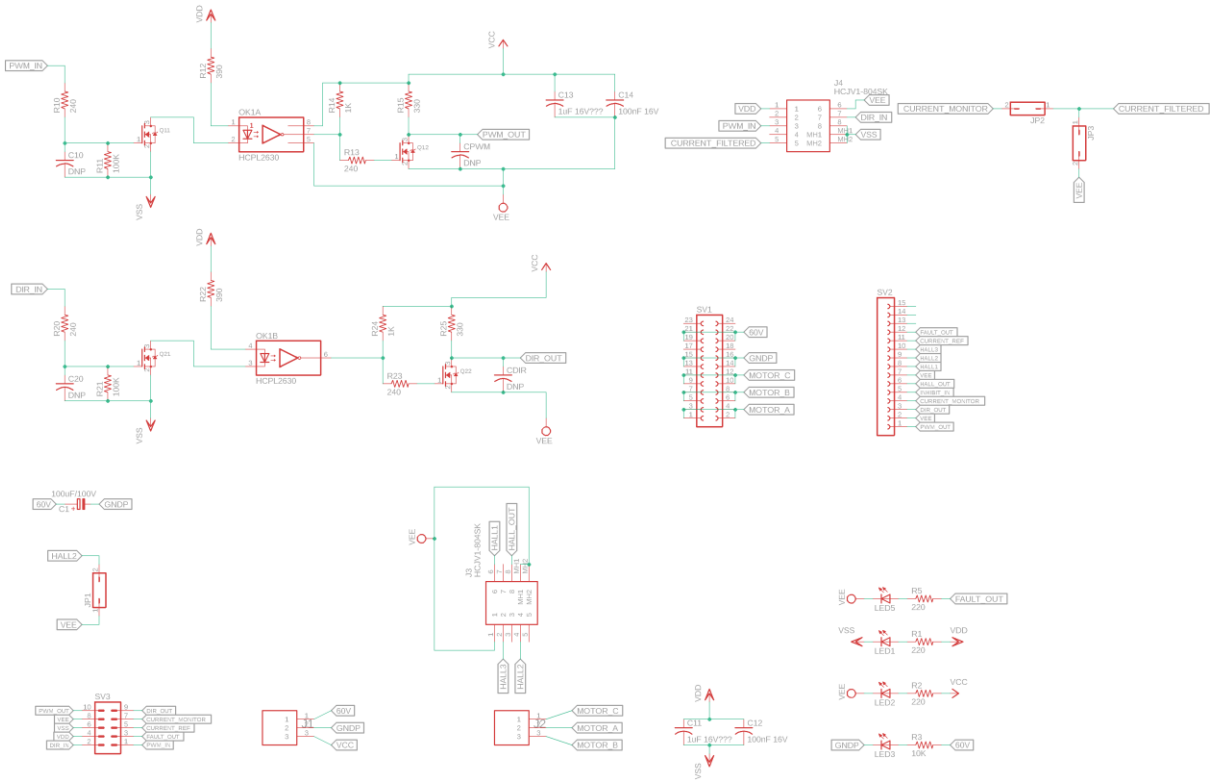
Η καινούρια πλακέτα είναι συμβατή και με την παλαιότερη μονάδα AZBDC12A8. Όσον αφορά την συνδεσμολογία τους η μόνη διαφορά που έχουν είναι ότι ο καινούριος ελεγκτής λόγω της μεγαλύτερης ισχύος του έχει διπλή σειρά από headers για τις γραμμές ισχύος, οπότε έχει γίνει και η αντίστοιχη τροποποίηση στην καινούρια πλακέτα.

Έχει προστεθεί επίσης η λειτουργία ένδειξης σφάλματος μέσω της εξόδου του motor controller FAULT_OUT. Πιο συγκεκριμένα στην πλακέτα επέκτασης υπάρχει η φωτεινή ένδειξη FAULT και ανάβει όταν απενεργοποιείται η ισχύς στον κινητήρα λόγω υπέρτασης, βραχυκυκλώματος, υπερθέρμανσης, εσφαλμένο Hall-state, inhibit ή power-up reset.

Τέλος, έχει προστεθεί η λειτουργία καταγραφής του ρεύματος ισχύος που αναλύεται στην παρακάτω παράγραφο. Η πληροφορία μεταφέρεται μέσω του pin 5 του RJ-45 κονέκτορα. Το συνολικό διάγραμμα του κυκλώματος της πλακέτας επέκτασης φαίνεται στο Σχήμα 2-99 ενώ η κάτοψη της πλακέτας φαίνεται στο Σχήμα 2-1010.



Σχήμα 2-9. Κάτοψη της πλακέτας επέκτασης AZBDC20A8.

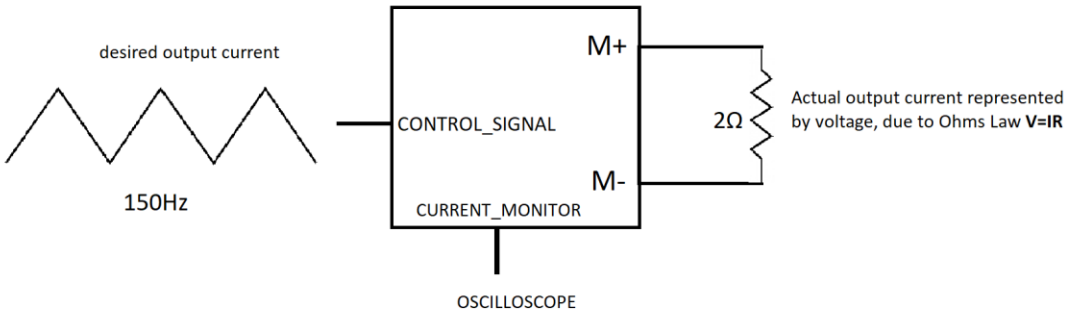


Σχήμα 2-10. Διάγραμμα κυκλώματος πλακέτας επέκτασης AZBDC20A8.

2.3.4 Καταγραφή ρεύματος των κινητήρων μέσω του σήματος CURRENT_MONITOR

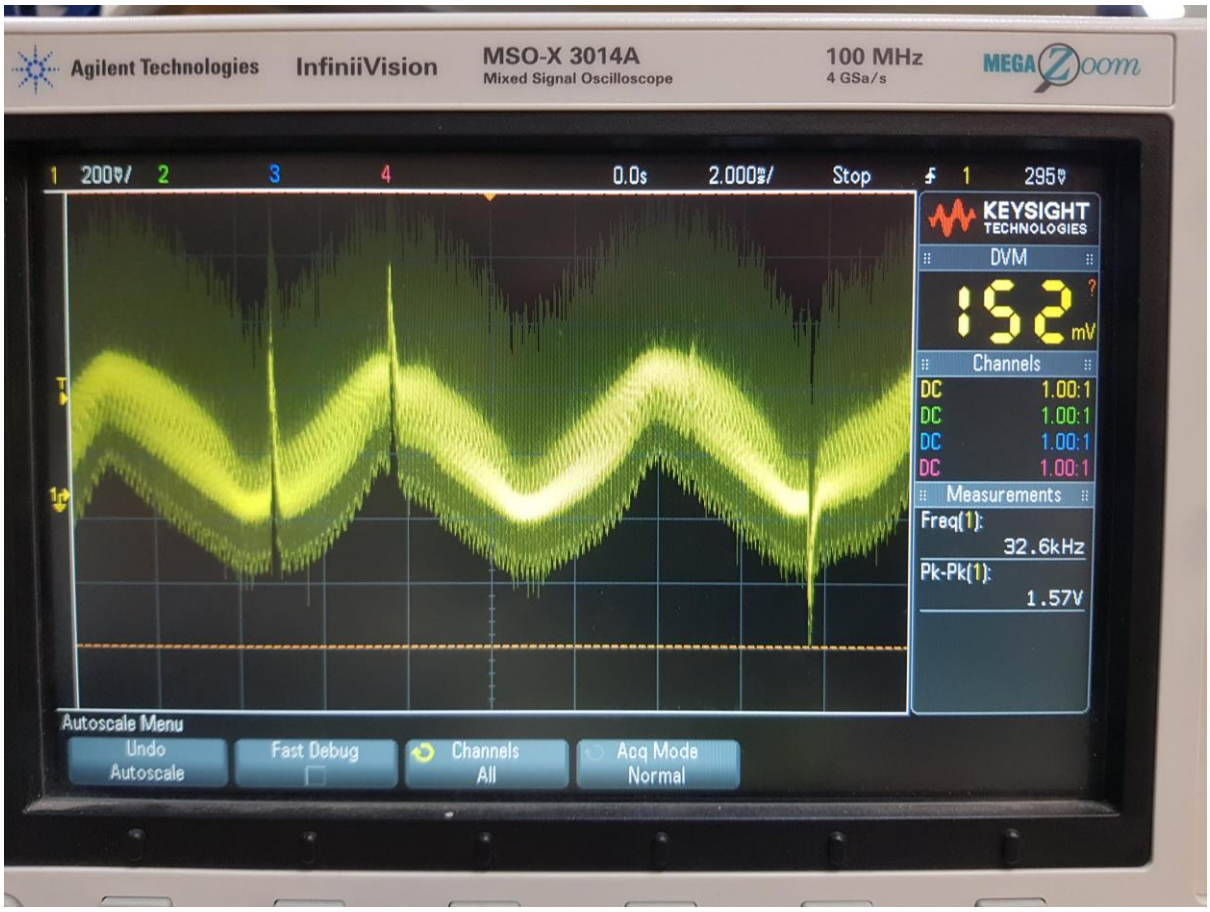
Για καλύτερη εποπτεία της λειτουργίας τον ηλεκτρικών κινητήρων στην παρούσα έκδοση του Laelaps προστέθηκε η λειτουργία καταγραφής του ρεύματος που καταναλώνουν. Η λειτουργία αυτή είναι ενσωματωμένη στον motor controller μέσω της εξόδου CURRENT_MONITOR. Η έξοδος αυτή έχει σήμα πλάτους 0-5V που είναι ανάλογο του ρεύματος κατανάλωσης με λόγο 6.4 A/V. Παρόλα αυτά, το σήμα αυτό δεν είναι φιλτραρισμένο και ως εκ τούτου δεν είναι ιδιαίτερα χρήσιμο χωρίς την εφαρμογή κάποιου φίλτρου πρώτα.

Έγινε λοιπόν ανάλυση του σήματος με έναν ψηφιακό παλμογράφο για να εντοπιστούν οι κύριες συνιστώσες της συχνότητας του σήματος. Για να κατανοηθεί ευκολότερα αυτό το σήμα εξόδου, τοποθετήθηκε ένα στατικό φορτίο στην θέση του κινητήρα, μία αντίσταση 2Ω, δημιουργώντας έτσι μία 1-1 αντιστοιχία στο ζητούμενο ρεύμα εξόδου και στο πραγματικό ρεύμα εξόδου (Σχήμα 2-111). Το ζητούμενο ρεύμα εξόδου ή αλλιώς η είσοδος PWM στον motor controller, ορίστηκε σαν ένας τριγωνικός παλμός με μία πολύ χαμηλή συχνότητα 150Hz, ώστε να την ξεχωρίζουμε εύκολα από τυχόν συχνότητες που παράγονται από τον motor controller και επίσης να παρατηρήσουμε τυχόν αλλαγές στις συνιστώσες της συχνότητας αναλόγως την ζητούμενη έξοδο ρεύματος.



Σχήμα 2-11. Διάταξη ανάλυσης του σήματος CURRENT_MONITOR.

Παρατηρήσαμε μία συχνότητα 32kHz, όπως φαίνεται και στο Σχήμα 2-12, η οποία προφανώς αντιστοιχεί στην διακοπτόμενη συχνότητα (switching frequency) του motor controller. Η παρατήρηση επιβεβαιώθηκε, καθώς στο datasheet του συγκεκριμένου motor controller αναφέρεται ότι λειτουργεί με συχνότητα 31kHz, κάτι πολύ κοντά σε αυτό που παρατηρήσαμε.

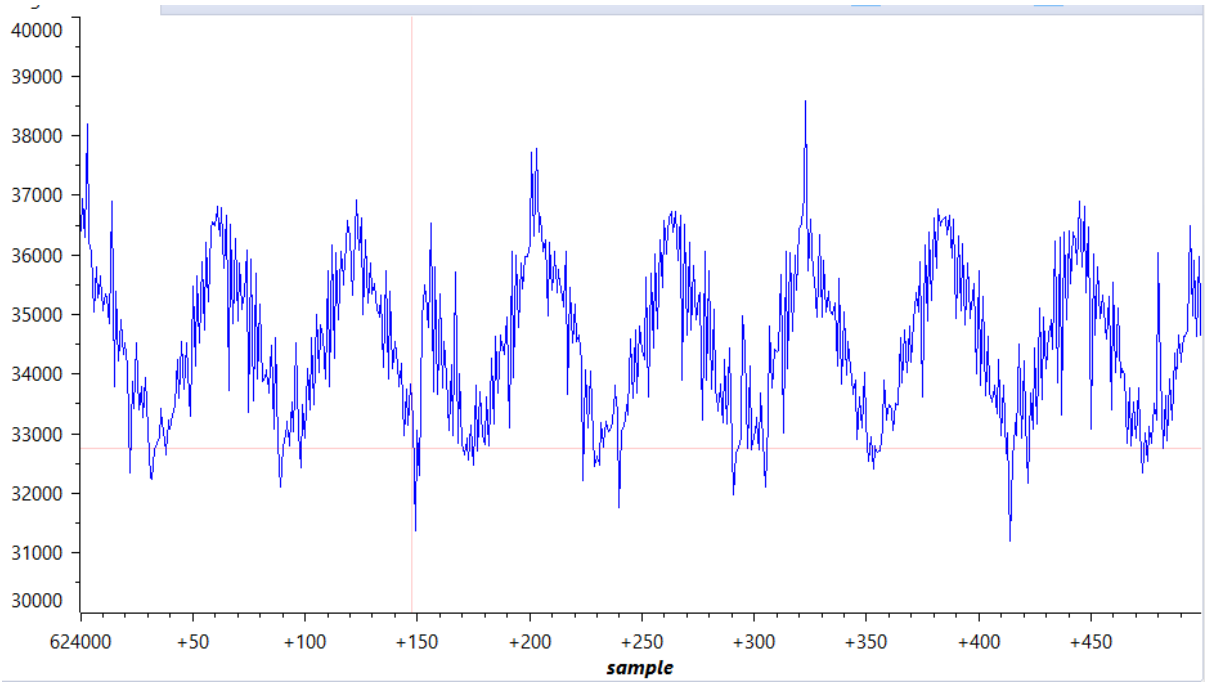


Σχήμα 2-12. Κυματομορφή του αφιλτράριστου σήματος CURRENT_MONITOR στον ψηφιακό παλμογράφο.

Το σήμα αυτό μπορεί να φιλτραριστεί με ένα απλό βαθυπερατό RC φίλτρο ή/και με ένα ψηφιακό φίλτρο ολισθαίνοντος μέσου όρου (digital moving average filter), εφόσον έχουμε ένα MCU υψηλών επιδόσεων στην διάθεσή μας. Το ψηφιακό φίλτρο μέσου όρου έχει συμπεριφορά παρόμοια με εκείνη του RC φίλτρου, όπου και τα δύο έχουν μία μικρή καθυστέρηση στην έξοδο τους σε σχέση με την είσοδο. Παρόλα αυτά στην εφαρμογή μας δεν είναι κάτι που μας απασχολεί.

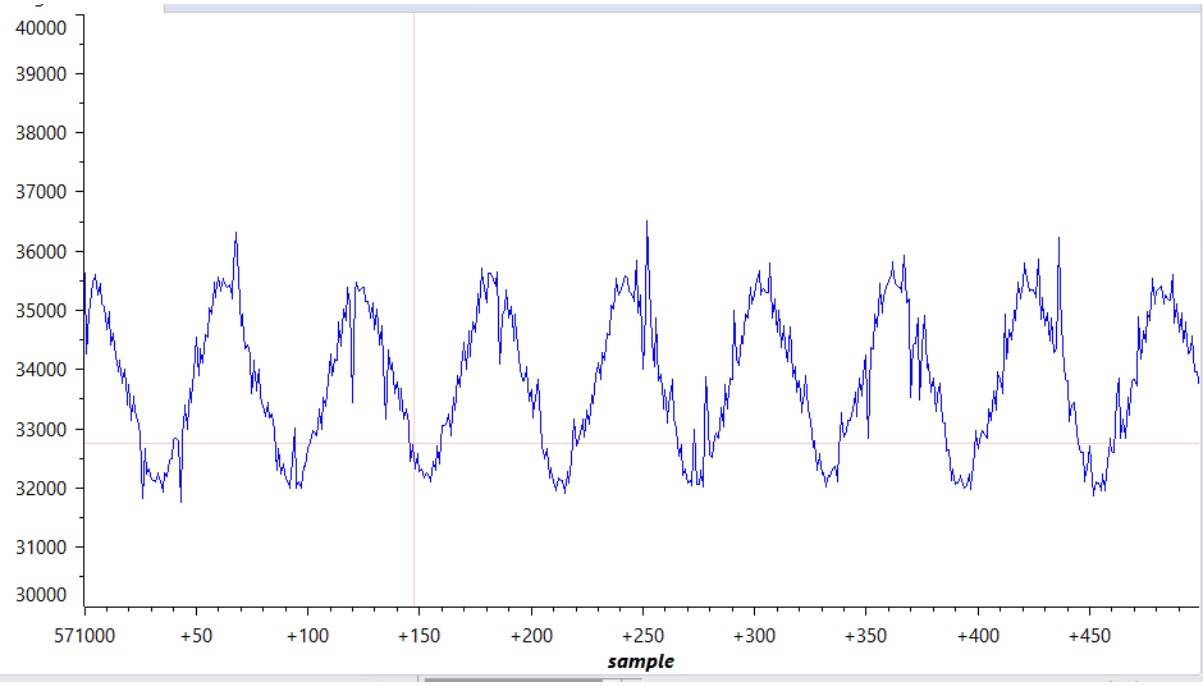
Αρχικά πρέπει να βεβαιωθούμε πως η δειγματοληψία γίνεται σωστά από τον μικροελεγκτή. Σύμφωνα με το θεώρημα του Nyquist η συχνότητα δειγματοληψίας του ψηφιακού φίλτρου θα πρέπει να είναι τουλάχιστον 2 φορές μεγαλύτερη από την συχνότητα που θέλουμε να φιλτράρουμε, άρα 64kHz. Χρησιμοποιώντας τον Debugger του μικροελεγκτή μετρήσαμε ότι απαιτούνται περίπου 60 κύκλοι για την εκτέλεση μιας εντολής δειγματοληψίας και αποθήκευσης της εξόδου στην μνήμη. Για συχνότητα συστήματος 200MHz η δειγματοληψία εκτελείται με συχνότητα 200MHz/60=3.3MHz, πολύ παραπάνω από ότι απαιτεί το θεώρημα.

Στο Σχήμα 2-13 έχουμε αποτυπώσει το σήμα που έχει δειγματοληφθεί από τον MCU, μέσω της μονάδας ADC, χωρίς την εφαρμογή κάποιου ψηφιακού ή αναλογικού φίλτρου. Παρατηρούμε παρόμοια έξοδο με αυτή του παλμογράφου. Το διάγραμμα αυτό δημιουργήθηκε αποθηκεύοντας τις τελευταίες 10000 τιμές της εξόδου της μονάδας ADC σε μία μεταβλητή και έπειτα η γραφική παράστασή της μέσω του CCS Debugger.



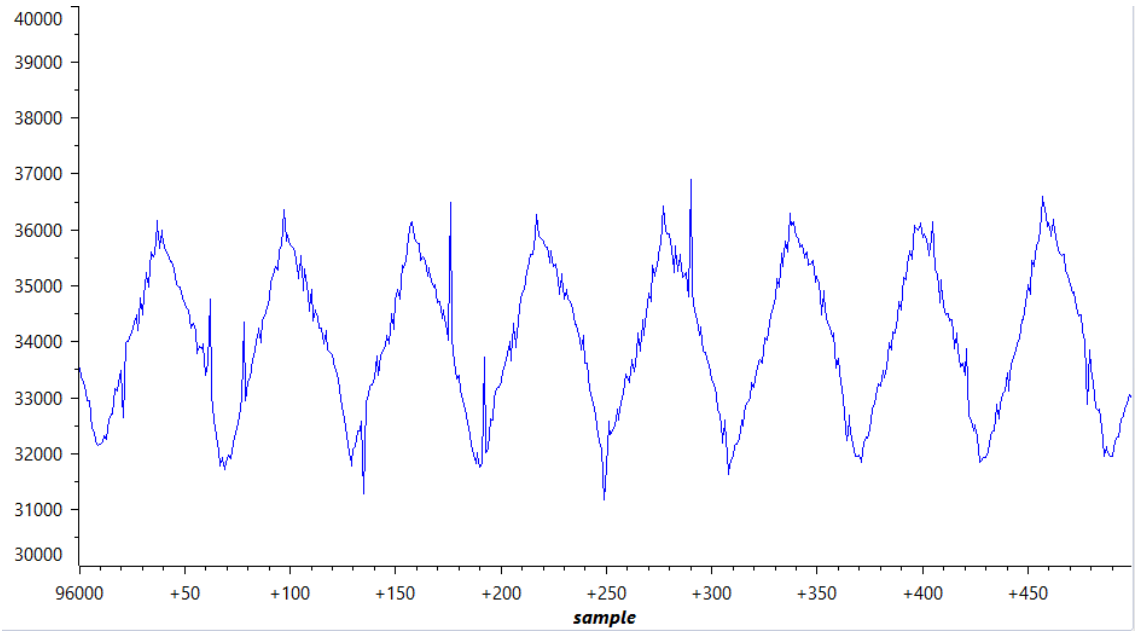
Σχήμα 2-13. Αφιλτράριστο σήμα CURRENT_MONITOR όπως έχει δειγματοληφθεί από τον MCU.

Έπειτα καταγράψαμε το σήμα με την χρήση μόνο αναλογικού RC φίλτρου που απαρτίζεται από μία αντίσταση 1000Ω και έναν πυκνωτή 0.016μF με συχνότητα αποκοπής 10kHz. Όπως παρατηρούμε και στο Σχήμα 2-144, η συνιστώσα των 32kHz έχει μειωθεί αλλά δεν έχει εξαλειφθεί τελείως.



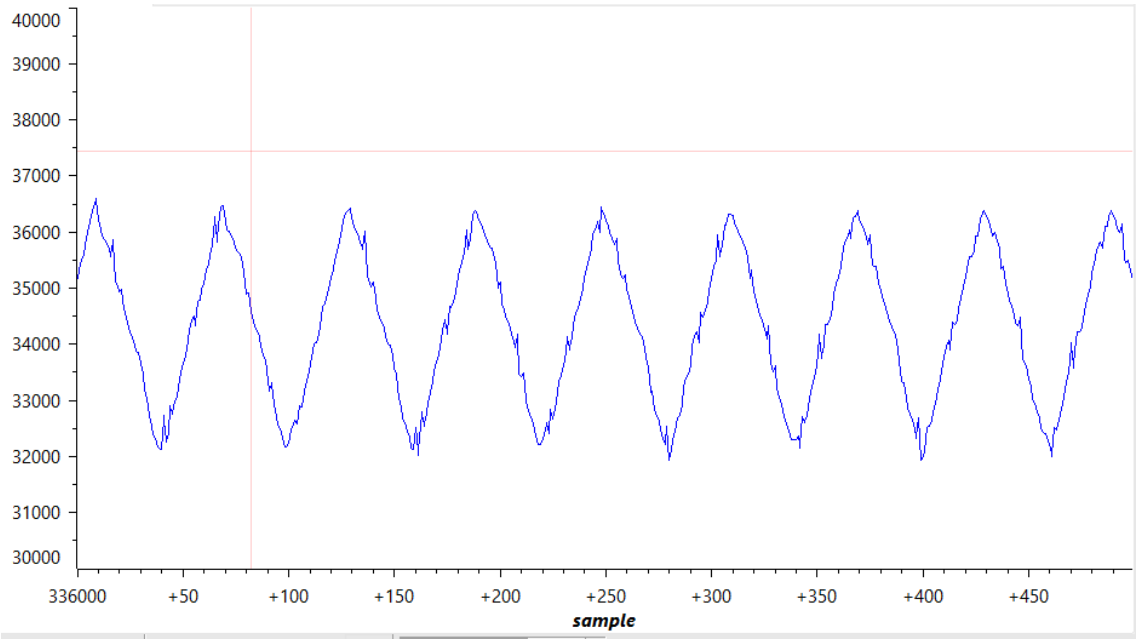
Σχήμα 2-14. Φιλτραρισμένο σήμα CURRENT_MONITOR όπως έχει δειγματοληφθεί από την MCU με RC φίλτρο αποκοπής 10kHz.

Θα εξετάσουμε και το ψηφιακό φίλτρο μέσου όρου. Με την εφαρμογή του με δειγματοληψία 10 δειγμάτων έχουμε την έξοδο που απεικονίζεται στο Σχήμα 2-155. Παρατηρούμε πιο καθαρό σήμα από το αναλογικό φίλτρο.



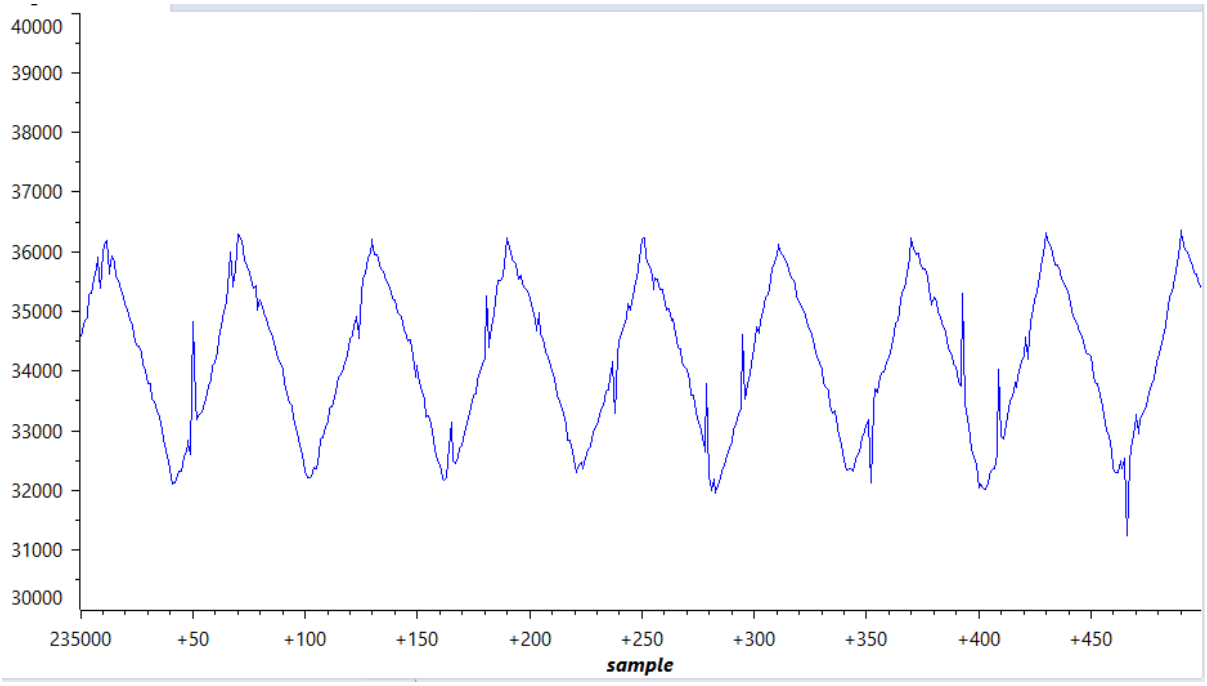
Σχήμα 2-15. Φιλτραρισμένο σήμα CURRENT_MONITOR όπως έχει δειγματοληφθεί από τον MCU με ψηφιακό φίλτρο μέσου όρου x10.

Καταγράφοντας 100 τιμές στο ψηφιακό φίλτρο μέσου όρου, έχουμε σχεδόν εξαλείψει τελείως τις μη επιθυμητές συχνότητες, όπως φαίνεται και στο Σχήμα 2-166. Παρόλα αυτά, λόγω των πολλών μετρήσεων που απαιτούνται, ο MCU επιβαρύνεται με αρκετούς κύκλους και ενδέχεται αυτή η επιβάρυνση να μην είναι ανεχτή στον σχεδιαστή του συστήματος.



Σχήμα 2-16. Φιλτραρισμένο σήμα CURRENT_MONITOR όπως έχει δειγματοληφθεί από την MCU με ψηφιακό φίλτρο μέσου όρου x100.

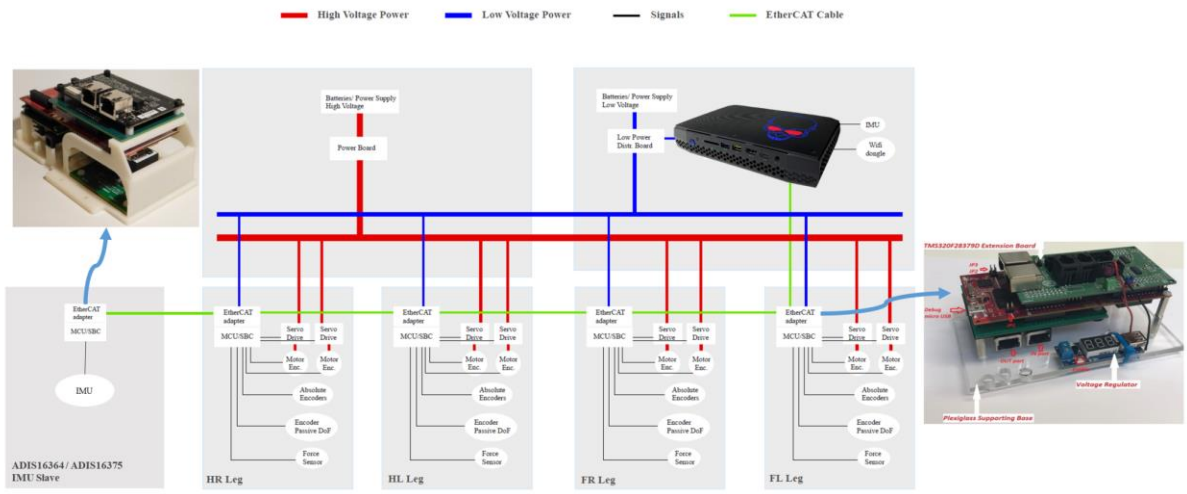
Φυσικά μπορούμε να χρησιμοποιήσουμε και συνδυασμό των δύο παραπάνω φίλτρων, που φαίνεται να έχει το καλύτερο αποτέλεσμα, χωρίς μεγάλη επιβάρυνση σε κύκλους στον MCU. Το αποτέλεσμα σε αυτή την περίπτωση φαίνεται στο Σχήμα 2-177. Όπως και να έχει στην πλακέτα επέκτασης υπάρχουν υποδοχές για την απαιτούμενη αντίσταση και πυκνωτή που χρειάζονται για το RC φίλτρο. Παρόλα αυτά, αν ο σχεδιαστής δεν επιθυμεί να το χρησιμοποιήσει μπορεί απλώς να ανοιχτοκυκλώσει τον πυκνωτή και να βραχυκυκλώσει την αντίσταση.



Σχήμα 2-17. Φιλτραρισμένο σήμα CURRENT_MONITOR όπως έχει δειγματοληφθεί από την MCU με RC φίλτρο αποκοπής 10kHz και ψηφιακό φίλτρο μέσου όρου x10.

2.4 Ανάπτυξη λογισμικού ελέγχου κίνησης

Το λογισμικό του συστήματος απαρτίζεται από το firmware που τρέχει στους MCUs και ουσιαστικά τρέχει στους ESCs (Ethercat Slave Controllers) καθώς και το λογισμικό TwinCat αποτελώντας τον κεντρικό Ethercat Master, ο οποίος τρέχει σε έναν κεντρικό υπολογιστή και ελέγχει τους slaves, όπως στο Σχήμα 2-18. Εκεί ορίζονται οι παράμετροι του ελέγχου όπως η συχνότητα βαδίσης, το μήκος του κάθε βήματος κ.ο.κ. Γενικότερα ο λόγος που επιλέχτηκε το πρωτόκολλο Ethercat για την επικοινωνία μεταξύ των υποσυστημάτων, είναι ότι επιτρέπει γρήγορο και κυριως ακριβή συγχρονισμό της μεταφοράς των απαιτούμενων δεδομένων, μέσω του DC (Distributed Clock). Αυτό είναι κάτι πολύ σημαντικό καθώς επιτρέπει τα τοπικά σχήματα ελέγχου που εκτελούνται στο κάθε άκρο να είναι τέλεια συγχρονισμένα μεταξύ τους, πετυχαίνοντας άριστή απόδοση και ντετερμινισμό στις πολύπλοκες διαδικασίες που χρειάζεται να εκτελεί το τετράποδο ρομπότ, όπως βάδιση, τρέξιμο ή πήδημα.



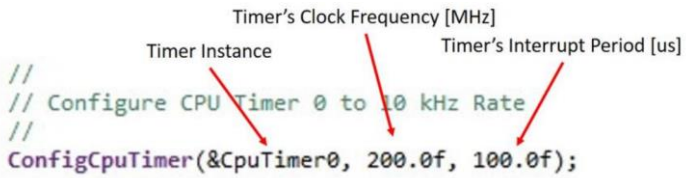
Σχήμα 2-18. Συνδεσμολογία των Ethercat master/slave.

2.4.1 Firmware

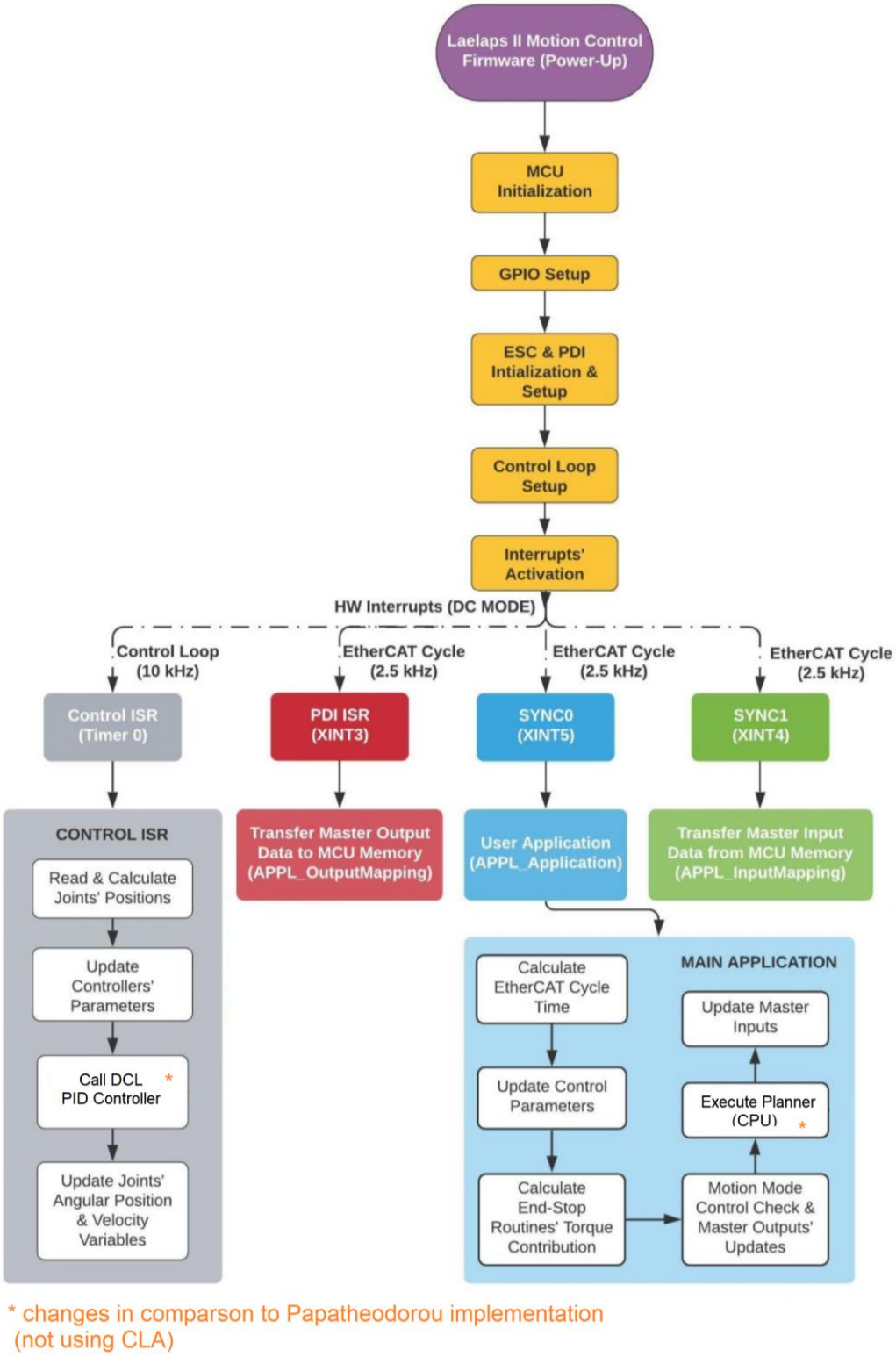
Το firmware βασίζεται στην τελευταία έκδοση του Laelaps [15] με κάποιες μικρές τροποποιήσεις ώστε να είναι συμβατό με την νέα πλακέτα. Στην επόμενη παράγραφο θα περιγράφουν οι αλλαγές αυτές.

Με λίγα λόγια όλες η λειτουργίες του εκτελούνται μέσα σε 4 διαφορετικά ISRs όπως φαίνεται και στο Σχήμα 2-20. Τρία από αυτά ενεργοποιούνται από τη μονάδα συγχρονισμού DC (Distributed Clock) της μονάδας ESC (Ethercat Slave Controller), ενώ το τέταρτο (επισημασμένο με γκρι) ενεργοποιείται από τον Timer 0. Να σημειωθεί ότι στην default υλοποίηση του HAL (Hardware Abstraction Layer) stack ο Timer 0 χρησιμοποιείται ως μετρητής για εξυπηρέτηση κάποιων λειτουργιών του Ethercat. Παρόλα αυτά, για λόγους προτεραιότητας interrupt, όπως και στην προηγούμενη MCU, λόγω χρήσης του Timer 0 για την εκτέλεση του κύριου βρόχου ελέγχου, αναθέσαμε την λειτουργία αυτή στον Timer 1.

Η συχνότητα εκτέλεσης των τριών ESC ISRs που αναφέρθηκαν ορίζεται από τον Ethercat Master, δηλαδή μέσω του προγράμματος TwinCat. Η συχνότητα εκτέλεσης ορίζεται όπως στην εντολή που φαίνεται στο Σχήμα 2-19.



Σχήμα 2-19. Ορισμός συχνότητας εκτέλεσης του Timer 0.



Σχήμα 2-20. Σχηματική παράσταση του firmware.

2.4.2 Αλλαγές στο firmware σε σχέση με την προηγούμενη έκδοση

Όπως και στην υλοποίηση του Παπαθεοδώρου η επικοινωνία με το σύστημα Ethercat γίνεται μέσω ενός EtherCAT stack. Η χρήση της καινούριας πλακέτας F28388 απαιτεί την χρήση διαφορετικού stack. Για την ανάπτυξη του κώδικα ουσιαστικά δημιουργήθηκε ένα κενό Ethercat Application από την αρχή πάνω στο καινούριο stack μέσω του SSC (Slave Stack Tool) όπως περιγράφεται αναλυτικά και στο Κεφάλαιο 11 της διπλωματικής εργασίας του Παπαθεοδώρου [15] . Έπειτα στο καινούριο project που δημιουργήθηκε ενσωματώθηκε ο υπάρχων κώδικας ελέγχου του Παπαθεοδώρου με κάποιες μικρές αλλαγές.

Αρχικά, η εκτέλεση των μαθηματικών πράξεων ελέγχου δεν γίνεται με την μονάδα CLA (Control Law Accellerator), αλλά απλώς με την CPU και την FPU, όπως φαίνεται και στο Σχήμα 2-20. Η αλλαγή αυτή δεν έγινε για λόγους απόδοσης, αλλά για απλοποίηση του firmware ώστε να είναι πιο εύκολη η ανάπτυξη του, καθώς δεν υπήρχε αρκετό χρονικό περιθώριο για την ενσωμάτωση αυτής της λειτουργίας. Παρόλα αυτά οι λειτουργίες αυτές υπάρχουν ήδη μέσα στον κώδικα του Παπαθεοδώρου και έχουν περαστεί και στον νέο, αρκεί να γίνει η διασύνδεση με τις διαφορετικές cla εντολές, λόγω της χρήσης καινούριας πλακέτας. Για την αλλαγή αυτή απλώς αφαιρέθηκαν όλα τα CLA predefined symbols από την επιλογή **Build->Compiler->Predefined symbols.**

Λόγω της ελαφρά διαφορετικής αρχιτεκτονικής της νέας MCU η μεταφόρτωσή ορισμένων βιβλιοθηκών στην μνήμη της, μέσω του linker, γίνεται αλλιώς. Πιο συγκεκριμένα θα πρέπει να γίνουν ορισμένες προσθήκες στο **linker command file** σχετικά με την εκχώρηση χώρου μνήμης στις λειτουργίες DCL. Αυτό είναι σημαντικό σε περίπτωση που όπως αναφέραμε στην προηγούμενη παράγραφο οι ρουτίνες DCL εκτελούνται στην CPU και όχι στην CLA (οικογένεια DCL_runPID_Cx). Οι ακόλουθες γραμμές κώδικα τοποθετούνται στο τέλος της ενότητας **SECTIONS** του αρχείου **2838x_flash_lnk_cpu1.cmd** και δίνουν εντολή στον linker να φορτώσει τη βιβλιοθήκη DCL σε ένα συγκεκριμένο μπλοκ στη μνήμη RAM (μπλοκ SG15).

```
.TI.memcrc      : type = COPY
  dclfuncs : > RAMGS15, PAGE = 0
```

Επιπλέον έγινε άλλη μία τροποποίηση στο linker command file, προκειμένου να δοθεί εντολή στον linker να χωρίσει τις ενότητες.bss και.data,.sysmem σε δύο διαφορετικά μπλοκ μνήμης, καθώς ένα μπλοκ 2048 byte δεν χωράει όλες αυτές τις ενότητες ταυτόχρονα. Αυτό επιτυγχάνεται αντικαθιστώντας την προεπιλεγμένη εντολή.bss με την ακόλουθη:

```
.bss      : >> RAMLS4|RAMLS5
```

2.4.3 Εγκατάσταση του firmware στην MCU

Για την εγκατάσταση του firmware κατεβάζουμε πρώτα το project directory από το repository του εργαστηρίου. Έπειτα το εισάγουμε στο CCS (Code Composer Studio) μέσω της επιλογής Project->Import CCS Projects, με ενεργοποιημένη την επιλογή Copy projects into workspace.

Για να μεταγλωττιστεί ο κώδικας, χρησιμοποιεί διάφορες βιβλιοθήκες του C2000Ware. Παρόλα αυτά στην έκδοση **C2000Ware 4.03.00** που χρησιμοποιήθηκε χρειάστηκαν να γίνουν δύο τροποποιήσεις για την επιτυχή μεταγλώττιση του κώδικα. Πρώτον, στο αρχείο

f2838_device.h έγινε comment out η γραμμή **#include f2838_cla.h**. Δεύτερον στο αρχείο **f2838x_globalprototypes.h** η συνάρτηση **IDLE** μετονομάστηκε σε **_IDLE.**

Αφού βεβαιωθούμε πως όλα τα include paths στις καρτέλες **Resource->Linked Resources**, **Build->Compiler->Include Options** και **Build->C2000 Linker->File Search Path** εντοπίζονται επιτυχώς, ορίζουμε στην καρτέλα **Build->Compiler->Predefined symbols** τον όρο **LEFT_LEG** ή **RIGHT_LEG** αναλόγως ποια μεριά του ρομπότ θέλουμε να ελέγξουμε. Τέλος μεταφορτώνουμε το firmware στον MCU μέσω του κουμπιού Debug.

3 Στοιχεία Θεωρίας: Μηχανική μάθηση & Τμηματοποίηση εικόνας

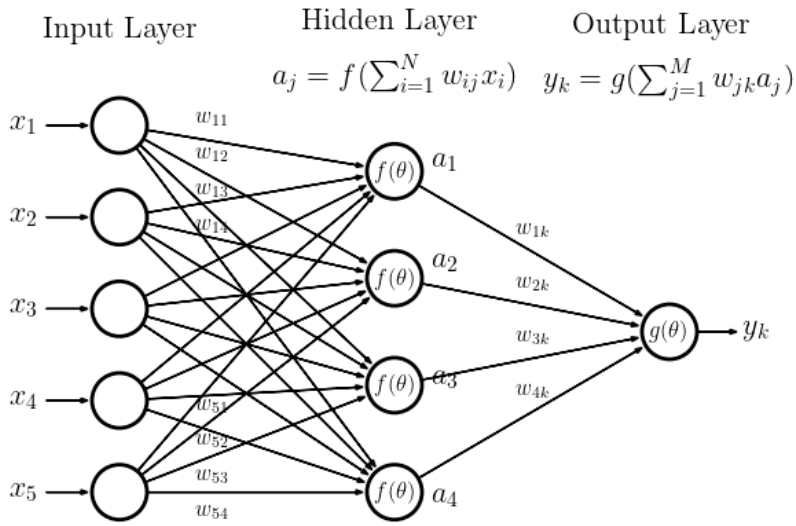
3.1 Μηχανική μάθηση με συνελικτικά νευρωνικά δίκτυα

3.1.1 Εισαγωγή στα νευρωνικά δίκτυα

Ένα νευρωνικό δίκτυο είναι ένας υπολογιστικός αλγόριθμος που μέσω μίας διαδικασίας μάθησης μαθαίνει να αναγνωρίζει μοτίβα. Ο λόγος που ονομάζεται «νευρωνικό» είναι ότι δουλεύει από ένα δίκτυο διασυνδεδεμένων νευρώνων όπως ο ανθρώπινος εγκέφαλος. Όταν π.χ. βλέπουμε μία γάτα και μας δίνεται υπόδειξη ότι αυτό που βλέπουμε είναι γάτα, δημιουργείται μία νέα σύναψη στον εγκέφαλο μας για το δεδομένο ερέθισμα που δημιουργεί το οπτικό νεύρο όταν βλέπουμε γάτες, η οποία ενεργοποιεί το τμήμα της μνήμης μας που έχουμε αποθηκεύσει πληροφορίες για γάτες. Στο αντίστοιχο μηχανικό νευρωνικό δίκτυο που θα περιγράψουμε παρακάτω, το ερέθισμα του οπτικού νεύρου αποτελεί την είσοδο, η σύναψη αντιστοιχεί στα βάρη των συνδέσεων.

Το δίκτυο αποτελείται από μια σειρά από "επίπεδα" $a_1, a_2, a_3, ..$ κάθε ένα από τα οποία αποτελείται από έναν αριθμό "νευρώνων". Κάθε νευρώνας λαμβάνει είσοδο από άλλους νευρώνες ή από εξωτερικές πηγές $x_1, x_2, x_3, ..$ (input layer) και δίνει μια έξοδο, η οποία είναι συνήθως συνδεδεμένη με την είσοδο ενός άλλου νευρώνα στο επόμενο επίπεδο (hidden layer), εκτός και αν αποτελείται από το τελικό επίπεδο (output layer) όπου μας δίνεται η τελική απάντηση, δηλαδή η έξοδος y_k. Η διαδοχή των επιπέδων επιτρέπει στο δίκτυο να μάθει σύνθετα μοτίβα και συνδέσεις μεταξύ των δεδομένων εισόδου και των αντίστοιχων εξόδων. Το δίκτυο μαθαίνει από τα δεδομένα εισόδου μέσω μιας διαδικασίας που ονομάζεται "εκπαίδευση", κατά την οποία προσαρμόζονται τα βάρη w_{xy} των συνδέσεων μεταξύ των νευρώνων, ώστε το δίκτυο να παράγει τα επιθυμητά αποτελέσματα. Κατά τη διάρκεια της εκπαίδευσης, το δίκτυο προσπαθεί να μάθει μια αναπαράσταση των δεδομένων που του δίνονται, ώστε να μπορεί να κάνει προβλέψεις για νέα δεδομένα που δεν έχει δει προηγουμένως. Αυτή η διαδικασία εκπαίδευσης βασίζεται στη μείωση του σφάλματος μεταξύ των προβλέψεων του δικτύου και των πραγματικών αποτελεσμάτων. Κατά τη διάρκεια της εκπαίδευσης, το δίκτυο "μαθαίνει" να αναγνωρίζει συγκεκριμένα χαρακτηριστικά των δεδομένων εισόδου και να εξάγει σημαντικές πληροφορίες από αυτά. Αφού ολοκληρωθεί η εκπαίδευση, το δίκτυο μπορεί να χρησιμοποιηθεί για να κάνει προβλέψεις για νέα δεδομένα που του δίνονται.

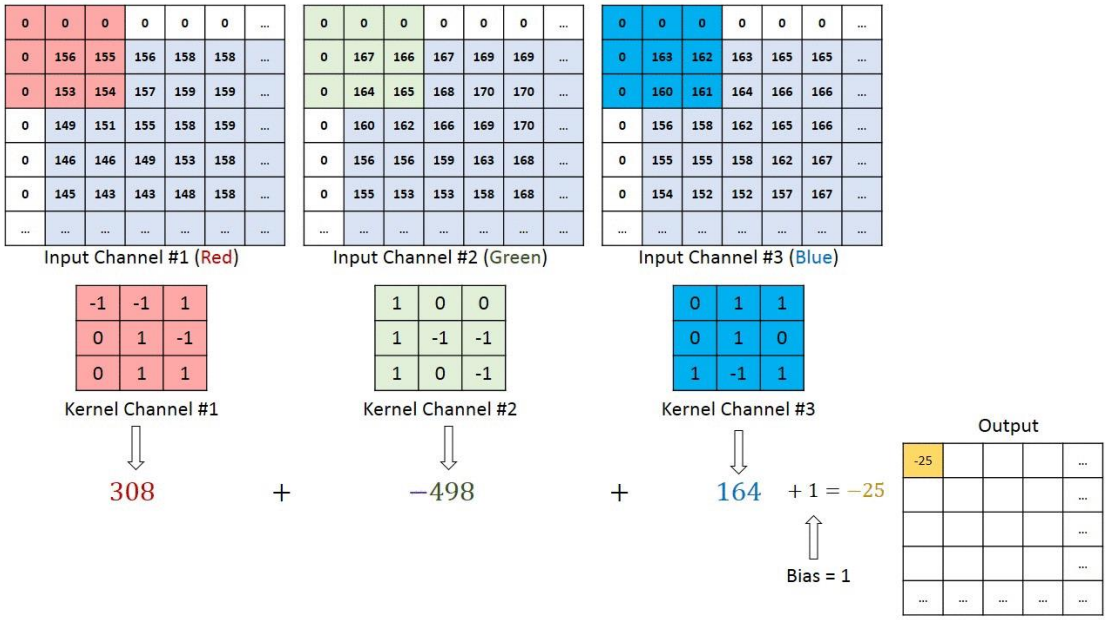
Τα νευρωνικά δίκτυα είναι εξαιρετικά χρήσιμα σε πολλούς τομείς, όπως η αναγνώριση φωνής, η εικόνα και η αναγνώριση προτύπων και πολλά άλλα. Τα νευρωνικά δίκτυα μπορούν να αντιμετωπίσουν πολύπλοκα προβλήματα που είναι δύσκολο να λυθούν με παραδοσιακές μεθόδους προγραμματισμού. Στην παρούσα διπλωματική εργασία θα χρησιμοποιήσουμε νευρωνικά δίκτυα για την τμηματοποίηση εικόνας, όπου και θα χρησιμοποιήσουμε **συνελικτικά** νευρωνικά δίκτυα.



Σχήμα 3-1. Παράσταση ενός απλού νευρωνικού δικτύου.

3.1.2 Συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks) ή CNNs

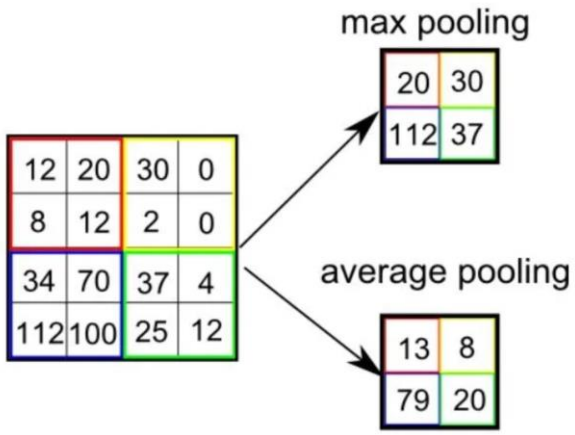
Τα συνελικτικά νευρωνικά δίκτυα είναι ένα είδος νευρωνικού δικτύου που ειδικεύεται κυρίως στην αναγνώριση προτύπων σε δεδομένα εικόνας. Η βασική λειτουργία τους είναι η επεξεργασία της εισόδου με ένα ορθογώνιο συνελικτικό φίλτρο σε όλα τα σημεία της εικόνας, έτσι ώστε να εξαχθούν χαρακτηριστικά που είναι κοινά σε όλο το σύνολο των δεδομένων, όπως φαίνεται στο Σχήμα 3-2. Μέσω της εφαρμογής των σχετικών φίλτρων το δίκτυο είναι σε θέση να αναγνωρίζει χωρικές και χρονικές εξαρτήσεις σε μια εικόνα. Η αρχιτεκτονική προσαρμόζεται καλύτερα στο σύνολο δεδομένων εικόνας λόγω της μείωσης του αριθμού των παραμέτρων που εμπλέκονται και της επαναχρησιμοποίησης των βαρών, καθώς τα μόνα βάρη που μαθαίνονται στο συνελικτικό επίπεδο είναι οι τιμές του ορθογώνιου φίλτρου. Με απλά λόγια, το δίκτυο μπορεί να εκπαιδευτεί ώστε να κατανοεί καλύτερα την πολυπλοκότητα της εικόνας.



Σχήμα 3-2. Σχηματική αναπαράσταση της εφαρμογής ενός συνελικτικού φίλτρου σε μία RGB εικόνα [7] .

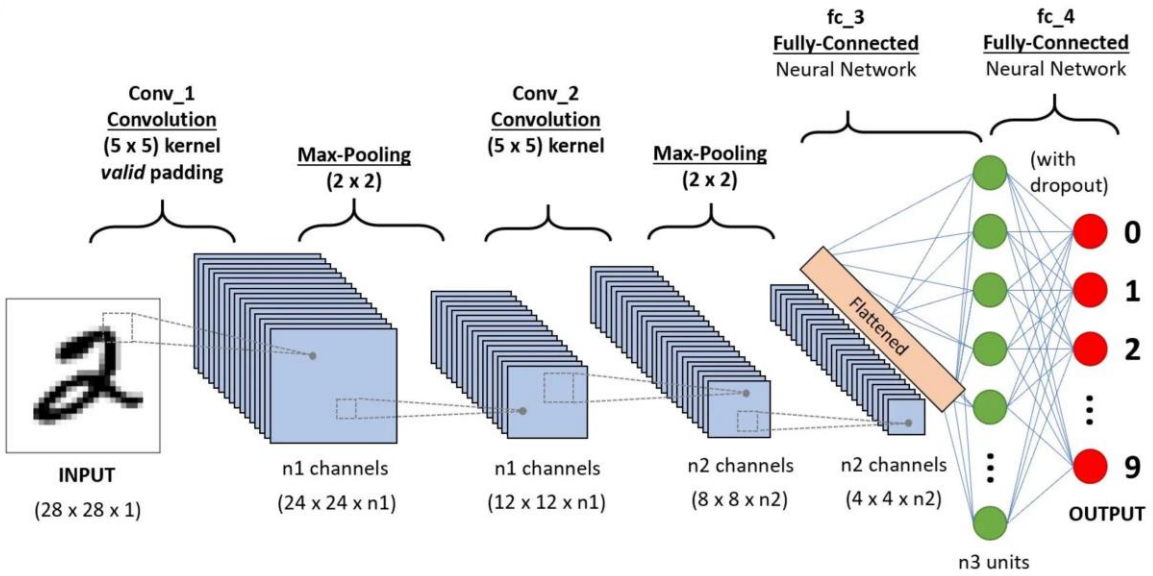
Ο στόχος της λειτουργίας συνέλιξης είναι να εξαγάγει τα χαρακτηριστικά **υψηλού επιπέδου**, όπως οι ακμές, από την εικόνα εισόδου. Τα συνελικτικά δίκτυα δεν χρειάζεται να περιορίζονται σε ένα μόνο συνελικτικό επίπεδο. Συμβατικά, το πρώτο επίπεδο είναι υπεύθυνο για την καταγραφή των χαρακτηριστικών χαμηλού επιπέδου, όπως άκρες, χρώμα, προσανατολισμό κλίσης, κ.λπ. Με πρόσθετα επίπεδα, η αρχιτεκτονική προσαρμόζεται και στα χαρακτηριστικά υψηλού επιπέδου, δίνοντάς μας ένα δίκτυο που έχει πλήρη κατανόηση των εικόνων στο σύνολο δεδομένων, παρόμοια με αυτό που θα κάναμε.

Παρόμοια με το επίπεδο συνέλιξης (Convolutional layer), το **Pooling** layer είναι υπεύθυνο για τη μείωση του χωρικού μεγέθους της εξόδου του προηγούμενου. Αυτό γίνεται για να μειωθεί η υπολογιστική ισχύς που απαιτείται για την επεξεργασία των δεδομένων μέσω μείωσης διαστάσεων. Επιπλέον, είναι χρήσιμο για την εξαγωγή **κυρίαρχων χαρακτηριστικών** που είναι ανεξάρτητα θέσης και προσανατολισμού, διατηρώντας έτσι τη διαδικασία αποτελεσματικής εκπαίδευσης του μοντέλου. Η διαδικασία του pooling γίνεται παρόμοια με ένα ορθογώνιο φίλτρο όπως στο συνελικτικό επίπεδο, αλλά σε αυτή την περίπτωση η έξοδος του φίλτρου είναι είτε ο μέσος όρος των τιμών του ή το μέγιστο εξ αυτών, όπως φαίνεται και στο σχήμα Σχήμα 3-3. Αναλόγως ποια λειτουργία έχουμε αυτή αντιστοιχεί σε **average pooling** και max **pooling αντίστοιχα**.



Σχήμα 3-3. Τύποι pooling [7] .

Όπως φαίνεται και στο Σχήμα **3-4** μετά το τελευταίο pooling επίπεδο, η εικόνα-πίνακας-έξοδος που έχει προκύψει, έχει πλέον πολύ μικρότερο μέγεθος από την αρχική εικόνα και μπορούμε να τον κάνουμε “flatten” σε μορφή διανύσματος στήλης. Η flattened έξοδος τροφοδοτείται σε ένα παραδοσιακό νευρωνικό δίκτυο feed-forward σαν αυτό που περιγράψαμε στην προηγούμενη ενότητα και εφαρμόζεται backpropagation σε κάθε επανάληψη της εκπαίδευσης. Σε μια σειρά εποχών, το μοντέλο είναι σε θέση να διακρίνει μεταξύ κυρίαρχων και ορισμένων χαρακτηριστικών χαμηλού επιπέδου σε εικόνες και να τα ταξινομήσει χρησιμοποιώντας την τεχνική ταξινόμησης Softmax.



Σχήμα 3-4. CNN για την αναγνώριση χειρόγραφων ψηφίων [7] .

3.2 Τμηματοποίηση εικόνας (Semantic segmentation)

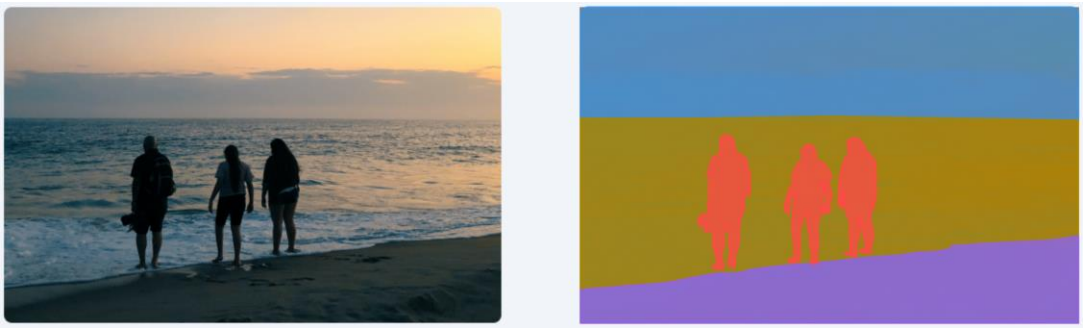
3.2.1 Τμηματοποίηση εικόνας με χρήση CNNs

Η τμηματοποίηση εικόνας ή semantic segmentation αναφέρεται σε μια διαδικασία ανάλυσης εικόνων, στην οποία κάθε εικονοστοιχείο (pixel) της εικόνας αντιστοιχίζεται σε μια κατηγορία ανάλογα με το περιεχόμενο της εικόνας. Για παράδειγμα στο Σχήμα 3-5, όπου έχουμε μία εικόνα μίας πόλης, κάθε pixel της εικόνας επισημαίνεται ως pixel του δρόμου, του ουρανού, του οδοστρώματος, του πεζοδρόμιου, των φαναριών, των πεζών και άλλων στοιχείων που μπορούν να αναγνωριστούν στην εικόνα. Φυσικά το πλήθος των κατηγοριών επιλέγεται από τον σχεδιαστή.

Σχήμα 3-5. Παράδειγμα semantic segmentation σε μία φωτογραφία μίας πόλης.

Για την επίτευξη αυτού του στόχου, χρησιμοποιούνται συνήθως συνελικτικά νευρωνικά δίκτυα όπως αυτά που περιγράψαμε παραπάνω, με ορισμένες τροποποιήσεις στο φίλτρο συνέλιξης καθώς και επίπεδα επέκτασης (upsampling). Κατά τη διάρκεια της εκπαίδευσης,

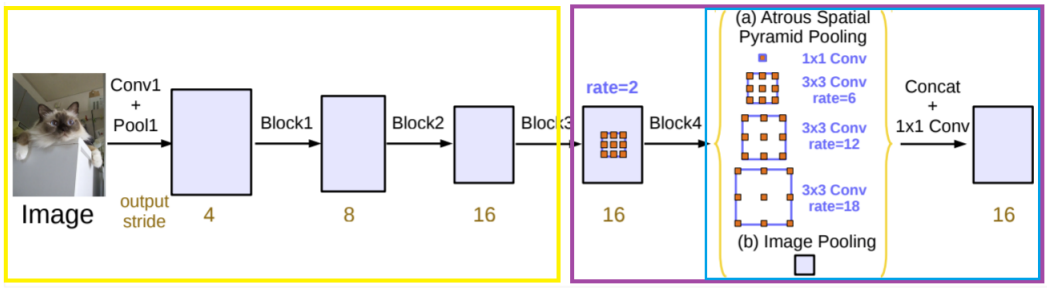
το νευρωνικό δίκτυο τροφοδοτείται με ζευγάρια εικόνων και των αντίστοιχων annotation map τους. Σε αντίθεση με ένα παραδοσιακό CNN όπου έχουμε και εκεί ως είσοδο στο δίκτυο μία εικόνα και στην έξοδο απλώς μία κατηγορία, εδώ έχουμε στην έξοδο μία εικόνα ίδιων διαστάσεων με της εισόδου, ενός καναλιού, όπου η τιμή κάθε pixel αντιστοιχεί στην κατηγορία που ανήκει. Στο Σχήμα 3-6 φαίνεται αριστερά η είσοδος και δεξιά η αναμενόμενη έξοδος.



Σχήμα 3-6. Ζευγάρι εκπαίδευσης για δίκτυο τμηματοποίησης εικόνας.

3.2.2 Η δομή ενός CNN τμηματοποίησης εικόνας – DeepLabV3

Όπως φαίνεται και στο Σχήμα 3-7 τα αρχικά επίπεδα (κίτρινα) ενός CNN τμηματοποίησης εικόνας είναι παρόμοια με ενός παραδοσιακού CNN, καθώς έχουν τον ίδιο τελικό στόχο, την εξαγωγή χαρακτηριστικών. Παρόλα αυτά όπως αναφέραμε και στην προηγούμενη παράγραφο η έξοδος είναι πάλι μία εικόνα. Υπάρχουν λοιπόν πρόσθετα επίπεδα (μοβ) αποτελούμενα από διευρυμένες συνελίξεις, με σκοπό την εξαγωγή χαρακτηριστικών με μικρό και μεγάλο οπτικό πεδίο. Τα επίπεδα στο μπλε κουτί είναι υπεύθυνα για την δημιουργία της τελικής τμηματοποιημένης εικόνας, και αντιστοιχίζουν τις κατηγορίες που έχουν εντοπιστεί στα pixel που ανήκουν. Στις παρακάτω παραγράφους θα αναλύσουμε από τι αποτελείται το κάθε επίπεδο καθώς και γιατί.



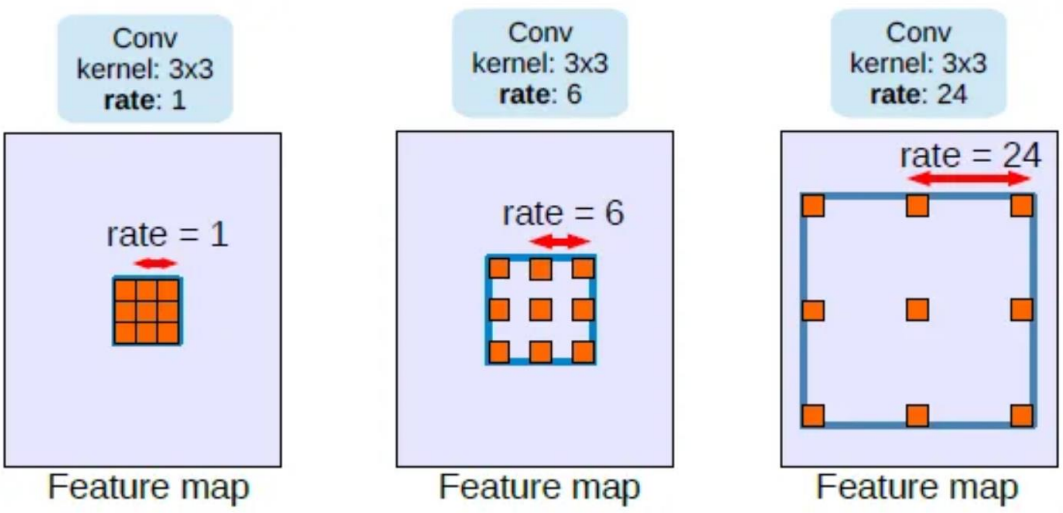
Σχήμα 3-7. Τα επίπεδα του DeepLabV3 για τμηματοποίηση εικόνας [25] .

3.2.3 Διευρυμένες συνελίξεις / Atrous(dilated) convolutions

Ένα φίλτρο διευρυμένης συνέλιξης εκτελεί τις ίδιες αριθμητικές πράξεις με το κλασσικό φίλτρο συνέλιξης που περιγράφηκε παραπάνω. Η διαφορά του έγκειται στις θέσεις των pixel εισόδου. Αναλόγως με τον συντελεστή r (rate) ο δείκτης εισόδου για το pixel εισόδου μετατοπίζεται r θέσεις τόσο οριζόντια, όσο και κατακόρυφα.

Όπως φαίνεται και στο Σχήμα 3-8 αυτό έχει ως αποτέλεσμα, μία διευρυμένη συνέλιξη να έχει πολύ πιο ευρύ οπτικό πεδίο (field of view). Μέσου της παραμέτρου r λοιπόν έχουμε έναν αποτελεσματικό μηχανισμό για τον έλεγχο του οπτικού πεδίου, που μας επιτρέπει να

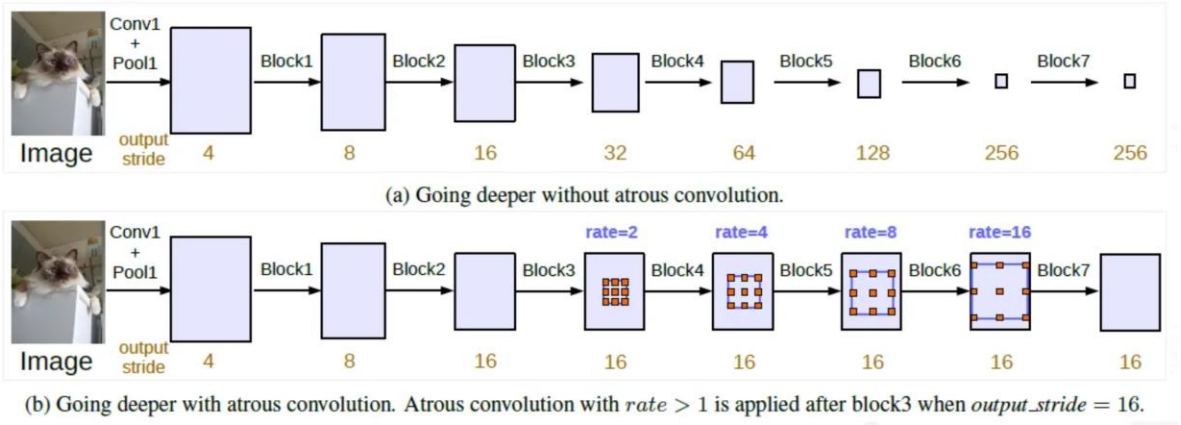
βρούμε την καλύτερη αντιστάθμιση μεταξύ ακριβούς εντοπισμού (μικρό οπτικό πεδίο) και αφομοίωσης περιβάλλοντος (μεγάλο οπτικό πεδίο).



Σχήμα 3-8. Σχηματική επεξήγηση μίας διευρυμένης συνέλιξης [25] .

3.2.4 Εμβάθυνση μέσω της διευρυμένης συνέλιξης και της τεχνικής Multi-Grid

Ένα άλλο πλεονέκτημα της διευρυμένης συνέλιξης είναι η ικανότητα αύξησης του οπτικού πεδίου, χωρίς την μείωση του μεγέθους της εξόδου, που είναι κάτι που επιθυμούμε σε εφαρμογές τμηματοποίησης εικόνας. Γενικά μας συμφέρει να διατηρήσουμε το μέγεθος των βαθύτερων επιπέδων όσο πιο κοντά στην αρχική εικόνα, καθώς η έξοδος του δικτύου είναι μία εικόνα ίδιων διαστάσεων με της εισόδου. Με άλλα λόγια το **output stride** παραμένει σταθερό. Η παρατήρηση αυτή διατυπώνεται και εύκολα στο Σχήμα 3-9. Επιπλέον, σε σχέση με ένα παραδοσιακό φίλτρο συνέλιξης, για δεδομένο μέγεθος οπτικού πεδίου, μειώνεται το πλήθος των παραμέτρων που πρέπει να υπολογιστούν, άρα και το υπολογιστικό φορτίο.

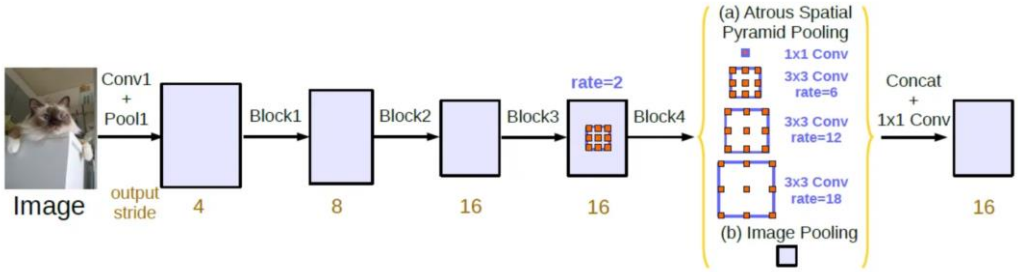


Σχήμα 3-9. Σύγκριση παραδοσιακής συνέλιξης (a) και διευρυμένης συνέλιξης (b) [25] .

3.2.5 Atrous Spatial Pyramid Pooling (ASPP)

Τα επίπεδα από δω και πέρα χρησιμοποιούνται για την δημιουργία της τελικής τμηματοποιημένης εικόνας που θα είναι η έξοδος του νευρωνικού δικτύου, βασιζόμενα στα χαρακτηριστικά που εξήχθησαν από τα προηγούμενα επίπεδα. Πιο συγκεκριμένα, στην έξοδο του προηγούμενου επιπέδου εφαρμόζεται Atrous Spatial Pyramid Pooling (ASPP).

Ουσιαστικά εφαρμόζονται παράλληλα πολλές διαφορετικές (4 στην περίπτωση του DeepLabV3) διευρυμένες συνελίξεις με διαφορετικό rate η κάθε μία, για τη διαχείριση της τμηματοποίησης του αντικειμένου σε διαφορετικές κλίμακες. Έπειτα στην έξοδο της κάθε μίας εφαρμόζεται pooling ώστε να έχουν όλες το ίδιο μέγεθος. Άρα θα έχουμε K διαφορετικά feature maps WxH διαστάσεων. Στην συνέχεια αυτά συνενώνονται (concat operation) και έτσι το τελικό feature map θα έχει διαστάσεις WxHxK. Τέλος σε αυτό εφαρμόζονται F (όπου F το πλήθος των κλάσεων) συνελικτικά φίλτρα 1x1 ώστε να δημιουργηθούν οι τελικές πιθανότητες των προβλέψεων του μοντέλου για κάθε κλάση (logits). Έπειτα τα logits θα περαστούν από μία συνάρτηση ενεργοποίησης για την κανονικοποίηση των πιθανοτήτων αυτών, ώστε το δίκτυο να έχει την τελική του έξοδο, δηλαδή μία τμηματοποιημένη εικόνα, με κατηγορία κάθε pixel την πιο πιθανή κατηγορία. Η διαδικασία αυτή αναπαρίσταται στο Σχήμα 3-10.



Σχήμα 3-10. Atrous Spatial Pyramid Pooling (ASPP) [25] .

4 Αλγόριθμος ελέγχου κατεύθυνσης μέσω τμηματοποίησης εικόνας για τετράποδα ρομπότ

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε τα βήματα που έγιναν για την ανάπτυξη ενός αλγόριθμου ελέγχου της κατεύθυνσης ενός τετράποδου γεωργικού ρομπότ με την χρήση νευρωνικών δικτύων βαθιάς μάθησης. Εδώ γίνεται το πρώτο βήμα προς αυτή την κατεύθυνση, δηλαδή αξιολογείται η ιδέα και η πρώτη υλοποίηση του αλγορίθμου σε μία απλούστερη ρομποτική πλατφόρμα η οποία επιτρέπει κίνηση σε κάθε κατεύθυνση χρησιμοποιώντας τροχούς τύπου “mecanum” και είναι ό,τι πιο κοντά υπάρχει στις δυνατότητες κίνησης ενός τετράποδου ρομπότ. Το τελικό βήμα της αξιοποίησης του αλγορίθμου στο τετράποδο Laelaps αλλά και σε αλλά τετράποδα ρομπότ της ομάδας αφήνεται ως μελλοντική εργασία. Αρχικά θα περιγράψουμε το υλικό που χρησιμοποιήθηκε, τόσο για την εκπαίδευση του μοντέλου μηχανικής μάθησης, όσο και για την ρομποτική πλατφόρμα πάνω στην οποία θα εκτελεστεί. Έπειτα θα αναλύσουμε τον κώδικα εκπαίδευσης του μοντέλου, καθώς και τα δεδομένα εκπαίδευσης και θα καταλήξουμε με τον τρόπο που χρησιμοποιούμε την έξοδο του, δηλαδή μία τμηματοποιημένη εικόνα σε πραγματικό χρόνο, για τον έλεγχο της ρομποτικής πλατφόρμας.

4.2 Απαιτούμενο υλικό

4.2.1 Απαιτούμενο υλικό για την εκπαίδευση του μοντέλου μηχανικής μάθησης

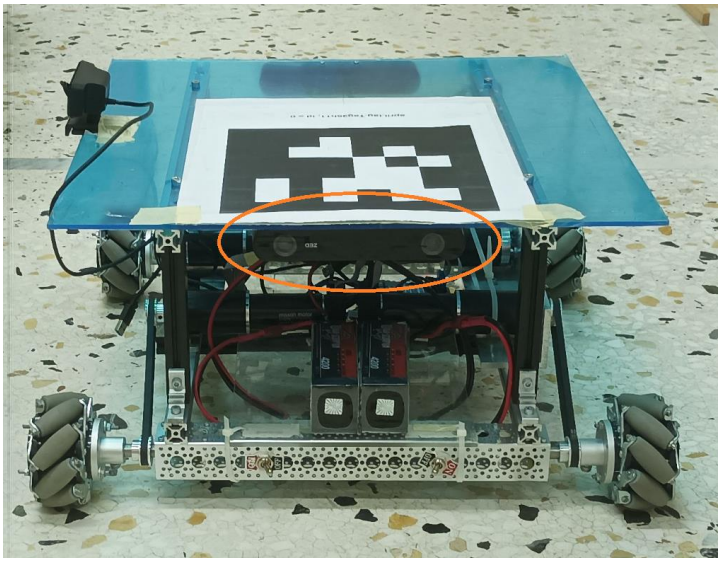
Η βιβλιοθήκη μηχανικής μάθησης PyTorch υποστηρίζει εκπαίδευση του μοντέλου τόσο μέσω της ΚΜΕ (CPU), όσο και μέσω της κάρτας γραφικών (GPU). Η εκπαίδευση σε GPU έχει γίνει ιδιαίτερα δημοφιλής καθώς οι GPU έχουν πολλούς πυρήνες επεξεργασίας που μπορούν να εκτελέσουν παράλληλα πολλές πράξεις, επιταχύνοντας έτσι τον χρόνο εκπαίδευσης, σε αντίθεση με την CPU η οποία από την φύση της περιορίζεται στην εκτέλεση των πράξεων σειριακά. Για την εκπαίδευση απλών νευρωνικών δικτύων με λίγες παραμέτρους η CPU είναι επαρκής. Παρόλα αυτά για την εκπαίδευση συνελικτικών νευρωνικών δικτύων, τα οποία έχουν πολύ μεγαλύτερο αριθμό παραμέτρων προς εκπαίδευση η GPU είναι απαραίτητη, καθώς ο χρόνος εκπαίδευσης είναι πολύ μικρότερος. Ενδεικτικά μιλώντας, συγκρίνοντας μια σύγχρονη CPU ενάντια σε μια σύγχρονη GPU παρατηρούμε ταχύτητες έως 6 φορές πιο γρήγορες. Φυσικά για την εκπαίδευση του μοντέλου η κάρτα γραφικών πρέπει να υποστηρίζεται από την βιβλιοθήκη PyTorch. Γενικότερα η PyTorch χρησιμοποιεί την βιβλιοθήκη παράλληλης επεξεργασίας δεδομένων CUDA Toolkit. Η κάθε έκδοση της PyTorch έχει μία ελάχιστη απαίτηση στην έκδοση αυτής της βιβλιοθήκης. Γενικότερα όσο πιο καινούρια η έκδοση της PyTorch, τόσο και πιο πρόσφατη η ελάχιστή έκδοση του CUDA Toolkit που απαιτείται. Αυτό συνεπάγεται και πιο πρόσφατες κάρτες γραφικών σαν ελάχιστη απαίτηση, κάτι που θα μας απασχολήσει και στην επόμενη παράγραφο. Η εκπαίδευση του μοντέλου στην παρούσα εργασία έγινε σε μία κάρτα γραφικών NVidia RTX 3070, όπου χρησιμοποιήθηκε η PyTorch 2.0.0 και το NVidia Toolkit 12.0.

4.2.2 Απαιτούμενο υλικό για εκτέλεση του μοντέλου μηχανικής μάθησης στο ρομπότ

Για την εκτέλεση των αρχικών πειραμάτων χρησιμοποιήθηκε ένα laptop ASUS που διαθέτει κάρτα γραφικών NVidia GeForce 940MX. Λόγω της παλαιότητας της συγκεκριμένης κάρτας γραφικών χρειάστηκε να χρησιμοποιηθεί μία παλαιότερη έκδοση του PyTorch (1.5.1), ώστε να είναι συμβατή με την έκδοση του CUDA toolkit (10.2.89) που υποστηρίζει η συγκεκριμένη κάρτα. Παρόλα αυτά δεν αντιμετωπίστηκε κάποιο πρόβλημα στην εκτέλεση του μοντέλου. Η εικόνα εισόδου στο μοντέλο μηχανικής μάθησης λαμβάνεται από την κάμερα Zed 2i της Stereolabs (Σχήμα 4-1. Η κάμερα Stereolabs Zed 2i.).

Σχήμα 4-1. Η κάμερα Stereolabs Zed 2i.

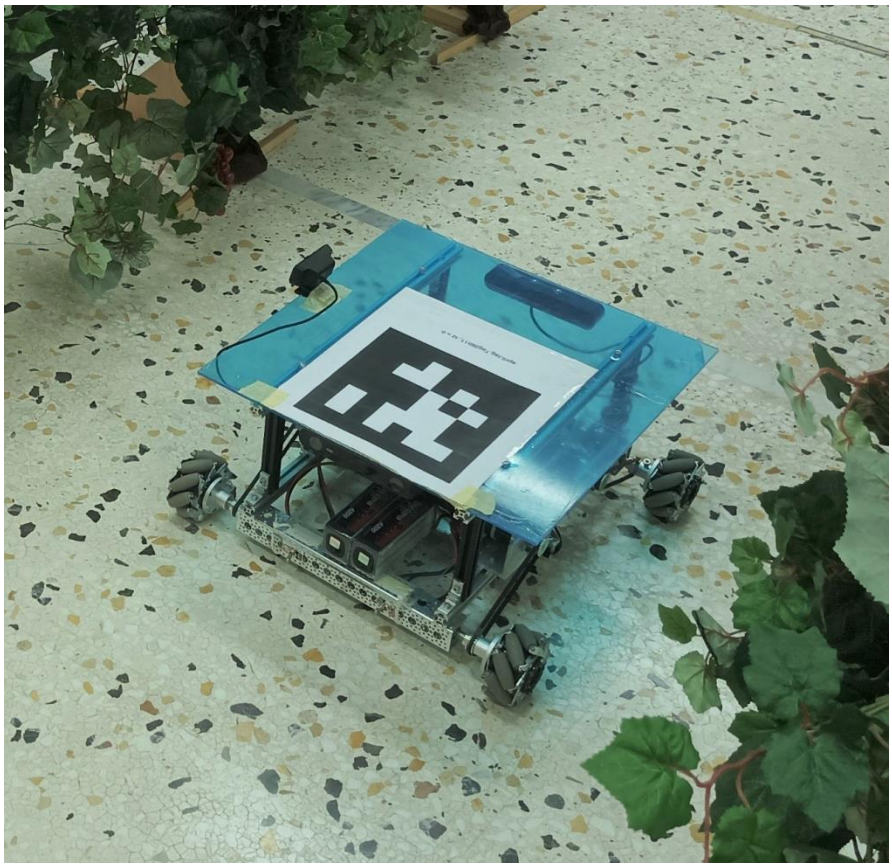
Η επιλογή της συγκεκριμένης κάμερας οφείλεται κυρίως στο γεγονός πως το εργαστήριο μας αναπτύσσει παράλληλα και άλλους αλγόριθμους κατεύθυνσης του ρομπότ μέσω εικόνων βάθους οι οποίοι μακροπρόθεσμα μπορεί να συνδυαστούν με την μεθοδολογία ελέγχου που παρουσιάζεται στην παρούσα εργασία, οπότε την διατηρήσαμε και στην παρούσα εργασία, για λόγους συμβατότητας και απλοποίησης, σε περίπτωση που πράγματι χρειαστεί να τρέξουν παράλληλα. Επιπρόσθετα, η συγκεκριμένη κάμερα διαθέτει πλήθος αισθητήρων, μεταξύ άλλων και ένα επιταχυνσιόμετρο (IMU) 3 βαθμών ελευθερίας, παρέχοντας μας πληροφορίες προσανατολισμού της οποίες χρησιμοποιούμε κατά την διάρκεια εκτέλεσης στροφών για την αλλαγή σειράς αμπελιού, όπως θα περιγραφεί σε παρακάτω κεφάλαια. Η κάμερα είναι τοποθετημένη στο εμπρόσθιο τμήμα του ρομπότ όπως φαίνεται και στο Σχήμα 4-2.



Σχήμα 4-2. Εμπρόσθια όψη του ρομπότ, δείχνοντας την θέση της κάμερας.

4.2.3 Το σύστημα ελέγχου πορείας του robot

Μια τροχήλατη ρομποτική πλατφόρμα (RP) χρησιμοποιήθηκε για την επικύρωση της ιδέας (Σχήμα 4-3). Το RP έχει σχεδιαστεί και κατασκευαστεί για ερευνητικούς σκοπούς, αποτελούμενο από ειδικά εξαρτήματα κατασκευασμένα στο εργαστήριο καθώς και έτοιμα εξαρτήματα (π.χ. προφίλ αλουμινίου, μονάδες ρουλεμάν κ.λπ.). Το σύστημα κίνησής του διαθέτει τέσσερις τροχούς τύπου mecanum για να παρέχει στο ρομπότ δυνατότητες κίνησης προς όλες τις κατευθύνσεις. Οι τροχοί τροφοδοτούνται από τέσσερις κινητήρες Maxon DC (RE 35) σε συνδυασμό με πλανητικούς μειωτήρες (GP 42) και αυξητικούς κωδικοποιητές (HEDL 5540), παρέχοντας 5 Nm συνεχούς ροπής ανά τροχό. Για τη μετάδοση ισχύος στους τροχούς χρησιμοποιούνται ιμάντες και τροχαλίες GT2 ενώ παράλληλα προστατεύουν τους άξονες του κινητήρα από τυχόν αυξημένα φορτία. Δύο ελεγκτές DC κινητήρων RoboClaw 2x30A χρησιμοποιούνται για των έλεγχο δύο κινητήρων ο καθένας. Οι κωδικοποιητές που είναι προσαρτημένοι στους κινητήρες διαβάζονται από τους ελεγκτές, οι οποίοι εκτελούν τοπικά σχήματα ελέγχου PID που μπορούν να ακολουθήσουν με ακρίβεια εντολές ταχύτητας για όλους τους τροχούς. Οι δύο ελεγκτές συνδέονται μέσω USB στον κύριο υπολογιστή του συστήματος, ο οποίος είναι ένα Raspberry Pi 2 μοντέλο B (RPi) με το Raspbian OS. Ο χειριστής μπορεί να συνδεθεί στο RPi χρησιμοποιώντας το πρωτόκολλο δικτύου WiFi και Secure Shell (SSH) ή UDP. Το σύστημα τροφοδοτείται από δύο μπαταρίες LiPo για τους ελεγκτές roboclaw και ένα powerbank για τον κύριο υπολογιστή.



Σχήμα 4-3. Η ρομποτική πλατφόρμα με ρόδες Mecanum που χρησιμοποιήσαμε.

4.3 Το συνελικτικό νευρωνικό δίκτυο μηχανικής μάθησης

4.3.1 Η βιβλιοθήκη PyTorch και η γλώσσα προγραμματισμού Python

Για την βιβλιοθήκη που θα θεμελιώσει την υλοποίηση του μοντέλου μηχανικής μάθησης έχουμε να διαλέξουμε ανάμεσα στις βιβλιοθήκες TensorFlow και PyTorch. Αποτελούν δύο από τις πιο δημοφιλείς βιβλιοθήκες που χρησιμοποιούνται στον τομέα της μηχανικής μάθησης και της αναγνώρισης προτύπων. Και οι δύο βιβλιοθήκες έχουν τα δικά τους πλεονεκτήματα και χαρακτηριστικά, και η επιλογή εξαρτάται από τις ανάγκες της εφαρμογής.

Η PyTorch αποτέλεσε ξεκάθαρη επιλογή για αρκετούς λόγους. Αρχικά, προσφέρει ευκολία χρήσης μέσω μιας ευέλικτης και ευανάγνωστης σύνταξης, που καθιστά την ανάπτυξη μοντέλων γρήγορη και ευκολότερη, ιδιαίτερα στον τομέα της έρευνας όπου υλοποιούνται πρωτότυπες εφαρμογές, σε αντίθεση με την σύνθετη φύση της TensorFLow, η οποία χρησιμοποιείται κυρίως από την βιομηχανία για την παραγωγή τελικών προϊόντων. Το μοντέλο DeepLabV3+ [11] που χρησιμοποιούμε στην παρούσα εργασία περιλαμβάνεται σε αυτή την βιβλιοθήκη και η χρήση του καθίσταται ιδιαίτερα εύκολη. Επιπλέον, η PyTorch είναι σχεδιασμένη έτσι ώστε να εκμεταλλεύεται τις δυνατότητες της GPU για ταχύτερη επεξεργασία δεδομένων, κάτι σημαντικότερα πιο δύσκολο να υλοποιηθεί στην βιβλιοθήκη TensorFlow. Η χρήση GPU μπορεί να επιταχύνει σημαντικά τον χρόνο εκτέλεσης των αλγορίθμων μηχανικής μάθησης, επιτρέποντας την εκτέλεση μεγαλύτερων και πιο περίπλοκων μοντέλων, κάτι που είναι πολύ σημαντικό για την παρούσα εργασία λόγω των πολλαπλών αναλύσεων που χρειάστηκε να κάνουμε για να καταλήξουμε στο βέλτιστο μοντέλο με τις βέλτιστες παραμέτρους. Γενικότερα, η PyTorch παρέχει μια ευέλικτη και φιλική προς τον χρήστη πλατφόρμα, που επιτρέπει στους επιστήμονες δεδομένων και στους προγραμματιστές να εργάζονται με τη βιβλιοθήκη με ευκολία και να αναπτύξουν γρήγορα τα μοντέλα τους.

Συνολικά, η PyTorch είναι μια ισχυρή επιλογή για την επίλυση προβλημάτων που σχετίζονται με την επεξεργασία εικόνων. Το λογισμικό της παρούσας εργασίας έχει γραφτεί σε περιβάλλον Python, καθώς πάνω σε αυτή έχει γραφτεί η βιβλιοθήκη PyTorch. Φυσικά υπάρχουν εκδόσεις της για άλλες γνωστές γλώσσες προγραμματισμού όπως η C++, αλλά διαλέξαμε το περιβάλλον της Python, λόγω της μεγάλη απλότητας του, χωρίς να υστερεί σημαντικά σε δυνατότητες. Επιπλέον διαθέτει την πολύ ισχυρή βιβλιοθήκη επεξεργασίας πινάκων NumPy [12] καθώς και την βιβλιοθήκη επεξεργασίας εικόνων opencv [13] τις οποίες χρησιμοποιούμε εκτενώς στον κώδικα της παρούσας εργασίας.

Η γλώσσα αυτή λοιπόν είναι ιδανική για την επίλυση πρωτότυπων προβλημάτων, όπου ο κύριος στόχος είναι η ανάπτυξη του μοντέλου μηχανικής μάθησης. Συνήθως η χρήση άλλων γλωσσών προγραμματισμού πιο χαμηλού επιπέδου όπως η C, C++ γίνεται σε επόμενο στάδιο, όπου έχει ολοκληρωθεί η ανάπτυξη του μοντέλου, με σκοπό την βελτιστοποίηση της αλληλεπίδρασης υλικού, λογισμικού και μοντέλου μάθησης για γρηγορότερη και πιο αποδοτική εκτέλεση.

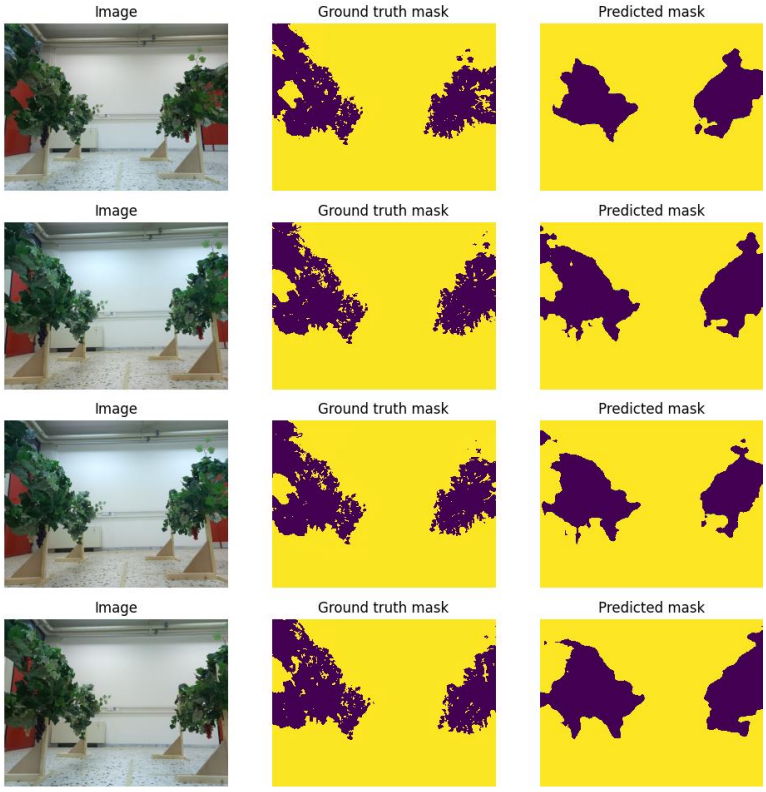
4.3.2 Σύγκριση διαθέσιμων μοντέλων μάθησης

Η PyTorch διαθέτει μερικά από τα πιο ισχυρά μοντέλα τμηματοποίησης εικόνας. Τόσο το U-Net όσο και το DeepLabv3 είναι δύο αρχιτεκτονικές που χρησιμοποιούνται για την τμηματοποίηση εικόνας, αλλά έχουν διαφορετικές προσεγγίσεις και χαρακτηριστικά. Το U-Net χρησιμοποιεί μια αρχιτεκτονική encoder-decoder, η οποία επιτρέπει την αναγνώριση και την ανακατασκευή της εικόνας, διατηρώντας τη λεπτομέρεια. Από την άλλη πλευρά, το

DeepLabv3 χρησιμοποιεί τεχνικές atrous convolutions και atrous spatial pyramid pooling για την αύξηση του πεδίου θέασης και την ακριβή αναγνώριση των αντικειμένων. Και τα δύο μοντέλα έχουν επιτύχει εξαιρετικά αποτελέσματα σε προβλήματα τμηματοποίησης εικόνας, Το U-Net είναι κατάλληλο για προβλήματα που απαιτούν τη διατήρηση λεπτομερειών, ενώ το DeepLabv3 έχει επιδόσεις όταν απαιτείται ακρίβεια και αναγνώριση αντικειμένων σε διάφορες κλίμακες. Στην δική μας περίπτωση η επιλογή είναι ξεκάθαρη, υπέρ του DeepLabV3 καθώς έχουμε έντονες αλλαγές στις κλίμακες των αμπελιών αναλόγως με την απόσταση τους από το ρομπότ, ενώ δεν μας ενδιαφέρει η αποτύπωση της λεπτομέρειας στην τμηματοποιημένη εικόνα. Ακόμα το DeepLabv3, μπορεί και έρχεται προεκπαιδευμένο σε μεγάλα σύνολα δεδομένων, όπως το COCO ή το PASCAL VOC, για να αποκτήσει γενική αναγνώριση και γενικευτική ικανότητα τμηματοποίησης, κάτι το οποίο αποδεικνύεται ιδιαίτερα σημαντικό στο επόμενο κεφάλαιο.

4.3.3 Αξιολόγηση απόδοσης του μοντέλου DeepLabV3+ με προεκπαιδευμένα βάρη

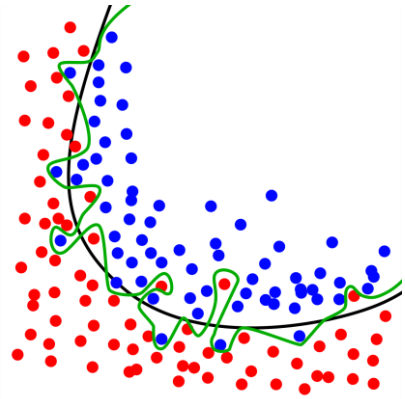
Αρχικά ορίσαμε ένα μοντέλο στην PyTorch με βάση το δίκτυο DeepLabV3+ με τα βάρη με τα οποία έχει προεκπαιδευτεί, και δύο κλάσεις συνολικά. Η μία αντιπροσωπεύει τα αμπέλι και η άλλη την υπόλοιπη εικόνα. Στο Σχήμα 4-4 η μεσαία στήλη αντιπροσωπεύει την σωστή τμηματοποιημένη εικόνα, ενώ η δεξιά την πρόβλεψη που έκανε το μοντέλο. Προς έκπληξή μας, το δίκτυο, χωρίς καμία εκπαίδευση αντιστοίχησε τις κλάσεις με εντυπωσιακή ακρίβεια. Όπως είναι φυσικό, λόγω του ότι δεν είχε προηγηθεί κάποια εκπαίδευση στο μοντέλο, παρουσίασε κάποιες αδυναμίες, όπως φαίνεται στο ζευγάρι εικόνων πάνω δεξιά, όπου λανθασμένα ταξινομήθηκε μέρος του αμπελιού σαν παρασκήνιο. Με βάση αυτά το μοντέλο DeepLabV3 κρίθηκε κατάλληλο και ακολούθησε η εκπαίδευσή του.



Σχήμα 4-4. Σύγκριση των προβλέψεων του μοντέλου με προεκπαιδευμένα βάρη (δεξιά στήλη) με το σωστό αποτέλεσμα (μεσαία στήλη).

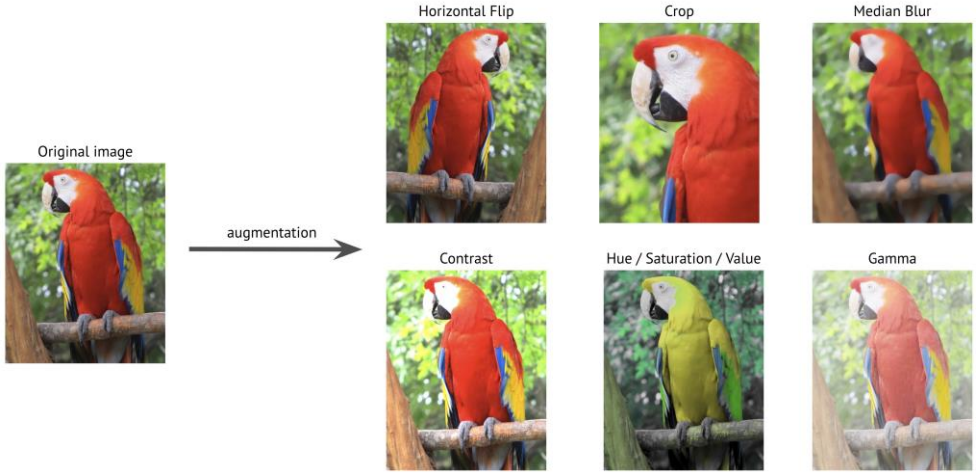
4.3.4 Επαύξηση δεδομένων εκπαίδευσης με χρήση της βιβλιοθήκης Albumentations

Η επαύξηση δεδομένων εικόνων είναι μια τεχνική που χρησιμοποιείται στη μηχανική μάθηση για τη βελτίωση της απόδοσης των μοντέλων εκπαίδευσης στην ανάλυση εικόνων, περιορίζοντας το φαινόμενο του overfitting σε μοντέλα μηχανικής μάθησης για εικόνες. Πιο συγκεκριμένα, κατά τη διαδικασία εκπαίδευσης, ένα μοντέλο μπορεί να μάθει πολύ καλά τα δεδομένα εκπαίδευσης και να υπερ-γενικεύει κακά τα δεδομένα που δεν έχει δει ποτέ. Όπως π.χ. φαίνεται στο Σχήμα 4-5 η ανοιχτή πράσινη γραμμή έχει υπερ-εξειδικευτεί στα συγκεκριμένα δεδομένα εκπαίδευσης, ενώ το σωστό αποτέλεσμα θα ήταν η σκούρα πράσινη γραμμή.



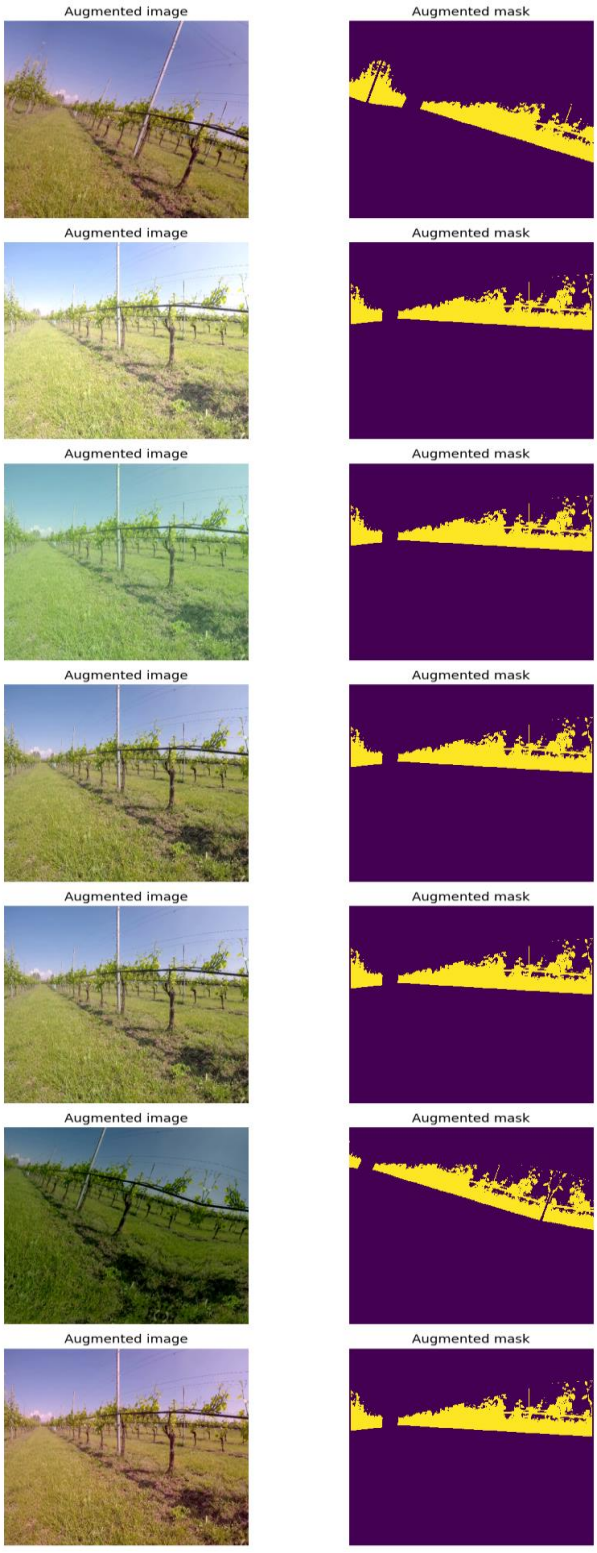
Σχήμα 4-5. Παράδειγμα φαινομένου overfitting σε ένα σύνολο δυσδιάστατων δεδομένων.

Η επαύξηση δεδομένων εικόνας μπορεί να βοηθήσει στην αντιμετώπιση αυτού του προβλήματος εφαρμόζοντας διάφορους μετασχηματισμούς στο αρχικό σύνολο εικόνων, προκειμένου να το επαυξήσει με νέες εικόνες που παρουσιάζουν παρόμοια χαρακτηριστικά με τις αρχικές. Όμως λόγω των μικρών διαφορών που έχουν σε χαρακτηριστικά, που δεν επηρεάζουν το περιεχόμενο που πρέπει να «μαντέψει» το δίκτυο, μειώνεται το πλήθος της πληροφορίας που θα μπορούσε να υπερ-γενικεύσει το δίκτυο βασισμένο σε αυτή. Αυτό επιτυγχάνεται με μικρές τροποποιήσεις, όπως αναστροφή, περιστροφή, μετατόπιση, κλιμάκωση, περικοπή, αλλαγή της φωτεινότητας ή και απόχρωσης και προσθήκη θορύβου, όπως φαίνεται στο Σχήμα 4-6. Καθώς το μοντέλο δεν έχει δει αυτές τις εικόνες κατά τη διαδικασία εκπαίδευσης, αντιμετωπίζει περισσότερη ποικιλία στα δεδομένα και μπορεί να γενικεύσει καλύτερα στα νέα δεδομένα εικόνας που δεν έχει δει ποτέ.



Σχήμα 4-6. Εφαρμογή διάφορων μετασχηματισμών επαύξησης εικόνας σε μία εικόνα.

Στην δική μας περίπτωση όμως η έξοδος του δικτύου αποτελεί μία τμηματοποιημένη εικόνα. Άρα για κάθε μετασχηματισμό εικόνας εισόδου στα δεδομένα εκπαίδευσης θα πρέπει να μετασχηματίσουμε και την τμηματοποιημένη εικόνα εξόδου, σε περίπτωση που έχουμε χωρικούς μετασχηματισμούς, όπως φαίνεται και στο Σχήμα 4-7. Σε αυτή την διαδικασία μας βοηθάει η βιβλιοθήκη Albumentations.



Σχήμα 4-7. Παραδείγματα μετασχηματισμών σε ένα ζευγάρι δεδομένων εκπαιδευσης τμηματοποίησης εικόνας.

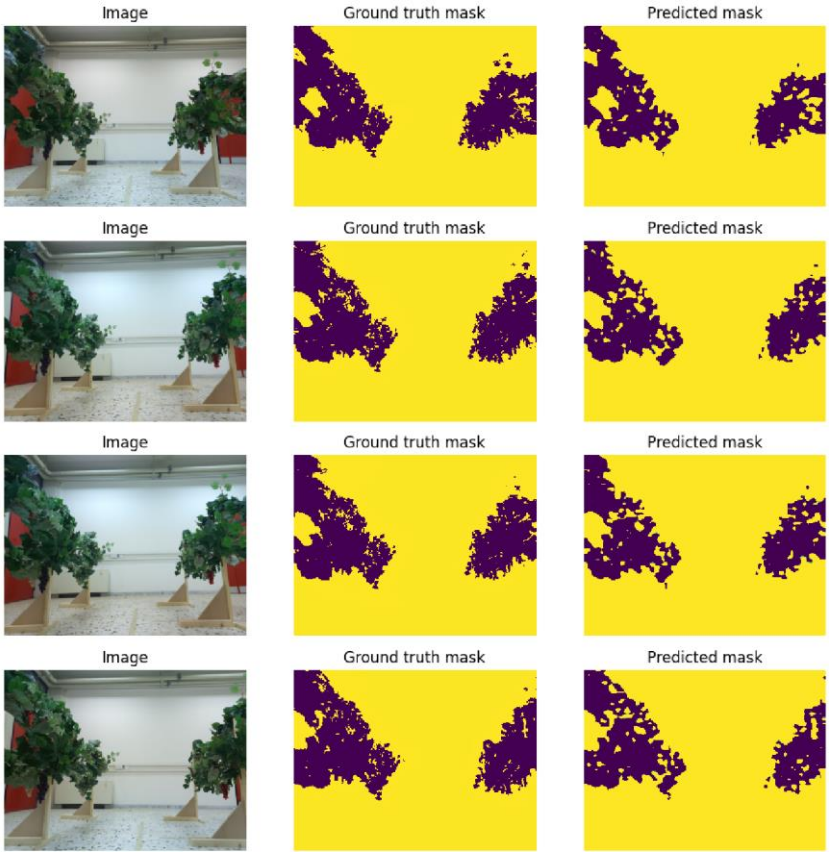
Μέσω αυτής ορίζουμε ένα pipeline μετασχηματισμών, όπου κάθε μετασχηματισμός δέχεται τιμές παραμετροποίησης. Το pipeline που εφαρμόσαμε στο δικό μας μοντέλο απεικονίζεται στο Σχήμα 4-8. Π.χ. μπορούμε να ορίσουμε την μέγιστη περιστροφή που πραγματοποιείται, το μέγιστο ζουμ ή το όριο φωτεινότητας/αντίθεσης. Επίσης για κάθε μετασχηματισμό στο pipeline υπάρχει η τιμή πιθανότητας p. Αυτή ορίζει την πιθανότητα να εφαρμοστεί ο μετασχηματισμός όταν καλείται με είσοδο ένα αρχικό ζευγάρι εκπαίδευσης.

```
train_transform = A.Compose(
    [
        A.Resize(600, 600),
        A.ShiftScaleRotate(shift_limit=0.2, scale_limit=0.2, rotate_limit=30, p=0.5),
        A.RGBShift(r_shift_limit=25, g_shift_limit=25, b_shift_limit=25, p=0.5),
        A.RandomBrightnessContrast(brightness_limit=0.3, contrast_limit=0.3, p=0.5),
        A.Normalize(mean=(0.49698316, 0.51454199, 0.43739734), std=(0.20042156, 0.20729762, 0.30175445)),
        ToTensorV2(),
    ]
)
```

Σχήμα 4-8. Pipeline μετασχηματισμού που εφαρμόσαμε στο δικό μας μοντέλο.

4.3.5 Αξιολόγηση της απόδοσης του μοντέλου DeepLabV3+ μετά την εκπαίδευση

Το δίκτυο εκπαιδεύτηκε συνολικά για 10 εποχές. Για την αξιολόγηση της απόδοσης του δικτύου, μέρος των δεδομένων εκπαίδευσης (10%) χρησιμοποιήθηκε σαν validation set, ώστε να δούμε σε ποια εποχή έχουμε την καλύτερη ακρίβεια. Όπως φαίνεται και από τα ζευγάρια των predicted/ground truth, το δίκτυο πλέον έχει πάρα πολύ καλή ακρίβεια όπου οι δύο αυτές εικόνες σχεδόν ταυτίζονται μεταξύ τους.



Σχήμα 4-9. Σύγκριση των προβλέψεων του μοντέλου μετά την εκπαίδευση (δεξιά στήλη) με το σωστό αποτέλεσμα (μεσαία στήλη).

4.4 Τα δεδομένα εκπαίδευσης του δικτύου και η δημιουργία τους

4.4.1 Δημιουργία ετικετών εκπαίδευσης στον χώρο του εργαστηρίου

Οι πρώτες δοκιμές και η αξιολόγηση του αλγορίθμου έγιναν σε μία αίθουσα όπου έχουν τοποθετηθεί διάδρομοι τεχνητών αμπελιών. Πέρα από τις σταθερές και ελεγχόμενες συνθήκες πειράματος που προσφέρει αυτός ο χώρος, έχει και ένα άλλο σημαντικό πλεονέκτημα. Όπως φαίνεται και στο Σχήμα 4-10, το λευκό χρώμα των τοίχων έχει έντονη αντίθεση με το σκούρο πράσινο χρώμα των αμπελιών. Αυτή η αντίθεση επιτάχυνε την διαδικασία της δημιουργίας ετικετών για την κάθε φωτογραφία.

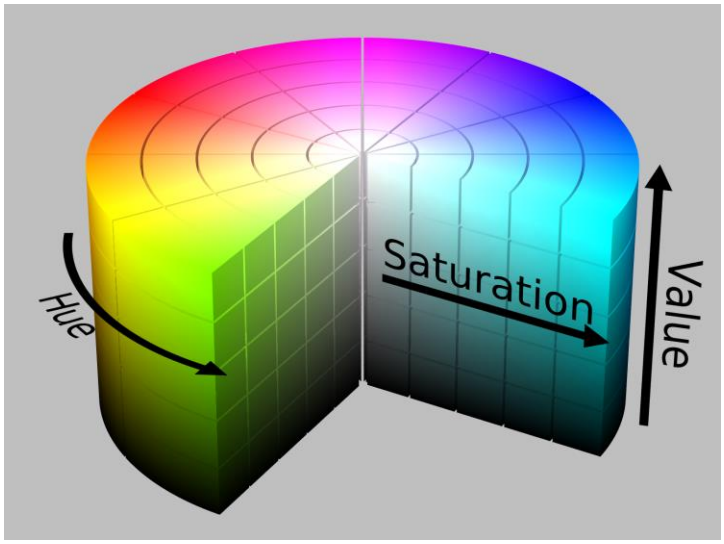
Σχήμα 4-10. Στιγμιότυπο εκπαίδευσης από τον χώρο του εργαστηρίου.

Παραδοσιακά σε εφαρμογές τμηματοποίησης εικόνων κάποιος είναι υπεύθυνος για την σκιαγράφηση των εκατοντάδων ή και χιλιάδων στιγμιότυπων εικόνων ένα προς ένα. Η διαδικασία αυτή φυσικά είναι πολύ χρονοβόρα και θέλει ιδιαίτερη λεπτομέρεια. Για αυτό τον λόγο, συνήθως μία εταιρεία που ειδικεύεται σε τέτοιες υπηρεσίες αναλαμβάνει την διαδικασία. Παρόλα αυτά η χρήση της μεθοδολογίας που θα περιγραφεί παρακάτω μας επέτρεψε την γρήγορη και ακριβή σκιαγράφηση όλων των εικόνων χωρίς κάποιο επιπλέον κόστος.

4.4.2 Η διαδικασία της δημιουργίας των ετικετών εκπαίδευσης με αξιοποίηση του χρωματικού χώρου HSV

Για την επιλογή των πράσινων αμπελιών σε μία φωτογραφία, αρκεί να διαλέξουμε με κάποιον τρόπο ό,τι βρίσκεται κοντά στην απόχρωση του σκούρου πράσινου. Δουλεύοντας στον χρωματικό χώρο του HSV (Hue-απόχρωση Saturation-κορεσμός Value-Τιμή), μπορούμε να πετύχουμε το επιθυμητό αποτέλεσμα. Το πλεονέκτημα του HSV είναι ότι μεταφράζει σε αυτές τις 3 τιμές τον τρόπο με τον οποίο οι άνθρωποι αντιλαμβάνονται το χρώμα, όπως φαίνεται και στο Σχήμα 4-11. Ως εκ τούτου, είναι η πιο ακριβής απεικόνιση του πώς αισθανόμαστε τα χρώματα στην οθόνη του υπολογιστή. Παρακάτω αναλύεται και πως η κάθε τιμή επηρεάζει την τελική επιλογή χρώματος:

- Απόχρωση: Η απόχρωση αντιπροσωπεύει την γωνία του κυλινδρικού δίσκου. Η απόχρωση αντιπροσωπεύει το χρώμα. Η τιμή απόχρωσης κυμαίνεται από 0 έως 360 μοίρες.
- Κορεσμός: Η τιμή κορεσμού μας λέει πόση ποσότητα αντίστοιχου χρώματος πρέπει να προστεθεί. Κορεσμός 100% σημαίνει ότι προστίθεται πλήρες καθαρό χρώμα, ενώ κορεσμός 0% σημαίνει ότι δεν προστίθεται χρώμα, με αποτέλεσμα να έχουμε γκρι χρώμα.
- Τιμή: Η τιμή αντιπροσωπεύει τη φωτεινότητα που αφορά τον κορεσμό του χρώματος. Η τιμή 0 αντιπροσωπεύει το απόλυτο μαύρο σκοτάδι, ενώ η τιμή 100 θα σημαίνει πλήρη φωτεινότητα και εξαρτάται από τον κορεσμό.



Σχήμα 4-11. Αναπαράσταση χρωματικού χώρου HSV.

Για την επιλογή του αμπελιού αρκεί να διαλέξουμε μία απόχρωση του πράσινου (Hue) που θα είναι ουσιαστικά η μέση τιμή του χρώματος των αμπελιών. Για της άλλες δύο τιμές (Saturation, Value) έχουμε ένα επιτρεπτό εύρος επιλογής, ώστε το επιλεγμένο εύρος χρώματος να περιλαμβάνει ολόκληρο το αμπέλι με όσο το δυνατόν λιγότερα άλλα αντικείμενα.

4.4.3 Η χρήση του λογισμικού paint.NET καθώς και του plugin HSV Eraser για την επιλογή των αμπελιών σε ένα στιγμιότυπο

Για την εφαρμογή της διαδικασίας που περιγράψαμε στην ενότητα 4.4.2 θα χρησιμοποιήσουμε το δωρεάν λογισμικό επεξεργασίας εικόνας paint.NET καθώς και το plugin HSV Eraser.

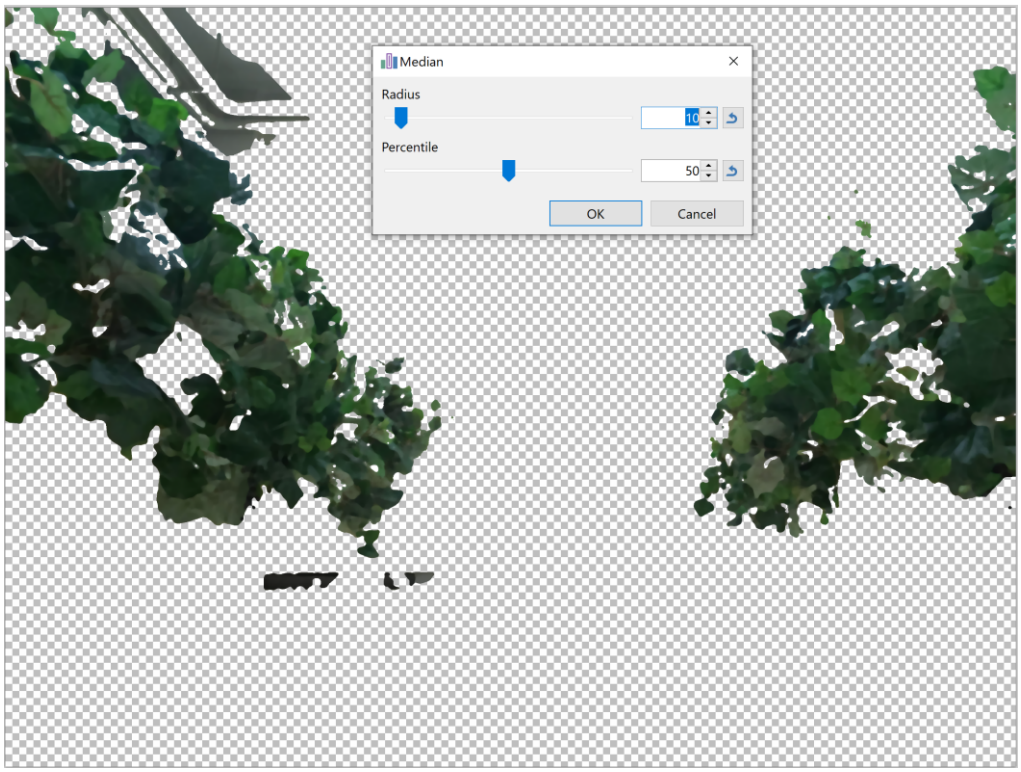
Βήμα 1: Άνοιγμα εικόνας και ορισμός παραμέτρων στο HSV Eraser

Σε κάθε εικόνα διαλέγουμε πολύ πρόχειρα τα τμήματα των αμπελιών με το lasso selection tool και εφαρμόζουμε το HSV Eraser (από το menu Adjustments) με τις τιμές που φαίνονται στο Σχήμα 4-12. Μετά αντιγράφουμε το επεξεργασμένο τμήμα των αμπελιών με Ctrl-C, διαλέγουμε όλη την εικόνα με Ctrl-A, διαγράφουμε τα περιεχόμενά της με Del και επικολλούμε το τμήμα των αμπελιών με Ctrl-V. Πατάμε Esc στο τέλος ώστε να εφαρμοστεί το επόμενο βήμα σε όλη την εικόνα.

Σχήμα 4-12. Γρήγορη επιλογή αμπελιών με lasso tool και εφαρμογή του HSV με τις προκαθορισμένες τιμές.

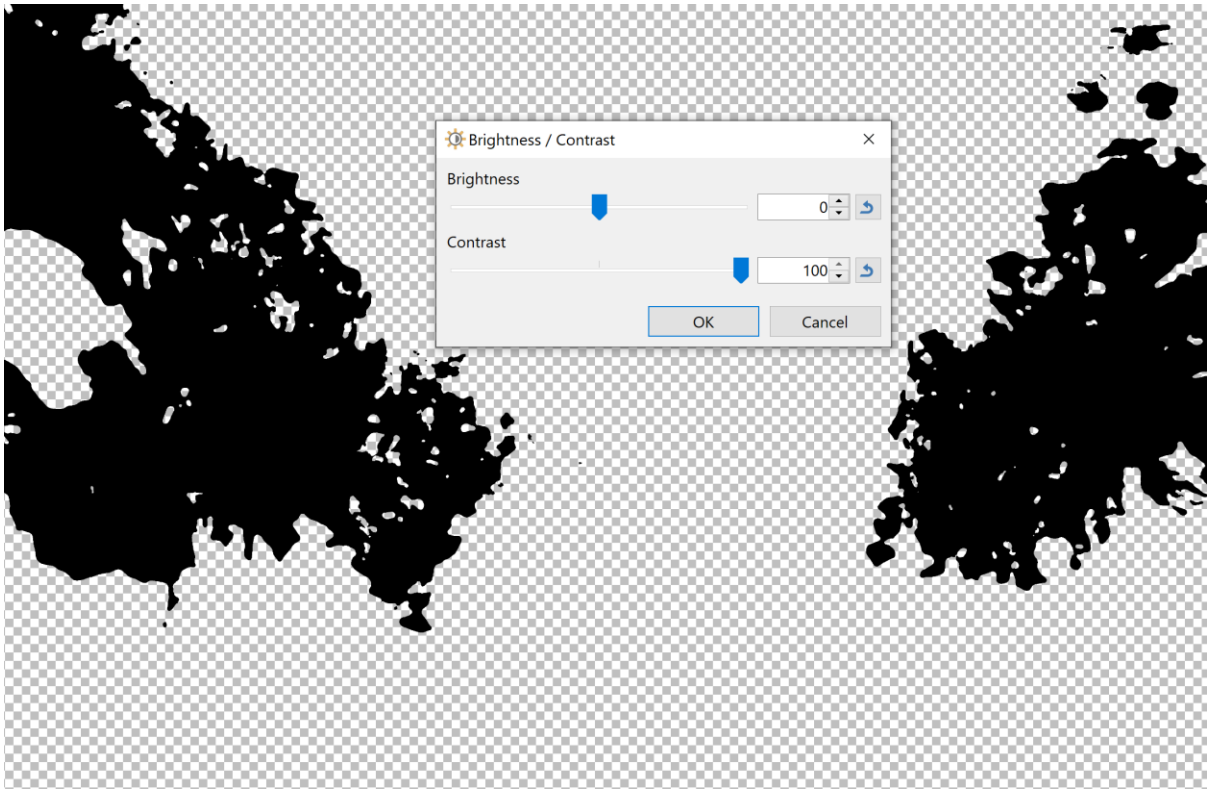
Βήμα 2: Εφαρμογή Median Blur και διαγραφή τμημάτων που έχουν απομείνει και δεν αποτελούν αμπέλια.

Εφαρμόζουμε median blur με τις default τιμές όπως φαίνεται στο Σχήμα 4-13, ώστε να καθαριστούν τυχόν κομμάτια που έχει αφήσει το προηγούμενο βήμα και να γεμιστούν διάφορα μικρά κενά που βρίσκονται στα αμπέλια. Έπειτα με το lasso selection tool διαγράφουμε προσεκτικά όλα τα τμήματα της εικόνας που δεν είναι μέρος του αμπελιού, όπως π.χ. το ταβάνι όπως φαίνεται στο Σχήμα 4-13 πάνω αριστερά.



Σχήμα 4-13. Εφαρμογή Median Blur.

Βήμα 3: Μετατροπή της εικόνας σε δυαδική με εφαρμογή 100% αντίθεσης.
Από το μενού adjustments επιλέγουμε Brightness/Contrast και ορίζουμε το contrast σε 100. Έτσι η εικόνα θα μετατραπεί σε δυαδική όπου τα αμπέλια είναι μαύρα και το υπόλοιπο είναι λευκό, όπως φαίνεται και στο Σχήμα 4-14. Αποθηκεύουμε το τελικό αποτέλεσμα ως JPEG στον φάκελο labels χωρίς να αλλάξουμε όνομα αρχείου. Έτσι έχουμε δημιουργήσει ένα ζευγάρι image/label. Η διαδικασία αυτή για κάθε εικόνα διαρκεί περίπου 2 λεπτά.



Σχήμα 4-14. Τελικό αποτέλεσμα τμηματοποιημένης εικόνας μετά την εφαρμογή αντίθεσης (contrast) 100%.

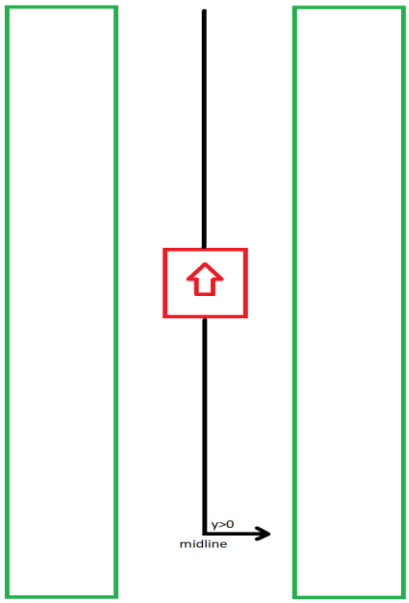
4.5 Χρήση τμηματοποιημένης εικόνας για έλεγχο κατεύθυνσης του ρομπότ

4.5.1 Ανάλυση εξόδου τμηματοποιημένης εικόνας σε διάφορες περιπτώσεις

Στόχος είναι να διατηρήσουμε την πορεία και τον προσανατολισμό του ρομπότ στο κέντρο εκατέρωθεν των δύο σειρών αμπελιών. Μέσω της τμηματοποιημένης εικόνας μπορούμε να εξάγουμε πληροφορία σχετικά με τον προσανατολισμό του ρομπότ, καθώς και την y-θέση του (ως προς τον άξονα που είναι κάθετος στον διάδρομο του αμπελιού) στον χώρο. Θα εξετάσουμε διαφορετικές περιπτώσεις θέσης και προσανατολισμού στον χώρο, καθώς και την αντίστοιχη τμηματοποιημένη εικόνα που προκύπτει σε κάθε περίπτωση.

1η Περίπτωση: Ρομπότ στο κέντρο του διαδρόμου με ουδέτερο προσανατολισμό.
Εδώ το ρομπότ έχει μηδενική απόσταση ως προς y, δηλαδή βρίσκεται στο κέντρο του διαδρόμου, ισαπέχοντας από τα αμπέλια εκατέρωθεν, όπως φαίνεται και στο Σχήμα 4-15. Το ζευγάρι εικόνας εισόδου και η έξοδος της τμηματοποιημένης εικόνας που προκύπτουν για αυτή την περίπτωση φαίνονται στο Σχήμα 4-16. Παρατηρούμε πως στην

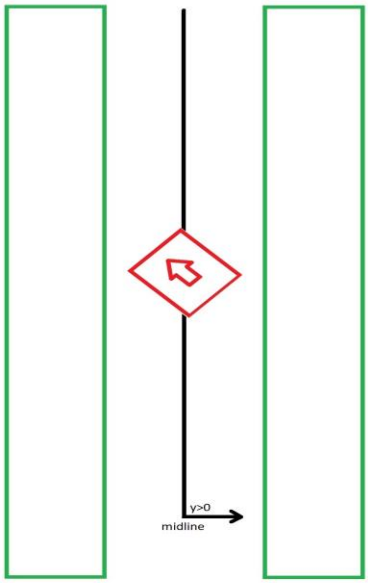
τμηματοποιημένη εικόνα η νοητή μέση γραμμή μεταξύ των αμπελιών ταυτίζεται με την νοητή μέση γραμμή που χωρίζει την εικόνα εξόδου σε δύο ίσα τμήματα.



Σχήμα 4-15. Σχηματική παράσταση του ρομπότ με ισαπέχουσες αποστάσεις από τα αμπέλια και ουδέτερο προσανατολισμό.

Σχήμα 4-16. Στιγμιότυπο της κάμερας του ρομπότ και η αντίστοιχη τμηματοποιημένη εικόνα όταν έχει ισαπέχουσες αποστάσεις από τα αμπέλια και ουδέτερο προσανατολισμό.

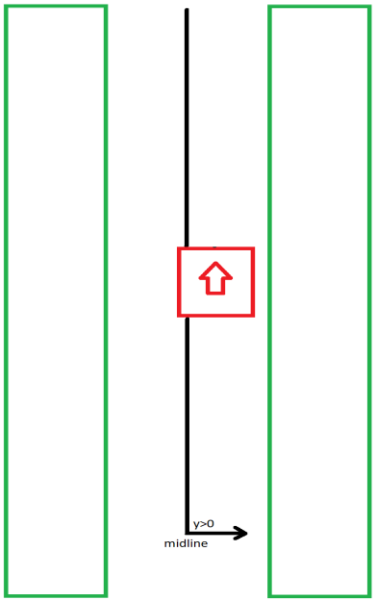
<u>2η Περίπτωση</u>: Ρομπότ στο κέντρο του διαδρόμου περιστραμμένο κατά Φ μοίρες. Εδώ το ρομπότ έχει μηδενική απόσταση ως προς y όπως και στην προηγούμενη περίπτωση, αλλά είναι περιστραμμένο Φ μοίρες γύρω από τον εαυτό του, όπως φαίνεται και στο Σχήμα 4-17. Το ζευγάρι εικόνας εισόδου και η έξοδος της τμηματοποιημένης εικόνας που προκύπτουν για αυτή την περίπτωση φαίνονται στο Σχήμα 4-18. Παρατηρούμε πως παρόλο που το ρομπότ βρίσκεται στο κέντρο του διαδρόμου, λόγω του ότι το ρομπότ «βλέπει» πλάγια, στην τμηματοποιημένη εικόνα η νοητή μέση γραμμή μεταξύ των αμπελιών αποκλίνει από την νοητή μέση γραμμή που χωρίζει την εικόνα εξόδου σε δύο ίσα τμήματα.



Σχήμα 4-17. Σχηματική παράσταση του ρομπότ με ισαπέχουσες αποστάσεις από τα αμπέλια και περιστροφή γύρω από τον εαυτό του.

Σχήμα 4-18. Στιγμιότυπο της κάμερας του ρομπότ και η αντίστοιχη τμηματοποιημένη εικόνα όταν έχει ισαπέχουσες αποστάσεις από τα αμπέλια και περιστροφή γύρω από τον εαυτό του.

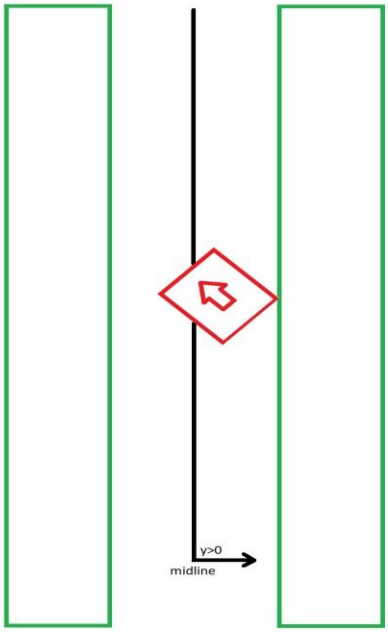
3η Περίπτωση: Ρομπότ μετατοπισμένο από το κέντρο του διαδρόμου κατά Υ με ουδέτερο προσανατολισμό. Εδώ το ρομπότ έχει μετατοπιστεί κατά Υ από την αρχή του άξονα y, με ουδέτερο προσανατολισμό, όπως φαίνεται και στο Σχήμα 4-19. Το ζευγάρι εικόνας εισόδου και η έξοδος της τμηματοποιημένης εικόνας που προκύπτουν για αυτή την περίπτωση φαίνονται στο Σχήμα 4-20. Παρατηρούμε πως παρόλο που το ρομπότ έχει ουδέτερο προσανατολισμό και κοιτάει εντελώς ευθεία, λόγω του ότι το ρομπότ είναι πιο κοντά στο ένα αμπέλι, αυτό φαίνεται μεγαλύτερο στην τμηματοποιημένη εικόνα σε σχέση με το άλλο, οπότε και έχει μετατοπιστεί η νοητή μέση γραμμή μεταξύ των αμπελιών. Έτσι αποκλίνει και σε αυτή την περίπτωση από την νοητή μέση γραμμή που χωρίζει την εικόνα εξόδου σε δύο ίσα τμήματα.



Σχήμα 4-19. Σχηματική παράσταση του ρομπότ με μετατόπιση από το κέντρο του διαδρόμου και ουδέτερο προσανατολισμό.

Σχήμα 4-20. Στιγμιότυπο της κάμερας του ρομπότ και η αντίστοιχη τμηματοποιημένη εικόνα όταν είναι μετατοπισμένο από το κέντρο του διαδρόμου και έχει ουδέτερο προσανατολισμό.

<u>4η Περίπτωση</u>: Ρομπότ μετατοπισμένο από το κέντρο του διαδρόμου κατά Υ περιστραμμένο κατά Φ μοίρες. Εδώ το ρομπότ έχει ταυτόχρονα μετατοπιστεί Υ από την αρχή του άξονα y, αλλά είναι κιόλας περιστραμμένο Φ μοίρες γύρω από τον εαυτό του, όπως φαίνεται και στο Σχήμα 4-21. Παρατηρούμε πως ο συνδυασμός της μετατόπισης και της περιστροφής του έχει μετατοπιστεί ακόμα περισσότερο την νοητή μέση γραμμή μεταξύ των αμπελιών. Έτσι αποκλίνει ακόμα παραπάνω από την νοητή μέση γραμμή που χωρίζει την εικόνα εξόδου σε δύο ίσα τμήματα.



Σχήμα 4-21. Σχηματική παράσταση του ρομπότ με μετατόπιση από το κέντρο του διαδρόμου και περιστροφή γύρω από τον εαυτό του.

4.5.2 Συμπεράσματα σχετικά με το νόημα της θέσης της νοητής γραμμής στην τμηματοποιημένη εικόνα και χρήση της ως τιμή σφάλματος

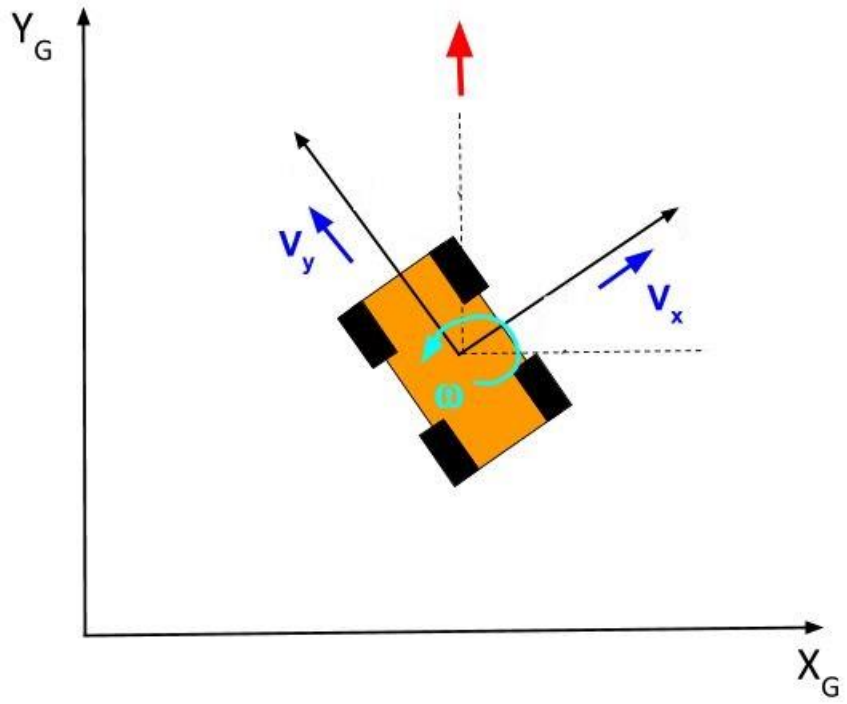
Όπως παρατηρούμε η απόκλιση της νοητής μέσης γραμμής από το κέντρο της τμηματοποιημένης εικόνας επηρεάζεται ανεξαρτήτως τι είδους κίνηση πραγματοποιεί το ρομπότ. Ουσιαστικά περιέχει ταυτόχρονα πληροφορία για την απόκλιση του ρομπότ από το κέντρο του άξονα y καθώς και την απόκλιση προσανατολισμού του. Μπορούμε λοιπόν να πραγματοποιήσουμε έλεγχο της κίνησης του ρομπότ με proportional controller θεωρώντας ως σφάλμα την διαφορά της θέσης της νοητής της μέσης γραμμής στην τμηματοποιημένη εικόνα από το μέτρο του ημι-πλάτους της εικόνας, το οποίο φυσικά είναι σταθερό. Διαιρώντας αυτή την ποσότητα με το μέτρο του ημι-πλάτους της εικόνας έχουμε πλέον μία τιμή ποσοστιαίου σφάλματος. Έχουμε δηλαδή:

$$error = \frac{DetectedMidlinePosition - 0.5ImageWidth}{0.5ImageWidth} \quad (4\text{-}1)$$

Αν το ποσοστό αυτό έχει θετικό/αρνητικό πρόσημο και πάνω από ένα κατώφλι ευαισθησίας που έχουμε ορίσει σημαίνει ότι το ρομπότ έχει μετατοπιστεί ή/και είναι προσανατολισμένο προς το δεξί/αριστερό αμπέλι αντίστοιχα. Στην περίπτωση αυτή πρέπει να επενεργηθούν οι κινητήρες ώστε το ρομπότ να κινηθεί προς την αντίθετη κατεύθυνση. Αντιθέτως αν η απόλυτη τιμή του ποσοστού αυτού βρίσκεται εντός του κατωφλίου σημαίνει ότι ρομπότ κοιτάει σχεδόν ή και τελείως ευθεία, άρα δεν χρειάζεται επέμβαση στην πορεία του και αυτή διατηρείται ευθεία. Στην επόμενη παράγραφο θα αναλύσουμε πώς χρησιμοποιούμε την τιμή του ποσοστιαίου σφάλματος για τον έλεγχο της πορείας του ρομπότ πειραματικής διάταξης στο εργαστήριο.

4.5.3 Έλεγχος πορείας

Όπως περιγράφηκε και στην ενότητα 4.2.3 για τις πειραματικές δοκιμές χρησιμοποιήθηκε ένα ρομπότ με ρόδες mecanum. Το συγκεκριμένο ρομπότ ελέγχεται μέσω πακέτων UDP και έχει την ικανότητα μετατόπισης τόσο ευθεία κατά μήκος του άξονα x, όσο και κάθετα κατά μήκος του άξονα y, ενώ μπορεί να περιστραφεί κιόλας γύρω από τον εαυτό του, όπως φαίνεται και στο Σχήμα 4-22. Κάθε εντολή που αποστέλλεται λοιπόν αποτελείται από 3 ορίσματα ταχύτητας, τα οποία αντιστοιχούν στην γραμμική ταχύτητα v_x του ρομπότ κατά μήκος του άξονα x, την γραμμική ταχύτητα v_y του ρομπότ κατά μήκος του άξονα y, καθώς και την γωνιακή ταχύτητα ω γύρω από τον εαυτό του. Μόλις σταλεί μία εντολή το ρομπότ διατηρεί σταθερές τις δοσμένες ταχύτητες, μέχρι να δοθεί εντολή με διαφορετικό διάνυσμα ταχυτήτων ή να δοθεί εντολή διακοπής κίνησης.



Σχήμα 4-22. Παράσταση των συνιστωσών γραμμικής και γωνιακής ταχύτητας με τις οποίες μπορεί να κινηθεί το ρομπότ.

Όπως περιγράψαμε στην ενότητα 4.5.2, το ποσοστιαίο σφάλμα που επιστρέφει ο αλγόριθμος εύρεσης της μέσης γραμμής μπορεί να οφείλεται στην απόκλιση του ρομπότ από το κέντρο του άξονα ή/και την απόκλιση προσανατολισμού του. Έτσι για την διόρθωση αυτού του σφάλματος πρέπει να εκτελούμε αντίστοιχα μία κίνηση που μπορεί να το διορθώσει ανεξαρτήτως από το ποια περίπτωση το προκάλεσε. Με τον συνδυασμό μόνο εμπρόσθιας γραμμικής ταχύτητας v_x καθώς και γωνιακής ταχύτητας ω διορθώνεται αυτομάτως το σφάλμα ανεξαρτήτως από το ποια περίπτωση το έχει προκαλέσει. Εκτελούμε λοιπόν proportional control με κέρδος K_p στην γωνιακή ταχύτητα ω του ρομπότ με σφάλμα το ποσοστιαίο σφάλμα $error$, διατηρώντας σταθερή γραμμική εμπρόσθια ταχύτητα μετατόπισης x. Παρόλα αυτά για την αποφυγή ταλαντώσεων και επειδή στην παρούσα εφαρμογή δεν είναι επιτακτική ανάγκη το ρομπότ να ακολουθεί πολύ αυστηρά το κέντρο του αμπελιού, το proportional control εκτελείται αν το σφάλμα υπερβαίνει ένα κατώφλι

steer_sensitivity. Στις δοκιμές που κάναμε είχε οριστεί 5%. Έτσι αν η σταθερά ταχύτητας που έχει οριστεί είναι V_{const} θα ισχύουν για τα 3 διανύσματα ταχύτητας του ρομπότ τα εξής

$$V_y = 0$$
$$V_x = V_{const} \tag{4-2}$$
$$\omega = \begin{cases} 0, |error| < \text{steer\_sensitivity} \\ K_p error V_{const}, |error| \geq \text{steer\_sensitivity} \end{cases}$$

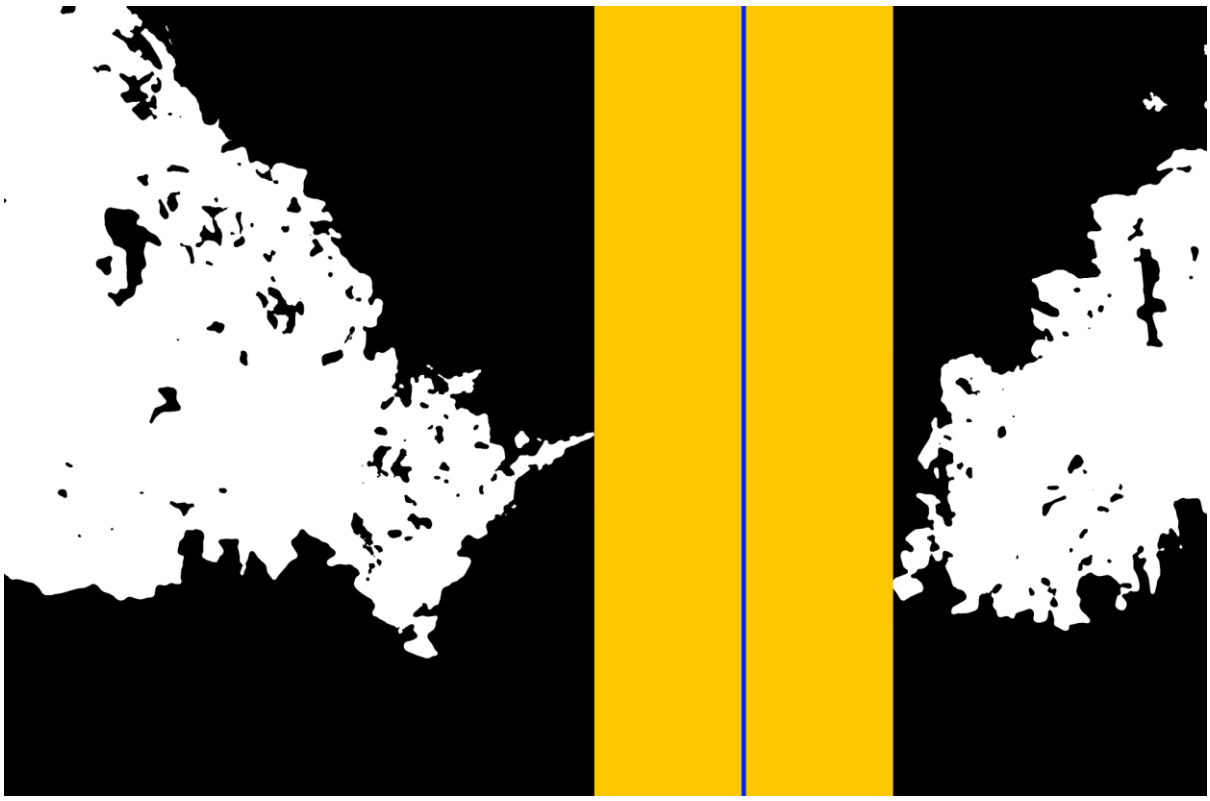
4.5.4 Συνάρτηση υπολογισμού μέσης γραμμής από την τμηματοποιημένη εικόνα

Για την εύρεση της μέσης γραμμής ανάμεσα στα αμπέλια θα θεωρήσουμε ότι η τμηματοποιημένη εικόνα αποτελείται από ένα πλήθος στηλών ίσο με το πλάτος (σε pixel) της εικόνας. Έτσι κάθε στήλη θα έχει μέγεθος ίσο με το ύψος (σε pixel) της εικόνας. Επειδή η εικόνα είναι τμηματοποιημένη με 2 κατηγορίες μόνο θα αποτελείται μόνο από άσπρα και μαύρα pixel. Θα θεωρήσουμε πως τα μαύρα pixel (υπόλοιπος χώρος) αντιστοιχούν σε 0 και τα άσπρα pixel (αμπέλια) σε 1, όπως στο Σχήμα 4-23. Έτσι και κάθε στήλη θα αποτελείται από 0 και 1. Μπορούμε λοιπόν να αθροίσουμε τα 1 σε κάθε στήλη και έτσι να εξάγουμε πληροφορία που θα μας οδηγήσει στην εύρεση της μέσης γραμμής.

Σχήμα 4-23. Η τμηματοποιημένη εικόνα που θα εφαρμόσουμε την συνάρτηση υπολογισμού μέσης γραμμής.

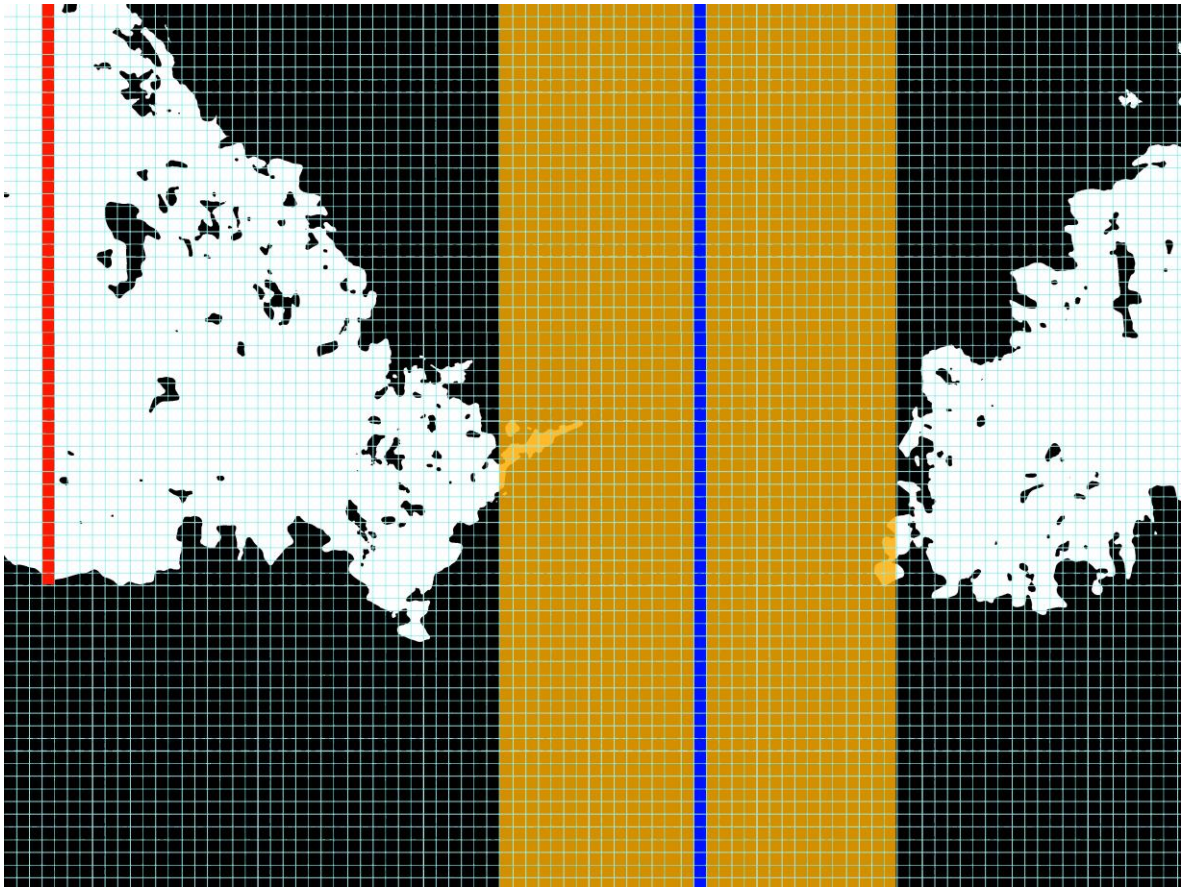
Για την εύρεση του κενού μπορούμε απλώς να διαλέξουμε όλες της στήλες με μηδενικό άθροισμα των «1». Παρόλα αυτά η προσέγγιση αυτή έχει ένα σημαντικό πρόβλημα. Πολλές φορές λεπτά φύλλα προεξέχουν από το αμπέλι τα όποια φυσικά φαίνονται ως άσπρα στην

τμηματοποιημένη εικόνα. Έτσι το διάστημα του «κενού» αναγνωρίζεται λανθασμένα πολύ μικρότερο όπως φαίνεται στο Σχήμα 4-24.



Σχήμα 4-24. Παράδειγμα λανθασμένης λειτουργίας συνάρτησης υπολογισμού μέσης γραμμής σε μία τμηματοποιημένη εικόνα.

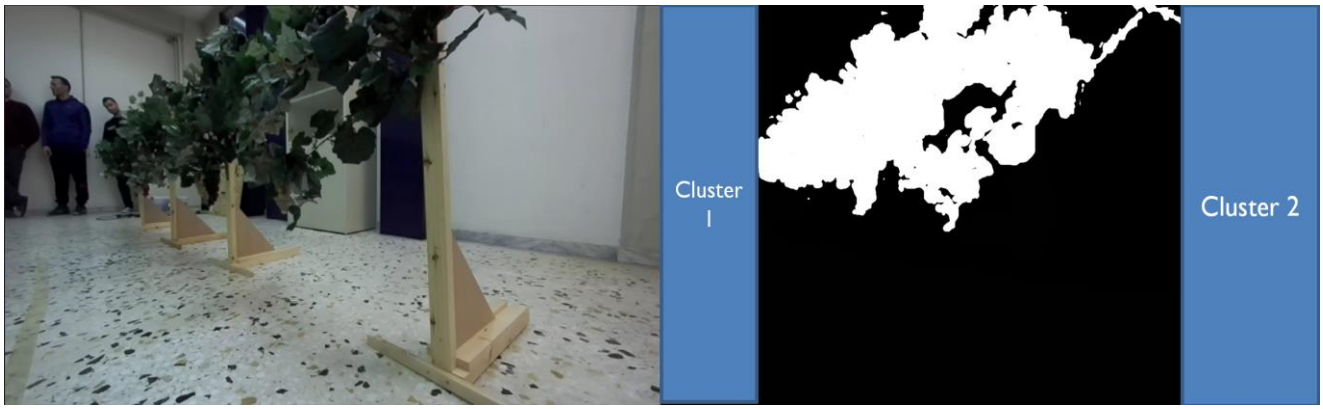
Η λύση σε αυτό το πρόβλημα δίνεται από την εξής διαδικασία και οι λειτουργίες της απεικονίζονται με χρώματα στο Σχήμα 4-25: Πιο συγκεκριμένα βρίσκουμε την στήλη που αποτελεί το πιο «μεγάλο» σημείο του αμπελιού (σημαδεμένη με κόκκινο στο Σχήμα 4-25), αθροίζοντας διαδοχικά τα 1 για την κάθε στήλη και διαλέγοντας εκείνη με το μεγαλύτερο άθροισμα. Στην συνέχεια θεωρούμε «κενό» ανάμεσα στα δύο αμπέλια, οποιαδήποτε στήλη της οποίας το άθροισμα των 1 είναι κάτω από ένα κατώφλι το οποίο ισούται με ένα ποσοστό του μεγαλύτερου αθροίσματος που υπολογίσαμε προηγουμένως (στα πειράματα που πραγματοποιήσαμε το ποσοστό αυτό ορίστηκε 10%). Στο Σχήμα 4-25 είναι όλες οι κίτρινες στήλες. Παρατηρούμε πως τα φύλλα που εξέχουν στο αριστερό αμπέλι δεν εξαιρέθηκαν όπως στην προηγούμενη εικόνα. Έτσι για τυχόν φύλλα που δημιουργούν μικρό πλήθος από «1» στην εκάστοτε στήλη, το πλήθος των «1» σε αυτή την στήλη είναι κάτω από το κατώφλι που ορίστηκε και έτσι αυτομάτως η συγκεκριμένη στήλη ταξινομείται ορθά ως «κενό». Τέλος για την εύρεση της μέσης γραμμής από το πλήθος των στηλών που έχουν προστεθεί διαδοχικά σε μία λίστα, διαλέγουμε απλώς την median τιμή της λίστας αυτής. Στο Σχήμα 4-25 η στήλη αυτή έχει υπογραμμιστεί με μπλε.



Σχήμα 4-25. Παράδειγμα σωστής λειτουργίας συνάρτησης υπολογισμού μέσης γραμμής σε μία τμηματοποιημένη εικόνα.

Θα πρέπει να εξεταστεί φυσικά και η εδική περίπτωση που έχουμε πολλαπλά clusters. Σε περίπτωση που έχουμε πολλαπλά clusters από κενά, διαλέγουμε εκείνο που είναι πιο κόντα με αυτό της προηγούμενης εκτέλεσης του κώδικα.Για παράδειγμα στην παραπάνω εικόνα, αν υποθέσουμε πως στο προηγούμενο το ρομπότ είχε ουδέτερο προσανατολισμό πριν στρίψει δεξιά, διαλέγουμε το cluster 1.

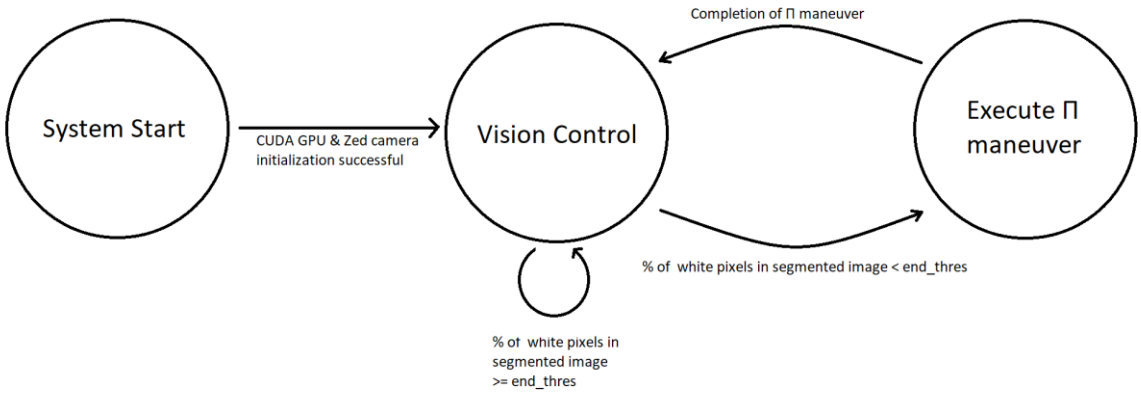
Αν δεν υπάρχει προηγούμενη εκτέλεση, όπως π.χ. στην περίπτωση που ξεκινάει το ρομπότ να εισέρχεται στο αμπέλι διαλέγουμε το κεντρικό cluster



Σχήμα 4-26. Παράδειγμα ειδικής περίπτωσης που έχουμε πολλαπλά cluster.

4.5.5 Γενικός αλγόριθμος κίνησης του ρομπότ

Ο κώδικάς του ρομπότ λειτουργεί σύμφωνα με το state machine που απεικονίζεται στο Σχήμα 4-27. Αρχικά αμέσως μετά την ενεργοποίηση του ρομπότ, ελέγχεται αν υπάρχει κάρτα γραφικών CUDA συμβατή με την έκδοση της PyTorch που χρησιμοποιείται καθώς και έχει συνδεθεί η Zed Camera και επικοινωνεί σωστά. Εφόσον αυτοί οι έλεγχοι ολοκληρωθούν επιτυχώς το πρόγραμμα εισέρχεται στο κύριο βρόχο ελέγχου. Κάθε βήμα σε αυτό τον βρόχο ξεκινάει με την καταγραφή μία εικόνας από την κάμερα και την είσοδό της στο μοντέλο μηχανικής μάθησης. Αν το ποσοστό του πλήθους των άσπρων pixel προς των συνολικό αριθμό pixel στην εικόνα αυτή είναι μεγαλύτερο από την παράμετρο **end_thres**, το ρομπότ εκτελεί την συνάρτηση υπολογισμού μέσης γραμμής και πραγματοποιεί έλεγχο της κατεύθυνσης. Αντιθέτως αν το ποσοστό αυτό βρίσκεται κάτω από το end_thres ξεκινά την διαδικασία αλλαγής σειράς, όπως περιγράφεται στην ενότητα 4.5.6. Μόλις ολοκληρωθεί η διαδικασία αυτή γίνεται ξανά επιστροφή στον κύριο βρόχο ελέγχου. Το πλήθος των άσπρων pixels επιλέχθηκε σαν κριτήριο τερματισμού της σειράς του αμπελιού καθώς όταν το ρομπότ φτάνει στο τέλος είναι το μοναδικό σημείο όπου η κάμερα αρχίζει να μην βλέπει αμπέλια οπότε και το πλήθος των άσπρων pixel στην τμηματοποιημένη εικόνα θα είναι πολύ μικρό. Στις δοκιμές που έγιναν στο εργαστήριο το ποσοστό end_thres ορίστηκε πειραματικά **5%**.



Σχήμα 4-27. Σχηματική παράσταση του state machine που εκτελεί το ρομπότ.

4.5.6 Η διαδικασία αλλαγής σειράς αμπελιού

Για την αλλαγή της σειράς το ρομπότ εκτελεί μία κίνηση σχήματος Π. Ουσιαστικά μόλις αναγνωρίσει ότι έφτασε στο τέλος της σειράς του αμπελιού, ξεκινά την διαδικασία της κίνησης αυτής η οποία είναι προκαθορισμένη. Μόλις την τελειώσει επιστρέφει στην διαδικασία ελέγχου μέσω της τμηματοποιημένης εικόνας όπως περιγράφεται και στην ενότητα 4.5.5. Η κίνηση αυτή αποτελείται από τέσσερις διαδοχικές μετατοπίσεις και περιστροφές. Για την ακριβή εκτέλεση των περιστροφών χρησιμοποιείται το επιταχυνσιόμετρο της Zed. Πιο συγκεκριμένα, το ρομπότ περιστρέφεται γύρω από τον εαυτό του μέχρι η απόλυτη διαφορά προσανατολισμού γύρω από τον άξονα z που επιστρέφει η μονάδα IMU να ισούται με 90^{o} μοίρες.

5 Συμπεράσματα και Μελλοντική Εργασία

5.1 Συμπεράσματα

5.1.1 Συμπεράσματα πρώτου μέρους εργασίας: Ανάπτυξη συστήματος ελέγχου κίνησης βασισμένο σε EtherCAT

Στόχος του πρώτου μέρους της παρούσας διπλωματικής εργασίας ήταν ο σχεδιασμός και η υλοποίηση του νέου ηλεκτρονικού συστήματος του τετράποδου ρομπότ Laelaps.

Αρχικά εφαρμόστηκαν επιτυχώς νέες, για το εργαστήριο μας, τεχνικές κόλλησης. Αυτό μας επέτρεψε την διασύνδεση του πιο εξελιγμένου μικροελεγκτή F28388 με ενσωματωμένη μονάδα Ethercat, απλοποιώντας την δομή του ηλεκτρονικού συστήματος και αυξάνοντας την αξιοπιστία του.

Όσον αφορά τον ελεγκτή κινητήρων, η χρήση του νέου και ισχυρότερου AZBDC20A8 μειώνει σημαντικά την πιθανότητα υπερθέρμανσης, ενώ προστέθηκε και η λειτουργία καταγράφης ρεύματος για την αντιμετώπιση προβλημάτων στο μέλλον.

Εν κατακλείδι έγιναν στοχευμένες παρεμβάσεις στο ήδη αρκετά αξιόπιστο ηλεκτρονικό σύστημα του Laelaps, με απώτερο στόχο τα επίπεδα απόδοσης και αξιοπιστίας που απαιτεί ένα σύγχρόνο ρομπότ.

5.1.2 Συμπεράσματα δεύτερου μέρους εργασίας: Αλγόριθμος ελέγχου κατεύθυνσης μέσω τμηματοποίησης εικόνας για τετράποδα ρομπότ

Στόχος του δεύτερου μέρους της παρούσας διπλωματικής εργασίας ήταν ο σχεδιασμός και η υλοποίηση ενός αυτόνομου συστήματος πλοήγησης για γεωργικά ρομπότ με την βοήθεια νευρωνικών δικτύων βαθιάς μάθησης.

Αρχικά μελετήθηκε η διαδικασία σχεδιασμού και εκπαίδευσης ενός μοντέλου βαθιάς μάθησης. Η απόδοση του μοντέλου DeepLabV3, με τα προεκπαιδευμένα βάρη, πάνω στις εικόνες του εργαστηρίου ήταν εντυπωσιακή και μας έδωσε μία εικόνα των ισχυρών δυνατοτήτων και τις ευελιξίας των σύγχρονων μοντέλων μηχανικής μάθησης. Μάλιστα, με ένα πολύ μικρό σύνολο δεδομένων εικόνας, και με την βοήθεια απλών τεχνικών επαύξησης, το μοντέλο πέτυχε εξαιρετική ακρίβεια στον χώρο του εργαστηρίου, ώστε να μας επιτρέψει να πραγματοποιήσουμε επιτυχώς έλεγχο του ρομπότ με βάση την έξοδο του.

Η χρήση της τεχνικής τμηματοποίησης εικόνας αποδείχτηκε ιδαίτερα αποδοτική, καθώς η αξιοποίηση της για τον έλεγχο κατεύθυνσης του ρομπότ ήταν πολύ αποτελεσματική, χωρίς να απαιτούνται πολύπλοκες τεχνικές για την εξαγωγή της πληροφορίας.

Αναμφίβολα η ταχύτητα και η απόδοση της υλοποίησης της παρούσας εργασίας οφείλεται και στη γλώσσα Python. Η απλότητα της σύνταξής της, καθώς και η ύπαρξη της μαθηματικής βιβλιοθήκης numpy και η άψογη συνεργασίας της με την βιβλιοθήκη μηχανικής μάθησης PyTorch μας επέτρεψε αρχικά να εφαρμόσουμε εύκολα τους απαιτούμενους μετασχηματισμούς για τις εικόνες εισόδου και εξόδου στο μοντέλο. Επιπρόσθετα οι δυνατότητες πράξεων σε πίνακες της βιβλιοθήκης αυτής μας επέτρεψαν εύκολα να εξάγουμε πολύτιμη πληροφορία από την τμηματοποιημένη εικόνα για να πραγματοποιήσουμε επιτυχώς έλεγχο της κατεύθυνσης του ρομπότ.

Εν κατακλείδι, η ραγδαία τεχνολογική πρόοδος στον τομέα της μηχανικής μάθησης και ιδιαιτέρως στην όραση υπολογιστών, την καθιστά ένα πολύ ελκυστικό εργαλείο ιδιαίτερα σε

εφαρμογές αυτόνομης κίνησης ρομπότ. Όσον αφορά την πλοήγηση γεωργικών ρομπότ, στο παρελθόν, για την αυτόνομη κίνηση τους απαιτούνταν η χρήση πολύ ακριβών αισθητήρων για την επίτευξη παρόμοιας απόδοσης. Η χρήση μηχανικής μάθησης, είναι μία πολύ προσιτή και αποδοτική επιλογή που δεν απαιτεί χρήση εξωτικών αισθητήρων, ενώ παράλληλα είναι ιδιαίτερα προσαρμόσιμη, χωρίς να απαιτείται χρήση ριζικά διαφορετικού υλικού αναλόγως την εφαρμογή.

5.2 Μελλοντική Εργασία

5.2.1 Μελλοντική εργασία πρώτου μέρους εργασίας: Ανάπτυξη συστήματος ελέγχου κίνησης βασισμένο σε EtherCAT

Στο πρώτο μέρος της εργασίας έγινε ο σχεδιασμός και η υλοποίηση του νέου ηλεκτρονικού συστήματος του τετράποδου ρομπότ, από την σκοπιά κυρίως του hardware. Φυσικά έγιναν μεγάλα βήματα για την μεταφορά και του software στο καινούριο ηλεκτρονικό σύστημα, εντούτοις πρέπει να γίνουν αρκετά ακόμη για να έχουμε ένα ολοκληρωμένο αποτέλεσμα.

Όσον αφορά το firmware του MCU, πρέπει να υλοποιηθεί η χρήση του CLA και να δοκιμαστούν οι επιπρόσθετες λειτουργίες Auto Startup και End-Stop όπως περιγράφονται στην εργασία του Παπαθεοδώρου [15] . Ακόμα, πρέπει να ενσωματωθούν οι μετρήσεις του current monitoring, εκτελώντας την λειτουργία αυτή παράλληλα με την εκτέλεση του ελέγχου του ρομπότ και αποστέλοντας τις μετρήσεις μέσω μηνυμάτων Ethercat. Τέλος, πρέπει να δοθεί και έμφαση στη διασφάλιση της ορθής λειτουργίας του λογισμικού, με την υιοθέτηση αυστηρών προτύπων ανάπτυξης κώδικα, όπως το MISRA.

5.2.2 Μελλοντική εργασία δεύτερου μέρους εργασίας: Αλγόριθμος ελέγχου κατεύθυνσης μέσω τμηματοποίησης εικόνας για τετράποδα ρομπότ

Στο δεύτερο μέρος της παρούσας εργασία έγιναν τα αρχικά βήματα για την ανάπτυξη ενός αυτόνομου συστήματος πλοήγησης για γεωργικά ρομπότ με την βοήθεια νευρωνικών δικτύων βαθιάς μάθησης, με πολύ ενθαρρυντικά αποτελέσματα. Παρόλα αυτά πρέπει γίνουν αρκετά βήματα ακόμη για την αξιοποίηση του συστήματος σε πραγματικές συνθήκες.

Αρχικά, θα πρέπει να χρησιμοποιηθεί ένας ειδικευμένος υπολογιστής για την εκτέλεση του μοντέλου μηχανικής μάθησης πάνω στην ρομποτική πλατφόρμα. Αυτό όχι μόνο θα επιταχύνει την ταχύτητα του συστήματος, αλλά και θα μειώσει την συνολική κατανάλωση ισχύος, βελτιώνοντας την αυτονομία. Παράδειγμα τέτοιου συστήματος αποτελεί ο υπολογιστής Jetson της NVidia.

Έπειτα πρέπει να γίνουν δοκιμές σε πραγματικά αμπέλια. Η εκπαίδευση του μοντέλου μάθησης πρέπει να γίνει πάνω σε αυτά. Ακόμα το τραχύ έδαφος που έχει το χώμα μπορεί να φέρει νέες δυσκολίες στον έλεγχο κατεύθυνσης. Επιπρόσθετα λόγω της μεγάλης έκτασης των αμπελιών η διαδικασία αλλαγής σειρών θα πρέπει αναμφίβολα να συνοδευτεί με κάποιο σύστημα GPS.

Φυσικά θα πρέπει να γίνει και κάποια υλοποίηση για τις ακριανές σειρές των αμπελιών, όπου δεν έχουμε εκατέρωθεν αμπέλια και η διαδικασία εξαγωγής της μέσης γραμμής θα είναι κάπως διαφορετική.

Ακόμα, η ακρίβεια του μοντέλου μάθησης μπορεί να βελτιωθεί και με χρήση δεδομένων από εικόνα βάθους. Η συγκεκριμένη τεχνική είναι κάτι που έχει εφαρμοστεί από πολλούς [2] , [4] σε παρόμοιες εφαρμογές και αξίζει να εξερευνηθεί.

Τέλος, μακροπρόθεσμος στόχος του εργαστηρίου μας είναι η ανάπτυξη ενός τετράποδου ρομπότ για την κίνηση στο αμπέλι. Φυσικά η επιλογή της παρούσας ρομποτικής πλατφόρμας η οποία επιτρέπει κίνηση σε κάθε κατεύθυνση, είναι ό,τι πιο κοντά υπάρχει στις δυνατότητες κίνησης ενός τετράποδου ρομπότ. Παρόλα αυτά είναι κάτι που απαιτεί ιδιαίτερη μελέτη. Ένα σημαντικό πρώτο βήμα, που θα συμβάλλει ιδιαίτερα στην μακροπρόθεσμη ανάπτυξη τους συστήματος, είναι η ενσωμάτωσή του σε περιβάλλον ROS. Το ROS θα μας επιτρέψει εύκολα να χρησιμοποιήσουμε το σύστημα ελέγχου μέσω της τμηματοποιημένης εικόνας σε ρομπότ με διαφορετικές αρχιτεκτονικές, χωρίς να χρειάζεται να τροποποιήσουμε τον βασικό αλγόριθμο. Επίσης θα επιταχύνει ιδιαίτερα την ανάπτυξη του ρομπότ καθώς διαθέτει εξελιγμένες λειτουργίες επικοινωνίας και καταγραφής δεδομένων.

6 Βιβλιογραφία

[1] Pytorch segmentation models
https://github.com/qubvel/segmentation_models.pytorch

[2] Diego Aghi, Simone Cerrato, Vittorio Mazzia, Marcello Chiaberge ”Deep Semantic Segmentation at the Edge for Autonomous Navigation in Vineyard Rows”, *IROS, 2021*, https://arxiv.org/abs/2107.00700

[3] Xia Li, Junhao Su, Zhenchao Yue, Fangtao Duan, “Adaptive Multi-ROI Agricultural Robot Navigation Line Extraction Based on Image Semantic Segmentation”, *MDPI Sensors*, Oct 2022,
https://www.mdpi.com/1424-8220/22/20/7707

[4] Yan Song, Feiyang Xu, Qi Yao, Jialin Liu, Shuai Yang, “Navigation algorithm based on semantic segmentation in wheat fields using an RGB-D camera”, *Information Processing in Agriculture*, May 2022
https://www.sciencedirect.com/science/article/pii/S2214317322000488

[5] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, Qiang Yang, “Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN”, *WWW '18: Proceedings of the 2018 World Wide Web Conference*, April 2018, Pages 1063–1072,
https://doi.org/10.1145/3178876.3186005

[6] Bolonakis, G., “Electronic system design and manufacturing of quadruped robot”, NTUA diploma thesis, Athens, 2018

[7] A Comprehensive Guide to Convolutional Neural Networks,
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[8] Ganyu Deng, Jianwen Luo, Caiming Sun, Dongwei Pan, Longyao Peng, Ning Ding, Aidong Zhang, “Vision-based Navigation for a Small-scale Quadruped Robot Pegasus-Mini”, *IEEE ROBIO*, 2021
https://arxiv.org/pdf/2110.04426.pdf

[9] Z. HongChao, G. HaiBo, D. ZongQuan, D. Liang and L. Zhen, "A review of heavy-duty legged robots", Science China Technological Sciences, 2013.

[10] Claudio Semini, Angelo Bratta, Michele Focchi, Matteo Gatti, Stefano Poni, Victor Barasuol, “First Field Tests of a Legged Robot in a Vineyard”, *Italian Conference on Robotics and Intelligent Machines*, 2021

[11] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, Hartwig Adam, “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”, *ECCV* 2018
https://arxiv.org/pdf/1802.02611v3.pdf

[12] NumPy Python Library, https://numpy.org/about/

[13] OpenCV Library, https://opencv.org/about/

[14] Εργαστήριο αυτόματου Ελέγχου CSL-EP http://csl-ep.mech.ntua.gr/

[15] Papatheodorou A., “Design & implementation of a real-time distributed EtherCAT-based motion control system for a multi-DoF quadruped robot”,
Diploma Thesis, School of Mechanical Engineering, NTUA, Athens, 2021

[16] Stamatis Athiniotis, “Firmware design for microcontrollers on EtherCAT network for quadruped robot motion control,” *Diploma Thesis*, School of Mechanical Engineering, NTUA, Athens, 2018.

[17] Κ. Δ. Ασημακόπουλος, «Σχεδιασμός και Κατασκευή Ηλεκτρικών και Ηλεκτρονικών Υποσυστημάτων Τετράποδου Ρομπότ», *Μεταπτυχιακή Εργασία ΕΜΠ*, Αθήνα, 2017.

[18] M. Karamousadakis, “Real-time programming of EtherCAT master in ROS for a quadruped robot,” *Diploma Thesis*, National Technical University of Athens, Athens, 2019.

[19] “Boston Dynamics Atlas,” Boston Dynamics. https://www.bostondynamics.com/atlas.
[20] “Boston Dynamics Spot,” Boston Dynamics. https://www.bostondynamics.com/spot.
[21] “Boston Dynamics Legacy Robots,” Boston Dynamics. https://www.bostondynamics.com/legacy.
[22] “ANYbotics ANYmal C,” ANYbotics. https://www.anybotics.com/anymal-legged-robot/.
[23] “Biomimetic Robotics Lab,” MIT MECHE. https://biomimetics.mit.edu.
[24] “Agility Robotics DIGIT,” Agility Robotics. https://www.agilityrobotics.com/digit
[25] Review: DeepLabv3 — Atrous Convolution (Semantic Segmentation), https://towardsdatascience.com/review-deeplabv3-atrous-convolution-semantic-segmentation-6d818bfd1d74

7 Datasheets

[26] AZBDC20A8, Analog Servo Drive
[27] Texas Instruments F28388D controlCARD, Realtime MCU, Information Guide
[28] HEDS-5540, Absolute Encoder
[29] SN75175, Quadruple Line Receiver
[30] RMF44, Angular magnetic encoder module
[31] 2N7002, N-Channel Enhancement Mode Mosfet
[32] 6N137, VO2630, High Speed Optocoupler, Single and Dual, 10 MBd.
[33] LM7805, 5V Voltage Regulator
[34] MAX5035, High-Efficiency Switching regulator
[35] MECHANIC NS38 138C 40g Lead-free High TEMP Solder Paste

8 Παράρτημα Α: Κατάλογος Εξαρτημάτων του Laelaps III

Πίνακας 8-1. Λίστα εξαρτημάτων για την πλακέτα επέκτασης F28388D του Laelaps III.

DESCRIPTION	QUANTITY	Designator	Footprint
HSEC 120 pin	1	J1	
HSEC 60 pin	1	J2	
RJ-45 Connector Top mount	2	J3,J4	
RJ-9 Connector Top mount	2	J6,J7	
SN75175D IC	3	U1,U2,U3	SOIC-16
120Ohm Resistor	9	R1,R2,R3,R4,R5,R6,R7,R8,R9	R0805
4.7uF capacitor	3	C1,C3,C5	C0603
0.1uF capacitor	3	C2,C4,C6	C0603
3.3k resitor	4	RD3,RD4,RD12,RD22	R0805
1.8k resitor	4	RD1,RD2,RD21,RD11	R0805
Molex 3 pin power connector	1	J5	
2x5 female connector	3	SV1,SV2,SV3	
LM7805 5V LDO Regulator	1	IC1	

Πίνακας 8-2. Λίστα εξαρτημάτων για την πλακέτα του AZBDC20A8 motor controller του Laelaps III.

DESCRIPTION	QUANTITY	Designator	Footprint
2N7002 N-Channel MOSFET	4	Q11,Q12,Q21,Q22	SOT23
1KΩ Resistor	2	R14,R24	R0603
220Ω Resistor	5	R1,R2,R3,R4,R5	R0603
100KΩ Resistor	2	R11,R21	R0603
10KΩ Resistor	1	R3	R0603
330Ω Resistor	2	R15,R25	R0603
240Ω Resistor	4	R10,R13,R20,R23	R0603
390Ω Resistor	2	R12,R22	R0603
1uF Capacitor	2	C11,C13	C0805
100nF Capacitor	2	C12,C14	C0603
RED LED	5	LED1,LED2,LED3,LED4,LED5	R0603
2x5 female connector	1	SV3	
Molex 3 pin power connector	2	J1,J2	
100V 100uF Capacitor	1	C1	

9 Παράρτημα Β: Κώδικας εκπαίδευσης μοντέλου μηχανικής μάθησης

Η ανάπτυξη του κώδικα έγινε αρχικά σε περιβάλλον Notebook, πιο συγκεκριμένα στο Jupyter Notebook, καθώς είναι πιο εύκολη η τροποποίηση επιμέρους υπολειτουργιών του κώδικά. Αρχίζουμε εισάγοντας τις απαραίτητες βιβλιοθήκες, PyTorch, OpenCV, NumPy, Albumentations, tqdm καθώς και κάποιες λειτουργίες τους:

```
from collections import defaultdict
import copy
import random
import os
import shutil
from urllib.request import urlretrieve

import albumentations as A
import albumentations.augmentations.functional as F
from albumentations.pytorch import ToTensorV2
import cv2
import matplotlib.pyplot as plt
import numpy as np
from tqdm import tqdm
import torch
import torch.backends.cudnn as cudnn
import torch.nn as nn
import torch.optim
from torch.utils.data import Dataset, DataLoader

import torchvision.transforms as tf
import torchvision.models.segmentation
```

Στην συνέχεια διαβάζουμε τα δεδομένα εκπαίδευσης και τα αποθηκεύουμε σε λίστες. Έπειτα τυχαιοποιούμε την σειρά των δεδομένων στην λίστα.

Η τυχαιοποίηση της σειράς των δεδομένων εισόδου σε ένα νευρωνικό δίκτυο είναι σημαντική καθώς μπορεί να βοηθήσει στην αποτροπή της μάθησης των δεδομένων με συγκεκριμένη σειρά, που μπορεί να οδηγήσει σε υπερεκπαίδευση (overfitting) στο σύνολο δεδομένων εκπαίδευσης.

Θα χρησιμοποιήσουμε τις πρώτες X εικόνες για εκπαίδευση (training set) και τις υπόλοιπες Y για επικύρωση (validation set). Όταν εκπαιδεύουμε ένα μοντέλο μηχανικής μάθησης, χρησιμοποιούμε ένα σύνολο δεδομένων για να το εκπαιδεύσουμε (training set) και ένα άλλο σύνολο δεδομένων για να το αξιολογήσουμε (validation set). Η διαίρεση των δεδομένων σε training και validation sets είναι απαραίτητη για την αξιολόγηση της απόδοσης του μοντέλου μας και την αποφυγή του overfitting. Το training set χρησιμοποιείται για να εκπαιδεύσει το μοντέλο μηχανικής μάθησης. Δηλαδή, το μοντέλο εκπαιδεύεται πάνω σε αυτά τα δεδομένα και προσαρμόζει τα βάρη του έτσι ώστε να μπορεί να κάνει προβλέψεις σε νέα δεδομένα. Το validation set χρησιμοποιείται για να αξιολογηθεί η απόδοση του μοντέλου σε δεδομένα που δεν έχουν χρησιμοποιηθεί για την εκπαίδευσή του. Έτσι, μπορούμε να ελέγξουμε εάν το μοντέλο έχει μάθει καλά τα χαρακτηριστικά των δεδομένων και μπορεί να γενικεύσει σε νέα δεδομένα.

Επίσης ορίζουμε μία συνάρτηση (Η έξοδός της φαίνεται στο Σχήμα 9-1) για την απεικόνιση τους, για να βεβαιωθούμε πως είναι στην σωστή μορφή για είσοδο και έξοδο στο μοντέλο:

```
original_width = 2704
original_height = 2032

dataset_directory = os.path.join("./lab_pics")
root_directory = os.path.join(dataset_directory)
images_directory = os.path.join(root_directory)
masks_directory = os.path.join(root_directory, "labels")

images_filenames = list(sorted(os.listdir(masks_directory)))
correct_images_filenames = [i for i in images_filenames if
cv2.imread(os.path.join(images_directory, i)) is not None]

random.seed(40)
random.shuffle(correct_images_filenames)

train_images_filenames = correct_images_filenames[:60]
val_images_filenames = correct_images_filenames[60:]
test_images_filenames = images_filenames[-10:]

print(len(train_images_filenames), len(val_images_filenames),
len(test_images_filenames))

def display_image_grid(images_filenames, images_directory, masks_directory,
predicted_masks=None):
  cols = 3 if predicted_masks else 2
```

```
  rows = len(images_filenames)
  figure, ax = plt.subplots(nrows=rows, ncols=cols, figsize=(10, 24))
  for i, image_filename in enumerate(images_filenames):
    image = cv2.imread(os.path.join(images_directory, image_filename))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    mask = cv2.imread(os.path.join(masks_directory, image_filename),0)
    ax[i, 0].imshow(image)
    ax[i, 1].imshow(mask, interpolation="nearest")

    ax[i, 0].set_title("Image")
    ax[i, 1].set_title("Ground truth mask")

    ax[i, 0].set_axis_off()
    ax[i, 1].set_axis_off()

    if predicted_masks:
      predicted_mask = predicted_masks[i]
      ax[i, 2].imshow(predicted_mask, interpolation="nearest")
      ax[i, 2].set_title("Predicted mask")
      ax[i, 2].set_axis_off()
  plt.tight_layout()
  plt.show()
```

Σχήμα 9-1. Έξοδος της συνάρτησης display_image().

Έπειτα υπολογίζουμε την μέση τιμή και την διασπορά των pixels σε όλο το σύνολο εκπαίδευσης των εικόνων εισόδου. Στην επόμενη παράγραφο θα εξηγήσουμε γιατί χρειάζονται αυτές οι τιμές.

```
mean = np.array([0.,0.,0.])
stdTemp = np.array([0.,0.,0.])
std = np.array([0.,0.,0.])

numSamples = len(images_filenames)

for i,image_filename in enumerate(images_filenames):
  im = cv2.imread(os.path.join(images_directory, image_filename))
  im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
  im = im.astype(float) / 255.0

  for j in range(3):
    mean[j] += np.mean(im[:,:,j])

mean = (mean/numSamples)

print(mean) # [0.5359456 0.57724129 0.54770349]

for i,image_filename in enumerate(images_filenames):
  im = cv2.imread(os.path.join(images_directory, image_filename))
  im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
  im = im.astype(float) / 255.
  for j in range(3):
    stdTemp[j] += ((im[:,:,j] - mean[j])**2).sum()/(im.shape[0]*im.shape[1])

std = np.sqrt(stdTemp/numSamples)

print(std) #[0.23438126 0.23149222 0.26260811]
```

Στην συνέχεια ορίζουμε μία κλάση **officeDataset** για τα δεδομένα εκπαίδευσης από το αμπέλι του εργαστηρίου. Η δημιουργία της κλάσης μας βοηθάει ώστε ανεξάρτητα από την αρχική μορφή των δεδομένων εκπαίδευσης να επιστρέφονται στο μοντέλο σε μία μορφή με την οποία είναι συμβατά. Για παράδειγμα μπορεί σε κάποια καινούρια δεδομένα εκπαίδευσης να υπάρχει διαφορετική αντιστοιχία μεταξύ των ονομάτων αρχείων εικόνων εισόδου και εξόδου ή να θέλει κάποια διαφορετική προεπεξεργασία η τμηματοποιημένη εικόνα. Δημιουργούμε λοιπόν για κάθε διαφορετική μορφή δεδομένων εκπαίδευσης την δική της κλάση. Όλες οι κλάσεις ανεξαρτήτως της αρχικής μορφής των δεδομένων τα επιστρέφουν σε μορφή συμβατή με το μοντέλο.

Αφού έχουμε ορίσει την κλάση officeDataset, δημιουργούμε τα αντικείμενα της κλάσης αυτής **train_dataset** και **validation_dataset** που περιλαμβάνουν τα δεδομένα για την εκπαίδευση και την αξιολόγηση του μοντέλου αντίστοιχα. Για τα δύο αυτά σετ δεδομένων ορίζουμε και τους αντίστοιχους μετασχηματισμούς **train_transform** και **val_transform** όπως περιγράφονται στο στο κεφάλαιο 4.3.4 μέσω της βιβλιοθήκης Albumentations. Θα

επαυξήσουμε προφανώς μόνο τα δεδομένα εκπαίδευσης, οπότε στο train_transform ορίζουμε μετασχηματισμούς περιστροφής, μετατόπισης, ζουμ, φωτεινότητας/αντίθεσης καθώς και τις αντίστοιχες τιμές τους παραμετροποίησης. Τέλος, κανονικοποιούμε την μέση τιμή και την διασπορά των pixel της κάθε εικόνας στις αντίστοιχες τιμές που υπολογίσαμε για το σύνολο των δεδομένων εκπαίδευσης.

Η κανονικοποίηση των δεδομένων εικόνων σε εφαρμογές μηχανικής μάθησης είναι σημαντική καθώς μειώνει την διακύμανση των δεδομένων εικόνας και επιτρέπει στο μοντέλο να συγκεντρωθεί σε κοινές ομοιότητες/διαφορές σε όλο το σύνολο δεδομένων, αντί να επηρεαστεί από την έλλειψη ομοιογένειας. Μπορεί λοιπόν να βοηθήσει στην αποφυγή του overfitting. Ουσιαστικά, η κανονικοποίηση μπορεί να εξομαλύνει τα δεδομένα εικόνας και να μειώσει τις ακραίες τιμές, κάτι που μπορεί να βοηθήσει στην αποφυγή της υπερβολικής προσαρμογής του μοντέλου.

Ο μετασχηματισμός του συνόλου των δεδομένων αξιολόγησης προφανώς πραγματοποιεί μόνο κανονικοποίηση στα δεδομένα με τις ίδιες τιμές, χωρίς καμία επαύξηση.

```
class officeDataset(Dataset):
  def __init__(self, images_filenames, images_directory, masks_directory,
transform=None):
    self.images_filenames = images_filenames
    self.images_directory = images_directory
    self.masks_directory = masks_directory
    self.transform = transform

  def __len__(self):
    return len(self.images_filenames)

  def __getitem__(self, idx):
    image_filename = self.images_filenames[idx]
    image = cv2.imread(os.path.join(self.images_directory,image_filename))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    mask = cv2.imread(os.path.join(masks_directory, image_filename),0)
    mask = mask.astype(float) / 255
    mask[mask>0.5]=1.0
    mask[mask<=0.5]=0
    if self.transform is not None:
      transformed = self.transform(image=image, mask=mask)
      image = transformed["image"]
      mask = transformed["mask"]
    return image, mask

train_transform = A.Compose([A.Resize(600,600),
A.ShiftScaleRotate(shift_limit=0.2, scale_limit=0.2,rotate_limit=30,p=0.5),
A.RGBShift(r_shift_limit=25,g_shift_limit=25, b_shift_limit=25, p=0.5),
A.RandomBrightnessContrast(brightness_limit=0.3, contrast_limit=0.3, p=0.5),
A.Normalize(mean=(0.49698316, 0.51454199, 0.43739734),
std=(0.20042156,0.20729762, 0.30175445)),ToTensorV2(),])
train_dataset = officeDataset(train_images_filenames, images_directory,
masks_directory, transform=train_transform,)
```

```
val_transform = A.Compose(
  [A.Resize(600, 600), A.Normalize(mean=(0.49698316, 0.51454199, 0.43739734),
std=(0.20042156, 0.20729762, 0.30175445)), ToTensorV2()]
)
val_dataset = officeDataset(val_images_filenames, images_directory,
masks_directory, transform=val_transform,)
```

Για την οπτικοποίηση των μετασχηματισμών και την επιβεβαίωση ότι πραγματοποιούνται σωστά, ορίζουμε την συνάρτηση visualize_augmentations(). Η έξοδός της φαίνεται στο Σχήμα 4-7. Η συνάρτηση αυτή επιστρέφει μία επιλεγμένη εικόνα όπως επηρεάζεται από τον κάθε μετασχηματισμό:

```
def visualize_augmentations(dataset, idx=0, samples=30):
  dataset = copy.deepcopy(dataset)
  dataset.transform = A.Compose([t for t in dataset.transform if not
isinstance(t, (A.Normalize, ToTensorV2))])
  figure, ax = plt.subplots(nrows=samples, ncols=2, figsize=(10, 24))
  for i in range(samples):
    image, mask = dataset[idx]
    ax[i, 0].imshow(image)
    ax[i, 1].imshow(mask, interpolation="nearest")
    ax[i, 0].set_title("Augmented image")
    ax[i, 1].set_title("Augmented mask")
    ax[i, 0].set_axis_off()
    ax[i, 1].set_axis_off()
  plt.tight_layout()
  plt.show()

visualize_augmentations(train_dataset, idx=5)
```

Έπειτα ορίζουμε ένα πλήθος συναρτήσεων και κλάσεων για την εκπαίδευση του μοντέλου. Η κλάση **MetricMonitor** βοηθά στην παρακολούθηση μετρήσεων όπως η ακρίβεια (accuracy) ή η συνάρτηση κόστους (Loss function) κατά τη διάρκεια της εκπαίδευσης και της επικύρωσης.

```
class MetricMonitor:
  def __init__(self, float_precision=3):
    self.float_precision = float_precision
    self.reset()

  def reset(self):
    self.metrics = defaultdict(lambda: {"val": 0, "count": 0, "avg": 0})

  def update(self, metric_name, val):
    metric = self.metrics[metric_name]

    metric["val"] += val
    metric["count"] += 1
    metric["avg"] = metric["val"] / metric["count"]

  def __str__(self):
    return " | ".join(
      [
        "{metric_name}: {avg:.{float_precision}f}".format(
          metric_name=metric_name, avg=metric["avg"],
float_precision=self.float_precision
        )
        for (metric_name, metric) in self.metrics.items()
      ]
    )
```

Ορισμός συναρτήσεων για εκπαίδευση και επικύρωση:

```
def train(train_loader, model, criterion, optimizer, epoch, params):
  metric_monitor = MetricMonitor()
  model.train()
  stream = tqdm(train_loader)
  for i, (images, target) in enumerate(stream, start=1):
    images = images.to(params["device"], non_blocking=True)
    target = target.to(params["device"], non_blocking=True)
    output = model(images)['out']
    loss = criterion(output, target.long())
    metric_monitor.update("Loss", loss.item())
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    stream.set_description(
      "Epoch: {epoch}. Train.   {metric_monitor}".format(epoch=epoch,
metric_monitor=metric_monitor)
    )
```

```
def validate(val_loader, model, criterion, epoch, params):
  metric_monitor = MetricMonitor()
  model.eval()
  stream = tqdm(val_loader)
  with torch.no_grad():
    for i, (images, target) in enumerate(stream, start=1):
      images = images.to(params["device"], non_blocking=True)
      target = target.to(params["device"], non_blocking=True)
      output = model(images)['out']
      loss = criterion(output, target.long())
      metric_monitor.update("Loss", loss.item())
      stream.set_description(
        "Epoch: {epoch}. Validation. {metric_monitor}".format(epoch=epoch,
metric_monitor=metric_monitor)
      )

def create_model(params):
  model = torchvision.models.segmentation.deeplabv3_resnet50(pretrained=True)
# Load net
  model.classifier[4] = torch.nn.Conv2d(256, 2, kernel_size=(1, 1), stride=(1,
1)) # Change final layer to 2 classes
  model = model.to(params["device"])
  return model

def train_and_validate(model, train_dataset, val_dataset, params):
  train_loader = DataLoader(
    train_dataset,
    batch_size=params["batch_size"],
    shuffle=True,
    num_workers=params["num_workers"],
    pin_memory=True,
  )
  val_loader = DataLoader(
    val_dataset,
    batch_size=params["batch_size"],
    shuffle=False,
    num_workers=params["num_workers"],
    pin_memory=True,
  )
  criterion = torch.nn.CrossEntropyLoss().to(params["device"])
  optimizer = torch.optim.Adam(model.parameters(), lr=params["lr"])
  for epoch in range(1, params["epochs"] + 1):
    train(train_loader, model, criterion, optimizer, epoch, params)
    validate(val_loader, model, criterion, epoch, params)
    print("Saving Model" +str(epoch) + ".torch")
    torch.save(model.state_dict(),  str(epoch) + ".torch")
  return model
```

```
def predict(model, params, test_dataset, batch_size):
  test_loader = DataLoader(
    test_dataset, batch_size=batch_size, shuffle=False,
num_workers=params["num_workers"], pin_memory=True,
  )
  model.eval()
  predictions = []
  with torch.no_grad():
    for images, (original_heights, original_widths) in test_loader:
      images = images.to(params["device"], non_blocking=True)
      output = model(images)['out']
      probabilities = torch.sigmoid(output)
      predicted_masks = (probabilities >= 0.5).float() * 1
      predicted_masks = predicted_masks.cpu().numpy()
      for predicted_mask, original_height, original_width in zip(
        predicted_masks, original_heights.numpy(), original_widths.numpy()
      ):
        predictions.append((predicted_mask, original_height, original_width))
  return predictions
```

10 Παράρτημα Γ: Κώδικας ελέγχου κατεύθυνσης του ρομπότ

Η ανάπτυξη του κώδικα έλεγχου του robot έγινε στο Visual Studio Code απευθείας σε ένα αρχείο Python (infer_zed.py). Ο κώδικας αυτός εκτελεί το state machine που περιγράψαμε στο κεφάλαιο 4.5.5. Αρχίζουμε εισάγοντας τις απαραίτητες βιβλιοθήκες για την τμηματοποίηση της εικόνας και εξαγωγή της απαιτούμενης πληροφορίας: PyTorch, OpenCV, NumPy καθώς και την βιβλιοθήκη PyZed για την επικοινωνία με την Zed camera και τους αισθητήρες της:

```
import cv2
import torchvision.models.segmentation
import torch
import torchvision.transforms as tf
import matplotlib.pyplot as plt
import os
import numpy as np
from torch.utils.data import Dataset, DataLoader
import random
import albumentations as A
from albumentations.pytorch import ToTensorV2
import pyzed.sl as sl
import time
import sys
import math
import socket
```

Έπειτα ορίζονται διάφοροι παράμετροι. Το πρώτο ζευγάρι **udp παραμέτρων** ορίζει την διεύθυνση και την θύρα του μικροϋπολογιστή Raspberry Pi στον οποίο στέλνονται οι εντολές κινήσεις μέσω του πρωτόκολλου UDP. Έπειτα με την παράμετρο **VELOCITY** ορίζεται η γραμμική ταχύτητα του robot σε οποιαδήποτε κίνηση και αν κάνει. Η παράμετρος **analysis_size** ορίζει το μέγεθος που γίνεται resize η εικόνα της κάμερας πριν την είσοδό της στο μοντέλο μηχανικής μάθησης. Μικρότερο μέγεθος οδηγεί σε ταχύτερη εκτέλεση του μοντέλου με κόστος χειρότερη ακρίβεια. Για analysis_size=480 παρατηρήσαμε πολύ καλή ακρίβεια, στον χώρο του εργαστηρίου με ταχύτητα εκτέλεσης του κύριου βρόχου ελέγχου γύρω στο 1.2Hz.

```
UDP_IP = "192.168.0.100"
UDP_PORT = 1234

#SPEED
VELOCITY = 3000

#IN VINEYARD PARAMETERS
#image size fed to CNN (smaller is faster, but loss on detail)
analysis_size = 480
```

```
#%threshold of deviation from center to begin steering
steer_sensitivity=7


#TURNING PARAMETERS
#percentage of white pixels to background to initiate rotation
end_thres=5

#for how much time to move forward during Π maneuver
forward_interval=10
backward_interval=0

#for how much time to move side during Π manuevre (depends on row spacing)
sideways_interval=10

#direction of rotation
ROT = False


#MODEL SETUP
#path of model
model_name = "entire_model_70samples.pt"

#for image transformations
model_mean = (0.49698316, 0.51454199, 0.43739734)
model_std = (0.20042156, 0.20729762, 0.30175445)
```

Έπειτα ορίζουμε τις συναρτήσεις για την αποστολή πακέτων UDP με τις εντολές κίνησης στο RPi που ελέγχει το ρομπότ. Η συνάρτηση **send_speed** δέχεται 3 παραμέτρους την γραμμική ταχύτητα ως προς **x** και **y** καθώς και την γωνιακή ταχύτητα **ω**. Το robot μόλις δοθεί μία τέτοια εντολή διατηρεί τις δοσμένες αυτές ταχύτητες.

```
def print_udp(msg):
  sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  sock.sendto(bytes(msg, "utf-8"), (UDP_IP, UDP_PORT))
  sock.close()
  print("UDP -> " + msg)

def send_speed(x,y,rot):
  print_udp('ix' + str(x) + 'y' + str(y) + 'r' + str(int(rot)))
```

Έπειτα ορίζουμε μία συνάρτηση για την μετατροπή του συστήματος προσανατολισμού από τετραδόνια (quaternions) σε γωνίες Euler. Η συνάρτηση αυτή χρησιμοποιείται καθώς η βιβλιοθήκη PyZed που διαβάζει τους αισθητήρες της Zed επιστρέφει τον προσανατολισμό μόνο σε τετραδόνια.

```
def euler_from_quaternion(w,x, y, z):
  ysqr = y * y

  t0 = +2.0 * (w * x + y * z)
  t1 = +1.0 - 2.0 * (x * x + ysqr)
  X = np.degrees(np.arctan2(t0, t1))

  t2 = +2.0 * (w * y - z * x)

  t2 = np.clip(t2, a_min=-1.0, a_max=1.0)
  Y = np.degrees(np.arcsin(t2))

  t3 = +2.0 * (w * z + x * y)
  t4 = +1.0 - 2.0 * (ysqr + z * z)
  Z = np.degrees(np.arctan2(t3, t4))

  return X, Y, Z
```

Τέλος ορίζουμε μία συνάρτηση που μας επιτρέπει να εισάγουμε στο μοντέλο μηχανικής μάθησης μία εικόνα την φορά αντί για ένα batch. Η συνάρτηση αυτή απλώς προσθέτει άλλη μία διάστασή στην εικόνα με μέγεθος 1, μέσω της εντολής `unsqueeze(0)`, δημιουργώντας ένα batch με μία μόνο εικόνα.

```
#function for predicting single images
def predict_single(model, params, img):
  test_transform = A.Compose(
  [A.Resize(analysis_size, analysis_size), A.Normalize(mean=model_mean,
std=model_std), ToTensorV2()])

  img = test_transform(image=img)
  img = img["image"]
  img = img.unsqueeze(0)

  model.eval()
  predictions = []
  with torch.no_grad():
    img = img.to(params["device"], non_blocking=True)
    output = model(img)['out']
    probabilities = torch.sigmoid(output)
    predicted_masks = (probabilities >= 0.5).float() * 1
    predicted_masks = predicted_masks.cpu().numpy()
  return predicted_masks
```

Ο παρακάτω κώδικάς φορτώνει το μοντέλο μηχανικής μάθησης, ενεργοποιεί την λειτουργία της κάμερας Zed σε ανάλυση 720p και αρχικοποιεί διάφορές μεταβλητές που χρησιμοποιούνται στον κύριο βρόχο του προγράμματος

```
#load neural model to gpu
device = torch.device('cuda')
Net = torch.load(model_name)
Net = Net.to(device)
Net.eval()

#ZED SETUP
# Create a ZED camera object
zed = sl.Camera()

# Set configuration parameters
input_type = sl.InputType()
init = sl.InitParameters(input_t=input_type)
init.camera_resolution = sl.RESOLUTION.HD720
# Open the camera
err = zed.open(init)
if err != sl.ERROR_CODE.SUCCESS :
  print(repr(err))
  zed.close()
  exit(1)

# Set runtime parameters after opening the camera
runtime = sl.RuntimeParameters()
runtime.sensing_mode = sl.SENSING_MODE.STANDARD

# Prepare new image size to retrieve half-resolution images
image_size = zed.get_camera_information().camera_resolution
image_size.width = image_size.width
image_size.height = image_size.height

# Declare your sl.Mat matrices
image_zed = sl.Mat(image_size.width, image_size.height, sl.MAT_TYPE.U8_C4)

#sensor data
sensors_data = sl.SensorsData()

#master switch for robot control
master=True

#variable for measuring FPS
prev_time = time.time()

#pixel count variable
pixel_count=0
init_turn = False
init_turn_master = True
init_angle = 0
```

Ακολουθεί ο κύριος βρόχος του προγράμματος, που αποτελείται από ένα while loop. Ουσιαστικά εδώ εκτελούνται οι λειτουργίες που περιγράφτηκαν στο state machine. Αρχικά, το πρόγραμμα αρχικά ελέγχει αν πατήθηκαν κάποια πλήκτρα q, space, t και αντίστοιχα τερματίζει, ενεργοποιεί/απενεργοποιεί τον έλεγχο κατεύθυνσης, ενεργοποιεί/απενεργοποιεί την διαδικασία αλλαγής σειράς αμπελιού.

```
key = ''
while key != 115 :
  if (key == 113): #press q to terminate udp session
    print_udp('b')
    print('q pressed - exiting..')
    exit(1)


  if (key == 32):
    print('space pressed - toggling master')
    if (master == True):
      master = False
      print_udp('b')
    else:
      master = True

  if (key == 116):
    print('t pressed - toggling turning function')
    if (init_turn_master == True):
      init_turn_master = False
    else:
      init_turn_master = True
```

Έπειτα καταγράφει την εικόνα από την κάμερα, γίνεται resize στο μέγεθος που έχει οριστεί και δίνεται ως είσοδος στο μοντέλο μηχανικής μάθησης, ενώ παράλληλα απεικονίζεται και σε ένα παράθυρο με επιπρόσθετες πληροφορίες στον τίτλο, όπως την ανάλυση, FPS. Η τελική έξοδος αποθηκεύεται στην μεταβλητή img.

```
  err = zed.grab(runtime)
  if err == sl.ERROR_CODE.SUCCESS :
    # Retrieve the left image, depth image in the half-resolution
    zed.retrieve_image(image_zed, sl.VIEW.LEFT, sl.MEM.CPU, image_size)

    # To recover data from sl.Mat to use it with opencv, use the get_data()
method
    # It returns a numpy array that can be used as a matrix with opencv
    image_ocv = image_zed.get_data()
    cv2.imshow("In", cv2.resize(image_ocv, (840,540), interpolation =
cv2.INTER_AREA))
    time_now=time.time()
```

```
    cv2.setWindowTitle("In", "In - RES=" + str(analysis_size) + " FPS=" +
str(round((1.0 / (time_now - prev_time)),2)) + " Fill%=" +
str(pixel_count/(840*540)*100.0))
    prev_time=time_now
    image_ocv=cv2.cvtColor(image_ocv, cv2.COLOR_BGR2RGB)

    #predict classes with CNN
    out = predict_single(Net, image_ocv)[0][1]

    img = out.copy()
    width=img.shape[1]
    img = 1-img #flip 0 & 1
```

Ακολουθεί η συνάρτηση υπολογισμού της μέσης γραμμής που περιγράψαμε στο κεφάλαιο 4.5.4. Εδώ θα αναλύσουμε πώς εκτελούνται οι υπολειτουργίες της προγραμματιστικά. Για την εύρεσης της στήλης με τα περισσότερα άσπρα pixels ελέγχουμε διαδοχικά όλες τις στήλες και αποθηκεύουμε το μεγαλύτερο πλήθος που έχουμε βρει.

```
  #find widest column of image
    max_col=0
    for i in range(width):
      c=np.count_nonzero(img[:,i])
      if (c>max_col): max_col=c
```

Με βάση αυτό τον αριθμό, δημιουργούμε το κατώφλι επιλογής των ενδιάμεσων στηλών ως ένα ποσοστό αυτού του αριθμού. Για οποιαδήποτε στήλη που έχει πλήθος άσπρων pixel μικρότερο από αυτόν τον αριθμό, αποθηκεύουμε την θέση της στην λίστα **gap**. Έπειτα βρίσκουμε την μέση γραμμή που είναι το στοιχείο διάμεσος (median) αυτής της λίστας. Τέλος υπολογίζουμε την τιμή ποσοστιαίου σφάλματος όπως την περιγράψαμε στο κεφάλαιο 4.5.3 και την αποθηκεύουμε στην μεταβλητή **divergence.**

```
 #select all columns smaller than percent_thres*100% of widest column
    percent_thres=0.1
    threshold=max_col*percent_thres
    gap = []
    for i in range(img.shape[1]):
      if (np.count_nonzero(img[:,i])<threshold):
        gap.append(i)
    gap= np.array(gap)
    if (gap.any()):
      mid = gap[int(gap.shape[0]/2)] # find mid
    else:
      mid = width/2
    divergence=(mid-width/2)/(width/2)*100 # find divergence from center
(append /(width/2)*100 for percentage)
```

Το πρόγραμμα έπειτα πρέπει να αποφασίσει ποιο θα είναι το επόμενο state στο state machine όπως περιγράφεται στο κεφάλαιο 4.5.5. Υπολογίζει λοιπόν την πλήθος των άσπρων pixel μέσω της συνάρτησης NumPy **count_nonzero** και αναλόγως αν βρίσκονται πάνω ή κάτω από το κατώφλι μεταβαίνει και στην αντίστοιχη λειτουργία αλλάζοντας την τιμής της μεταβλητής **init_turn**.

```
    #check if we have reached end
    size = img.shape[0]*img.shape[1]*(end_thres/100.0)
    pixel_count = np.count_nonzero(img)
    if (pixel_count < size):
      init_turn = True
```

Αν το πρόγραμμα μεταβεί σε vision control state, εκτελεί proportional control στην γωνιακή ταχύτητα ω του robot βάση του ποσοστού σφάλματος που είχαμε προηγουμένως αποθηκεύσει στην μεταβλητή divergence, διατηρώντας σταθερή γραμμική εμπρόσθια ταχύτητα μετατόπισης x, όπως περιγράφεται και στο κεφάλαιο 4.5.3.

```
    if (master == True and init_turn == False):
      print(divergence)
      if (divergence>steer_sensitivity):
        print('a')
        send_speed(VELOCITY,0,divergence/100*VELOCITY)
      if (divergence<-steer_sensitivity):
        print('b')
        send_speed(VELOCITY,0,divergence/100*VELOCITY)
      if (abs(divergence)<=steer_sensitivity):
        send_speed(VELOCITY,0,0)
```

Αντιθέτως, αν το πρόγραμμα μεταβεί σε maneuver execution state, εκτελεί πρώτα μία μετατόπιση ευθεία ώστε να είναι αρκετά απομακρυσμένο από το αμπέλι για να μπορεί να στρίψει εκατέρωθεν προς την επόμενη σειράς, καταγράφοντας και τον αρχικό του προσανατολισμό μέσω του επιταχυνσιομετρού της Zed.

```
print("INITIALISED TURNING SEQUENCE")
      #get sensors
      zed.get_sensors_data(sensors_data, sl.TIME_REFERENCE.CURRENT)
      q = sensors_data.get_imu_data().get_pose().get_orientation().get()
      x_deg,y_deg,_z_deg = euler_from_quaternion(q[0],q[1],q[2],q[3])
      init_angle = y_deg

      #now we perform Π turn

      #move forward
      print("MOVING FORWARD")
      send_speed(VELOCITY,0,0)
      time.sleep(forward_interval)
```

Έπειτα εκτελεί μία περιστροφή 90 μοιρών γύρω από τον εαυτό του με βάση τον προσανατολισμό του επιταχυνσιομέτρου.

```
#start rotating
print("STARTING FIRST ROTATION")
print("initial angle = " + str(init_angle))
if (ROT):
  send_speed(0,0,VELOCITY)
else:
  send_speed(0,0,-VELOCITY)

#rotate up to 90 degrees
total=90
prev_angle=init_angle

while (total>0):
  zed.grab(runtime)
  zed.get_sensors_data(sensors_data, sl.TIME_REFERENCE.CURRENT)
  q = sensors_data.get_imu_data().get_pose().get_orientation().get()
  x_deg,y_deg,_z_deg = euler_from_quaternion(q[0],q[1],q[2],q[3])
  if (int(y_deg) != int(prev_angle)):
    diff = abs(int(y_deg)-int(prev_angle))
    prev_angle = int(y_deg)
    total=total-diff
  print("current angle=" + str(int(y_deg)) + " total=" + str(total))
```

Παρομοίως εκτελεί μία μετατόπιση ευθεία αλλά επειδή πλέον βρίσκεται κάθετα στο αμπέλι θα αλλάξει σειρά. Έπειτα κάνει ξανά μία περιστροφή 90 μοιρών για να ευθυγραμμιστεί με την επόμενη σειρά και μεταβαίνει ξανά στο Vision Control state, ορίζοντας την μεταβλητή **init_turn** σε **False**. Έτσι αν το ρομπότ λόγω ανωμαλιών στο έδαφος δεν έχει βρεθεί ακριβώς στο κέντρο της σειράς ή έχει αποκλίνων προσανατολισμό θα ευθυγραμμιστεί αυτόματα καθώς μπαίνει στο αμπέλι.

```
 #move sideways
      print("MOVING SIDEWAYS")
      send_speed(VELOCITY,0,0)
      time.sleep(sideways_interval)

      #start second rotation
      zed.get_sensors_data(sensors_data, sl.TIME_REFERENCE.CURRENT)
      q = sensors_data.get_imu_data().get_pose().get_orientation().get()
      x_deg,y_deg,_z_deg = euler_from_quaternion(q[0],q[1],q[2],q[3])
      init_angle = y_deg
      print("STARTING SECOND ROTATION")
      print("initial angle = " + str(init_angle))
      if (ROT):
        send_speed(0,0,VELOCITY)
      else:
        send_speed(0,0,-VELOCITY)

      #rotate up to 90 degrees
      total=90
      prev_angle=init_angle
      while (total>0):
        zed.grab(runtime)
        zed.get_sensors_data(sensors_data, sl.TIME_REFERENCE.CURRENT)
        q=sensors_data.get_imu_data().get_pose().get_orientation().
get()
        x_deg,y_deg,_z_deg = euler_from_quaternion(q[0],q[1],q[2],q[3])
        if (int(y_deg) != int(prev_angle)):
          diff = abs(int(y_deg)-int(prev_angle))
          prev_angle = int(y_deg)
          total=total-diff
        print("current angle=" + str(int(y_deg)) + " total=" + str(total))

      #move backwards
      print("MOVING BACKWARD")
      send_speed(VELOCITY,0,0)
      time.sleep(backward_interval)
      init_turn=False

      print("RE-ENTERING IMAGE CONTROL")
```